МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования
«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)

Кафедра автоматизации обработки информации (АОИ)

Утверждаю:
Зав. кафедрой АОИ
профессор
_____Ю.П.Ехлаков
«___»_____2013 г.

Методические указания для выполнения лабораторных работ и организации самостоятельной работы по дисциплине Тестирование программного обеспечения

для студентов направления подготовки
231000.62 «Программная инженерия»

Разработчик:
_____ С.В. Лучкова

Томск - 2013

Содержание

Введение

В процессе подготовки и выполнения лабораторных работ формируются следующие компетенции:

1) умение применять основы информатики и программирования к проектированию, конструированию и тестированию программных продуктов (ПК-10);

2) навыки чтения, понимания и выделения главной идеи прочитанного исходного кода, документации (ПК-11);

3) понимание концепций и атрибутов качества программного обеспечения (надежности, безопасности, удобства использования), в том числе роли людей, процессов, методов, инструментов и технологий обеспечения качества (ПК-18);

4) понимание методов контроля проекта и умение осуществлять контроль версий (ПК-25).

В качестве технологии интерактивного обучения выбрана «работа в команде» и «мозговая атака», используемые для второй приведенной в методических указаниях лабораторной работе.

Форма контроля компетенций: устные опросы на лабораторных работах, проверка программного кода.

Лабораторная работа №1
Технологии разработки программного обеспечения: "разработка через тестирование"

Количество аудиторных часов - 2

Цель работы

Знакомство с технологией "разработка через тестирование". Изучение инструментов позволяющих применять данную технологию.

Общие сведения

Разработка через тестирование (англ. *test-driven development, TDD*) — техника разработки программного обеспечения, которая основывается на повторении очень коротких циклов разработки: сначала пишется тест, покрывающий желаемое изменение, затем пишется код, который позволит пройти тест, и под конец проводится рефакторинг нового кода к соответствующим стандартам.

TDD требует от разработчика создания автоматизированных модульных тестов, определяющих требования к коду непосредственно перед написанием самого кода. Тест содержит проверки условий, которые могут либо выполняться, либо нет. Когда они выполняются, говорят, что тест пройден. Прохождение теста подтверждает поведение, предполагаемое программистом. Разработчики часто пользуются библиотеками для тестирования (англ. *testing frameworks*) для создания и автоматизации запуска наборов тестов. На практике модульные тесты покрывают критические и нетривиальные участки кода. Это может быть код, который подвержен частым изменениям, код, от работы которого зависит работоспособность большого количества другого кода, или код с большим количеством зависимостей.

Среда разработки должна быстро реагировать на небольшие модификации кода. Архитектура программы должна базироваться на использовании множества сильно связанных компонентов, которые слабо сцеплены друг с другом, благодаря чему тестирование кода упрощается.

TDD не только предполагает проверку корректности, но и влияет на дизайн программы. Опираясь на тесты, разработчики могут быстрее представить, какая функциональность необходима пользователю. Таким образом, детали интерфейса появляются задолго до окончательной реализации решения

Задание

Изучить библиотеки для тестирования
Рассмотреть применение NUnit, ReSharper.

Лабораторная работа №2
Модульное тестирование

Количество аудиторных часов - 6
Цель работы
Овладение навыками модульного тестирования.

Общие сведения

Модульное тестирование - это тестирование программы на уровне отдельно взятых модулей, функций или классов. Цель модульного тестирования состоит в выявлении локализованных в модуле ошибок в реализации алгоритмов, а также в определении степени готовности системы к переходу на следующий уровень разработки и тестирования. Модульное тестирование проводится по принципу "белого ящика", то есть основывается на знании внутренней структуры программы, и часто включает те или иные методы анализа покрытия кода.

Модульное тестирование обычно подразумевает создание вокруг каждого модуля определенной среды, включающей заглушки для всех интерфейсов тестируемого модуля. Некоторые из них могут использоваться для подачи входных значений, другие для анализа результатов, присутствие третьих может быть продиктовано требованиями, накладываемыми компилятором и сборщиком.

На уровне модульного тестирования проще всего обнаружить дефекты, связанные с алгоритмическими ошибками и ошибками кодирования алгоритмов, типа работы с условиями и счетчиками циклов, а также с использованием локальных переменных и ресурсов. Ошибки, связанные с неверной трактовкой данных, некорректной реализацией интерфейсов, совместимостью, производительностью и т.п. обычно пропускаются на уровне модульного тестирования и выявляются на более поздних стадиях тестирования.

Задание

1) "Мозговая атака":
Этапы:

- Получить вопрос (задание) для обсуждения.
- Задать вопросы относительно не понятных моментах в вопросе (задании).
- Высказать свои мысли по данному вопросу (заданию).
- Записать все прозвучавшие высказывания с уточнениями.
- По окончанию прочитать все, что было записано.
- Обсудить все варианты ответов.
- Выяснить, как можно использовать полученные результаты при выполнении данной лабораторной работы.

2) "Работа в команде"
Этапы:
- Разбиться на команды.
- Реализовать полученный вопрос (задание), согласно технологии TDD.
- Представить результаты.

Лабораторная работа №3
Интеграционное тестирование

Количество аудиторных часов - 2
Цель работы

Овладение навыками интеграционного тестирования.

Общие сведения

Интеграционное тестирование называют еще тестированием архитектуры системы. С одной стороны, это название обусловлено тем, что интеграционные тесты включают в себя проверки всех возможных видов взаимодействий между программными модулями и элементами, которые определяются в архитектуре системы - таким образом, интеграционные тесты проверяют полноту взаимодействий в тестируемой реализации системы. С другой стороны, результаты выполнения интеграционных тестов - один из основных источников информации для процесса улучшения и уточнения архитектуры системы, межмодульных и межкомпонентных интерфейсов. Т.е., с этой точки зрения, интеграционные тесты проверяют корректность взаимодействия компонент системы.

В результате проведения интеграционного тестирования и устранения всех выявленных дефектов получается согласованная и

целостная архитектура программной системы, т.е. можно считать, что интеграционное тестирование - это тестирование архитектуры и низкоуровневых функциональных требований.

Интеграционное тестирование, как правило, представляет собой итеративный процесс, при котором проверяется совокупность модулей, возрастающая от итерации к итерации. В интеграционном тестирование выделяют три методы выполнения: восходящее тестирование; монолитное тестирование; нисходящее тестирование.

Задание

Согласно варианту провести один из методов интеграционного тестирования.

Лабораторная работа №4
Системное тестирование

Количество аудиторных часов - 4

Цель работы

Овладение навыками системного тестирования.

Общие сведения

Системное тестирование - один из самых сложных видов тестирования. На этапе системного тестирования проводится не только функциональное тестирование, но и оценка характеристик качества системы - ее устойчивости, надежности, безопасности и производительности. На этом этапе выявляются многие проблемы внешних интерфейсов системы, связанные с неверным взаимодействием с другими системами, аппаратным обеспечением, неверным распределением памяти, отсутствием корректного освобождения ресурсов и т.п.

После завершения системного тестирования разработка переходит в фазу приемо-сдаточных испытаний (для программных систем, разрабатываемых на заказ) или в фазу альфа- и бета-тестирования (для программных систем общего применения).

Системное тестирование проводится в несколько фаз, на каждой из которых проверяется один из аспектов поведения системы, т.е. проводится один из типов системного тестирования. Все эти фазы могут протекать одновременно или последовательно. Следующий

раздел посвящен рассмотрению особенностей каждого из типов системного тестирования на каждой фазе.

Виды системного тестирования:

1) функциональное тестирование;
2) тестирование производительности;
3) нагрузочное или стрессовое тестирование;
4) тестирование конфигурации;
5) тестирование безопасности;
6) тестирование надежности и восстановления после сбоев;
7) тестирование удобства использования.

Задание

Согласно варианту провести несколько видов системного тестирования.

Лабораторная работа №5
Ручное тестирование, генерация тестов

Количество аудиторных часов - 2

Цель работы

Овладение навыками ручного тестирования и составление тестовых случаев.

Общие сведения

Ручное тестирование заключается в выполнении задокументированной процедуры, где описана методика выполнения тестов, задающая порядок тестов и для каждого теста - список значений параметров, который подается на вход, и список результатов, ожидаемых на выходе. Поскольку процедура предназначена для выполнения человеком, в ее описании для краткости могут использоваться некоторые значения по умолчанию, ориентированные на здравый смысл, или ссылки на информацию, хранящуюся в другом документе.

Описание тестов разрабатывается для облегчения анализа и поддержки тестового набора. Описание может быть реализовано в произвольной форме, но при этом должны выполнять следующие задачи:

1. Анализировать степень покрытия продукта тестами на основании описания тестового набора.

2.Для любой функции тестируемого продукта найти тесты, в которых функция используется.

3.Для любого теста определить все функции и их сочетания, которые данный тест использует (затрагивает).

4.Понять структуру и взаимосвязи тестовых файлов.

5.Понять принцип построения системы автоматизации тестирования

Задание

Подготовить тестовый случай, выполнить и задокументировать результаты.

Лабораторная работа №6
Документация

Количество аудиторных часов - 2

Цель работы

Овладение навыками документирования результатов тестирования.

Общие сведения

Каждый дефект, обнаруженный в процессе тестирования, должен быть задокументирован и отслежен. При обнаружении нового дефекта его заносят в базу дефектов. При занесении нового дефекта рекомендуется указывать, как минимум, следующую информацию:

1) Наименование подсистемы, в которой обнаружен дефект.

2) Версия продукта (номер build), на котором дефект был найден.

3) Описание дефекта.

4) Описание процедуры (шагов, необходимых для воспроизведения дефекта).

5) Номер теста, на котором дефект был обнаружен.

6) Уровень дефекта, то есть степень его серьезности с точки зрения критериев качества продукта или заказчика.

Тестовый отчет обновляется после каждого цикла тестирования и должен содержать следующую информацию для каждого цикла:

1) Перечень функциональности в соответствии с пунктами требований, запланированный для тестирования на данном цикле, и реальные данные по нему.

2) Количество выполненных тестов – запланированное и реально исполненное.

3) Время, затраченное на тестирование каждой функции, и общее время тестирования.

4) Количество найденных дефектов.

5) Количество повторно открытых дефектов.

6) Отклонения от запланированной последовательности действий, если таковые имели место.

7) Выводы о необходимых корректировках в системе тестов, которые должны быть сделаны до следующего тестового цикла.

Задание

Задокументировать результаты тестирования. Для выполнения работы использовать тестовые случае из лабораторной работы №5.

Руководство к выполнению самостоятельной работы

Согласно рабочей программе самостоятельная работа студентов заключается в следующем:

№	Наименование работы	Форма контроля
1	Подготовка к контрольным работам	Контрольный опрос на лекции
2	Проработка лекционного материала	Тестовый опрос на лекции
3	Подготовка устных тематических докладов	Опрос на лекции
4	Изучение тем теоретической части дисциплины, вынесенных для самостоятельной проработки	Опрос на лекции
5	Подготовка к лабораторным работам	Программирование

Темы, отводимые на самостоятельное изучение:
1) Стратегии тестирования
2) Нефункциональные требования
3) Управление тестированием
4) Исследовательское тестирование
5) Жизненный цикл ошибки
6) Виды отчетностей и показателей

Темы для устных тематических докладов:
1) Разработка через тестирование
2) Гибкое тестирование

Список рекомендуемой литературы

1. Майерс Гленфорд Дж. Искусство тестирования программ. - М.: Финансы и статистика, 1982. - 176 с.

2. Липаев В.В. Тестирование компонентов и комплексов программ. - М.: Синтег, 2010. - 399 с.

3. Винниченко И. В. Автоматизация процессов тестирования: производственно-практическое издание. - СПб. : Питер, 2005. - 202 с.

4. Бек К. Экстремальное программирование: разработка через тестирование. - СПб. : Питер, 2003. - 224 с.