БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра информатики

Факультет НиДО

Специальность ИиТП

Контрольная работа № 2

по дисциплине «Проектирование программного обеспечения»

Выполнил студент: Дегтярев А.А.

группа 393551

Зачетная книжка № 952004-7

Минск 2015

**Вариант 3  
Обзор методов тестирования ПО**

**Тестирование программного обеспечения**— процесс исследования, испытания программного обеспечения с целью получения информации о качестве продукта.

ПО – комплексный продукт, и однозначно оценить его качество невозможно. Чтобы упростить процедуру и повысить точность оценки выделяют определенные критерии качества, по которым проверяют продукт. Так, например, стандарт ISO-9126 выделяет следующие компоненты:

- Надёжность

- Сопровождаемость

- Практичность

- Эффективность

- Мобильность

- Функциональность

Все они в совокупности и характеризуют качество оцениваемого программного продукта.  
  
Методы тестирование ПО можно классифицировать по следующим критериям:

**По объекту тестирования**

**Функциональное тестирование** - это тестирование ПО в целях проверки реализуемости функциональных требований (функциональная пригодность, точность, способность к взаимодействию, соответствие стандартам и правилам, защищённость), то есть способности ПО в определённых условиях решать задачи, нужные пользователям. Функциональные требования определяют, что именно делает ПО, какие задачи оно решает.

**Тестирование производительности -** проводится с целью определения, как быстро работает 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.) **Подразделяется на:**

**Нагрузочное тестирование** — это простейшая форма тестирования производительности. обычно проводится для того, чтобы оценить поведение приложения под заданной ожидаемой нагрузкой. Этой нагрузкой может быть, например, ожидаемое количество одновременно работающих пользователей приложения, совершающих заданное число транзакций за интервал времени. Такой тип тестирования обычно позволяет получить время отклика всех самых важных бизнес-транзакций. В случае наблюдения за базой данных, сервером приложений, сетью и т. д., этот тип тестирования может также идентифицировать некоторые узкие места приложения.

**Стресс-тестирование** обычно используется для понимания пределов пропускной способности приложения. Этот тип тестирования проводится для определения надёжности системы во время экстремальных или диспропорциональных нагрузок и отвечает на вопросы о достаточной производительности системы в случае, если текущая нагрузка сильно превысит ожидаемый максимум.

**Тестирование стабильности** проводится с целью убедиться в том, что приложение выдерживает ожидаемую нагрузку в течение длительного времени. При проведении этого вида тестирования осуществляется наблюдение за потреблением приложением памяти, чтобы выявить потенциальные утечки. Также важным, но часто незамеченным, фактором является деградация производительности, смысл которого сводится к тому, чтобы скорость обработки информации и/или время ответа приложения через длительное время работы были такими же или лучше, чем в самом начале теста.

**Юзабилити-тестирование** (***тестирование эргономики***) — исследование, выполняемое с целью определения, удобен ли некоторый искусственный объект (такой как веб-страница,пользовательский интерфейс или устройство) для его предполагаемого применения. Таким образом, проверка эргономичности измеряет эргономичность объекта или системы. Проверка эргономичности сосредоточена на определённом объекте или небольшом наборе объектов, в то время как исследования взаимодействия человек-компьютер в целом — формулируют универсальные принципы.

**Тестирование безопасности** — оценка уязвимости программного обеспечения к различным атакам.В ходе тестирования безопасности испытатель играет роль взломщика. Ему разрешено все:

попытки узнать пароль с помощью внешних средств;

атака системы с помощью специальных утилит, анализирующих защиты;

подавление, ошеломление системы (в надежде, что она откажется обслуживать других клиентов);

целенаправленное введение ошибок в надежде проникнуть в систему в ходе восстановления;

просмотр несекретных данных в надежде найти ключ для входа в систему.

**Тестирование совместимости**

Тестирование совместимости — вид нефункционального тестирования, основной целью которого является проверка корректной работы продукта в определенном окружении.

**По знанию системы**

**Тестирование чёрного ящика -** в этой технике программа рассматривается как чёрный ящик. Целью тестирования ставится выяснение обстоятельств, в которых поведение программы не соответствует спецификации. Для обнаружения всех ошибок в программе необходимо выполнить *исчерпывающее тестирование*, то есть тестирование на всевозможных наборах данных. Для большинства программ такое невозможно, поэтому тестирование программы ограничивается небольшим подмножеством всевозможных наборов данных. При этом необходимо выбирать наиболее подходящие подмножества, подмножества с наивысшей вероятностью обнаружения ошибок.

*Основные техники чёрного ящика*

Эквивалентное разбиение.

Анализ граничных значений.

Анализ причинно-следственных связей.

Предположение об ошибке.

**Тестирование белого ящика -** тестирование кода на предмет логики работы программы и корректности её работы с точки зрения компилятора того языка на котором она писалась. также называемая **техникой тестирования управляемой логикой программы**, позволяет проверить внутреннюю структуру программы. Исходя из этой стратегии тестировщик получает тестовые данные путем анализа логики работы программы.

Техника Белого ящика включает в себя следующие методы тестирования:

покрытие операторов

покрытие решений

покрытие условий

покрытие решений и условий

комбинаторное покрытие условий

**По степени автоматизации**

**Ручное тестирование -** генерируют ошибочные ситауции, нагрузку, вводят некорректные данные, проводят проверку по граничным условиям и т.д.

**Автоматизированное тестирование -** Существует два основных подхода к автоматизации тестирования: тестирование на уровне кода и тестирование пользовательского интерфейса (в частности, GUI-тестирование). К первому типу относится, в частности, модульное тестирование. Ко второму — имитация действий пользователя с помощью специальных тестовых фреймворков.

**По степени изолированности компонентов:**

**Модульное тестирование**, или **юнит-тестирование** — процесс в программировании, позволяющий проверить на корректность отдельные модули исходного кода программы. Идея состоит в том, чтобы писать тесты для каждой нетривиальной функции или метода. Это позволяет достаточно быстро проверить, не привело ли очередное изменение кода к *регрессии*, то есть к появлению ошибок в уже оттестированных местах программы, а также облегчает обнаружение и устранение таких ошибок. Цель модульного тестирования — изолировать отдельные части программы и показать, что по отдельности эти части работоспособны.Этот тип тестирования обычно выполняется программистами.

**Интеграционное тестирование** — одна из фаз тестирования программного обеспечения, при которой отдельные программные модули объединяются и тестируются в группе. Обычно интеграционное тестирование проводится после модульного тестирования и предшествует системному тестированию.

Интеграционное тестирование в качестве входных данных использует модули, над которыми было проведено модульное тестирование, группирует их в более крупные множества, выполняет тесты, определённые в плане тестирования для этих множеств, и представляет их в качестве выходных данных и входных для последующего системного тестирования.

Целью интеграционного тестирования является проверка соответствия проектируемых единиц функциональным, приёмным и требованиям надежности. Тестирование этих проектируемых единиц — объединения, множества или группы модулей — выполняется через их интерфейс, с использованием тестирования «чёрного ящика».

**Системное тестирование** программного обеспечения — это тестирование ПО, выполняемое на полной, интегрированной системе, с целью проверки соответствия системы исходным требованиям. Системное тестирование относится к методам тестирования чёрного ящика, и, тем самым, не требует знаний о внутреннем устройстве системы. Основной задачей системного тестирования является проверка как функциональных, так и не функциональных требований к системе в целом. При этом выявляются дефекты, 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.

Можно выделить **два подхода к системному тестированию**:

-на базе требований (requirements based)

Для каждого требования пишутся тестовые случаи (test cases), проверяющие выполнение данного требования.

-на базе случаев использования (use case based)

Альфа-тестирование и бета-тестирование являются подкатегориями системного тестирования.

**По времени проведения тестирования**

**Альфа-тестирование** — имитация реальной работы с системой штатными разработчиками, либо реальная работа с системой потенциальными пользователями/заказчиком. Чаще всего альфа-тестирование проводится на ранней стадии разработки продукта, но в некоторых случаях может применяться для законченного продукта в качестве внутреннего приёмочного тестирования. Иногда альфа-тестирование выполняется под отладчиком или с использованием окружения, которое помогает быстро выявлять найденные ошибки. Обнаруженные ошибки могут быть переданы тестировщикам для дополнительного исследования в окружении, подобном тому, в котором будет использоваться ПО.

**Тестирование при приёмке**

**Smoke Test** означает минимальный набор тестов на явные ошибки и обычно выполняется самим программистом; не проходящую этот тест программу не имеет смысла отдавать на более глубокое тестирование.Их легче автоматизировать, что снижает количество ручного труда и поэтому позволяет проводить эти тесты чаще. Автоматизация тестирования часто выполняется с помощью средств непрерывной интеграции.

**Регрессио́нное тести́рование** — собирательное название для всех видов тестирования программного обеспечения, направленных на обнаружение ошибок в уже протестированных участках исходного кода. Такие ошибки — когда после внесения изменений в программу перестает работать то, что должно было продолжать работать, — называют *регрессионными ошибками*. Регрессионное тестирование может быть использовано не только для проверки корректности программы, часто оно также используется для оценки качества полученного результата. Так, при разработке компилятора, при прогоне регрессионных тестов рассматривается размер получаемого кода, скорость его выполнения и время компиляции каждого из тестовых примеров.

**Бета-тестирование** — в некоторых случаях выполняется распространение предварительной версии (в случае проприетарного ПО иногда с ограничениями по функциональности или времени работы) для некоторой большей группы лиц, с тем чтобы убедиться, что продукт содержит достаточно мало ошибок. Иногда бета-тестирование выполняется для того, чтобы получить обратную связь о продукте от его будущих пользователей.Бета-версия не является финальной версией продукта, поэтому разработчик не гарантирует полного отсутствия ошибок, которые могут нарушить работу компьютера и/или привести к потере данных.