МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования
Томский государственный университет систем управления и радиоэлектроники

Кафедра автоматизированных систем управления

УТВЕРЖДАЮ
Зав. кафедрой АСУ
профессор, д-р. техн. наук
 А.М. Кориков
«6» сентября 2011 г.

Елизаров А.И.

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ К ЛАБОРТОРНЫМ ЗАНЯТИЯМ И САМОСТОЯТЕЛЬНОЙ РАБОТЕ
По дисциплине «Технология разработки программного обеспечения»

для студентов специальности 080801 «Прикладная информатика (в экономике)»

Томск 2012

Методические указания по практическим занятиям и самостоятельной работе по дисциплине «Технология разработки программного обеспечения» составлены в соответствии с программой и включают в себя тематику лабораторных работ, их содержание, список тем предложенных для самостоятельного изучения, список вопросов для подготовки к экзамену и рекомендуемую литературу. Методические указания предназначены для студентов специальности 080801 «Прикладная информатика (в экономике)»

Практикум по предмету «Технология разработки программного обеспечения» имеет целью закрепить теоретические знания по вопросам:

- жизненного цикла программного обеспечения (ПО);
- метрология и качества ПО;
- критериям качества (сложность, корректность, надежность, трудоемкость);
- процесса производства ПО (методов, технологий и инструментальных средства);
- тестирования и отладки;
- документирования;
- проектирования программного обеспечения;
- технологического цикла разработки программных систем;
- коллективной работы по созданию программ;
- организации процесса разработки и инструментальные средства поддержки;
- автоматизация проектирования программных продуктов.

Лабораторные работы представляют собой разработку и реализацию программной документации по циклу разработки. Разработанная документация должна быть продемонстрирована преподавателю.

В результате выполнения лабораторных работ студенты должны уметь:

самостоятельно выполнять цикл проектирования программного обеспечения;

- разрабатывать спецификации и абстрактные типы данных на основе анализа требований, предъявляемых к программному обеспечению;
- пользоваться стандартными терминами и определениями,
- читать научные статьи и пользоваться литературой для самостоятельного решения научно - исследовательских задач, связанных с разработкой программных систем.

В ходе выполнения лабораторно-практических работ должен быть оформлен отчет, содержащий:

- цель работы;
- набор программной документации;
- выводы.

Лаборторная работа №1 «Назначение и содержание соглашения о требованиях» – 6 час.

Цель работы: научиться подготавливать программную документацию по организации коллективной разработки ПО.

Краткие теоретические сведения:

В соглашении о требованиях должно содержаться письменное изложение того, что будет сделано и что не будет делаться при выпуске программного обеспечения.

Документ «Соглашение о требованиях» является основным средством управления разработкой программного обеспечения или генеральным планом его разработки.

Все участники разработки программного обеспечения должны выполнять то, что установлено в документе «Соглашение о требованиях» или запрашивать и получать разрешение на его изменение.

Предполагается, что все утверждения, включенные в соглашение о требованиях, являются требованиями, если они не определены как цели.

Каждый документ «Соглашение о требованиях» должен точно соответствовать установленной форме. Тогда каждый раздел можно будет найти в одном и том же месте аналогичного документа любой разработки программного обеспечения. В документ целесообразно включить заголовки всех предусмотренных разделов, если только специально не оговариваются условия, при которых какой-либо раздел может быть опущен. Тогда при рассмотрении документа будет решаться вопрос, действительно ли такие разделы нужны.

Соглашения о требованиях пишутся на естественном языке в терминах понятных и пользователю и разработчику программного обеспечения. Стороны должны четко представлять каждое требование.

Следует напомнить, что пользователь несет ответственность за проверку требований на полноту и точность, а разработчик - за проверку их на осуществимость и понятность.

Подробное описание разделов, которые должны присутствовать в любом соглашении о требованиях, приведено в соответствующем методическом пособии [5].

Задание

Разработать соглашение о требованиях на разрабатываемый программный продукт.

Лабораторная работа №2 « Методы написания спецификаций» - 6 часов.
Цель работы: научиться определять спецификации ПО
Краткие теоретические сведения:

На этапе определения спецификаций осуществляется точное описание функций, реализуемых ЭВМ, а также задаются структуры входных и выходных данных, методы и средства их размещения. Определяются алгоритмы обработки данных.

Центральным вопросом определения спецификаций является проблема организации базы данных. При этом решается комплекс вопросов, имеющих отношение к структуре файлов, организации доступа к ним, модификации и удаления.

В случае, когда новая система создается на основе существующих, составной частью спецификаций является схема (алгоритм) приведения существующей базы данных к новому формату. Такое преобразование может потребовать разработку

специальной программы, которая становится ненужной после ее первого и единственного использования.

Все эти вопросы должны быть отражены в функциональных спецификациях, которые представляют собой документ, являющийся основополагающим в процессе разработки системы, так как содержит конкретное описание последней. Чем подробней составлены спецификации, тем меньше вероятность возникновения ошибок.

В спецификациях должны быть представлены данные для тестирования элементов системы и системы в целом. Это требование является объективным и обязательным, так как на данном этапе на параметры тестирования не будет оказывать влияние конкретная реализация системы.

Так как функциональные спецификации описывают принятые решения в целом, данный документ можно использовать для начальных оценок временных затрат, числа специалистов и других ресурсов, необходимых для проведения работ.

В общем случае спецификации определяют те функции, которые должна выполнять система, не указывая, каким образом это достигается. Составление подробных алгоритмов на этом этапе преждевременно и может вызвать нежелательные осложнения.

Задание

На основании технического задания и соглашения о требованиях определить внешнюю и внутреннюю спецификации.

Лабораторная работа №3 «Доказательство правильности программ**» - 2часа.**

Цель работы: научиться доказывать правильность элементарных программ относительно спецификации.

Краткие теоретические сведения:

Одним из важных методов повышения эффективности проектирования программ является верификация программ или математическое доказательство того, что программа работает правильно. Для доказательства правильности программ используется аксиоматический подход, при котором применяется теория перечисления предикатов. Предполагается, что каждый оператор в программе выполняет заранее определенные действия, зависящие только от синтаксиса языка. Для двух предикатов P и Q и оператора S необходимо определить истинно ли выражение:

«Если P истинно и если выполняется оператор S, то Q истинно».

Предикат P является спецификацией правильного выполнения оператора S, а предикат Q будет истинным после выполнения оператора S и является спецификацией следующего за S оператора.

Если это утверждение распространить на все операторы программы и если P является спецификацией первого оператора, а Q истинно после окончания программы, то будет доказана правильность всей программы относительно предикатов P и Q. Эту конструкцию можно записать в следующем виде:

$$\{P\}S\{Q\},$$

где P называется предусловием истинности Q после выполнения программы S.

Доказательство правильности программы заключается в определении, является ли выражение $\{P\}S\{Q\}$ истинным относительно входных спецификаций P, выходных спецификаций Q и операторов S программы. Если $\{P\}S\{Q\}$ истинно, то это означает, что доказана правильность программы S относительно P и Q.

Задание

Используя расширенные аксиомы исчисления предикатов доказать правильность (или ошибочность) предложенной преподавателем программы.

Лабораторная работа №4 «Технология написания тестов» – 4 часа**.**

Цель работы: Научиться определять классы эквивалентности в спецификациях и подготавливать необходимый набор тестов по ним.

Краткие теоретические сведения:

Общие принципы тестирования

Этап тестирования обычно в финансовых затратах составляет половину расходов на создание системы. Плохо спланированное тестирование приводит к существенному увеличению сроков разработки системы и является основной причиной срывов графиков разработки.

В процессе тестирования используются данные, характерные для системы в рабочем состоянии, т.е. данные для тестирования выбираются случайным образом. План проведения испытаний должен быть составлен заранее, обычно на этапе проектирования.

Тестирование подразумевает три стадии:

- автономное;
- комплексное;
- системное.

При автономном тестировании модуль проверяется с помощью данных, подготовленных программистом. При этом программная среда модуля имитируется с помощью программ управления тестированием, содержащих фиктивные программы вместо реальных подпрограмм, к которым имеется обращение из данного модуля (заглушки). Подобную процедуру называют программным тестированием, а программу тестирования – UUT (тестирующей программой). Модуль, прошедший автономное тестирование, подвергается комплексному тестированию.

В процессе комплексного тестирования проводится совместная проверка групп программных компонентов. В результате имеем полностью проверенную систему. На данном этапе тестирование обнаруживает ошибки, пропущенные на стадии автономного тестирования. Исправление этих ошибок может составлять до четверти от общих затрат.

Системное (или оценочное) тестирование – это завершающая стадия проверки системы, т.е. проверка системы в целом с помощью независимых тестов. Независимость тестов является главным требованием. Обычно Заказчик на

стадии приемки работ настаивает на проведении собственного системного тестирования. Для случая, когда сравниваются характеристики нескольких систем (имеется альтернативная разработка), такая процедура известна как сравнительное тестирование.

Технология тестирования, классы эквивалентности

Одним из способов изучения данного вопроса является исследование стратегии тестирования, называемой стратегией черного ящика, *тестированием с управлением по данным* или *тестированием с управлением по входу-выходу*. При использовании этой стратегии программа рассматривается как черный ящик. Иными словами, такое тестирование имеет целью выяснение обстоятельств, в которых поведение программы не соответствует ее спецификации. При таком подходе обнаружение всех ошибок в программе является критерием *исчерпывающего входного тестирования*. Последнее может быть достигнуто, если в качестве тестовых наборов использовать все возможные наборы входных данных.

Разработка тестов методом эквивалентного разбиения осуществляется в два этапа:

1) выделение классов эквивалентности;
2) построение тестов.

Классы эквивалентности выделяются путем выбора каждого входного условия (обычно это предложение или фраза в спецификации) и разбиением его на две или более группы. Для проведения этой операции используют таблицу, подобную табл.

Таблица Форма таблицы для перечисления классов эквивалентности

Входные условия	Классы эквивалентности	
	Правильные	Неправильные

Различают два типа классов эквивалентности: правильные классы эквивалентности, представляющие правильные входные данные программы, и неправильные классы эквивалентности, представляющие все другие возможные состояния условий (т.е. ошибочные входные значения). Таким образом, придерживаются одного из принципов необходимости сосредоточивать внимание на неправильных или неожиданных условиях.

Задание

Идентифицировать входные условия и по ним определить классы эквивалентности, построить соответствующие тесты. В заключении дать рекомендации по устранению ошибок, выявленных в результате тестирования.

Проработка лекционного материала (34 часов)

Самостоятельная работа с материалами лекций и литературой для более глубокого и детального изучения разделов дисциплины, подготовка к их обсуждению на практических занятиях.

Подготовка к лабораторным работам (8 часа)

Самостоятельная работа с материалами лекций и литературой по темам практических занятий, выполнение практических заданий.

Список вопросов к экзамену по курсу «Технология разработки программного обеспечения»

1. Этапы разработки программного обеспечения.
2. Анализ требований, предъявляемых к системе.
3. Жизненный цикл программного обеспечения. Функциональные спецификации. Определение спецификаций. Проектирование. Кодирование.
4. Тестирование: программное, системное, оценочное и сравнительное тестирование. Сбой системы, выброс, ошибка. Испытания. Верификация системы.
5. Правильность и надежность программ.
6. Эксплуатация и сопровождение. Периоды обновления.
7. Организация интерфейса между модулями, написанными разными программистами. Выполнение проекта. Бригада главного программиста.
8. Методика оценки затрат. Методика инженерно-технической оценки затрат.
9. Методика экспертных оценок. Метод алгоритмического анализа. Пошаговый анализ. Закон Паркинсона. Затраты на завершения разработки.
10. Оценка длительности разработки на основе распределения Рэлея.
11. Контрольные точки. Средства обработки. Надежность. Концептуальная целостность.
12. "Уровни правильности" программ. Методы программирования.
13. Определение спецификаций.
14. Язык определения задач и анализатор определения задач (PSL/PSA).
15. Система структурного проектирования SADT. Структурное проектирование. Методика Джексона.
16. Стратегия объединения различных методов проектирования.
17. Язык проектирования программ PDL. Операторы выбора. Операторы цикла. Операторы описания данных. Операторы ввода вывода и вызова процедур. Оператор leave. Предложения на естественном языке.
18. Нисходящее проектирование и нисходящая разработка.
19. Пошаговое совершенствование. Восходящее проектирование.
20. Структурное проектирование. Простая программа. Элементарная программа. Управляющие структуры, способы их описания.
21. Скалярные и агрегативные типы данных. Массивы. Структуры. Списки. Очереди. Стеки. Множества. Графы. Деревья.
22. Абстрактные конструкции. Фиксированные данные абстрактного типа. Размещение указателей. Защита данных от несанкционированного доступа.
23. Правильность программ.
24. Аксиомы: правила следствия; аксиома присвоения; аксиома следования; аксиома цикла; аксиома выбора. Правила целочисленной арифметики -

коммутативность, ассоциативность, дистрибутивность, вычитания, обработка констант.

25. Стратегия тестирования. Имена переменных. Константы. Входные данные. Списки параметров. Проверка спецификаций.
26. Данные для тестирования. Формализация тестирования программ.
27. Стандартные методы проектирования. Разбиение задачи на независимые подзадачи. Разбиение задачи на одинаковые по сложности части. Рекурсия. Динамическое программирование. Моделирование. Алгоритм выбора из конечного числа состояний.
28. Стратегия распределения памяти. Сопрограммы.
29. Понятие изделия, как средства общения.
30. Нисходящий анализ процесса управления созданием программного изделия.
31. Установление целей и средства их достижения. Подбор и обучение кадров.
32. Организация планирования разработки программного изделия. Виды планов. Декомпозиция планов.
33. Организационная структура группы планирования.
34. Виды планов, связанных с созданием программного изделия.
35. Организация планирования разработки программного изделия.
36. Вопросы, рассматриваемые в фазовых обзорах группой планирования,
37. Управление проектом.
38. Организация работы группы разработки в фазах создания программного изделия.
39. Организация работы группы обслуживания в фазах создания программного изделия.
40. Организация работы группы выпуска документации в фазах создания программного изделия.
41. Организация испытаний программного изделия.
42. Психология и экономика тестирования программ
43. Принципы тестирования
44. Инспекции, сквозные просмотры и обзоры программы
45. Список вопросов для выявления ошибок при инспекции
46. Тестирование путем покрытия логики программы
47. Эквивалентное разбиение
48. Анализ граничных значений
49. Применение функциональных диаграмм
50. Предположение об ошибке. Стратегия
51. Понятие изделия, как средства общения.
52. Нисходящий анализ процесса управления созданием программного изделия.
53. Установление целей и средства их достижения.
54. Подбор и обучение кадров.
55. Организация планирования разработки программного изделия. Виды планов. Декомпозиция планов.
56. Организационная структура группы планирования.

57. Виды планов, связанных с созданием программного изделия.
58. Организация планирования разработки программного изделия.
59. Вопросы, рассматриваемые в фазовых обзорах группой планирования,
60. Управление проектом.
61. Организация работы группы разработки в фазах создания программного изделия.
62. Организация работы группы обслуживания в фазах создания программного изделия.
63. Организация работы группы выпуска документации в фазах создания программного изделия.
64. Организация испытаний программного изделия

Основная литература:

1. Технология разработки программного обеспечения: Учебное пособие / Калайда В. Т., Романенко В. В. – 2012. 220 с. [Электронный ресурс] - Научно-образовательный портал ТУСУР – 2012. – Режим доступа: http://edu.tusur.ru/training/publications/2076

Дополнительная литература

1. Калайда В. Т. Технология разработки программного обеспечения : Учебное пособие / В. Т. Калайда, В. В. Романенко; Федеральное агентство по образованию, Томский государственный университет систем управления и радиоэлектроники. - Томск: ТУСУР, 2007. - 237[1] с. (90 экз.).
2. Брауде Э. Д. Технология разработки программного обеспечения. - СПб.: Питер, 2004. - 654 с. (22 экз.).
3. Вендров А. М. Проектирование программного обеспечения экономических информационных систем. – М.: Финансы и статистика, 2002. – 176 с. (34 экз.)
4. Орлов, С.А. Технологии разработки программного обеспечения. Разработка сложных программных систем : Учебное пособие для вузов / Сергей Александрович Орлов. - СПб. : Питер, 2002. - 464 с. (26 экз.)+
5. Елизаров, А.И. Технология разработки программного обеспечения : учебное методическое пособие для студентов специальности 230105 / А. И. Елизаров, В. В. Романенко ; Федеральное агентство по образованию, Томский государственный университет систем управления и радиоэлектроники, Кафедра автоматизированных систем управления. - Томск : ТМЦДО, 2007. - 119 с. (8 экз.) +