Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «КубГУ»)

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по выполнению лабораторных работ по дисциплине
«Программная инженерия»:
часть I – технология разработки программного обеспечения

для бакалавров направления подготовки 02.03.03 «Математическое обеспечение и администрирование информационных систем» (профиль «Технологии разработки программных систем») и 09.03.04 «Программная инженерия» (профиль «Разработка программного обеспечения и приложений искусственного интеллекта»)

Краснодар,
2025

УДК 629.113(071):004.01:004.4

Методические указания к выполнению лабораторных работ по дисциплине «Программная инженерия»: часть I – технология разработки программного обеспечения, для бакалавров направления подготовки 02.03.03 «Математическое обеспечение и администрирование информационных систем» (профиль «Технологии разработки программных систем») и 09.03.04 «Программная инженерия» (профиль «Разработка программного обеспечения и приложений искусственного интеллекта») / Составители: доц. каф. ИТ Полетайкин А.Н., доц. каф. ИТ Добровольская Н.Ю. − Краснодар: КубГУ, 2025. – 55 с.

Методические указания содержат теоретические положения, задания и методические указания к выполнению лабораторных работ по дисциплине «Программная инженерия» в первом семестре изучения дисциплины. Предметом изучения является технология разработки программного обеспечения.

СОДЕРЖАНИЕ

1 Общие положения

Дисциплина «Программная инженерия» рассматривает методы, способы, модели и инструментальные средства для решения задач создания качественного программного обеспечения (ПО), программных продуктов (ПП) и программных систем (ПС) в разных сферах деятельности человека с применением современных компьютерных информационных технологий.

Изучаются основные понятия программной инженерии, составляющие процесса разработки ПО и ПП, управление требованиями к ПО, управление конфигурацией и качеством ПП. Рассматриваются и применяются на практике методы, способы и инструментальные средства для анализа предметных областей, постановки задачи на разработку ПП и ПС, формулирования требований к ним, компилятивной сборки и комплексного тестирования специального ПО для потребностей цифровой экономики с учетом задач будущей профессиональной деятельности студентов.

1.1 Организация занятий по курсу

Дисциплина «Программная инженерия» читается студентам направлений подготовки 02.03.03 «Математическое обеспечение и администрирование информационных систем» (профиль «Технологии разработки программных систем») и 09.03.04 «Программная инженерия» (профиль «Разработка программного обеспечения и приложений искусственного интеллекта») на третьем курсе обучения в осеннем и весеннем семестрах. Базовыми учебными дисциплинами для изучения дисциплины «Программная инженерия» являются «Методы программирования», «Низкоуровневое программирование», «Объектно-ориентированное программирование», «Алгоритмы и анализ сложности», «Компьютерные сети», «Базы данных», «Анализ, проектирование и разработка БД», «Операционные системы».

Целью курса является формирование у студентов знаний, умений и практических навыков в области анализа и системного представления бизнес-процессов, разработки технического задания на разработку ПП и ПС для решения задач человека в разных областях, а также организации программного процесса для создания ПП и ПС специального назначения.

Предметом учебной дисциплины являются методы, подходы и инструментальные средства программной инженерии.

Задачами дисциплины является получение представления о жизненном цикле многоверсионного программного продукта, а также приобретения навыков применения знаний и умений, приобретенных на младших курсах бакалавриата, для создания и сопровождения сложных программных проектов, отвечающих требованиям цифровой экономики.

Изучаемый курс предусматривает выполнение 18 лабораторных работ (по 9 работ в каждом семестре). В осеннем семестре выполняется разработка ПП для

реализации обработки информации и вычислений на ЭВМ в рамках заданного бизнес-процесса (см. приложение А).

Рабочая схема выполнения лабораторных работ в осеннем семестре представлена в таблице 1.1.

Таблица 1.1. Рабочая схема выполнения лабораторных работ в осеннем семестре

№ л. р.	Тема лабораторной работы: материал для изучения и характер выполняемых работ	Объем, часов	Неделя выполнения
1.	Анализ предметной области: системное описание заданного бизнес-процесса, характеристика ручного режима решения задач и выделение его недостатков, обоснование усовершенствования существующей схемы решения задач	5	1–3
2.	Анализ существующих подобных программных решений: системное описание существующих подобных решений, выделение их достоинств и недостатков, сравнительная характеристика	3	3, 4
3.	Техническое задание на создание программного продукта: назначение и общая цель создания ПП, структура ПП, функциональные и нефункциональные требования к ПП, моделирование требований на языке UML	4	5, 6
4.	Разработка функциональной структуры программного продукта (функционально-ориентированный подход): построение и документирование функциональной модели ПП в виде иерархии диаграмм в нотации IDEF0	3	7, 8
5.	Разработка концептуальной структуры программного продукта (объектно-ориентированный подход): перечисление функций ПП, формирование сценариев работы ПП, построение диаграммы классов ПП на языке UML	4	8–10
6.	Разработка функциональной структуры программного продукта (объектно-ориентированный подход): построение модели ПП на языке UML в виде комплекса диаграмм состояний, деятельности и взаимодействия	3	10, 11
7.	Разработка архитектуры программного продукта: проектирование компонентного состава ПП, разработка базы данных для ПП, построение диаграммы компонентов UML и диаграммы развертывания UML	4	12, 13
8.	Разработка специального программного обеспечения: выбор технологии программирования, разработка и отладка компонентов ПП, анализ сложности и эффективности кода, проведение рефакторинга	4	14, 15
9.	Комплексное тестирование программного продукта: ручное и автоматическое тестирование, проверка работоспособности ПП, модульное, регрессионное и нагрузочное тестирование, анализ производительности	4	16, 17
	Всего за семестр:	34	

В весеннем семестре выполняется проектирование и разработка программной системы по индивидуальной теме, выбранной и выполняемой студентом самостоятельно при консультационной поддержке преподавателя, ведущего лабораторные занятия по дисциплине, а также руководителя курсовой работы. Рабочая схема выполнения лабораторных работ в весеннем семестре представлена в таблице 1.2.

Таблица 1.2. Рабочая схема выполнения лабораторных работ в весеннем семестре

№ л. р.	Тема лабораторной работы: материал для изучения и характер выполняемых работ	Объем, часов	Неделя выполнения
1.	Анализ предметной области: согласование и утверждение индивидуальной темы, системный анализ объекта и процесса информатизации, характеристика решения задач и выделение ее недостатков, обоснование усовершенствования существующей схемы решения задач	6	1–3
2.	Анализ существующих компьютерных разработок: системное описание существующих подобных ПС, сравнительная характеристика описанных систем	3	4, 5
3.	Техническое задание на создание ПС: назначение и цель создания ПС, структура программы и состав функциональных задач, функциональные и нефункциональные требования к программе, моделирование требований на языке UML	4	5–7
4.	Проектирование функциональной структуры ПС: построение и документирование функциональной модели ПС в нотации IDEF0	3	7, 8
5.	Объектно-ориентированный подход к проектированию: Проектирование функциональной структуры ПС в нотации UML	4	9, 10
6.	Проектирование структур данных ПС: выбор программных средств моделирования структур данных ПС, проработка методов нормализации отношений в БД, создание БД ПС при помощи выбранной СУБД	3	11, 12
7.	Разработка программного обеспечения ПС: технология программирования прикладных задач, разработка интерфейсной части ПС, построение диаграмм UML физической реализации ПС, анализ производительности программы, оценивание эффективности кода	4	12–14
8.	Комплексное тестирование ПС: регрессионное и нагрузочное тестирование, системное тестирование, тестирование взаимодействия с другими системами, пользовательское тестирование	4	14–16
9.	Документирование и развертывание ПС: освоение методики документирования ПС, разработка функциональной спецификации ПС и руководства пользователя	3	16, 17
	Всего за семестр:	34	

По каждой лабораторной работе оформляется отчет. Отчеты сдаются на проверку преподавателю, ведущему лабораторные занятия, в течение курса по мере их выполнения и защищаются студентами в установленном порядке.

При защите отчета студенту могут быть заданы вопросы и дополнительные задания по теме лабораторной работы, в том числе из списка контрольных вопросов к данной лабораторной работе. При неудовлетворительной оценке знаний студента по теме данного отчета студент возвращается к повторному изучению соответствующих материалов, после чего допускается к повторной защите. Неудовлетворительно выполненный отчет также возвращается на доработку. Требования к содержанию отчетов о выполнении лабораторных работ представлены в разделе 1.2.

1.2 Требования к содержанию отчетов о выполнении лабораторных работ

В процессе изучения курса «Программная инженерия» должны быть подготовлены и сданы 18 отчетов о выполнении лабораторных работ (см. таблицы 1.1 и 1.2). Лабораторные работы выполняются в соответствии с заданием (раздел 2) в рамках индивидуальной темы.

Отчет предваряется титульным листом установленного образца (см. приложение В) и содержит тему лабораторной работы, цель, задание, индивидуальную тему, описание хода выполнения работы, необходимые прикладные материалы (схемы, макеты документов и т.п.), в соответствии с требованиями к содержанию, представленными в соответствующем подразделе раздела 2, и выводы по работе.

Отчет о выполнении каждой без исключения лабораторной работы предполагает включение в него иллюстраций (рисунков, схем, графиков, диаграмм и др.). Также большинство отчетов предполагает наличие таблиц.

Для корректного выполнения и успешной защиты отчетов необходимо перед выполнением каждой лабораторной работы внимательно и полностью прочитать задание и требования к содержанию отчета, и только после этого приступать к выполнению работы и оформлению отчета. После завершения работы над отчетом рекомендуется еще раз свериться с заданием (которое должно располагаться в начале отчета) и убедиться в том, что все пункты задания выполнены.

1.3 Требования к оформлению отчетов

Отчет должен быть выполнен печатным способом с использованием компьютера и принтера на одной стороне листа белой бумаги формата А4 через полтора интервала. Для создания текста работы рекомендуется использовать текстовый редактор MS Word. Шрифт на протяжении всего документа должен быть одинаковый: Times New Roman 14-го размера, за исключением оформления иллюстраций, таблиц и формул, в которых допускается использовать шрифт меньшего размера. Минимально допустимый размер шрифта – 12.

Текст отчета следует печатать, соблюдая следующие размеры полей: слева – 25-30 мм, справа – 10 мм, сверху и снизу – 20 мм.

При оформлении текста работы следует использовать абзацный отступ, который должен составлять 15-17 мм от левого поля документа

Вне зависимости от способа выполнения отчета качество напечатанного текста и оформления иллюстраций, таблиц, распечаток с ПЭВМ должно удовлетворять требованию их четкого воспроизведения.

Иллюстрации (чертежи, графики, схемы, компьютерные распечатки, диаграммы, фотоснимки) следует располагать в отчете непосредственно после текста, в котором они упоминаются впервые, или на следующей странице. Иллюстрации могут быть в компьютерном исполнении, в том числе и цветные. На все иллюстрации должны быть даны ссылки в отчете.

Иллюстрации следует нумеровать арабскими цифрами сквозной нумерацией. Если рисунок один, то он обозначается “Рисунок 1”. Слово “Рисунок” и его наименование располагают посередине строки. Допускается нумеровать иллюстрации в пределах раздела. В этом случае номер иллюстрации состоит из номера раздела и порядкового номера иллюстрации, разделенных точкой. Например, Рисунок 1.1. Иллюстрации, при необходимости, могут иметь наименование и пояснительные данные (подрисуночный текст). Слово “Рисунок” и наименование помещают после пояснительных данных и располагают следующим образом: Рисунок 1.1 — Функциональная схема.

При ссылках на иллюстрации следует писать “... в соответствии с рисунком 2” при сквозной нумерации и “... в соответствии с рисунком 1.2” при нумерации в пределах раздела.

Таблицы применяют для лучшей наглядности и удобства сравнения показателей. Название таблицы должно отражать ее содержание, быть точным, кратким. Название таблицы следует помещать над таблицей слева, без абзацного отступа в одну строку с ее номером через тире.

При переносе части таблицы название помещают только над первой частью таблицы, нижнюю горизонтальную черту, ограничивающую таблицу, не проводят.

Таблицу следует располагать в отчете непосредственно после текста, в котором она упоминается впервые, или на следующей странице. На все таблицы должны быть ссылки в отчете. При ссылке следует писать слово “таблица” с указанием ее номера.

Таблицу с большим количеством строк допускается переносить на другой лист (страницу). При переносе части таблицы на другой лист (страницу) слово “Таблица” и номер ее указывают один раз справа над первой частью таблицы, над другими частями пишут слово “Продолжение” и указывают номер таблицы, например: “Продолжение таблицы 1”. При переносе таблицы на другой лист (страницу) заголовок помещают только над ее первой частью.

Таблицу с большим количеством граф допускается делить на части и помещать одну часть под другой в пределах одной страницы. Если строки и графы таблицы выходят за формат страницы, то в первом случае в каждой части таблицы повторяется головка, во втором случае — боковик.

Если повторяющийся в разных строках графы таблицы текст состоит из одного слова, то его после первого написания допускается заменять кавычками; если из двух и более слов, то при первом повторении его заменяют словами “То же”, а далее — кавычками. Ставить кавычки вместо повторяющихся цифр, марок, знаков, математических и химических символов не допускается. Если цифровые или иные данные в какой-либо строке таблицы не приводят, то в ней ставят прочерк.

Таблицы следует нумеровать арабскими цифрами сквозной нумерацией. Допускается нумеровать таблицы в пределах раздела. В этом случае номер таблицы состоит из номера раздела и порядкового номера таблицы, разделенных точкой.

Заголовки граф и строк таблицы следует писать с прописной буквы в единственном числе, а подзаголовки граф – со строчной буквы, если они составляют одно предложение с заголовком, или с прописной буквы, если они имеют самостоятельное значение. В конце заголовков и подзаголовков таблиц точки не ставят. Таблицы слева, справа и снизу, как правило, ограничивают линиями. Допускается применять размер шрифта в таблице меньший, чем в тексте. Горизонтальные и вертикальные линии, разграничивающие строки таблицы, допускается не проводить, если их отсутствие не затрудняет пользование таблицей. Заголовки граф, как правило, записывают параллельно строкам таблицы. При необходимости допускается перпендикулярное расположение заголовков граф. Головка таблицы должна быть отделена линией от остальной части таблицы.

Формулы и уравнения следует выделять из текста в отдельную строку. Выше и ниже каждой формулы или уравнения должно быть оставлено не менее одной свободной строки. Если уравнение не умещается в одну строку, то оно должно быть перенесено после знака равенства (=) или после знаков плюс (+), минус (-), умножения (х), деления (:), или других математических знаков, причем знак в начале следующей строки повторяют. При переносе формулы на знаке, символизирующем операцию умножения, применяют знак “Х”.

Пояснение значений символов и числовых коэффициентов следует приводить непосредственно под формулой в той же последовательности, в которой они даны в формуле.

Формулы в отчете следует нумеровать порядковой нумерацией в пределах всего отчета арабскими цифрами в круглых скобках в крайнем правом положении на строке. Одну формулу обозначают – (1). Ссылки в тексте на порядковые номера формул дают в скобках. Пример – ... в формуле (1).

2 Методические указания

Лабораторная работа 1

Тема: анализ предметной области.

Цель: изучение и системное представление бизнес-процессов, подлежащих программированию, приобретение навыков системного анализа объектов и процессов реального мира на предмет организации программного управления.

Задание

1. Выполнить системное описание заданного бизнес-процесса. Построить модель «Черный ящик» и описать информационные потоки на ней.
2. Выполнить декомпозицию бизнес-процесса на задачи. Дать характеристику схеме решения выделенных задач в ручном режиме и выделить её недостатки.
3. Обосновать необходимость усовершенствования существующей схемы решения задач за счет разработки программного продукта.

Теоретические положения

Принцип системного анализа

При системном анализе необходимо определить целевую функцию – результат работы изучаемой бизнес-процесса (например, целевая функция работы фабрики – производство продукции, парикмахерская – выполнение стрижек и причесок, и т. д.).

Сущность системного анализа заключается в том, что система разделяется на ряд подсистем (частей), а каждая подсистема в свою очередь делится на задачи.

Понятие «подсистема» подразумевает, что выделяется относительно независимая часть системы, обладающая свойствами системы и, в частности, имеющая подцель, на достижение которой ориентирована подсистема, а также другие свойства – целостности, коммуникативности и т.п., определяемые закономерностями систем.

Система (процесс) может быть разделена на элементы (задачи) не сразу, а последовательным расчленением на подсистемы. Такое расчленение, как правило, производится на основе определения независимой функции, выполняемой данной совокупностью элементов совместно для достижения некой частной цели, которая обеспечивает достижение общей цели системы, и называется декомпозицией. Подсистема отличается от простой группы элементов, для которой не выполняется условие целостности.

Последовательное разбиение системы в глубину приводит к получению иерархии подсистем, нижним уровнем которых является элемент. С этой концепцией связано понятие структуры системы.

Системный подход к описанию бизнес-процессов

Бизнес-процесс можно представить в виде модели «Черный ящик» (рис. 2.1).

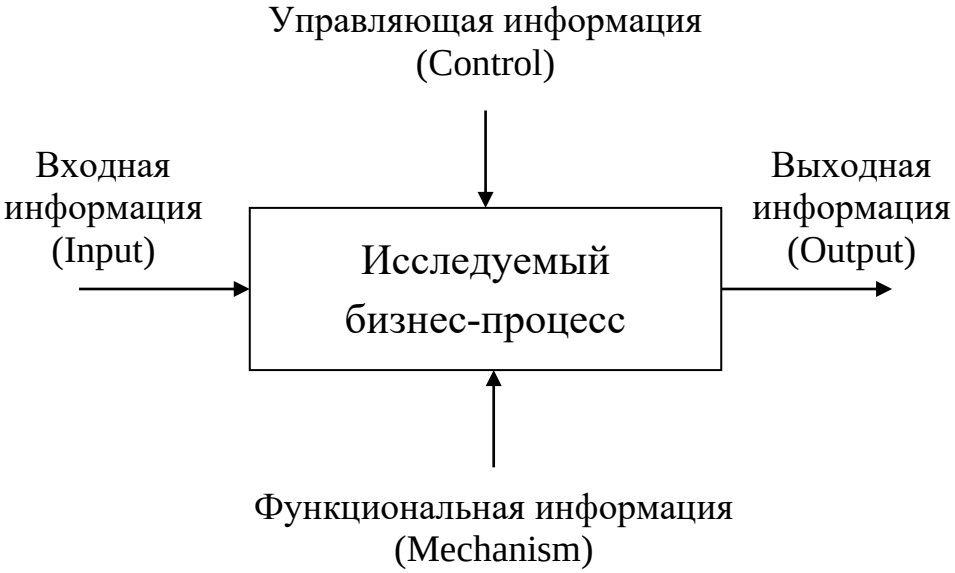

Рис. 2.1. Обобщенная схема модели «Черный ящик»

Разделение бизнес-процесса на подпроцессы (задачи) производится с учетом целевой функции бизнес-процесса, а деление операции – с учетом целевой функции подпроцесса, исходя из целевой функции бизнес-процесса. Такой принцип упрощает изучение сложных бизнес-процессов и является системным подходом. На рис. 2.2 представлен принцип декомпозиции, использующийся в системном анализе.

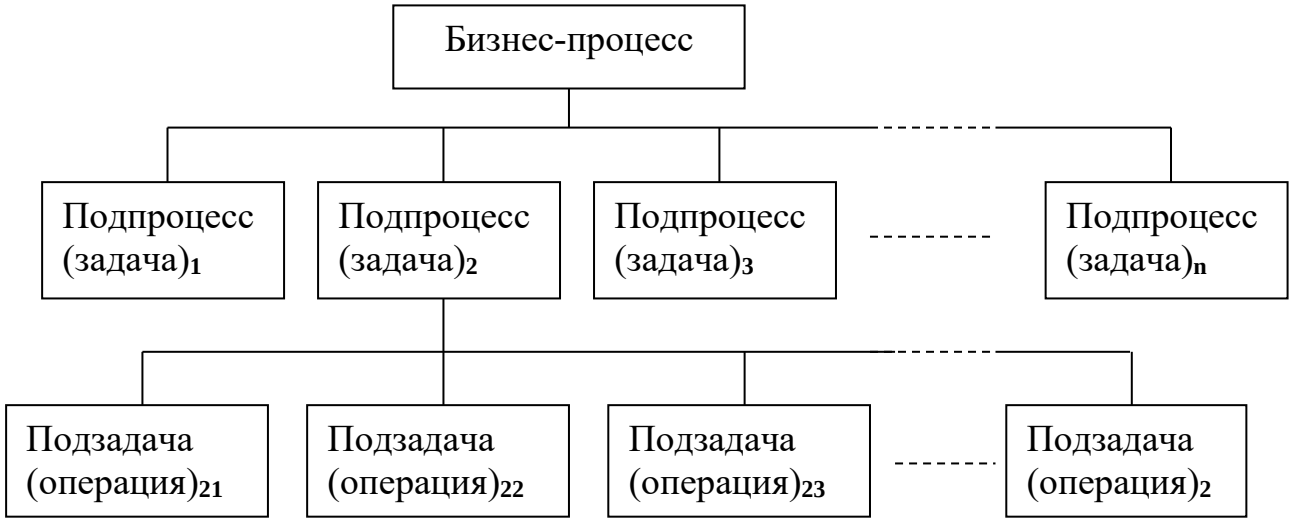

Рис. 2.2. Принцип декомпозиции бизнес-процесса

Требования к содержанию отчета

В подразделе 1 дается характеристика заданного бизнес-процесса (приложение А). Для этого выполняются следующие обязательные элементы:

- приводится подробное описание бизнес-процесса;
- определяется состав действующих лиц, задействованных в рассматриваемом процессе;
- определяется входная и выходная информация, строится структурная схема типа "черный ящик" (см. рис. 2.3).
- выполняется описание входных и выходных информационных потоков в виде табл. 2.1;
- приводятся правила обработки информации и возможные ограничения;
- определяется нормативно-справочная документация, регламентирующая бизнес-процесс.

Рис. 2.3. Пример модели «Черный ящик» для бизнес-процесса обмена валюты

Следует учитывать, что построенная модель бизнес-процесса – это информационно взаимосвязанный функциональный комплекс, назначение которого – переработка входной информацию в выходную.

Таблица 2.1. Описание информационных потоков

№	Наименование и назначение потока	Вид потока	Форма представления	Корреспондент (Откуда)	Периодичность, регламент

Наименование инфопотока должно быть лаконичным и конкретным (например, не «Отчет», а «Отчет о продажах за период»). Возможные виды инфопотоков показаны на рис. 1: Input, Output, Control, Mechanism.

Возможные формы представления информации: сигнал, устное сообщение, устное распоряжение, документ, датасет и др.

Если формой представления инфопотока является документ, целесообразно представить его макет, либо копию. При этом необходимо кратко описать такие реквизиты документа, смысл которых не является очевидным.

В целом описание бизнес-процесса должно давать представление о вычислительных задачах, предполагаемых заданным бизнес-процессом, величине информационного массива, подлежащего обработке, численности персонала, занятого в процессе создания специального ПО для решения выделенных задач.

Для более строгого структурирования данного материала рекомендуется использовать табличную форму подачи информации.

В подразделе 2 производится декомпозиция бизнес-процесса на подпроцессы (задачи). Фактически производится разукрупнение модели «Черный ящик». Строится древовидная структура, подобная показанной на рис. 2.2. При этом следует помнить, что задача программы – это формализованная совокупность действий, выполнение которых приводит к результату заданного вида. Поэтому в качестве задач надо выбирать такие, для которых можно четко сформулировать результат. Необходимо выделить такие задачи:

1) задача ввода входных данных;
2) задача сохранения данных в памяти ЭВМ;
3) задача формирования выходных данных;
4) задача вычисления некоторого итогового показателя;
5) задача статистического анализа данных, и др.

По каждой задаче необходимо дать общую информацию, достаточную для понимания следующих вещей:

1) для чего решается задача;
2) какой конкретный результат она дает;
3) какие исходные данные для этого необходимы.

В подразделе 3 в повествовательной форме простыми предложениями описываются основные операции, которые выполняются при сборе и обработке информации, а также связанные с этим действия персонала предприятия без использования специального программного обеспечения. Указываются те недостатки этой системы, которые приводят к снижению эффективности решения данных задач и бизнес-процесса в целом.

По результатам анализа недостатков дается обоснование необходимости создания специального ПО для автоматизации бизнес-процесса. Излагаются причины, вследствие которых создание программ для решения задач бизнес-процесса является необходимым.

В целом из материала отчета должно быть видно, что представляет собой заданный бизнес-процесс, какие задачи при этом решаются, какие из них выполняются не достаточно эффективно, почему и на каком уровне необходима программная реализация указанных задач.

Рекомендуемая литература

[1] – глава 2.
[2] – раздел 1.1, подразделы 1.3.5 и 2.1.6.
[3] – главы 1 и 2.

Контрольные вопросы

1. Что такое декомпозиция бизнес-процесса?
2. Какова типовая структура декомпозиции бизнес-процесса?
3. Что такое схема типа "черный ящик"?
4. Что такое документооборот бизнес-процесса?
5. Что такое нормативно-справочная документация, регламентирующая бизнес-процесс?
6. В чем заключаются цели и назначение выбранного бизнес-процесса и какова его динамика?
7. Что такое программа?
8. Что такое задача программы?
9. Чем определяется эффективность решения задач программы?
10. В чем различия между задачей бизнес-процесса и задачей программы?
11. Каковы могут быть основания для создания специального ПО для автоматизации бизнес-процесса?
12. Какие задачи выбранного бизнес-процесса решаются не достаточно эффективно и почему?
13. Какие критерии эффективности могут быть использованы для оценивания эффективности реализации бизнес-процесса?
14. Перечислите задачи в структуре заданного бизнес-процесса. Дайте краткую характеристику одной из задач, включая задействованные документы.
15. Насколько целесообразным является решение об автоматизации выделенных задач бизнес-процесса?
16. Приведите пример правил обработки информации при описании бизнес-процесса. Выполните описание основных операций, которые выполняются при сборе и обработке информации для конкретной задачи бизнес-процесса.
17. Укажите несколько недостатков, которые приводят к снижению эффективности решения задач бизнес-процесса.

Лабораторная работа 2

Тема: анализ существующих подобных программных решений.

Цель: освоение методики системного анализа программных решений поставленной задачи и выработка навыков анализа существующих программных продуктов на российском и зарубежном рынках.

Задание

1. Выполнить системное описание существующих подобных программных решений (не менее двух), которые могут быть применены в заданной предметной области, описанной при выполнении лабораторной работы 1. Выделить основные преимущества и недостатки представленных решений.
2. Выполнить сравнительную характеристику описанных ПП. Примерный набор показателей для сравнения (определения показателей см. в приложении Б):
   - назначение системы;
   - эффективность системы;
   - гибкость системы;
   - защищенность системы;
   - живучесть системы;
   - надежность системы;
   - открытость системы;
   - оптимальность использования ресурсов;
   - удобство пользовательского интерфейса системы;
   - стоимость системы (в том числе затраты на тех. поддержку);
   - эргономичность.
3. Сделать вывод о возможности и целесообразности использования этих решений на выбранном объекте автоматизации.

Требования к содержанию отчета

В подразделе 1 одно за другим приводятся описания существующих ПП, автоматизирующих объекты и процессы, подобные имеющим место в выбранной предметной области. Описание ПП выполняется *системно*, включая элементы, их взаимосвязи, цели, функции, ресурсы, а также допустимые режимы работы. Описание уместно сопровождать следующими иллюстрациями:

- функционально-структурная схема;
- обобщенная блок-схема алгоритма функционирования системы;
- экранные формы основных частей пользовательского интерфейса;
- таблицы и графики, отражающие статистические показатели функционирования ПП.

Для каждого продукта указываются преимущества и недостатки: вообще, а не применительно к выбранному объекту.

В подразделе 2 выполняется сравнение представленных систем по нескольким показателям. Сравнение подкреплять количественными показателями (например, сроки внедрения, объем дискового пространства, количество единиц техники, стоимость ПО и т.п.). Результаты сравнения уместно представить в табличном формате. Также здесь уместно приведение статистических оценок в виде таблиц, диаграмм, графиков.

В подразделе 3 делается заключение о возможности применения рассмотренных ПП к выбранному объекту автоматизации. Оценивается степень этой возможности на предмет удобства и скорости настройки системы на данную предметную область, а также оптимальности их внедрения с учетом затрат на дальнейшее обслуживание и поддержку.

В целом из материала отчета должно быть видно, какие из систем, автоматизирующих подобные выбранным объекты и процессы, существуют на отечественном и зарубежном рынке. Какой комплекс задач они позволяют решить, насколько оперативны и эффективны получаемые решения и насколько они соответствуют целям, поставленным в лабораторной работе 1.

Рекомендуемая литература

[1] – раздел 1.2.
[2] – подразделы 1.1.2 и 1.1.3.
[3] – глава 3.

Контрольные вопросы

1. Насколько широк спектр существующих информационных систем, которые уместно применить к выбранному объекту автоматизации?
2. Какая из рассмотренных систем наиболее близка для применения на выбранном объекте автоматизации, и почему?
3. Насколько глубокие изменения необходимо произвести в той или иной из описанных систем для ее использования в другой предметной области?
4. Как быстро произойдет окупаемость той или иной из описанных систем при их внедрении на выбранном объекте?
5. Почему какую-либо из рассмотренных систем целесообразно применить на выбранном объекте, а почему нецелесообразно?

Ответы должны подкрепляться количественными оценками.

Лабораторная работа 3

Тема: техническое задание на создание программного продукта.

Цель: освоение методик предварительного анализа разрабатываемой программы, формулирования функциональных и нефункциональных требований к программной реализации отдельных задач и к программе в целом, выработка навыков разработки технического задания.

Задание

1. Установить назначение и общую цель создания программы. Определить структуру программы и состав функциональных задач.
2. Разработать функциональные требования к программе и построить модель требований в нотации UML.
3. Разработать специальные требования к информационному и математическому обеспечению программной реализации задач.
4. Разработать требования к инструментальному ПО, необходимому для разработки и эксплуатации ПП.
5. Установить необходимые нефункциональные требования к ПП.

Требования к содержанию отчета

В подразделе 1 описывается назначение программы и цели ее создания. При описании назначения ПП указывают вид деятельности, которая автоматизируется (учет, расчет, анализ, управление, диагностика, проектирование и т.п.) и перечень объектов информатизации, на которых предполагается ее использовать. При описании цели создания ПП приводят наименование и необходимые значения технических, технологических, производственно-экономических или других показателей объекта автоматизации, которые должны быть достигнуты в результате создания ПС, и указывают критерии оценки достижения указанных целей. Цель должна отвечать на вопросы:

1) для чего нужен ПП?

2) что улучшается на объекте информатизации за счет использования ПП?

Также в этом подразделе указывается перечень задач, программную реализацию которых предполагается осуществить.

В подразделе 2 разрабатывается статическая объектная модель будущей программы в виде диаграммы вариантов использования. Диаграммы реализовать в пакете программ Rational Rose, Star UML или Altova UModel.

Диаграмму следует стрить дедуктивным методом – от общего к частному. В качестве базовых вариантов использования следует взять задачи, перечень которых указан в подразделе 1. Далее необходимо их детализировать, используя отношения включения, расширения и обобщения, принятые в нотации UML для данной диаграммы.

По ходу детализации необходимо формулировать функциональные требования к ПП. При этом необходимо учитывать, что потребуется обеспечить их выполнение при программировании соответствующих задач.

Изложение функциональных требований разбивается на несколько пунктов в зависимости от количества задач, например:

3.1 Требования к задаче “...”

3.2 Требования к задаче “...”

3.3 Требования к задаче “...”

В каждом пункте указывается полное название задачи. Кратко излагается, какие в точности действия программа должна выполнить для реализации данной задачи, в форме словесного или формульно-словесного описания алгоритма. Кроме того, по каждой задаче могут быть указаны следующие требования:

– к выходным (результатным) данным;
– к входным (исходным) данным;
– к преобразованию входной информации к машиночитаемому виду;
– к временному регламенту реализации каждой задачи;
– к качеству реализации каждой задачи;
– к возможной одновременности выполнения нескольких задач;
– к достоверности результатных данных.

При описании требований к входным данным приводится: перечень и описание входных данных (идентификатор, форма представления, сроки и частота поступления), перечень и описание структурных единиц информации входных сообщений или ссылка источники этих данных. В описании для каждой структурной единицы информации входных сообщений следует указывать:

– наименование;
– необходимую точность ее числового значения (при необходимости);
– источник информации (документ, видеокадр, устройство, кодограмма, информационная база на машинных носителях и т.д.);
– идентификатор источника информации.

Требования к программной реализации задач должны содержать:

– требования к организации хранения данных в виде таблиц базы данных, файлов или каких-либо других структур данных;
– предварительное описание (структуру) интерфейса для доступа к реализации отдельных задач и подзадач;
– требования к методу программирования, к структуре кода, к именованию программных и информационных объектов.

В итоге подраздел 2 должен содержать максимально полное описание того что и как должен делать ПП в виде функциональных требований к нему, а также соответствующую этому описанию хорошо детализированную диаграмму требований UML. Пример такой диаграммы показан на рис. 2.4.

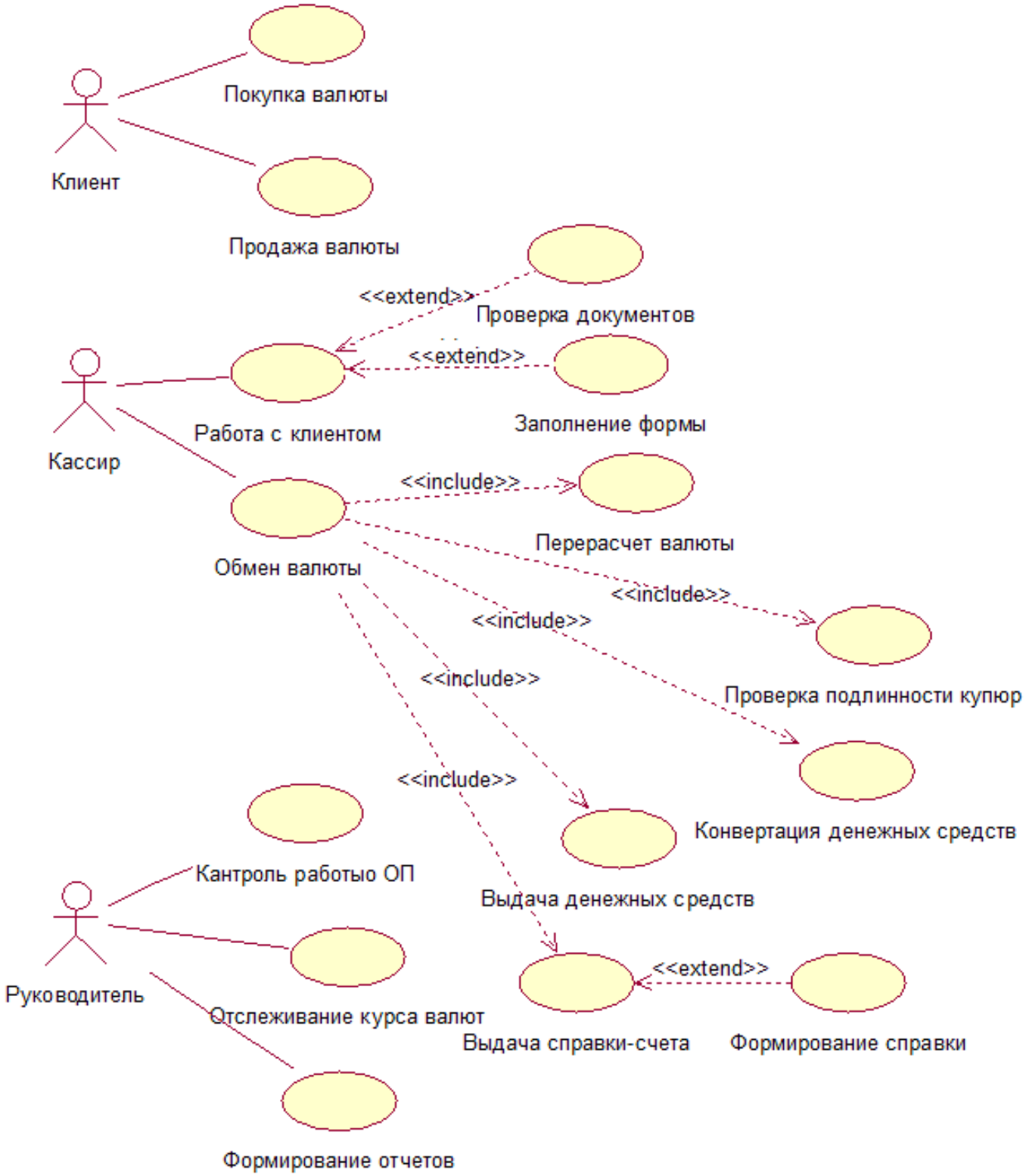

Рис. 2.4. Диаграмма требований для ПП выполнения операций обмена валюты

Подраздел 3. При формулировке специальных требований к информационному обеспечению ПП приводят требования к составу и структуре информации, обрабатываемой и генерируемой ПП. Предъявляются требования к структурированию данных и используемому для этого прикладному инструментальному ПО (СУБД, API и др.). Эти требования не предполагают исчерпывающей детализации. Не нужно приводить подробные

описания структур данных с указанием типов данных всех атрибутов, ограничений и значений по умолчанию – это содержание проектных решений. Исключение составляют принципиальные требования к обязательному наличию определенных атрибутов (напр., контактные данные клиента), необходимым типам и форматам данных (напр., формат JSON), формату и качеству графических данных и медиаданных, и т.п. Также могут быть предъявлены требования к режиму доступа к общим данным, к распределение информационных ресурсов (напр., по локальной сети или в облачной среде), а также к информационной безопасности ПП

При формулировке специальных требований к математическому обеспечению программной реализации задач приводят требования к составу, специфике применения, включая ограничения на применения, математических методов и моделей, типичных алгоритмов и специальных алгоритмов, подлежащих разработке. При необходимости предъявляются требования к используемому для этого прикладному инструментальному ПО (напр., MathLab, AnyLogic и др.). Детализация этих требований может касаться, например, параметризации математических моделей (напр., пороговые вероятности операторов генетического алгоритма) либо требования к подготовке данных (напр., требования к сэмплингу или инструменту обогащения данных), а также особенностей комплексирования методов и миграции данных между моделями, и т.п.

В подразделе 4 указываются:

– требования к операционной среде;

– требования к инструментальным средствам программной инженерии, обеспечивающих разработку ПО (CASE-средства и средства объектно-ориентированного моделирования);

– требования к инструментальным средствам разработки ПО;

– требования к использованию готовых программных пакетов;

– требования к вспомогательным программным средствам (сервисные программы и утилиты).

В подразделе 5 указывают нефункциональные требования к ПП. Требования этого вида не являются описанием функций программы и описывают качественные характеристики ПП. Их исчерпывающий перечень представлен в приложении Б. Поскольку нефункциональные требования часто относятся ко всей системе в целом важно, чтобы их формулировки были предельно конкретными и увязаны с требованиями функциональными по смыслу и терминологически.

В числе нефункциональных требований можно выделить требования к аппаратному обеспечению, в качестве которых указываются требования к функциональным, конструктивным и эксплуатационным характеристикам технических средств, задействованных в функционировании ПП. При формулировании аппаратных требований к ПП необходимо учитывать, что эти требования обусловленные спецификой задач, которые решаются в программой. При описании требований к аппаратному обеспечению необходимо указывать не конкретные значения характеристик аппаратных

компонентов или конкретные типы, а лишь определенные ограничения на эти характеристики.

Следует учитывать, что реализация нефункциональных требований часто требует больших затрат, чем функциональных. Так, сопровождаемость требует значительных усилий по поддержанию соответствия проекта исходному коду и применения специальных методов создания модифицируемых программ и структур данных. Надежность – дополнительных средств восстановления системы при сбоях. Эффективность – поиска специальных архитектурных решений и оптимизации структуры системы и программного кода. А удобство – проектирования не «интуитивно» понятного, а профессионально понятного интерфейса пользователя.

Рекомендуемая литература

[1] – глава 3.
[2] – раздел 1.4, подразделы 1.1.8, 1.2.6, 2.1.6
[3] – глава 4.

Контрольные вопросы

1. Что такое техническое задание и какова его структура?
2. Для чего и зачем разрабатываются компьютерные программы?
3. Какие положительные результаты могут быть получены в процессе использования компьютерной программы?
4. Что подразумевает программная реализация задач бизнес-процесса?
5. Каково назначение разрабатываемой компьютерной программы?
6. Назовите цели, в соответствии с которыми разрабатывается программа.
7. Дайте определение функционально-структурной и объектной модели компьютерной программы. Укажите принципиальные различия между этими моделями.
8. Перечислите базовые объектные модели компьютерной программы и компьютерные средства их реализации.
9. Что такое функциональные и нефункциональные требования к компьютерной программе?
10. Принципы формулировки требования к компьютерной программе при разработке технического задания.
11. Укажите основные аспекты формулирования функциональных требований к программной реализации задач бизнес-процессов.
12. Какие виды обеспечения компьютерной программы разрабатываются при ее создании?
13. Что такое прикладное обеспечение компьютерной программы?
14. Какие требования предъявляются к входным и выходным данным при программной реализации задач бизнес-процессов?

15.Какие требования предъявляются собственно к программной реализации задач бизнес-процессов?

16.Какие требования предъявляются к прикладному математическому обеспечению при программной реализации задач бизнес-процессов?

17.Улучшение каких технических, технологических, производственно-экономических или других показателей бизнес-процесса может быть достигнуто в результате создания и использования компьютерной программы?

18.Какова численность оперативного персонала, задействованного в автоматизируемых задачах до и после предполагаемого внедрения компьютерной программы?

Лабораторная работа 4

Тема: разработка функциональной структуры программного продукта: функционально-ориентированный подход.

Цель: изучение методики функционально-ориентированного подхода программной инженерии для разработки и описания функциональности разрабатываемого программного обеспечения.

Задание

1. Построить функциональную модель разрабатываемого ПП в виде контекстной диаграммы в нотации IDEF0 при помощи пакета BPWin.
2. На основе контекстной диаграммы и диаграммы требований UML, построенной при выполнении лабораторной работы 3, построить диаграмму декомпозиции A0 на дочерние подпроцессы (автоматизированные функции) ПП.
3. Опираясь на диаграмму A0 выполнить перечисление автоматизированных функций ПП. Для не простых функций построить диаграммы декомпозиции A2.

Требования к содержанию отчета

В подразделе 1 выполняется разработка контекстной диаграммы в нотации IDEF0 при помощи пакета программ BPWin 4.1. В основе модели лежит принцип «черного ящика» (см. рис. 2.1, 2.3). Построение диаграммы выполняется на основании разработанной при выполнении лабораторной работы 1 структурной схемы типа «черный ящик» с обозначенными входными (Input) выходными (Output) данными, а также списков нормативно-справочной документации (Control) и лиц, состав, задействованных в бизнес-процессе (Mechanism).

Следует иметь в виду, что при выполнении лабораторной работы 1 моделью черного ящика был описан процесс информатизации как есть. На этапе постановки задачи происходит переосмысление процесса информатизации и некоторые его подпроцессы в ПП реализуются полностью автоматически. В таком случае внешнее дополнение ПП на модели черного ящика может отличаться от исходной модели процесса информатизации. Например, взяв за основу модель, показанную на рис. 2.3, можно заметить, что эктор «Клиент» является лишним, так как не является пользователем ПП. К тому же очевидно, что эти данные о курсах валюты не вводятся в ПП как входные ни одним из действующих лиц. Вместо этого данные о курсах валют по логике работы программы извлекаются из корпоративной базы данных непосредственно при расчете суммы к выплате. Поэтому контекстная диаграмма для ПП обмена валют будет выглядеть так, как показано на рис. 2.5.

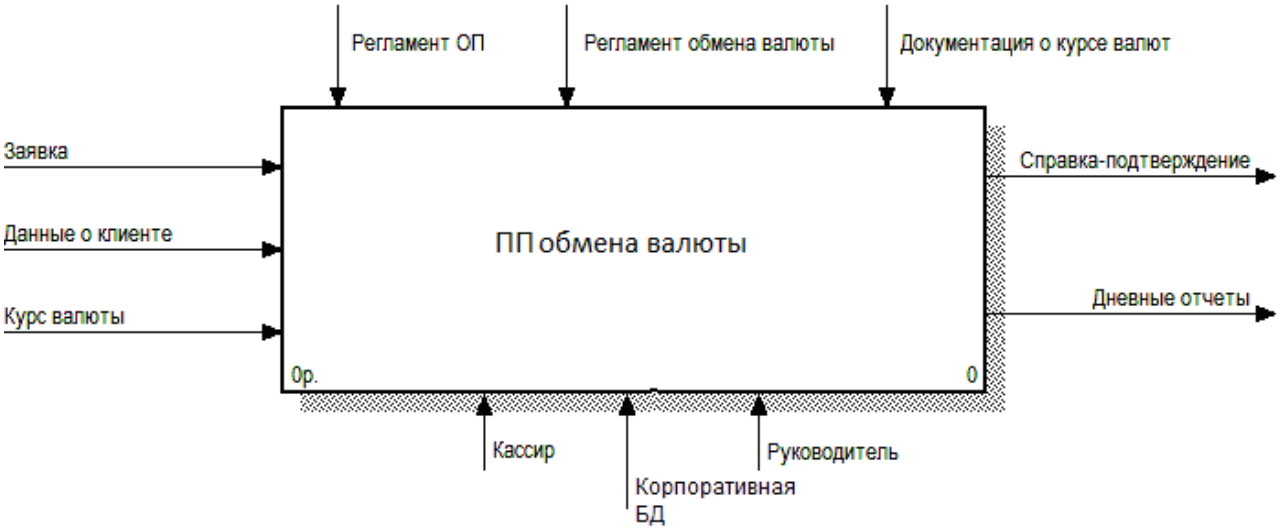

Рис. 2.5. Контекстная диаграмма ПП выполнения операции обмена валюты

В подразделе 2 так же при помощи CASE-средства BPwin 4.1 строится диаграмма декомпозиции A0. Дочерние процессы диаграммы должны в точности соответствовать задачам, заявленным при формулировании функциональных требований к ПО (лабораторная работа 3) и нашедших свое отражение на диаграмме требований UML, построенной при выполнении лабораторной работы 3. Связи (интерфейсные дуги) диаграммы должны учитывать функциональные требования к задачам, в частности к входной и выходной информации, к преобразованию данный, и др. При этом следует придерживаться требований к построению диаграмм:

- любая дуга может иметь ответвление;
- слияние дуг не допускается;
- все дуги должны быть именованы;
- во всей модели не должно быть дуг с одинаковыми именами.

Пример контекстной диаграммы, выполненной при помощи CASE-средства BPWin для декомпозиции черного ящика на рис. 2.5, представлен на рис. 2.6. Стрелка эктора «Корпоративная БД» отражает механизм автоматического запроса курса валюты при выполнении функции «Операция обмена валюты» (см. блок 2 на рис. 2.6).

В подразделе 3 выполняется перечисление выделенных на диаграмме A0 автоматизированных функций ПП. Перечисление выполняется для каждой функции в отдельности в виде описания простыми повествовательными предложениями. Объем текста описания для простой информационной функции обычно составляет 30–50 слов.

В случае описания сложной вычислительной или управляющей функции слов может быть больше в разы. При этом в описание при необходимости могут быть включены формулы, таблицы и дополнительные иллюстрации. Так, например, в описание функции операции обмена валюты может быть представлена формула для расчета суммы к выдаче. При этом возможны отсылки к материалам отчета о выполнении лабораторной работы 1.

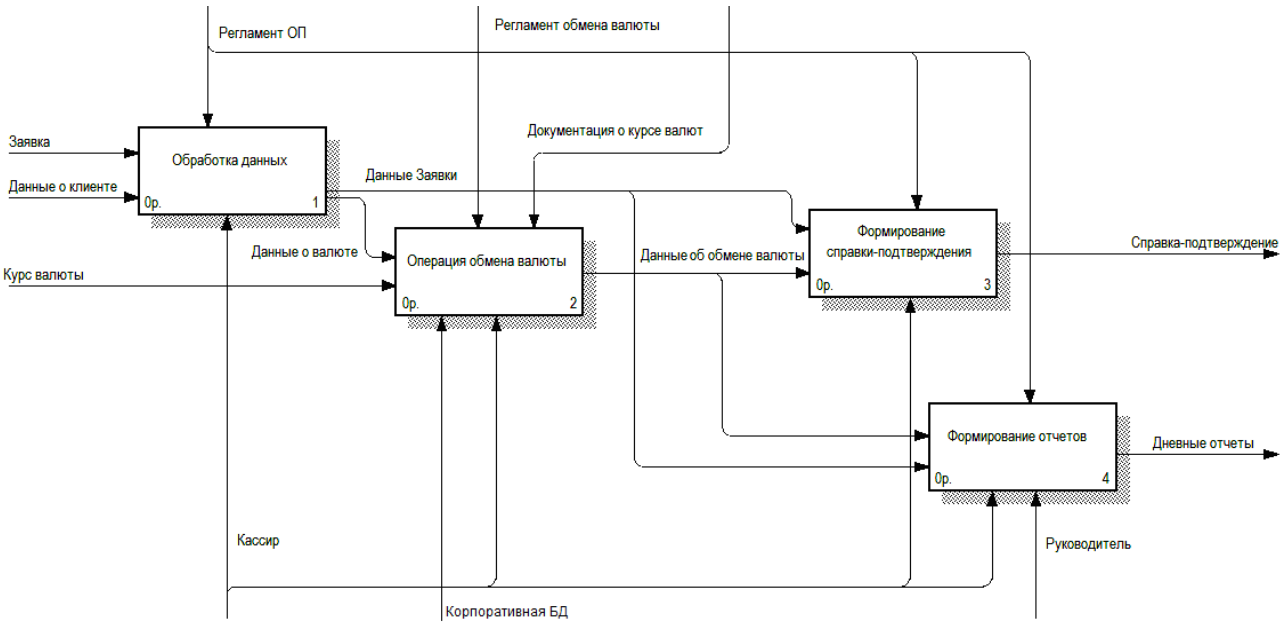

Рис. 2.6. Диаграмма декомпозиции A0 для ПС выполнения операции обмена валюты

Для не простых функций рекомендуется выполнять дальнейшую их декомпозицию на операции. В качестве операций следует определять такие, для которых можно четко сформулировать результат (Output). Отдельно рассматриваются действия, которые выполняются программными или техническими средствами и человеком (Mechanism). Каждая диаграмма декомпозиции A2 должна отражать все действия, направленные на программную обработку и хранение входной информации, а также удовлетворять вышеуказанным требованиям к построению диаграмм.

Рекомендуемая литература

[1] – глава 6.
[2] – подразделы 1.2.1, 1.2.2, раздел 1.3.
[3] – подразделы 2.2.1 и 5.1.2.

Контрольные вопросы

1. В чем заключается функционально-ориентированный подход к разработке и описанию функциональности компьютерной программы?
2. Каково назначение и структура функциональной модели компьютерной программы?
3. Что такое перечисление автоматизированных функций компьютерной программы?
4. Чем отличаются модели черного ящика, построенные при выполнении лабораторных работ 1 и 4?

5. Что такое нотация IDEF0?
6. Какова структура функциональной модели в нотации IDEF0?
7. В чем заключается принцип «черного ящика» и принцип декомпозиции при построении функциональной модели в нотации IDEF0?
8. Перечислите основные правила построения диаграмм в нотации IDEF0.
9. Как выражаются функциональные требования к компьютерной программе на диаграммах в нотации IDEF0?
10. Какие существуют ограничения на декомпозицию в нотации IDEF0?
11. Как документируются диаграммы в нотации IDEF0?
12. Выполните функциональное моделирование работы компьютерной программы в нотации IDEF0.

Лабораторная работа 5

Тема: Разработка концептуальной структуры программного продукта: объектно-ориентированный подход

Цель: освоение методики объектно-ориентированного анализа, приобретение навыков формирования сценариев работы ПП и построения диаграммы классов ПП на языке UML.

Задание

1. Используя материалы отчета о выполнении лабораторной работы 4 и методику объектно-ориентированного анализа, выполнить полное перечисление потока событий каждой автоматизированной функции и составить пошаговое описание ее реализации посредством ПП.
2. Для каждой автоматизированной функции перечислить и описать все возможные альтернативные варианты реализации функции, включая некорректные ситуации и аномалии, которые могут иметь место при реализации функции и оказывать существенное влияние на работу ПП в целом.
3. Скорректировать поток событий при нормальной реализации обследуемой функции так, чтобы последствия перечисленных аномалий оказались минимальными.
4. Проанализировать описанные потоки событий на предмет выявления набора абстракций предметной области проектируемого ПП. Для этого провести синтаксический анализ текстов потоков событий. В качестве предварительных кандидатов в абстракции принять подлежащие, выделенные из потоков событий.
5. Разделить выделенные абстракции на три типа: сущности, поведения, интерфейсы. Для каждой абстракции указать ее вид согласно принятой классификации (табл. 2.4). Результат представить в виде таблицы 2.2.

Таблица 2.2. Абстракции подсистемы

№	Абстракция	Тип	Вид	Описание

6. Проанализировать поведение выделенных абстракций. Абстракции, для которых уместно выделить возможное поведение в пределах функциональности ПП, представленной диаграммой требований UML, занести в табл. 2.3.

Таблица 2.3. Абстракции подсистемы и их поведение

№	Абстракция	Требование	Описание поведения

7. Опираясь на данные табл. 2.3 построить диаграмму классов UML (class diagram), указывая при этом лишь имена классов без указания свойств класса. Пример такой диаграммы приведен на рис. 2.7.
8. Выявить в потоках событий атрибуты классов и внести их в секции классов на диаграмме. В качестве атрибутов выбирать подлежащие,

выявленные в потоках событий и не вошедшие в табл. 2.2, а также абстракции из табл. 2.2, не вошедшие в табл. 2.3.

9. Выявить в потоках событий операции классов и внести их в секции классов на диаграмме. В качестве операций выбирать сказуемые, выделенные из текстов анализируемых потоков событий.

10. Установить между классами адекватные отношения и обозначить их кратность. При необходимости пометить отношения текстовыми метками.

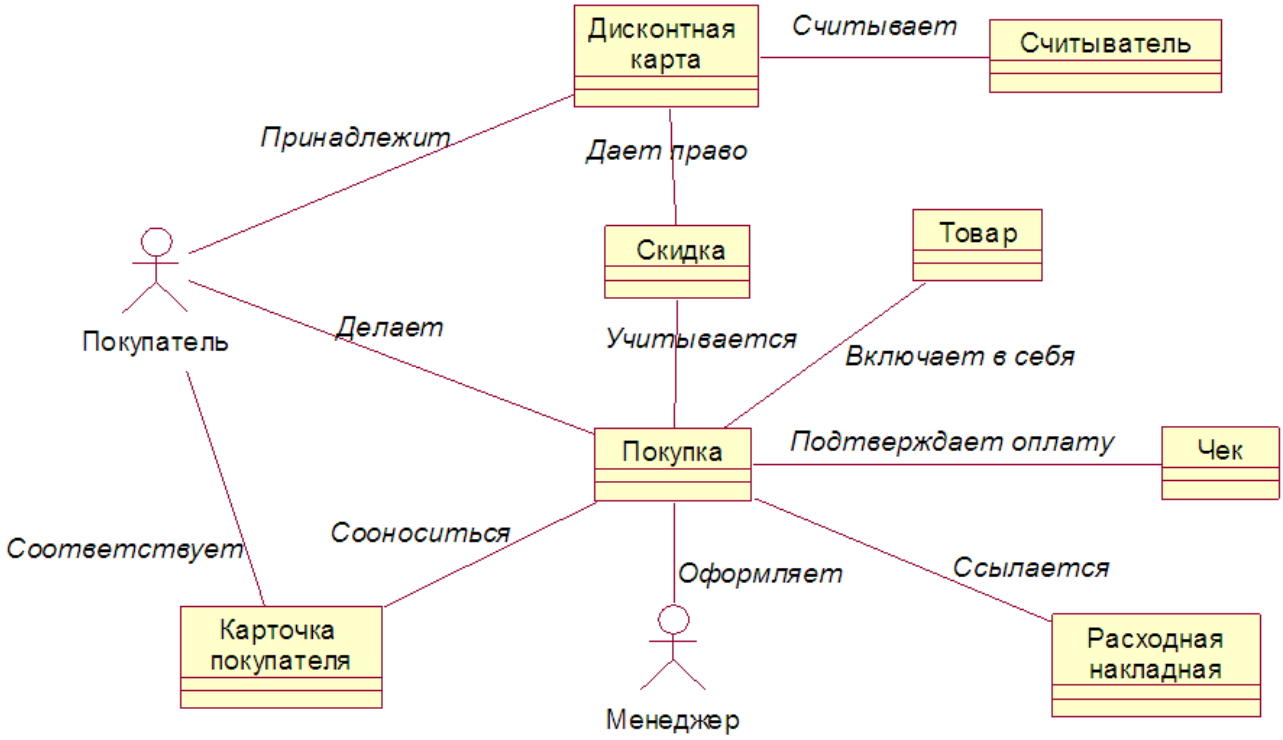

Рис. 2.7. Пример диаграммы классов UML с пустыми секциями атрибутов и операций

Теоретические положения

Объектно-ориентированный анализ – это методика, при которой требования к системе воспринимаются с точки зрения классов и объектов, выявленных в предметной области.

Для идентификации классов и объектов используют текстовые описания потоков событий проектируемой системы. Из этих описаний выделяют кандидатов в абстракции. Для этого используют синтаксический разбор предложений из текстов потоков событий и в качестве кандидатов в абстракции принимают подлежащие. Абстракции, которые обладают выраженным поведением, принимают в качестве классов. Остальные становятся атрибутами классов.

Все абстракции могут быть разделены на 3 типа:

– абстракция сущности – объект представляет собой полезную модель некой сущности в предметной области;

– абстракция поведения – объект состоит из обобщенного множества операций;

– абстракция-интерфейс – объект группирует операции, которые вместе используются более высоким уровнем управления как интерфейс системы.

Также абстракции могут быть разделены на виды в соответствии с классификатором, представленном в табл. 2.4. Результатом анализа являются характеристики абстракций, которые сводятся в табл. 2.2 и 2.3.

Таблица 2.4. Классификатор видов абстракций

Вид	Описание
Роли	Роли, которые исполняют пользователи, работающие с системой
Предметы	Осязаемый материальный объект или группа объектов
Места, локации	Области, связанные с людьми или предметами, существенные для работы системы
Инструменты	Средства деятельности, используемые определенными ролями
Организации	Формально организованная совокупность людей, ресурсов, оборудования, инструментов, которая имеет определенную цель и существование которой в целом не зависит от людей
Концепции	Принципы и идеи, сами по себе неосязаемые, но предназначенные для организации деятельности и/или общения, или же для наблюдения за ними
События	Нечто случающееся с чем-то/кем-то в заданное время или последовательно. Происшествия, которые должны быть зафиксированы
Показатели	Характеристики, определяющие деятельность, либо ее отдельные процессуальные компоненты
Устройства	Устройства, с которыми взаимодействует система
Другие системы	Внешние системы, с которыми взаимодействует проектируемая система

Требования к содержанию отчета

В подразделе 1 представляется основной поток событий для каждой автоматизированной функции ПП при нормальном его функционировании без учета возможных проблем, ошибок и некорректных ситуаций. Поток событий выполняется по пунктам нумерованным списком для каждой функции в отдельности. При этом следует обеспечивать максимально полное соответствие данного потока диаграмме требований UML, а также диаграммам A0 и A2. При обнаружении любых несоответствий следует их устранять посредством внесения модификаций в соответствующие проектные решения. При этом все модификации должны быть документированы (что, где и почему изменено).

По аналогичному принципу, но с учетом возможных проблем и некорректных ситуаций, разрабатываются альтернативные потоки. А для обработки возможных ошибок при реализации основных и альтернативных потоков разрабатываются потоки ошибок. После этого приводится корректировка основных потоков событий при нормальной реализации функций так, чтобы последствия обрабатываемых аномалий оказались минимальными.

В качестве результатов приводятся скорректированные основные потоки, альтернативные потоки и потоки ошибок, а также описание модификаций диаграмм, если таковые выполнялись при разработке потоков событий.

В подразделе 2 приводятся результаты объектно-ориентированного анализа, в том числе описание абстракций в формате табл. 2.2 и 2.3, а также полностью разработанная диаграмма классов UML с тремя заполненными секциями и установленными отношениями между классами. Диаграмма должна быть выполнена при помощи CASE-средства. Рекомендуемое CASE-средство: IBM Rational Rose.

Необходимо убедиться, что нанесенные на диаграмму классов UML операции представлены на диаграмме требований UML и на диаграммах декомпозиции A0 и A2. Обнаруженные несоответствия должны быть устранены, а соответствующие модификации задокументированы в отчете.

Процесс построения диаграммы классов должен быть откомментирован по тексту отчета. В частности, должен быть обоснован выбор абстракций поведения и атрибутов классов, а также отношений между классами и их множественности.

В конце отчета приводится диаграмма вариантов использования UML для разрабатываемого ПП, которая получена как модифицированная в результате выполнения лабораторной работы диаграмма требований UML.

Рекомендуемая литература

[1] – раздел 1.5.
[2] – подразделы 1.3.6, 2.1.7, 2.1.8.
[3] – раздел 1.2, 8.1, 8.2, 8.4.

Контрольные вопросы

1. В чем заключается метод объектно-ориентированного анализа?
2. В чем заключается метод объектно-ориентированного проектирования?
3. В чем заключается метод перечисления автоматизированных функций?
4. Что такое поток событий? Какова технология его построения?
5. Как выполняется идентификация абстракций?
6. Что такое абстракция поведения?
7. Что такое диаграмма классов UML?

8. Какие секции имеет класс в нотации UML?

9. Какие выделаются типы классов UML? В чем их принципиальное различие?

10. Какие отношения между классами устанавливаются на диаграмме классов UML?

11. В чем заключается установление отношений между классами на диаграмме классов UML?

12. Как соотносятся между собой базовые диаграммы UML: диаграмма требований и диаграмма классов?

13. Как соотносятся между собой базовые диаграммы UML и диаграммы в нотации IDEF0?

14. В чём отличия диаграммы вариантов использования UML от диаграммы требований UML.

15. Какие CASE-средства могут быть использованы для объектно-ориентированного проектирования ПО?

Лабораторная работа 6

Тема: разработка функциональной структуры программного продукта: объектно-ориентированный подход

Цель: изучение методики объектно-ориентированного подхода программной инженерии для разработки и описания функциональности разрабатываемого программного продукта, освоение средств языка UML для разработки функциональной модели ПО.

Задание

Опираясь на основные потоки событий, диаграмму вариантов использования UML и диаграмму классов UML, разработанные в результате выполнения лабораторной работы 5, построить функциональную модель ПП на языке UML. Для этого построить следующие диаграммы поведения UML (behavior diagram):

1. Выбрать в диаграмме классов несколько классов-сущностей (не более трёх), которые характеризуются наиболее частой сменой состояний, и построить для них диаграммы состояния UML (statechart diagram).
2. Для каждого из базовых вариантов использования построить диаграмму деятельности (activity diagram). Для вариантов использования, с которыми связаны несколько действующих лиц, диаграмму деятельности построить в виде дорожек с привязкой к исполнителям отдельных операций.
3. В диаграмме вариантов использования UML разрабатываемого ПП для каждого прецедента выделить список объектов, участвующих во взаимодействии в этом прецеденте, и заполнить таблицу 2.5.

Таблица 2.5. Список объектов для каждого потока событий

№ п.п.	Прецедент	Объект	Описание объекта

4. Построить диаграммы последовательности (sequence diagram) для перечисленных прецедентов. Для простого ПП (если в табл. 2.5 находится не более 5 объектов) может быть построена одна диаграмма последовательности для всех объектов из табл. 2.5.
5. Для вариантов использования, характеризующихся сложным поведением, построить кооперативные диаграммы (collaboration diagram).
6. Убедиться, что все построенные диаграммы поведения покрывают операции всех классов диаграммы классов UML.

Требования к содержанию раздела

В подразделе 1 определяются классы, которые характеризуются нетривиальным поведением (наиболее частой сменой состояний). Как правило, это классы-сущности, отражающие центральные объекты предметной области (для системы управления продажами – это товар, для системы бронирования

билетов – билет, для аналитической системы – предмет анализа, и т.п.), с которой выполняются действия по ходу автоматизируемого бизнес-процесса. Для данных классов строятся диаграммы состояний UML и выполняется их описания по тексту подраздела. Диаграммы состояний должны иметь начальное и конечное состояние (конечных состояний может быть несколько).

В подразделе 2 отражается построение одной или нескольких диаграмм деятельности, которые должны покрывать все функциональные требования и, соответственно, все варианты использования, представленные на диаграмме прецедентов UML, построенной при выполнении лабораторной работы 5. Все диаграммы должны быть описаны по тексту подраздела.

В подразделе 3 отражается построение диаграмм взаимодействия UML (interaction diagrams). При этом необходимо построенными диаграммами покрыть все классы и особенности их взаимодействия с другими классами и действующими лицами, которые представлены на базовых диаграммах UML, построенных при выполнении лабораторной работы 5. Все диаграммы должны быть описаны по тексту подраздела.

Следует внимательно проследить за тем, чтобы все построенные в результате выполнения данной лабораторной работы диаграммы поведения UML покрывали все операции диаграммы классов UML.

Рекомендуемая литература

[1] – разделы 1.5, 1.6.
[2] – подразделы 1.2.1, 1.2.2, 1.3.6, глава 2.
[3] – подразделы 5.1.1, 5.1.3.

Контрольные вопросы

1. Каково назначение диаграмм поведения UML?
2. Перечислите диаграммы поведения UML, опишите их особенности.
3. Как взаимосвязаны диаграммы поведения UML и базовые диаграммы UML?
4. Каков порядок построения диаграмм UML?
5. Откуда берутся исходные данные для построения диаграммы состояния UML?
6. Откуда берутся исходные данные для построения диаграммы деятельности UML?
7. Откуда берутся исходные данные для построения диаграмм взаимодействия UML?
8. Какие диаграммы UML входят в модель программного продукта на языке UML?
9. Сколько и каких диаграмм поведения должно быть построено, если диаграмма прецедентов состоит из трех базовых прецедентов?

Лабораторная работа 7

Тема: разработка архитектуры программного продукта.

Цель: приобретение навыков проектирования физической архитектуры ПП на языке UML, а также разработки базы данных для ПП.

Задание

1. Опираясь на специальные требования к информационному обеспечению, представленные в техническом задании, а также на диаграмму классов UML, разработать решения по организации структур данных в ПП.
2. Выполнить описание структур данных на уровне атомарных атрибутов с указанием типов данных, ограничений на значения и значений по умолчанию. Каждую структуру данных представить в формате таблицы 2.6.

Таблица 2.6. Структура данных

Атрибут	Тип данных	Размер, байт	Условие на значение*	Значение по умолчанию*	Примечание*

* Указанные столбцы не являются обязательными для заполнения

3. Если в качестве структуры данных разрабатывается база данных (БД), представить её нормализованную модель данных, список разработанных таблиц БД, а также описание связей между таблицами в формате табл. 2.7 и 2.8.

Таблица 2.7. Список разработанных таблиц БД

№ п/п	Имя таблицы	Описание

Таблица 2.8. Связи между таблицами БД

Родительская таблица		Дочерняя таблица		Тип связи*
Название	Атрибут PK	Название	Атрибут FK	

* в качестве типа связи указывать «1:1», «1:M», «M:M»

N.B. Если в ПП не используется БД, то п.3 задания следует пропустить.

4. Опираясь на специальные требования к математическому обеспечению, представленные в техническом задании, а также на диаграммы поведения UML, разработанные при выполнении лабораторной работы 6, перечислить и описать программные компоненты ПП в формате табл. 2.9. Стереотипы компонентов указать согласно табл. 2.10. Имена компонентов-файлов привести с указанием расширений.

Таблица 2.9. Перечень программных компонентов ПП

№	Имя	Стереотип	Описание

Таблица 2.10. Типы компонентов приложения

Стереотип	Тип компонента
«source»	файл исходного текста программы
«form»	файл интерфейсной формы или web-страницы
«executable»	исполняемый файл
«library»	статическая или динамическая библиотека
«document»	файл документации
«data»	файл данных
«table»	таблица базы данных
«view»	представление как объект базы данных
«file»	любой другой файл, кроме указанных выше в это таблице

5. Построить диаграмму компонентов UML, отражающую взаимодействие его программных компонентов между собой и с компонентами информационного обеспечения ПП.

6. Выполнить описание физических элементов комплекса технических средств, необходимых для функционирования ПП, в формате табл. 2.11. Стереотипы элементов указать согласно табл. 2.12.

Таблица 2.11. Перечень узлов комплекса технических средств

№	Имя	Стереотип	Описание

Таблица 2.12. Типы элементов диаграмм развертывания UML

Стереотип	Тип элемента
«processor»	Ресурсоемкий узел
«device»	Нересурсоемкий узел (устройство)
«sensor»	Измерительное устройство (датчик)
«printer»	Печатающее устройство (принтер)
«net»	Сетевая передающая среда
«database»	База данных
«node»	Узел, не попадающий под тип, указанный выше в этой таблице

7. Построить диаграмму развертывания UML, выражающую зависимости между узлами комплекса технических средств и развернутыми на них программными компонентами из табл. 2.9.

Требования к содержанию отчета

В подразделе 1 приводятся подробные описания структур данных, которые обрабатываются, формируются и хранятся ПП. При этом описания структур должны соответствовать требованиям к ним, представленным в ТЗ.

Как правило, структуры данных присутствуют среди классов-сущностей диаграммы классов UML.

Для каждой структуры указываются все атрибуты, их типы данных, ограничения и значения по умолчанию в формате табл. 2.6. В случае разработки реляционной базы данных необходимо также представить схему данных в виде связанных таблиц-отношений, а также таблицу с описанием связей. Схема БД должна находиться не менее чем в 3-й нормальной форме (3НФ).

В подразделе 2 представляется таблица программных компонентов ПП, строится диаграмма компонентов и выполняется её краткое описание. В описании должны раскрываться существенные детали реализации и взаимодействия компонентов (например, технологии реализации или обмен данными и др.). При этом следует избегать использования стандартного стереотипа «file», а максимально дифференцировать компоненты по типам согласно табл. 2.10.

В подразделе 3 представляется диаграмма развертывания и выполняется её краткое описание. В описании должны раскрываться существенные детали развертывания и взаимодействия узлов (например, важные с точки зрения понимания схемы технические характеристики, обмен данными между узлами, отношения узлов с компонентами и др.). При этом следует избегать использования стандартного стереотипа «node», а максимально дифференцировать компоненты по типам согласно табл. 2.12. Также следует различать на диаграмме узлы (представляются параллелепипедами) и реализованные на их базе компоненты (двумерные фигуры).

В целом в материалах должна быть ясно представлена архитектура ПП как её компонентный состав и структура взаимодействия этих компонентов между собой и с внешней средой, в том числе с пользователями и другими программными системами.

Рекомендуемая литература

[1] – разделы 2.1, 2.2, 2.9.
[2] – подраздел 1.3.6, раздел 1.2, глава 2.
[3] – подраздел 5.1.1, разделы 5.2, 5.4, 5.5, главы 6, 8.

Контрольные вопросы

1. Что такое архитектура программного продукта?
2. Что такое информационное и программное обеспечение?
3. В какой форме может быть структурирована информация, подлежащая переработке посредством ПП? Какова структурная организация инфопотоков в вашей ПП?
4. Каковы отличия между компонентами информационного и программного обеспечения? Какова между ними связь?
5. Что такое специальное ПО?

6. Какие программные компоненты относятся к специальному ПО?
7. Какие бывают типы компонентов специального ПО?
8. Каково назначение диаграммы компонентов UML?
9. Какие бывают типы физических элементов ПС?
10. Каково назначение диаграммы развертывания UML?

Лабораторная работа 8

Тема: разработка специального программного обеспечения.

Цель: приобретение навыков разработки и отладки программного обеспечения ПП при помощи современных технологий программирования.

Теоретические положения

Разработка специального ПО

К программному обеспечению, подлежащему разработке в процессе рабочего проектирования относится категория специального ПО, представляющего собой программную реализацию моделей и алгоритмов, заложенных в реализацию операций классов и программных компонентов ПП.

При разработке социальных и экономических ИС специальное ПО чаще всего включает в себя интерфейсные средства: формы, и отчеты, а также разрабатываемые для них запросы, представления, макросы и модули программного кода.

В общем случае материал рабочего проекта в части специального ПО предполагает разработку и документирование всех программных компонентов ПП, запроектированных при выполнении лабораторной работы 7 вплоть до отдельных методов, записанных в секциях операций классов на диаграмме классов UML, построенной при выполнении лабораторной работы 5 и доработанной при выполнении лабораторных работ 6–7.

Управление качеством специального ПО

При разработке ПО на практике используются такие распространенные способы контроля его качества:

1. Наладка качественного программного процесса. Это залог успеха проекта. Для комплексного улучшения процессов в компании компаниями-разработчиками ПО используются стандарты CMM/CMMI, а также стандарты серии ISO 9000 (с последующей официальной сертификацией).

2. Формальные (математизированные) методы и подходы – использование математических формализмов для доказательства корректности, спецификации, проверки формального соответствия, автоматической генерации и т.д. К таковым относятся, например, методика верификации Model Cheking для проверки определенных свойств программы на её моделях, либо модельно-ориентированное тестирование (model-based testing) для автоматической генерации тестов и тестового окружения по формальным спецификациям требований к системе. Имеют высокий порог вхождения. Эффективны в случае повышенных требований к качеству продукта.

3. Профилирование – исследование и анализ динамических свойств ПО: быстродействия (временна́я сложность), использования памяти и других ресурсов (пространственная сложность) путем запуска и непосредственных наблюдений в виде графиков, отчетов и пр. Основная цель – выявление узких

мест в производительности программы, идентификация областей в исходном тексте, где программа может быть оптимизирована для повышения ее эффективности и производительности. Также профилирование помогает обнаружить скрытые ошибки или проблемы, которые можно не заметить при обычном тестировании.

Профилирование может быть статическим и динамическим. *Статическое* включает анализ кода программы без ее выполнения, как правило, для понимания ее сложности. *Динамическое* профилирование отслеживает программу во время ее выполнения, чтобы собрать статистику за время выполнения.

Существует множество инструментов профилирования (профилировщиков). Любой современный язык и среда программирования предлагают свои решения. Например, практически любая IDE при запуске выдает время выполнения программ, современные IDE Visual Studio и IntelliJ IDEA включают встроенные профилировщики, ресурс[1] содержит технологию профилирования приложений под JVM, а в статье[2] рассмотрены инструменты профилирования ПО на Python.

Последующая работа над качеством ПО требует правильной интерпретации полученных данных. Это позволяет определить приоритетные области исходного текста, которые нуждаются в оптимизации. Среди методов оптимизации кода главным образом выделяют рефакторинг и тестирование. Тестирование подробно рассматривается в лабораторной работе 9.

4. Рефакторинг – регулярная деятельность по переписыванию кода с целью улучшения его структуры без изменения его внешнего поведения. Это может включать использование более эффективного алгоритма, сокращение использования памяти, минимизацию дискового или сетевого ввода-вывода или использование преимуществ параллелизма. Важные структурные характеристики текста программы, на которые оказывает влияние рефакторинг:

– жесткость – характеристика программы, выражающая затруднение внесения изменений в код;

– хрупкость – свойство программы повреждаться во многих местах при внесении единственного изменения;

– косность – программа содержит части, которые могут быть полезны в других системах, но усилия и риски их отделения от системы слишком велики;

– ненужная сложность – характеризуется тем, что программа содержит элементы, которые не используются в текущий момент;

– ненужные повторения – состоят в излишне повторяющихся фрагментах кода в программе;

– непрозрачность, которая характеризует трудность текста программы для понимания.

---

[1] Подробно о мониторинге и профилировании JVM. URL: https://habr.com/ru/companies/piter/articles/677212/

[2] Профилирование кода на Python: лучшие практики и инструменты. URL: https://tproger.ru/articles/profilirovanie-koda-na-python-luchwie-praktiki-i-instrumenty

5. Инспекция кода, например, техника peer code review – заключается в том, что код каждого участника проекта читается и обсуждается на специальных регулярно встречах (code review meetings). Практика показывает, что в целом код улучшается.

6. Вычитка кода – используется при разработке критических систем реального времени. Ею занимаются высококвалифицированные разработчики. При этом их роль в данном проекте – вычитка, а не разработка.

Задание

1. В соответствии с требованиями технического задания, разработанного при выполнении лабораторной работы 3, провести обоснованный выбор средства разработки специального ПО.

2. В соответствии с требованиями технического задания, разработанного при выполнении лабораторной работы 3, а также проектными решениями, разработанными при выполнении лабораторных работ 4–7, разработать специальное программное обеспечение ПП.

3. Осуществить пробный запуск приложения и убедиться в соответствии результатов требованиям, установленным в техническом задании. При обнаружении логических ошибок задокументировать их и устранить.

4. Представить экранные формы компонентов приложения в режиме запуска, а также отчетов, отражающих выходные данные.

5. Выполнить описание разработанных компонентов интерфейса приложения и отчетов в виде таблицы 2.13.

Таблица 2.13. Перечень разработанных компонентов приложения

№ пп	Имя компонента	Описание	Рисунок
			*

* даль ссылку на номер рисунка экранной формы компонента

6. Проанализировать исходные тексты программных компонентов приложения по критерию сложности. В качестве критериев использовать:

– число модулей (классов) приложения;

– суммарное число переменных подпрограмм (методов классов), включая их формальные параметры;

– суммарное количество операторов подпрограмм (методов классов);

– суммарное число строк текста программы;

– суммарное число запросов к базе данных (при наличии);

– глубину вложенности структурных операторов ветвления, повторения, рекурсий;

– глубину наследования классов.

7. Выполнить профилирование и рефакторинг исходных текстов программных компонентов. Оценить сложность модифицированных текстов программных компонентов.

Требования к содержанию отчета

В подразделе 1 проводится анализ и обоснованный выбор инструментальных средств для разработки специального ПО. Для анализа выбирается не менее 2-х современных систем (языков) программирования, которые пригодны для разработки программных компонентов ПС. Для каждого средства указываются основные возможности и характерные особенности без подробного описания языка программирования, интегрированной среды и т.п. В результате анализа делается вывод, на основании чего выбирается средство разработки специального ПО.

В подразделе 2 кратко документируется разработанное специальное ПО. Выполняется описание разработанных модулей и компонентов, в том числе:

– описание логических ошибок и процедур их устранения;

– описание компонентов в виде табл. 2.13 с приведением экранных копий;

– исходные тексты программ (листинг может быть представлен в приложении либо дана ссылка на репозиторий);

– тексты запросов к БД (при наличии), внедренные в компоненты, не являющиеся фалами исходного текста.

При этом следует убедиться, что разработаны все программные компоненты, запроектированные при выполнении лабораторной работы 7.

В подразделе 3 приводятся результаты анализа сложности исходных текстов специальных программ ПП и выполняется их качественный анализ. С учетом данных результатов выполняются процедуры профилирования и рефакторинга исходных текстов программных компонентов. Производится сравнительный анализ временно́й и пространственной сложности текстов программ, результаты которого представляются в виде графиков и/или таблиц и описываются по тексту.

Рекомендуемая литература

[1] – глава 4, раздел 5.14.
[2] – разделы 1.2, 1.4
[3] – глава 6, разделы 4.4, 8.4, подраздел 7.2.6.

Контрольные вопросы

1. Что такое программное обеспечение ПС?
2. Каковы отличия между сервисными программами и утилитами?
3. Какие типы программных средств относятся к специальному ПО.
4. Как происходит разработка программного кода ПС?
5. Что такое компиляция и построение программы?
6. Как в интегрированной среде программирования осуществляется обнаружение синтаксических ошибок?

7. Определить сложность заданного фрагмента программного кода.
8. Предложить способы оценивания производительности и эффективности программного обеспечения на базе данного кода.
9. Перечислить основные программные компоненты ПП и дать их краткие определения.
10. Какие известны методы анализ качества специального ПО?
11. Перечислить основные метрики качества исходного текста программы.
12. Что такое профилирование ПО? Как виды профилирования бывают?
13. Что такое рефакторинг ПО? Как он связан с профилированием ПО?
14. Как профилирование и рефакторинг влияют на качество ПО?
15. Что такое производительность программного обеспечения?
16. Перечислите показатели, по которым выполняется анализ кода.
17. Что понимается под эффективностью программного обеспечения?
18. Что понимается под сложностью программного обеспечения?
19. Каковы могут быть формы представления результатов оценивания эффективности программного обеспечения?
20. От чего зависит эффективность программного обеспечения?
21. Определить сложность заданного фрагмента программного кода. Предложить способы оценивания производительности и эффективности программного обеспечения на базе данного кода.

Лабораторная работа 9

Тема: комплексное тестирование программного продукта.

Цель: освоение методики тестирования разработанного ПП и проверки характеристик качества ПП.

Теоретические положения

Тестирование, как способ контроля качества ПО

Тестирование – это проверка соответствия между реальным и ожидаемым поведением программы в специально заданных условиях. Исходной информацией для тестирования является сведения о том, как программная система должна себя вести, то есть требования к ней или к ее отдельной части. Наиболее распространенным способом тестирования программ является метод *черного ящика*. При этом реализация системы недоступна тестировщикам, и тестируется только ее интерфейс посредством пользовательского воздействия. Противоположностью данному методу является тестирование методом *белого ящика*, когда исходный код доступен тестировщикам и используется в качестве источника информации о тестируемой системе. В этом случае на основе требований к системе помимо ее реализации создается (программируется) ее тестовая модель.

По способу реализации тесты могут быть ручными и автоматическими.

Ручной тест – это последовательность действий тестировщика (или разработчика), которую он может воспроизвести и при этом ошибка произойдет. Как правило, в средствах контроля ошибок такие последовательности действий содержатся в некоторой форме описания ошибки.

Автоматический тест – это некоторая программа, которая воздействует на систему и проверяет то или иное ее свойство (функциональное требование).

Критерии тестирования

В любом случае качество тестирования может оказаться неудовлетворительным – ошибки находятся редко или вообще не находятся ввиду того, что количество возможных состояний программы очень велико, и тестирование не в состоянии покрыть их все.

На практике в реальных проектах, определяют критерии тестирования, которые определяют тот уровень качества, который необходимо обеспечить в данном проекте. Хорошее качество стоит дорого и очевидно, что разное ПО имеет разное качество, например, система управления химическим реактором и текстовый редактор. В силу ограниченности ресурсов на тестирование также целесообразно определить те аспекты ПО, которые наиболее важны как для общей работоспособности системы, так и для удовлетворения заказчика.

При тестировании на некоторых (но не на всех возможных) входных данных применяют принцип факторизации – множество всех возможных

входных значений разбивается на значимые с точки зрения тестирования классы и "прогоняются" тесты не на всех возможных входных значениях, но берут по одному набору значений из каждого класса. Например, тестируют некоторую функцию системы на ее граничные значения – очень большие значения параметров, очень маленькие и т.п. Часто факторизацию удобно делать, исходя из требований к данной функции. Также бывает полезно посмотреть на ее реализацию и протестировать ее по разным логическим веткам (порождаемым, например, условными операторами).

Задание

Рассматривая разработанный при выполнении лабораторной работы 8 программный продукт как объект тестирования, выполнить тестирование в соответствии с заданием из таблицы 2.14.

Таблица 2.14. Задания на тестирование ПП

	Автоматическое тестирование	**Ручное тестирование**
Метод белого ящика	1. На основе функциональных требований, сформулированных при выполнении лабораторной работы 3, разработать несколько модульных тестов. Выполнить тесты. По проваленным тестам составить отчеты о несоответствии	3. На основе перечислений *альтернативных* потоков событий, разработанных при выполнении лабораторной работы 5, выполнить стрессовое тестирование программного продукта. В случае обнаружения некорректных ситуаций описать их, выявить причины и принять меры к их устранению
Метод черного ящика	2. Выполнить нагрузочное тестирование ПП и оценить эффективность разработанных при выполнении лабораторной работы 8 запросов к БД либо алгоритмов обработки данных исходных датасетов. Предварительно подготовить 4 набора данных на 10, 100, 1000 и 10000 записей. Построить графики зависимости времени вычислений от объема исходных данных	4. На основе перечислений *основных* потоков событий, разработанных при выполнении лабораторной работы 5, выполнить полную ручную проверку работоспособности ПП посредством воздействия на ее интерфейсную часть

При возникновении несоответствий задокументировать их, идентифицировать логические ошибки, при необходимости разработать дополнительные модульные тесты для их обнаружения. Устранить ошибки и провести рефакторинг.

Требования к содержанию отчета.

В подразделе 1 приводятся результаты тестирования методом черного ящика. Приводятся описания и тексты программ модульных тестов и результаты их выполнения. Для ручных тестов приводятся алгоритмы проверки работоспособности ПП по всем возможным ветвям. Для этого требуется инициировать срабатывание всех пунктов меню в определенном порядке.

Представить обнаруженные логические ошибки (при наличии таковых). Для каждой логической ошибки:

- описать ее суть;
- представить тестовую последовательность манипуляций для ее обнаружения (как правило, это подмножество пунктов полной тестовой последовательности);
- представить форму ее проявления в результате применения теста;
- определить форму ее выражения в коде;
- описать способ ее устранения.

В подразделе 2 приводятся результаты тестирования методом белого ящика. Необходимо представить технические характеристики машины, используемой для тестирования. Привести определенную последовательность действий пользователя (наподобие последовательности проверочных действий в подразделе 1). Также представить характеристики подготовленных наборов данных для нагрузочного тестирования. Указать источник данных и данные об объеме: число строк, слов, символов. При подготовке больших наборов данных допускается использование средств генеративного ИИ.

Привести временны́е и пространственные оценки проверок для каждой конфигурации и кратко их описать. Оценки нагрузочных проверок могут быть представлены в виде сводных таблиц и графиков на каждом наборе данных для каждого тестируемого метода. Описать результаты выполненного анализа. Результаты стрессового тестирования описать со ссылками на шифры альтернативных потоков событий, содержащих обработку некорректных ситуаций, причинами и способами их устранения.

Рекомендуемая литература

[1] – разделы 1.2, 1.3, 3.4, 3.5, главы 2, 5, 6.
[2] – подраздел 1.1.3–1.1.5, 1.2.3. разделы 1.4, 2.2.
[3] – глава 7.

Контрольные вопросы

1. Перечислите и охарактеризуйте методы обеспечения качества программного обеспечения.
2. Что такое тестирование компьютерной программы?
3. Что такое ожидаемое поведение программы?
4. Какие виды тестов и тестирования программного обеспечения применимы к разработанному программному приложению?
5. Укажите методы и способы оценивания производительности программного обеспечения.
6. В чем заключается идентификация ошибок в программе?
7. В чем разница между тестированием методами белого и черного ящика?
8. В чем разница между ручным и автоматическим тестами?
9. Какими способами можно обнаружить логически ошибки в коде?
10. В чем заключается идентификация ошибок в программе?

Список рекомендуемой литературы

Основная литература:

1. Полетайкин, А.Н. Методология и технология разработки программных систем: методы и модели программной инженерии: учебное пособие / А.Н. Полетайкин, Н.Ю. Добровольская; Министерство науки и высшего образования Российской Федерации, Кубанский государственный университет. − Краснодар: Кубанский гос. ун-т, 2025. − 229 с. Режим доступа: http://212.192.134.46/MegaPro/Download/MObject/1389.

2. Полетайкин, А.Н. Технологии разработки программных систем: учебное пособие / А.Н. Полетайкин, Н.Ю. Добровольская; Министерство науки и высшего образования Российской Федерации, Кубанский государственный университет. − Краснодар: Кубанский гос. ун-т, 2025. − 190 с.

3. Полетайкин А. Н. Социальные и экономические информационные системы: законы функционирования и принципы построения : Учеб. пособие / Сиб. гос. ун-т телекоммуникаций и информатики. - Новосибирск : СибГУТИ, 2016. - 241 с. Режим доступа: https://sibsutis.ru/upload/13e/up_is.pdf.

4. Ехлаков, Ю. П. Введение в программную инженерию: Учебное пособие [Электронный ресурс] / Ю. П. Ехлаков. — Томск: ТУСУР, 2011. — 148 с. — Режим доступа: https://edu.tusur.ru/publications/141.

5. Гагарина Л.Г., Кокорева Е.В., Виснадул Б.Д. Программная инженерия : учеб. пособие; под ред. Л.Г. Гагариной. – М.: ИД "ФОРУМ" : ИНФРА-М, 2012. – 399 с.

6. Батоврин В.К. Системная и программная инженерия. Словарь-справочник: учеб. пособие для вузов. - М.: ДМК Пресс, 2010. - 280 с.

7. Мартин Р. Чистый код. Создание, анализ и рефакторинг : монография. – СПб.: ПИТЕР, 2012. – 464 с.

8. Орлов С.А. Технологии разработки программного обеспечения : Учебник для вузов. – СПб. : ПИТЕР, 2002. – 463 с.

9. Кнут Д.Э. Искусство программирования: В 3 т. : Пер. с англ. Т.3:Сортировка и поиск : монография / Под ред. Ю.В. Козаченко. – 2-е изд. – М. : Издат.дом "Вильямс", 2003. – 822 с.

10. Иванова Г.С. Технология программирования : учебник – 3-е изд., перераб. и доп. – М. : Изд-во МГТУ им. Н. Э. Баумана, 2006. – 335 с.

11. Денис М. Ахен, Арон Клауз, Ричард Тернер. CMMI: Комплексный подход к совершенствованию процессов. Практическое введение в модель / Пер. с англ. – М.: «МФК», 2005. – 330 с.

12. Липаев В.В. Программная инженерия. Методологические основы: учебник. – М.: «ТЕИС», 2006 – 608 с.

13. Никитин В.А. Управление качеством на базе стандартов ИСО 9000:2000 : монография. – СПб. : ПИТЕР, 2002. – 262 с.

14. Босуэлл Д., Фаучер Т. Читаемый код или Программирование как искусство : монография. – СПб. : ПИТЕР, 2012. – 203 с.

Дополнительная литература:

15. Лавров С.С. Программирование. Математические основы, средства, теория : монография. – СПб. : БХВ-Петербург, 2001. – 317 с.

16. Вернер М. Основы кодирования : учебник / пер. с нем. Д.К. Зигангирова. – М. : Техносфера, 2004. – 286 с.

17. Макконелл Дж. Основы современных алгоритмов : учеб. пособия / пер. с англ. под ред. С.К. Ландо; доп. М.В. Ульянова. – М. : Техносфера, 2004. – 366 с.

18. Губарев А.В. Информационное обеспечение системы менеджмента качества : моногр. – Москва : Горячая линия-Телеком, 2013. – 132 с. : ил.

19. Лодон Дж. Управление информационными системами : учебник. – 7-е изд. – СПб. : ПИТЕР, 2005. – 910 с.

20. Кронрод А.С. Беседы о программировании : научно-популярная литература / предисл. Л.А. Кронрод; послеслов. В.Л. Арлазарова. – 2-е изд., стереотип. – М. : УРСС, 2004. – 246 с.

21. Кузнецов С.М., Малозёмов Б.В. Программирование и основы алгоритмизации : учеб. пособие. – Новосиб. гос. техн. ун-т. – Новосибирск : Изд-во НГТУ, 2006. – 199 с. : ил.

22. Кнут Д.Э. Искусство программирования: В 3 т. : Пер. с англ. Т.1:Основные алгоритмы : монография / Под общ. ред. Ю.В.Козаченко. – М. : Издат.дом "Вильямс", 2002. – 712 с.

23. Кнут Д.Э. Искусство программирования: В 3 т. : Пер. с англ. Т.2:Получисленные алгоритмы : монография / Под общ. ред. Ю.В.Козаченко. – 3-е изд. – М. : Издат.дом "Вильямс", 2003. – 828 с.

ГОСТы:

24. ГОСТ Р ИСО/МЭК 12207–99. ГОСУДАРСТВЕННЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ. Информационная технология. ПРОЦЕССЫ ЖИЗНЕННОГО ЦИКЛА ПРОГРАММНЫХ СРЕДСТВ.

25. ГОСТ Р ИСО/МЭК 15504-1-2009. Информационные технологии. Оценка процессов. Часть 1. Концепция и словарь Текст. Введ. 14.09.2009. – М. : Стандартинформ, 2010. – 19 с.

26. ГОСТ 34.601–90. Автоматизированные системы. Стадии создания. В кн.: Информационная технология. Комплекс стандартов и руководящих документов на автоматизированные системы. М.: Комитет стандартизации и метрологии СССР,1991. – с.45-52.

27. ГОСТ 34.602–89. Техническое задание на создание автоматизированной системы. В кн.: Информационная технология. Комплекс стандартов и руководящих документов на АС. М.: Комитет стандартизации и метрологии СССР,1991. – с.15-29

28. РД 50–34.698–90. Автоматизированные системы. Требования к содержанию документов. В кн.: Информационная технология. Комплекс стандартов и руководящих документов на АС. М.: Комитет стандартизации и метрологии СССР,1991. – с.66-104.

29. ГОСТ 19.701–90 (ИСО 5807-85). Схемы алгоритмов, программ, данных и систем. Госстандарт СССР, 1990. – 20 с.

Приложение А – Варианты индивидуальных заданий

1. Обменный пункт: сотрудники пункта, виды валют, курсы валют, операции обмена.
2. Ювелирный магазин: названия изделий, комитенты (кто сдал изделия на комиссию), журнал сдачи изделий на продажу, журнал покупки изделий.
3. Поликлиника: врачи, пациенты, виды болезней, журнал учета прихода пациентов.
4. Кондитерский магазин: виды конфет, поставщики, торговые точки, журнал поступления и отпуска товара.
5. Автобаза: автомашины, водители, рейсы, журнал выезда машин на рейсы.
6. Парикмахерская: клиенты, прайс услуг, сотрудники, кассовый журнал.
7. Школа: учителя, предметы, ученики, журнал успеваемости.
8. Оплата услуг на дачных участках: виды услуг, список владельцев, сотрудники управления, журнал регистрации оплат.
9. Гостиница: проживающие, сотрудники гостиницы, номера, журнал регистрации проживающих.
10. Книжный магазин: авторы, книги, продавцы, покупатели, регистрация продаж.
11. Ремонтная мастерская: виды работ, исполнители, заказы на ремонт, заказчики.
12. Аптечный киоск: номенклатура лекарств, работники аптеки, покупатели, журнал регистрации продаж.
13. Выставка: стенды, стендисты, экскурсии, посетители.
14. Охранная служба: список постов охраны, список охранников, журнал выхода на дежурство, журнал учета замечаний.
15. Столовая: продукты, блюда, меню, журнал заказов
16. Фото мастерская: заказчики работ, прайс работ, журнал поступления заказов, исполнители.
17. Ветеринарная лечебница: список животных, список болезней, список хозяев, журнал посещений.
18. Сельское хозяйство: список растений, список угодий, список работников, журнал посевной.
19. Холдинг: список регионов, список предприятий, список показателей, журнал учета отчетных данных.
20. Фонды предприятия: список основных средств, список категорий основных средств, список материально ответственных лиц, журнал учета состояния основных средств.
21. Учет расхода материалов в компании: список статей затрат, список сотрудников, журнал учета расхода канцтоваров, список департаментов.
22. Фильмотека: список фильмов, список клиентов, список библиотекарей, журнал выдачи фильмов.
23. Цирк: список категорий артистов, список артистов, журнал выхода артистов на работу, список цирковых площадок.

24. Спортивные заведения: список спортсменов, список видов спорта, список стадионов, журнал учета выступлений спортсменов.
25. Компьютерные занятия: список слушателей курсов, список предметов, список преподавателей, журнал учета успеваемости.
26. Сбор урожая: список видов продукции, список сборщиков, список бригад, журнал учета сбора урожая.
27. Фирма по обслуживанию населения: список заказчиков, список товаров, список разносчиков, журнал заказов.
28. Партийная работа: список членов партии, список мероприятий, журнал учета выхода на мероприятие, список городов
29. Экономическая база данных: список регионов, список показателей, список отраслей, отчетные статистические данные.
30. Журнальные статьи: список тем, список авторов, список названия статей, список журналов.
31. Анализ причин заболеваемости: список больных, список болезней, список районов, журнал учета заболевших.
32. Отдел кадров: список сотрудников, штатное расписание, список отделов, журнал перемещения сотрудников по службе.
33. Расчет нагрузки на преподавателя: список преподавателей, список кафедр, предметов, журнал нагрузки.
34. Проектные работы: список проектов, список специалистов, список должностей, журнал учета работ.
35. Учет компьютерного оборудования: список типов оборудования, список материально ответственных лиц, список департаментов, журнал регистрации выдачи оборудования.
36. Прививки детям: список прививок, список детей, список родителей, журнал учета сделанных прививок.
37. Начисление налогов в бюджет: виды налогов, список отраслей, список предприятий, журнал учета поступления налогов.
38. Экспертная система: список оцениваемых объектов, список экспертов, список регионов, журнал учета оценок.
39. Ремонтная мастерская электронного оборудования: список работ, список мастеров, список запасных частей, журнал учета выполненных работ, список поступившего оборудования.
40. Магазин по продаже автомобилей: список фирм производителей, список автомобилей, журнал поступления автомобиля, список водителя пригнавшего машину.
41. Автомобильный гараж: список владельцев, список автомобилей, список сторожей, журнал прихода и ухода автомобилей.
42. Учет криминогенной ситуации в городе: список районов, список типов преступлений, список дежурных, журнал регистрации преступлений.
43. Система здравоохранения: список регионов, список санаториев, список пенсионеров, журнал регистрации выдачи путевок в санатории.
44. Туристические агентства: список туров, список стран, список клиентов, журнал регистрации продаж туров.

45. Продажа билетов на рейсы: список рейсов, прайс билетов, список компаний, журнал продаж билетов.
46. Продажа пиломатериалов: виды пиломатериалов, регионы поставщики, список заказчиков, журнал учета продаж пиломатериалов.
47. Склад металлоконструкций: прайс товара металлоконструкций, список поставщиков, список продавцов, журнал учета продаж.
48. Система поддержки решений: список экспертов, список тем обсуждений, список департаментов, журнал учета предложений.
49. Детский сад: список родителей, список детей, список групп, журнал посещения детского сада.
50. Дом творчества молодежи: список кружков, список руководителей, список детей, журнал регистрации посещения кружков.

Приложение Б – Перечень показателей качества ПС

1. Целостность – способность системы противостоять (сохранять информационное содержание и однозначность интерпретации смысла) изменениям, искажениям или порче при возникновении сбоев или ошибок.
2. Рациональность – подразумевает, что при функционировании оптимальным образом используются имеющиеся в распоряжении системы ресурсы: время, оборудование (память), люди.
3. Численность задействованного персонала – количество лиц, задействованных в эксплуатации и обслуживании системы.
4. Работоспособность – способность системы выполнять свои функции с эксплуатационными показателями не ниже заданных.
5. Производительность – характеристика системы, отражающая ее способность производить определенный объем работ в единицу времени, например, пропускная способность, время ответа, доступность, число продуктов, полученная прибыль, быстродействие.
6. Гибкость – Возможность модификации обеспечивающей части системы, обычно возникает по двум причинам: чтобы отразить в системе изменение требований или чтобы исправить ошибки, внесённые ранее в процессе разработки. Гибкость заключается в возможности адаптации, наращивания, изменения средств.
7. Открытость – прозрачность функциональной части системы и возможность ее модификации без нарушения процесса функционирования.
8. Защищенность (Безопасность) – способность обеспечения защиты данных от разрушения, искажения или преднамеренных фальсификаций злоумышленником. Характеризует возможное отсутствие риска, связанного с нанесением некоторого ущерба
9. Потенциальная управляемость – возможность компенсировать возмущение быстрее, чем успеют измениться эти возмущения.
10. Наблюдаемость основных параметров управляемого процесса – характеризует степень и глубину отслеживания параметров функционирования системы и основных характеристик объекта управления на всех циклах и этапах работы системы/объекта.
11. Надёжность – комплексное свойство системы сохранять во времени в установленных пределах значения всех параметров, характеризующих способность системы выполнять требуемые функции в заданных режимах и эксплуатации. Надежность ПО включает такие элементы как:
    - отказоустойчивость – возможность восстановления системы и данных в случае сбоев в работе;
    - безопасность – сбои в работе системы не должны приводить к опасным последствиям (авариям);
    - защищенность от случайных или преднамеренных внешних воздействий (защита от «дурака», вирусов, спама).
12. Степень оперативности и надежности управления – характеризует способность системы и субъекта управления выполнять поставленные задачи в сроки, обеспечивающие эффективное управление объектом. Управление является

оперативным, если время, затрачиваемое на решение системой задач, в совокупности со временем, необходимым персоналу на подготовку к реализации управленческих решений, не будет превышать критического времени, диктуемого условиями сложившейся ситуации на объекте. О.у. достигается ускорением процесса сбора и обработки данных о состоянии объекта, принятия решений, планирования прогнозирования социально-экономических показателей.

13. Безотказность – свойство системы сохранять работоспособность в течение требуемого интервала времени непрерывно без вынужденных перерывов. Безотказность является наиболее важной компонентой надёжности, так как она отражает способность длительное время функционировать без отказов. Один из показателей надежности системы.
14. Живучесть – свойство системы сохранять работоспособность в условиях возмущающих воздействий внешней среды и отказов компонентов системы с минимальной частотой отказов, а в случае их возникновения эффективно восстанавливать утраченные функции и ресурсы.
15. Прогрессивность компьютерных и информационных технологий, задействованных в системе
16. Наглядность интерфейса – должен быть удобным, интуитивно понятным и продуманным с точки зрения инженерной психологии, эргономики и методов технической эстетики
17. Долговечность — свойство системы сохранять работоспособность до наступления предельного состояния. Зависит от долговечности технических средств и подверженности системы моральному старению.
18. Сопровождаемость – означает, что программа должна быть разработана с расчетом на дальнейшее развитие. Это критическое свойство системы, т.к. изменения системы неизбежны вследствие изменения бизнеса. Изменение может включать исправления, усовершенствования или адаптацию компонентов системы к изменениям в окружающей обстановке, требованиях и условиях функционирования, включая ресурсное обеспечение.
19. Эффективность – получение от функционирования системы существенного технико-экономического, социального или другого эффекта. Универсальным видом является техническая эффективность, которая оценивается следующими показателями:
    - время выполнения операций;
    - загруженность процессора;
    - объем требуемой памяти;
    - время отклика и др.
20. Мобильность – возможность перенесения системы из одного окружения в другое. Окружающая обстановка может включать организационное, техническое или программное окружение, а также условия эксплуатации.

Приложение В – Образец титульного листа
(масштабируется к формату А4)

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Кафедра информационных технологий

ОТЧЕТ

о выполнении лабораторной работы №__
по дисциплине «Программная инженерия»

Выполнил: ст. гр. ___
<Фамилия И.О.>
Проверил: <должность>
<Фамилия И.О.>

Краснодар
20__