Министерство науки и высшего образования Российской Федерации

Томский государственный университет
систем управления и радиоэлектроники

М. Е. Антипин

Программная инженерия

Методические указания по выполнению лабораторных работ

Томск
2022

УДК 004.02
ББК 3стд2-02
А 72

Рецензент:

Лобода Ю.О., доцент каф. управления инновациями ТУСУР,
канд. пед. наук

Антипин, Михаил Евгеньевич
А 72 Программная инженерия: Методические указания по выполнению лабораторных работ/ М.Е. Антипин. – Томск: Томск. гос. ун-т систем упр. и радиоэлектронники, 2022. – 17 с.

Методические указания содержат рекомендации и материалы, необходимые для выполнения лабораторных работ по дисциплине «Программная инженерия».
Для студентов высших учебных заведений.

Одобрено на заседании кафедры УИ, протокол № 1 от 31.08.2022.

УДК 004.02
ББК 3стд2-02

Оглавление

1. Общие положения

Данные методические указания разработаны для студентов, обучающихся в Томском государственном университете систем управления и радиоэлектроники (далее - Университет).

Структура дисциплины «Программная инженерия» предполагает проведение лабораторных работ. Лабораторные работы предназначены для закрепления материала, полученного в лекционном курсе, самостоятельного изучения материалов дисциплины, предусмотренных рабочей программой. Полученные навыки и знания могут быть полезны при проектировании, разработке и внедрении систем технического зрения. Рекомендации по выполнению самостоятельной работе студентов приведены в соответствующих методических указаниях.

В ходе выполнения лабораторных работ студенты приобретают навыки поиска информации, работы с учебно-методической документацией, умения увязывать теоретические знания с практикой, четко излагать свои мысли, отвечать на вопросы, оформлять и представлять результаты работы.

Рекомендации подготовлены с целью помочь студентам в успешном освоении дисциплины и подготовке и прохождении промежуточных этапов аттестации.

2 Общие требования к проведению лабораторных работ

Лабораторные работы по дисциплине «Современные методы оптимизации бизнес-процессов организации» проводятся согласно учебному расписанию. В ходе выполнения лабораторных работ студент выполняет задания, предусмотренные настоящими методическими указаниями. Набор входных данных определяется преподавателем с учетом текущих навыков и прогресса студента в изучении дисциплины. Это обеспечивает необходимую индивидуализацию выполняемых работ. Лабораторные работы выполняются студентами очной формы обучения индивидуально под контролем со стороны преподавателя. Все консультации осуществляются преподавателем.

Лабораторные работы выполняются студентами очной формы обучения индивидуально под контролем со стороны преподавателя. Все консультации осуществляются преподавателем. Число студентов, одновременно присутствующих на занятии не должно превышать 12 человек. Если в списочном составе группы студентов больше 12, то группа должна быть разделена на подгруппы численностью от 6 до 12 человек в каждой.

Для выполнения лабораторных работ целесообразно в учебном расписании выделять 4 академических часа подряд, без больших перерывов. Расписание также должно предусматривать раздельное проведение занятий у подгрупп, если группа была разделена.

Перед началом занятий студенты должны изучить инструкцию по охране труда, действующую в лаборатории. Преподаватель должен убедиться в знании инструкции, задавая студенту вопросы по ее содержанию, после чего сделать соответствующую запись в журнале охраны труда.

Во время выполнения лабораторных работ студентам в аудитории запрещается:

- Разговаривать между собой на любые темы без разрешения преподавателя.
- Консультировать друг друга.
- Передавать друг другу материалы, являющиеся результатом выполнения заданий.
- Производить шум, мешающий остальным сосредоточиться на выполнении задания.
- Пользоваться наушниками, берушами и другими приспособлениями, не позволяющими отчетливо слышать указания преподавателя.
- Читать литературу, конспекты и другие записи, не относящиеся к изучаемому предмету.
- Находиться в помещении аудитории в верхней одежде, если температура выше 18ºС.
- Приносить верхнюю одежду с собой и размещать ее на стуле/столе, если в учебном корпусе работает гардероб.

В случае однократного нарушения преподаватель должен предупредить студента. При повторном нарушении в течении одного занятия студент из аудитории удаляется.

Студент имеет право:

- Уточнять полученные задания у преподавателя.
- Пользоваться любыми доступными методическими материалами по данной дисциплине.
- Просить консультации у преподавателя, если он в текущий момент не распределяет задания, не принимает выполненные работы и не консультирует другого студента.
- Пользоваться для выполнения практических заданий собственным ноутбуком или планшетным компьютером.

Преподаватель, давая консультацию студенту, указывает раздел технической документации или методической литературы, в которой имеется ответ на вопрос студента.

Если необходимые сведения в документации и литературе отсутствуют, то преподаватель должен дать устные пояснения или продемонстрировать практические действия, приводящие к требуемому результату для повторения студентом.

Самостоятельная работа студентов над лабораторными работами осуществляется в той же аудитории (лаборатории), где проводятся практические занятия. Преподаватель должен согласовать со студентами расписание самостоятельной работы - не менее 2 астрономических часов в неделю. В указанное время по учебному расписанию студентов и в аудитории (лаборатории) не должны проводиться другие занятия. Преподаватель должен обеспечить доступ студентов в аудиторию (лабораторию) в указанные часы. Необходимость самостоятельной работы определяет студент.

Консультации, выдача лабораторных заданий и прием результатов выполнения осуществляется только во время аудиторных занятий. Задания выполняются последовательно. Правильное выполнение некоторых заданий возможно только, если студент корректно выполнил предыдущие задания. Поэтому приступать к следующему заданию студент может, только сдав преподавателю результат выполнения предыдущего.

3 Техническое обеспечение лабораторных работ

Для выполнения лабораторных работ студенту предоставляется индивидуальное рабочее место, в состав которого входят:

- Персональный компьютер с предустановленной операционной системой Windows7 и выше;
- Программный пакет для управления программными проектами (Scrum, Trello, Jira и т.п.);
- Пакет офисных приложений для разработки текста отчета.

Размещение и освещенность рабочих мест в учебной аудитории (лаборатории) должно удовлетворять действующим требованиям СанПиН.

4 Прием результатов выполнения лабораторных работ

Результаты выполнения лабораторных работ представляются преподавателю в виде электронного файла отчета, содержащего результат соответствующего выполненного задания.

Во время приема выполненной работы преподаватель вправе:

- Требовать у студента правильность заполнения всех полей элементов модели, в том числе и не визуализированных на итоговых диаграммах;
- Самостоятельно производить манипуляции с моделью без ее изменения;
- Требовать у студента пояснений, относящихся к отдельным элементам модели, исходной информации, способам ее получения и верификации.

Задание считается выполненным и принимается преподавателем только в том случае, если модель логически непротиворечива, не имеет несвязанных входов и выходов, корректна с точки зрения выбранного языка (нотации), исходная информация учтена полностью. Если эти условия не выполняются, то результат выполнения подлежит доработке. Студент должен работать над моделью максимально самостоятельно, использовать средства проверки синтаксиса, предоставляемые программным пакетом.

За выполнение каждого задания преподаватель выставляет студенту оценку. Оценка выполнения задания складывается из трех равнозначных компонентов:

- Время выполнения задания. Фиксируется с момента получения задания до момента сдачи отчета. Измеряется в астрономических часах. Сравнивается с нормативным временем выполнения.
- Полнота и правильность выполнения задания. Экспертная оценка преподавателя.
- Аккуратность при выполнении текстовых и графических материалов.

Во время приема выполненной работы преподаватель вправе требовать у студента обоснования представленных материалов.

Преподаватель должен объявить студенту поставленную ему оценку за выполнение задания, а в случае возникновения непонимания, объяснить причины ее выставления. В случае, если оценка неудовлетворительно, студент имеет право повторно предъявить результат выполнения, но не более двух раз в течение одного занятия. При этом для вычисления оценки время, затраченное на исправление, прибавляется к общему времени выполнения задания.

Выставленная оценка влияет на оценку студента по контрольной точке и среднюю оценку за практические занятия.

До конца семестра студент должен получить оценку по всем лабораторным работам, предусмотренным настоящими указаниями. За работы, результаты выполнения которых не были предъявлены преподавателю для оценивания, выставляется оценка неудовлетворительно. Студенты, имеющие среднюю оценку за практические занятия ниже удовлетворительной, к итоговой аттестации по предмету не допускаются.

5 Терминология дисциплины

Чтобы свободно ориентироваться в материалах дисциплины студенту следует ознакомиться с применяемой терминологией:

- Программа (program) – это набор операторов, который может быть представлен как единое целое в некоторой вычислительной системе и который используется для управления поведением этой системы.
- Программирование (в узком смысле) – процесс кодирования и отладки программы в рамках реального проекта.
- Программирование (programming) (в широком смысле) – все технические операции, необходимые для создания программы, включая анализ требований и все стадии разработки и реализации.
- Программная инжене́рия - системный подход к анализу, проектированию, оценке, реализации, тестированию, обслуживанию и модернизации программного обеспечения. Программная инженерия занимается разработкой систематических моделей и надежных методов производства высококачественного программного обеспечения, и данный подход распространяется на все уровни — от теории и принципов до реальной практики создания программного обеспечения. ПИ является отраслью информатики (computer science), это инженерная дисциплина, которая изучает вопросы построения компьютерных программ, отражает закономерности развития программирования, обобщает опыт программирования в виде комплекса знаний и правил регламентации инженерной деятельности разработчиков ПО.
- Технология программирования – процессы программной инженерии, методы программирования, инструментальные средства.
- Программное обеспечение – совокупность программ для обработки информации и комплект документации, необходимой для ее эксплуатации.
- Жизненный цикл – развитие системы, продукта, услуги, проекта начиная со стадии идеи и заканчивая завершением применения.
- Конфигурационное управление (англ. Software configuration management, SCM) в программной инженерии — комплекс методов, направленных на систематический учёт изменений, вносимых разработчиками в программный продукт в процессе его разработки и сопровождения, сохранение целостности системы после изменений, предотвращение нежелательных и непредсказуемых эффектов, формализацию процесса внесения изменений.

6 План выполнения лабораторных работ

1. Разработка технического задания на программную систему

Во время лабораторного практикума должна быть разработана программная система (проект) средней сложности, при этом в соответствии с требованиями сегодняшнего дня разработка системы должна вестись не единолично, а командой разработчиков, каждый из которых в дальнейшем будет выполнять порученную ему часть проекта.

Тема проекта выдается преподавателем (в дальнейшем это руководитель проекта) на первом занятии, в соответствии с которой в течение первых двух недель команда студентов разрабатывает техническое задание по форме, приведенной в приложении А. Разработку ТЗ можно считать началом первой фазы ЖЦ будущей ПС.

Техническое задание разрабатывается в соответствии с ГОСТ 34.602-89 «Техническое задание на создание автоматизированной системы» [2] и в дальнейшем должно стать основным документом, по которому студенты ведут разработку проекта. Любые изменения ТЗ на систему должны быть согласованы с преподавателем и заверены его подписью.

Техническое задание состоит из трех частей:

1.1. содержание задания;

1.2. исходные данные;

1.3. календарный план выполнения работ.

2. Описание и анализ предметной области

Для быстрой и эффективной разработки программной системы с минимальным браком требуется определить верное направление работы. Для того чтобы правильно построить систему, сначала необходимо построить ее модель (этим вы будете заниматься позднее). Моделирование - это устоявшаяся и повсеместно принятая инженерная методика. Хорошая модель всегда включает элементы, существенно влияющие на результат, и не включает те, которые малозначимы на данном уровне абстракции. Модели строятся для того, чтобы лучше понимать разрабатываемую систему. При этом нужно помнить об основных принципах моделирования, один из которых гласит: «Лучшие модели - те, что ближе к реальности». Поэтому первое, с чего нужно начать разработку системы – это досконально изучить предметную область, в которой Вы будете работать

Сложность предметной области определяет количество объектов и связей между ними, поэтому описание должно включать в себя базовые термины и определения, сопровождаться различными примерами, в нем могут приводиться различного рода классификации, поясняющие различные свойства описываемых объектов. Если в системе используются математические модели, то они также должны быть описаны с учетом специфики применения

Задание:

2.1. Выписать все термины и определения предметной области.

2.2. Создать модель предметной области.

2.3. В офисном пакете оформить отчет по лабораторной работе.

3. Постановка задачи

Постановка задачи – заключительный этап первой фазы ЖЦ системы. На данном этапе формулируются все требования, которым должна удовлетворять система. Постановка задачи пишется в повествовательной форме в будущем времени на основе ТЗ, в ней должны быть обязательно взаимоувязаны виды автоматизируемой деятельности (с привязкой к объекту(ам) автоматизации) со всеми ограничениями, накладываемыми на них, учтены особенности разрабатываемого информационного обеспечения и перечислены функции, которые должна выполнять система (с привязкой к процессам и информационному обеспечению).

Задание:

3.1. Составить модель требовний к программному продукту.

3.2. В офисном пакете оформить отчет по лабораторной работе.

4. Разработка структуры системы

Построение структурной схемы программной системы. На данном этапе система по функциональному признаку разделяется на основные подсистемы, между ними указываются информационные связи и/или связи по управлению, описывается основное назначение подсистем. При разработке структурной схемы используется методология структурного проектирования, в основе которой лежит алгоритмическая декомпозиция и иерархия вида «часть-целое», учитывающая, что внутренние связи элементов внутри подсистем сильнее, чем связь между подсистемами. Декомпозиция системы может повторяться многократно, вплоть до уровня конкретных процедур, при этом должна быть обеспечена целостность системы, а все составляющие компоненты взаимоувязаны. Для этого используются такие принципы разработки, как «сверху-вниз», «разделяй и властвуй», «иерархическое упорядочивание» и другие.

Система должна представлять собой совокупность элементов (объектов, субъектов), находящихся между собой в определенной зависимости и составляющих некоторое единство (целостность), направленное на достижение определенной цели. Система может являться элементом другой системы более высокого порядка (надсистема) и включать в себя системы более низкого порядка (подсистемы). То есть систему можно рассматривать как набор подсистем, организованных для достижения определенной цели и описанных с помощью набора моделей (возможно, с различных точек зрения), а подсистему – как группу элементов, часть которых составляет спецификацию поведения, представленного другими ее составляющими.

Полученная в результате декомпозиции структура системы должна сопровождаться кратким описанием включенных в нее подсистем. К типовым можно отнести следующие подсистемы:

4.1. подсистему управления;

4.2. подсистемы ввода-вывода:

- 4.2.1. подсистему настройки параметров;
- 4.2.2. файловую подсистему;
- 4.2.3. подсистему визуализации;
- 4.2.4. подсистему документирования;
- 4.2.5. подсистему взаимодействия с базой данных;

4.3. справочную подсистему.

5. Разработка спецификации требований

Разработка спецификации программного обеспечения является одним из фундаментальных процессов технологии разработки ПО. Этот процесс анализа, формирования, документирования и проверки функциональных возможностей и ограничений системы называется в терминологии программной инженерии «разработка требований» (спецификация требований). Он является критическим этапом в создании всех видов программных систем, что обусловлено тем, что ошибки, допущенные на этой стадии, ведут к возникновению серьезных проблем на этапах проектирования и разработки. Опыт индустрии информационных технологий однозначно показывает, что вопросы, связанные с управлением требованиями, оказывают критически-важное влияние на программные проекты, в определенной степени - на сам факт возможности успешного завершения проектов. Только систематичная работа с требованиями позволяет корректным образом обеспечить моделирование задач реального мира и формулирование необходимых приемочных тестов для

того, чтобы убедиться в соответствии создаваемых программных систем критериям, заданным реальными практическими потребностями.

Дадим некоторые определения.

Требования- это свойства, которыми должно обладать ПО для адекватного определения функций, условий и ограничений выполнения ПО, а также объемов данных, технического обеспечения и среды функционирования [12, 13].

На практике часто применяется подход, используемый в различных методологиях разработки ПО и базирующийся на определении групп требований к продукту. Такой подход обычно включает группы (типы, категории) требований, например: программные, системные, функциональные, нефункциональные (в частности, атрибуты качества) и т.п.

Программные требования(Software Requirements) - свойства программного обеспечения, которые должны быть надлежащим образом представлены в нём для решения конкретных практических задач [14]. Данная область знаний касается вопросов извлечения (сбора), анализа, специфицирования и утверждения требований к разрабатываемой ПС.

Функциональные требованиязадают «что» система должна делать;нефункциональные– с соблюдением «каких условий» (например, скорость отклика при выполнении заданной операции). При разработке этих требований в первую очередь необходимо учитывать потребности пользователя (заказчика).Пользовательские требования (User Requirements)– описывают цели/задачи пользователей системы, которые должны достигаться/выполняться пользователями при помощи создаваемой программной системы. Частопользовательские требования представляют в виде сценариев (вариантов использования) Use Case (см. лабораторную работу №5).

Среди нефункциональных требований на первый план выходят атрибуты качества и ограничения. Атрибуты качества (Quality Attributes) описывают дополнительные характеристики продукта в различных «измерениях», важных для пользователей и/или разработчиков. Атрибуты касаются вопросов портируемости, интероперабельности (прозрачности взаимодействия с другими системами), целостности, устойчивости и т.п. Данный вид требований мы будем называтьспецификацией качества.Ограничения (Constraints)включают в себя формулировки условий, модифицирующих требования или наборы требований, сужая выбор возможных решений по их реализации. В частности, к ним могут относиться параметры производительности, влияющие на выбор платформы реализации и/или развертывания (протоколы, серверы приложений, баз данных, ...), которые, в свою очередь, могут относиться, например, к внешним интерфейсам.

Спецификация требований к ПО(SRS)- процесс формализованного описания функциональных и нефункциональных требований, требований к характеристикам качества в соответствии со стандартом качества ISO/IEC 9126-94, которые будут отрабатываться на этапах ЖЦ ПО.

В спецификации требований отражается:

- структура ПО;
- требования к функциям, качеству и документации;
- задается в общих чертах архитектура системы и ПО, алгоритмы, логика управления и структуры данных.

Спецификация требований должна быть:

1. *Корректной*(каждое требование, изложенное в ней, является требованием, которому должно удовлетворять программное обеспечение);
2. *Однозначной*(каждое изложенное в ней требование может интерпретироваться только однозначно. Как минимум, для этого требуется, чтобы каждая характеристика конечного продукта была описана с использованием одного уникального термина);

3. *Полной*(все существенные требования должны быть подтверждены и обработаны любые внешние требования, налагаемые спецификацией системы; определены отклики программного обеспечения на все классы входных данных, которые могут быть реализованы, во всех возможных ситуациях, как на допустимые, так и недопустимые входные значения; полные обозначения и ссылки на все рисунки, таблицы и схемы в SRS и определение всех терминов и единиц измерения);
4. *Непротиворечивой*(никакой набор отдельных требований, описанных в ней, не находится в противоречии с ней или с каким-то документом более высокого уровня);
5. *Упорядоченной*по ее значимости и/или устойчивости (каждое требование должно иметь идентификатор и относиться к определенному классу требований: необходимые, условные и необязательные);
6. *Проверяемой*(каждое требование, изложенное в ней, может быть проверено.);
7. *Модифицируемой*(структура и стиль SRS таковы, что любые изменения требований могут быть выполнены легко, полностью и непротиворечивым образом при сохранении структуры и стиля);
8. *Отслеживаемой*(должен четко прослеживаться источник каждого из ее требований и SRS облегчает обращение к каждому из требований при дальнейшей разработке или модернизации документации).

В процессе работы с требованиями участвуют так называемые «*заинтересованные лица*», к числу которых мы будем относить:

- *Пользователей*(людей, кто будет непосредственно использовать ПО; пользователи могут описать задачи, которые они решают (планируют решать) с использованием ПС, а также ожидания по отношению к атрибутам качества, отображаемые в пользовательских требованиях, в этой роли выступает преподаватель);
- *Заказчиков*(людей, которые являются целевой аудиторией на рынке программного обеспечения, в этой роли выступает преподаватель);
- *Инженеров по программному обеспечению*(людей, ответственных за техническую оценку путей решения поставленных задач и последующую реализацию требований заказчиков, в этой роли выступают студенты - члены команды разработчиков).

Все заинтересованные лица должны прийти к соглашению о том, какая система должна быть создана, чтобы в ней воплотились необходимые дидактические и технологические свойства. Данные соглашения фиксируются в документе, с которым могут сверяться и на который могут ссылаться все участники проекта, - интегрированная спецификация требований, созданная на основе совместного использования традиционной функциональной спецификации и спецификации качества системы.

6. Разработка прототипа интерфейса пользователя

В этой фазе разработки Вы должны решить, какой интерфейс лучше всего будет подходить для достижения ваших целей - текстовый, графический или мультимедиа. Затем необходимо выбрать структуру взаимодействия, которая обеспечивает разные степени гибкости для пользователей. Обычно, чем гибче структура, тем больше она требует от пользователя обучения, понимания, и времени на работу с окнами (открыть, закрыть, разместить и т.д.).

Для выполнения начальной фазы разработки необходимо погрузиться целиком в задачи пользователей и создать бумажный прототип навигационной модели. Навигационная модель показывает, как необходимо распределять функции или задачи между окнами вашей программы, она определяет, как пользователи смогут перемещаться как между различными задачами, так и внутри отдельной задачи.

Хорошо выполненный дизайн выглядит чистым, простым и аккуратным. Его можно понять одним взглядом. Пользователь должен сразу распознавать, какие данные можно редактировать, какие нет; по каким объектам можно щелкать мышью и какие объекты можно перетаскивать

7. Разработка алгоритмов обработки данных

Выбор и обоснование алгоритмов(или разработка и описание алгоритмов). Если для организации работы системы можно использовать уже известные алгоритмы, то необходимо провести их сравнительный анализ (по эффективности) и выбрать наилучший для данной системы (при введенных ограничениях). В противном случае пользователь разрабатывает свои алгоритмы, обосновывая их необходимость. Описание алгоритма ведется в вербальной форме и с помощью схем алгоритмов [7].

Описание логической модели данных.Если в проекте данные необходимо хранить в базе данных (БД), то на данном этапе должна быть разработана концептуальная и логическая модель БД, выделены и описаны основные сущности, определены между ними отношения. Модели должны быть представлены в соответствующей нотации (ER-модель (сущность - связь),SHM-модель (семантическую иерархическую модель) [3]). Переход к реляционной модели производится в соответствии с правилами, приведенными в [4]. Обязательным условием является нормализация реляционной модели информационной базы системы.

Физическое проектирование программной системы - завершающий этап разработки системы. Он включает в себя:

- разработку пользовательского меню, которое должно быть ориентировано на структуру системы;
- описание интерфейса с обоснованием выбора того или иного стандарта оформления /1/.
- разработка модулей системы и описание их спецификаций, взаимодействие модулей должно быть представлено в виде диаграммы модулей с указанием иерархии модулей.

7 Оформление отчетов по лабораторным работам

Отчет по лабораторной работе должен включать:

1. Титульный лист, оформленный в соответствие с приложением А.
2. Введение, в котором указывается цель работы, схема лабораторной установки и описываются полученные исходные данные.
3. Ход работы, в которой описывается выполнение каждой задачи.
4. Заключение.

В целях завершения лабораторной работы в аудитории по решению преподавателя допускается сдача аккуратно оформленного рукописного отчета, включая титульный лист, со вставкой и вклейкой скриншотов, прочих рисунков и изображений графиков

Список рекомендуемой литературы

1. Программная инженерия. Визуальное моделирование программных систем : учебник для вузов / Е. А. Черткова. — 2-е изд., испр. и доп. — Москва : Издательство Юрайт, 2020. — 147 с.

2. Программная инженерия и технологии программирования сложных систем : учебник для вузов / Е. М. Лаврищева. — 2-е изд., испр. и доп. — Москва : Издательство Юрайт, 2020. — 432 с.

3. Технологии и методы программирования : учебное пособие для вузов / И. Г. Гниденко, Ф. Ф. Павлов, Д. Ю. Федоров. — Москва : Издательство Юрайт, 2020. — 235 с.

4. Практическое введение в программную инженерию : учебное пособие для вузов / В. К. Волк. — 2-е изд., стер. — Санкт-Петербург : Лань, 2022. — 100 с.

Приложение А

Образец титульного листа отчета по лабораторным работам

Министерство науки и высшего образования Российской Федерации

Томский государственный университет систем управления и радиоэлектроники

Факультет инновационных технологий
Кафедра управления инновациями

ОТЧЁТ
по лабораторной работе по дисциплинеНАИМЕНОВАНИЕ ДИСЦИПЛИНЫ
Тема лабораторной работы

Студент гр. 0ХХ

_____И.О. Фамилия

«___»_____201_г.

Преподаватель

Должность, ученая степень (еслиесть)
_____И. О. Фамилия

«___»_____201_г.

_____
оценка

Томск 201