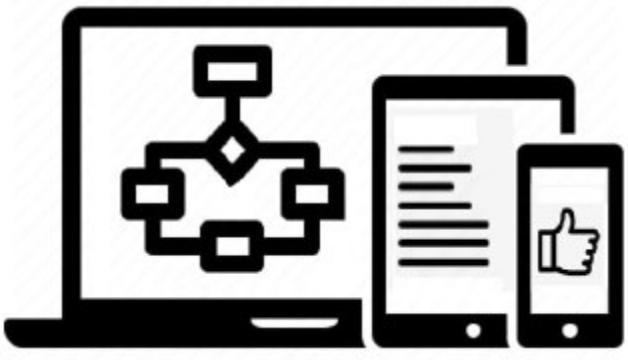
МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Программное обеспечение информационных технологий»

ТЕХНОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Методические рекомендации к курсовому проектированию для студентов направлений подготовки 09.03.01 «Информатика и вычислительная техника» и 09.03.04 «Программная инженерия» очной формы обучения



Могилев 2022

УДК 004.4
ББК 32.973.26-018.2
Т38

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой «Программное обеспечение информационных технологий» «16» сентября 2022 г., протокол № 2

Составители: канд. техн. наук, доц. К. В. Овсянников;
канд. техн. наук, доц. К. В. Захарченков;
канд. техн. наук, доц. Т. В. Мрочек

Рецензент канд. техн. наук, доц. В. М. Ковальчук

Методические рекомендации содержат указания, варианты индивидуальных заданий и требования к выполнению курсовой работы по дисциплине «Технологии разработки программного обеспечения» для студентов направлений подготовки 09.03.01 «Информатика и вычислительная техника» и 09.03.04 «Программная инженерия» очной формы обучения.

Учебно-методическое издание

ТЕХНОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Ответственный за выпуск	В. В. Кутузов
Корректор	И. В. Голубцова
Компьютерная верстка	Н. П. Полевничая

Подписано в печать . Формат 60×84/16. Бумага офсетная. Гарнитура Таймс.
Печать трафаретная. Усл. печ. л. . Уч.-изд. л. . Тираж 21 экз. Заказ ;

Издатель и полиграфическое исполнение:
Межгосударственное образовательное учреждение высшего образования
«Белорусско-Российский университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/156 от 07.03.2019.
Пр-т Мира, 43, 212022, г. Могилев.

Содержание

Введение

Современное состояние науки и техники требует от инженерно-технических и научных работников знания средств вычислительной техники и умения обращения с современными программно-техническими комплексами. Эффективное применение компьютеров для решения инженерных и научных задач невозможно без знаний основных методов составления схем алгоритмов, написания действенного программного обеспечения на различных языках программирования, использования пакетов программ инженерной графики и математических систем.

Целью изучения дисциплины является изучение системы инженерных принципов для создания программного обеспечения с заданными характеристиками.

Цели курсовой работы:

– участие в проектировании компонентов программного продукта в объеме, достаточном для их конструирования в рамках поставленного задания;

– создание компонент программного обеспечения (кодирование, отладка, модульное и интеграционное тестирование);

– выполнение измерений и рефакторинг кода в соответствии с планом;

– участие в интеграции компонент программного продукта;

– разработка и оформление эскизной, технической и рабочей проектной документации.

При выполнении курсовой работы рекомендуется использовать литературу, приведенную в [1–10].

1 Содержание курсовой работы

1.1 Содержание пояснительной записки

В состав пояснительной записки к курсовой работе входит следующее.

Введение.

Описание предметной области.

Видение проекта.

Математические (вербальные) модели.

Проектирование программного модуля.

Перечень функциональности (User Stories).

Сценарии взаимодействия (краткий и подробный).

Прототипирование интерфейса взаимодействия с пользователем (графического интерфейса).

Составление общего плана работы.

Составление индивидуального плана работы.

Разработка документации программного модуля.

Диаграмма классов.

Диаграмма деятельности.

Разработка программного модуля.

Выбор и обоснование технологий, используемых в программном модуле.

Настройка окружения и репозитория кода.

Описание используемых структур данных.

Тестирование программного модуля.

Заключение.

Приложение 1 – Код программных модулей.

Рекомендуемый объем пояснительной записки составляет 20–25 страниц.

1.2 Содержание графической части работы

Графическая часть работы выполняется на листе формата А1, где изображается следующее:

– снимки экрана;
– диаграмма классов;
– диаграмма взаимодействия.

Допускаются изменения размера формата и числа листов графической части по согласованию с руководителем курсовой работы.

2 Требования к содержанию пояснительной записки

2.1 Описание предметной области

В данном разделе студентом производится постановка заданной задачи, указываются необходимые математические выражения для её решения и прочие условия.

2.1.1 Видение проекта.

Видение проекта (vision) – краткое описание сути будущего продукта. В этом документе кратко описываются продукт, цели и задачи его создания, пользователи и основные возможности будущей системы.

Видение проекта (vision) позволяет получить общее представление о продукте, с помощью нескольких абзацев ознакомить с сутью проекта любое заинтересованное лицо.

Видение проекта (vision) позволяет собрать бизнес-требования. Документ дает общее представление о бизнес-целях, поставленных перед продуктом.

2.1.2 Математические (*вербальные*) *модели.*

При выполнении данного подраздела производится изучение и анализ литературных и иных источников по поставленной задаче. При анализе литературы необходимо получить итоговые математические выражения, которые будут использоваться. Здесь следует указать существующие математические пакеты программ и специализированных программ, которые применяются в настоящее время для решения данной задачи, показать их сильные и слабые стороны. При поиске имеющегося программного обеспечения для решения поставленной задачи рекомендуются поисковые службы сети Интернет. В записке следует указать, какие поисковые системы были использованы, а также общее число ссылок, напрямую относящихся к данной задаче.

При работе над подразделом обязательно указываются используемые литературные источники согласно требованиям ГОСТ 7.1–2003. Ссылка на информационные ресурсы сети Интернет оформляется в виде полного URL-адреса ресурса и названия источника.

В подразделе окончательно указываются математические выражения, применяемые для решения задачи, и литературные источники, используемые при ее математическом описании. Все записанные математические формулы и выражения должны иметь нумерацию, выполняемую в круглых скобках в правой части страницы, для организации последующих ссылок на них в тексте пояснительной записки. Можно самостоятельно вывести расчетные формулы. В этом случае указывается метод вывода полученных выражений.

Недопустимо приводить в составе подраздела доказательства математических теорем и лемм. Следует при необходимости указать лишь ссылку на них.

Также в данном подразделе описываются вербальные и знаковые модели, которые будут использованы при проектировании.

Вербальная модель – информационная модель в мысленной или разговорной форме.

Знаковая модель – информационная модель, выраженная специальными

знаками, т. е. средствами любого формального языка.

К информационным моделям можно отнести вербальные (от лат. verbalize – устный) модели, полученные в результате раздумий, умозаключений. Они могут так и остаться мысленными или быть выражены словесно. Примером такой модели может стать наше поведение при переходе улицы. Человек анализирует ситуацию на дороге (что показывает светофор, как далеко находятся машины, с какой скоростью они движутся и т. п.) и вырабатывает свою модель поведения. Если ситуация смоделирована правильно, то переход будет безопасным, если нет, то может произойти авария. К таким моделям можно отнести и идею, возникшую у изобретателя, и музыкальную тему, промелькнувшую в голове композитора, и рифму, прозвучавшую пока еще в сознании поэта.

Знаковые модели – это рисунки, тексты, графики и схемы. Вербальные и знаковые модели, как правило, взаимосвязаны. Мысленный образ, родившийся в мозгу человека, может быть облечен в знаковую форму. И наоборот, знаковая модель помогает сформировать в сознании верный мысленный образ.

По форме представления можно выделить следующие виды информационных моделей:

– геометрические модели – графические формы и объемные конструкции;

– словесные модели – устные и письменные описания с использованием иллюстраций;

– математические модели – математические формулы, отображающие связь различных параметров объекта или процесса;

– структурные модели – схемы, графики, таблицы и т. п.;

– логические модели – модели, в которых представлены различные варианты выбора действий на основе умозаключений и анализа условий;

– специальные модели – ноты, химические формулы и т. п.;

– компьютерные и некомпьютерные модели.

2.2 Проектирование программного модуля

В данном разделе студентом производится составление перечня функциональности, краткого и подробного сценариев взаимодействия, а также прототипирование интерфейса взаимодействия с пользователем.

2.2.1 Перечень функциональности (*User Stories*).

Перечень функциональности (таблица 1) (user stories) – это подробный список того, что пользователь может делать в системе. Вся функциональность будущего продукта разбивается на простейшие возможности в виде «<кто> <что делает> <с чем>». Каждая из функций имеет приоритет, определяющий важность для общего успеха продукта. Кроме того, для функции описываются критерии приемки – реакция на действия пользователя, при которых она считается правильно работающей.

Перечень функциональности пересекается со сценариями взаимодействия. Разница в том, что первые помогают в точном учете требований, а вторые – в понимании того, как работают функции и система в целом. User stories гово-

рят о том, что нужно сделать, а сценарии взаимодействия – как это работает.

Назначение перечня функциональности:

– точный сбор функциональных требований. Благодаря максимальной детализации все требования к системе учитываются и не забываются;

– точная оценка сроков и стоимости проекта. Планирование и учет трудозатрат более точны, когда работа над функцией занимает часы, а не дни;

– постановка заданий разработчикам. Реализовывать функции небольшими порциями проще;

– помощь в управлении проектом. User stories – один из главных инструментов управления проектами по гибким методикам (agile).

Эти методики дают более предсказуемый и качественный результат.

Таблица 1 – Пример перечня функциональности

Роль	Действие	Тип объекта	Объект	Дополнительные возможности	Сущность
1 Основной контент					
1.1 Видеоролики					
Пользователь	Просматривает	Сводная страница	Видеоролики	–	Видеоролик
Пользователь	Просматривает	Список	Видеоролики	Постраничная навигация, фильтрация по одному из параметров, сортировка по одному из параметров	Видеоролик
Пользователь	Просматривает	Страница	Видеоролик	Добавление комментария, добавление в альбом, добавление в канал, добавление в блог, отправление жалобы на содержание, копирование кода ролика в буфер обмена, отправка ссылки другу	Видеоролик, блог, канал, группа
Пользователь	Добавляет	Новый	Видеоролик	Выбор кадра из видеоролика для превью	Видеоролик
1.2 Альбомы					
Пользователь	Просматривает	Список	Альбомы	Постраничная навигация, фильтрация по одному из параметров, сортировка по одному из параметров	Альбом
Пользователь	Просматривает	Страница	Альбом	Удаление видеоролика из альбома	Альбом
Пользователь	Добавляет	Новый	Альбом	–	Альбом
Пользователь	Редактирует	Существующий	Альбом	–	Альбом
Пользователь	Удаляет	Существующий	Альбом	–	Альбом

2.2.2 Сценарии взаимодействия (*краткий и подробный*).

Сценарии взаимодействия – это описание того, как должны работать

функции системы. Они могут рассказывать о сути и особенностях работы функций как в общем виде, так и в подробном, алгоритмическом. Первый вариант нужен для того, чтобы понять, зачем нужна и что делает функциональность; второй по шагам расписывает все возможные сценарии использования продукта – что может сделать пользователь и как должна отреагировать на его действия система.

Сценарии взаимодействия пересекаются с перечнем функциональности (user stories). Разница в том, что первые помогают в понимании того, как работают функции и система в целом, а вторые – в точном учете требований. User stories говорят о том, что нужно сделать, а сценарии взаимодействия – как это работает.

Примеры сценариев взаимодействия.

Краткий сценарий взаимодействия.

Пользователи могут быть объединены в социальную сеть. Каждый пользователь имеет страницу персонального профиля (персональную страницу), с помощью которой другие пользователи получают представление о том, что это за пользователь. Пользователь может добавить другого пользователя в друзья. Это позволяет, во-первых, поддерживать контакты, а во-вторых, более удобно давать доступ другим пользователям к своим конспектам и учебным планам. Пользователя можно найти как по имени и фамилии, так и по вузу или специальности, на которой он обучается.

Подробный сценарий взаимодействия.

Подробный сценарий взаимодействия (выдержка из основного пути сценария).

1 Пользователь нажимает на ссылку на профиль одного из пользователей.

2 Система открывает страницу, которая, помимо прочего, содержит следующие интерактивные элементы:

– название вуза пользователя (ссылка);

– добавить в друзья (ссылка);

– блок «Конспекты пользователя». Блок состоит из списка элементов. Каждый элемент, помимо прочего, содержит следующие интерактивные элементы:

а) название конспекта (ссылка);

б) остальные конспекты (ссылка);

– блок «Книжная полка пользователя». Блок состоит из списка элементов. Каждый элемент, помимо прочего, содержит следующие интерактивные элементы:

а) название книги (ссылка);

б) остальные книги (ссылка);

– блок «Учебные планы пользователя». Блок состоит из списка элементов. Каждый элемент, помимо прочего, содержит следующие интерактивные элементы:

а) название учебного плана (ссылка);

б) остальные учебные планы (ссылка);

– блок «Информация о пользователе». Блок, помимо прочего, содержит следующие интерактивные элементы:

а) имя пользователя (ссылка);

б) название вуза пользователя (ссылка);

в) специальность пользователя (ссылка);

г) добавить в друзья (ссылка);

д) блок «Материалы пользователя». Блок, помимо прочего, содержит следующие интерактивные элементы:

1) конспекты (ссылка);
2) книжная полка (ссылка);
3) учебные планы (ссылка);

– блок «вузы и специальности». Блок, помимо прочего, содержит следующие интерактивные элементы:

а) вузы (ссылка);

б) специальности (ссылка);

в) города (ссылка);

г) пользователи (ссылка, заблокирована);

– блок «Друзья пользователя». Блок, помимо прочего, содержит следующие интерактивные элементы:

а) список друзей. Список элементов. Каждый элемент, помимо прочего, содержит следующие интерактивные элементы: имя пользователя (ссылка);

б) все друзья пользователя (ссылка).

Альтернативный сценарий. Пользователь уже находится в друзьях у пользователя. Система заменяет ссылки «Добавить в друзья» ссылками «Удалить из друзей». Перезагрузка страницы не производится.

3 Пользователь нажимает ссылку «Добавить в друзья».

4 Система вносит изменения в базу данных.

5 Система добавляет выбранного пользователя в список друзей пользователя. Система выводит сообщение об успешной операции. Система отсылает пользователю уведомление о том, что он был добавлен в друзья. Перезагрузка страницы не производится.

2.2.3 Прототипирование интерфейса взаимодействия с пользователем (*графического интерфейса*).

При прототипировании интерфейса взаимодействия с пользователем необходимо составить структурные схемы страниц.

Структурные схемы страниц (wireframes) (рисунок 1) – основной результат работ по проектированию. Они в деталях показывают, какая информация и элементы управления должны выводиться на каждой странице системы, а также расставляют акценты – какие из элементов страницы более, а какие – менее важны. Wireframes также описывают поведение динамических элементов страниц – как они должны реагировать на действия пользователя. Стоит помнить, что схемы страниц – это не конечный дизайн системы.

Назначение структурных схем страниц:

– постановка задания дизайнеру;

– постановка задания разработчикам. Команда разработки должна руководствоваться wireframes при создании функционального прототипа системы.

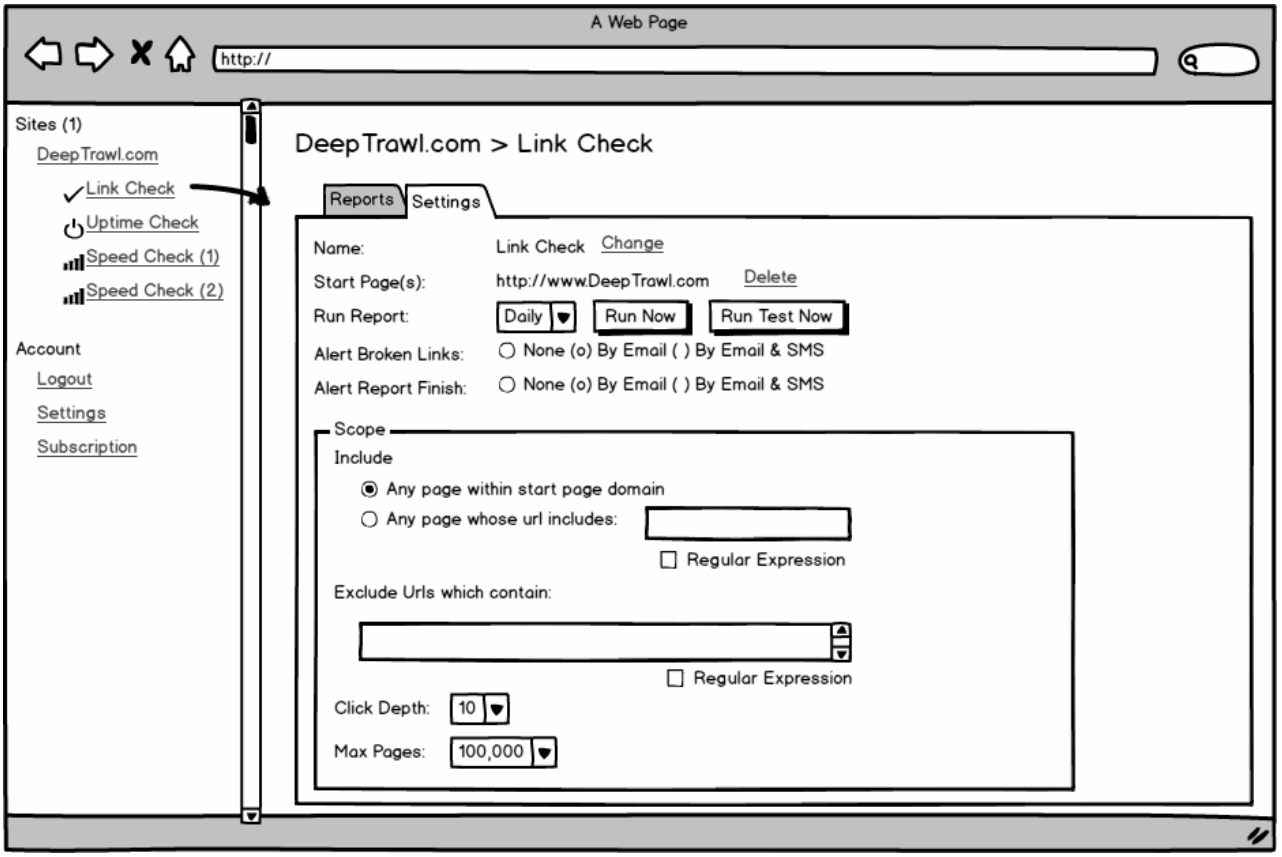


Рисунок 1 – Пример структурной схемы страницы системы

2.2.4 Составление общего плана работы.

В данном разделе приводятся общий план работы над всем программным решением, общие оценки трудоемкости и времени работы над каждой функциональной частью программной системы, диаграмма Ганта. Раздел должен содержать совокупный общий план всей группы разработчиков.

2.2.5 Составление индивидуального плана работы.

В данном разделе приводится индивидуальный план работы. Все дальнейшее изложение в пояснительной записке должно осуществляться по индивидуальному плану.

2.3 Разработка документации программного модуля

Проектирование алгоритмов и программ – наиболее ответственный этап жизненного цикла программных продуктов, определяющий, насколько создаваемая программа соответствует спецификациям и требованиям со стороны конечных пользователей. Данный раздел должен быть посвящен разработке документации программного модуля.

2.3.1 Диаграмма классов.

Диаграммы классов применяются для моделирования объектно-ориен-

тированных систем. На простых диаграммах показываются классы и отношения между классами; на сложных – классы, интерфейсы, кооперации и отношения между ними. Диаграммы классов дают статический вид системы. Можно также сказать, что они представляют собой взгляды разработчиков на статические состояния проектируемых систем. С помощью диаграмм классов составляется словарь системы, создаются диаграммы компонентов и развертывания. Следует особо подчеркнуть, что диаграммы классов важны не только для визуализации, специфицирования и документирования структурных моделей, но также для прямого и обратного проектирования исполняемых кодов систем. На рисунке 2 приведен пример простой диаграммы классов, моделирующей объекты системы регистрации курсов и отношения между ними.

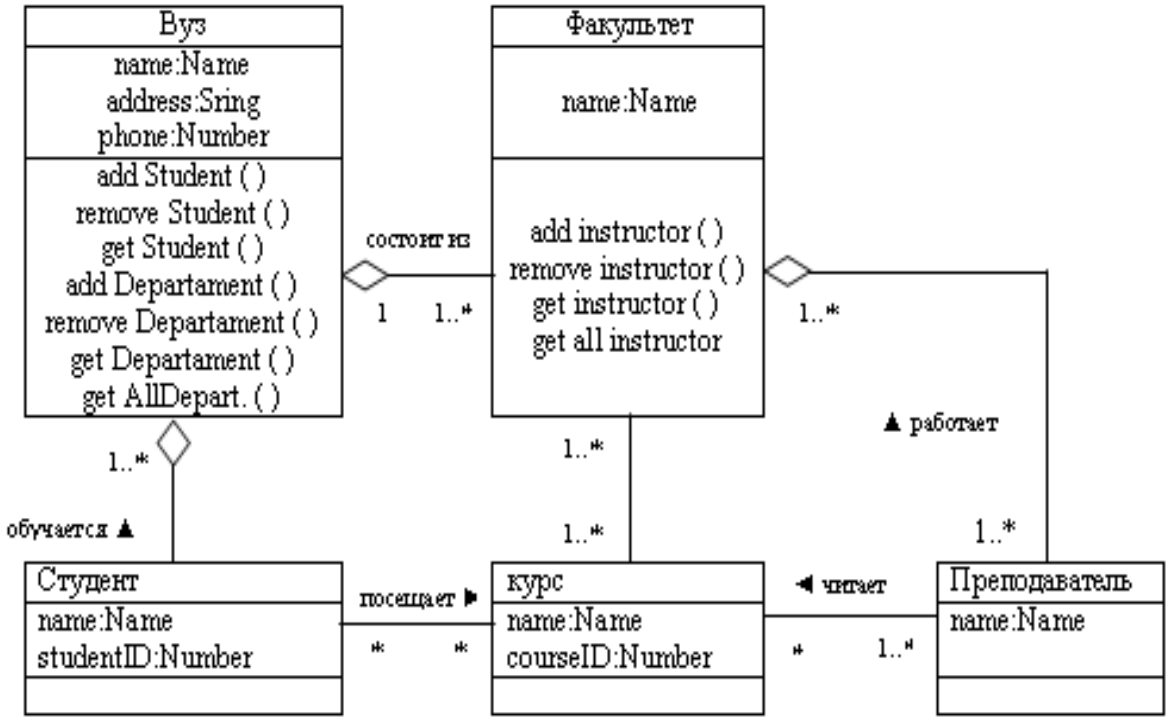


Рисунок 2 – Пример диаграммы классов

2.3.2 Диаграмма деятельности.

При моделировании поведения проектируемой или анализируемой системы возникает необходимость детализировать особенности алгоритмической и логической реализации выполняемых системой операций. Традиционно для этой цели использовались блок-схемы или структурные схемы алгоритмов. Каждая такая схема акцентирует внимание на последовательности совершения определенных действий или элементарных операций, которые в совокупности приводят к получению желаемого результата.

Важно подчеркнуть то обстоятельство, что с увеличением сложности системы строгое соблюдение последовательности выполняемых операций приобретает все более важное значение. Если попытаться заварить кофе холодной водой, то мы можем только испортить одну порцию напитка. Нарушение по-

следовательности операций при ремонте двигателя может привести к его поломке или выходу из строя. Еще более катастрофические последствия могут произойти в случае отклонения от установленной последовательности действий при взлете или посадке авиалайнера, запуске ракеты, регламентных работах на атомной электростанции.

Для моделирования процесса выполнения операций используются так называемые диаграммы деятельности (рисунок 3).

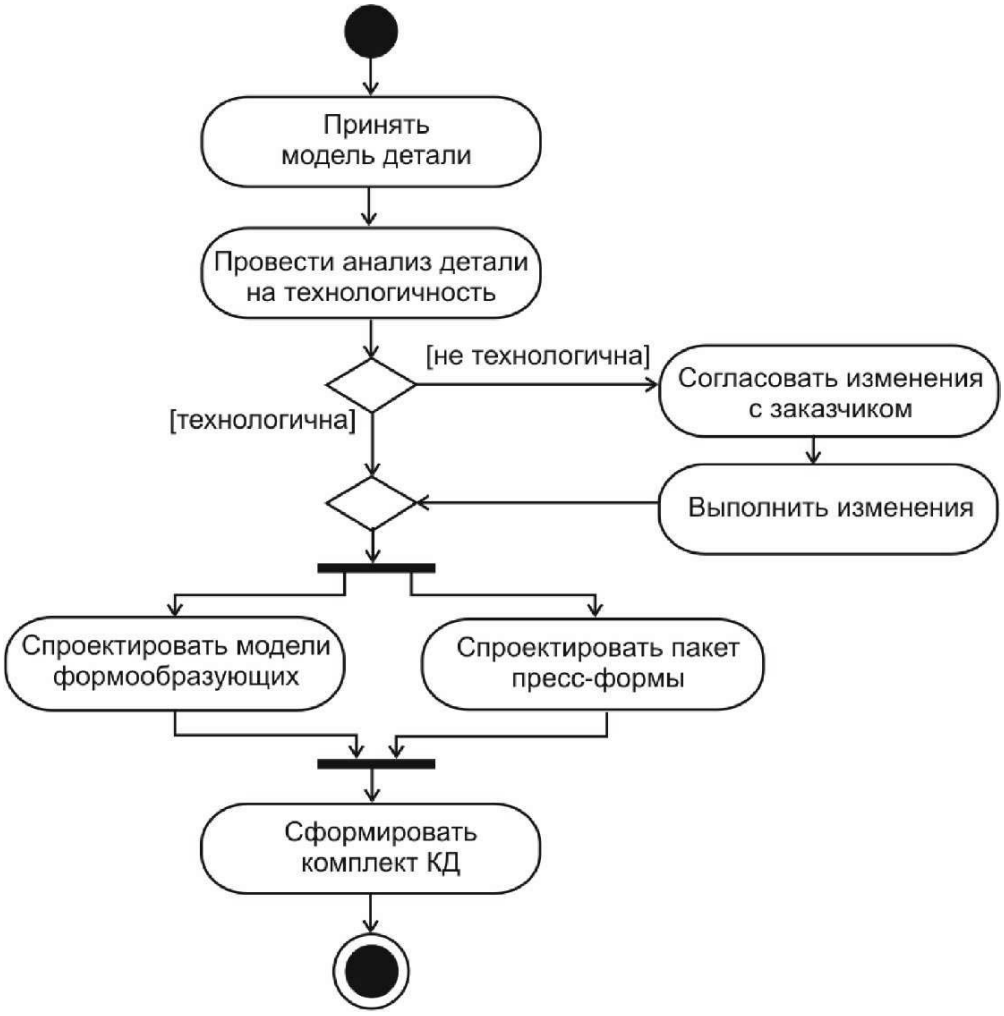


Рисунок 3 – Пример диаграммы деятельности

Применяемая в диаграммах деятельности графическая нотация во многом похожа на нотацию диаграммы состояний, поскольку на диаграммах деятельности также присутствуют обозначения состояний и переходов. Отличие заключается в семантике состояний, которые используются для представления не деятельностей, а действий, и в отсутствии на переходах сигнатуры событий. Каждое состояние на диаграмме деятельности соответствует выполнению некоторой элементарной операции, а переход в следующее состояние срабатывает только при завершении этой операции в предыдущем состоянии. Графически диаграмма деятельности представляется в форме графа деятельности, вершинами которого являются состояния действия, а дугами – переходы от одного состояния действия к другому.

2.4 Разработка программного модуля

В разделе рассматриваются вопросы построения исходного текста программного пакета для решения поставленной задачи в заданной системе программирования.

Для каждого подраздела описываются используемые константы, скалярные и составные типы данных, а также переменные. При задании состава переменных следует всегда сохранять вводимые значения исходных данных до конца программы, не допуская их переопределения в результате расчетов.

2.4.1 Выбор и обоснование технологий, используемых в программном модуле.

В данном разделе следует отметить, используются ли при существующей технологии решения задачи какие-либо технические и программные средства и каким образом. Если на рынке программных средств имеются готовые программные решения, желательно дать краткое описание и провести анализ хотя бы одной такой разработки, указав основные характеристики и функциональные возможности.

Необходимо произвести обзор рынка программных средств. Адреса используемых при обзоре ресурсов следует добавить в список литературы курсовой работы.

Затем нужно отметить, чем, с точки зрения реализации, должна и будет отличаться проектируемая система или технология решения задачи от существующей, а также почему необходимо разрабатывать новое решение и чем оно должно отличаться от уже имеющихся.

Далее следует дать краткую характеристику современных технологий проектирования информационных (телекоммуникационных) систем, перечислить их положительные черты и недостатки, основные факторы выбора, обосновать выбор применяемой технологии и указать особенности ее использования в данном проекте.

Обоснование проектных решений по программному обеспечению (ПО) задачи заключается в формировании требований к системному (общему) и специальному прикладному программному обеспечению и в выборе на основе этих требований его соответствующих компонентов.

При обосновании проектного решения по специальному ПО необходимо сформулировать требования, которым должны удовлетворять проектируемые программные средства.

Формулировка требований к специальному ПО должна происходить с учетом выдвинутых предложений по техническому обеспечению. При обосновании проектных решений по специальному программному обеспечению задачи необходимо определить возможности выбранных программных средств, при использовании которых достигаются требования к прикладному программному обеспечению (например, возможность организации удобного интерфейса администратора информационной системы, оптимизации запросов к данным и т. п.).

Выбор средств проектирования и разработки по возможности необходимо

аргументировать, сравнивая их с аналогичными средствами, существующими на рынке.

2.4.2 Настройка окружения и репозитория кода.

Система управления версиями (репозиторий кода) (от англ. Version Control System, VCS или Revision Control System) – программное обеспечение для облегчения работы с изменяющейся информацией. Она позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое.

Ситуация, в которой электронный документ за время своего существования претерпевает ряд изменений, достаточно типична. При этом часто бывает важно иметь не только последнюю версию, но и несколько предыдущих. В простейшем случае можно просто хранить несколько вариантов документа, нумеруя их соответствующим образом. Такой способ неэффективен (приходится хранить несколько практически идентичных копий), требует повышенного внимания и дисциплины и часто ведёт к ошибкам, поэтому были разработаны средства для автоматизации этой работы.

Рассматриваемые системы наиболее широко используются при разработке программного обеспечения для хранения исходных кодов разрабатываемой программы.

В данном разделе необходимо обосновать использование одного из вариантов репозитория кода, показать его настройку для применения в курсовой работе. Также следует привести сведения о настойке окружения для разработки программной системы.

2.4.3 Описание используемых структур данных.

Понятие структуры данных является настолько фундаментальным, что для него сложно подобрать простое определение. Задача упрощается, если попробовать сформулировать это понятие по отношению к типам данных и переменным. Как известно, программа представляет собой единство алгоритмов (процедур, функций) и обрабатываемых им данных. Данные, в свою очередь, определяются базовыми и производными типами данных – «идеальными» представлениями переменных фиксированной размерности с наборами известных операций над ними и их компонентами. Переменные – это именованные области памяти, в которые «отображаются» сконструированные типы данных.

В программе всегда можно выделить группы косвенно связанных (по использованию данных в одних и тех же процедурах и функциях) и непосредственно связанных (по наличию взаимосвязей через указатели) переменных. Их в первом приближении и можно считать структурами данных.

Различаются простые (базовые, примитивные) структуры (типы) данных и интегрированные (структурированные, композитные, сложные). Простыми называются такие структуры данных, которые не могут быть расчленены на составные части, большие, чем биты. С точки зрения физической структуры важным является то обстоятельство, что в определенной машинной архитектуре,

в определенной системе программирования всегда можно заранее сказать, каков будет размер данного простого типа и какова структура его размещения в памяти. С логической точки зрения простые данные – неделимые единицы. Интегрированными называются такие структуры данных, составными частями которых являются другие структуры данных – простые или, в свою очередь, интегрированные. Интегрированные структуры данных конструируются программистом с использованием средств интеграции данных, предоставляемых языками программирования.

В зависимости от отсутствия или наличия явно заданных связей между элементами данных следует различать несвязные (векторы, массивы, строки, стеки, очереди) и связные (связные списки) структуры.

В рассматриваемом разделе необходимо описать все используемые в программном модуле структуры данных.

2.5 Тестирование программного модуля

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

Тест – это процедура, которая позволяет либо подтвердить, либо опровергнуть работоспособность кода. Когда программист проверяет работоспособность разработанного им кода, он выполняет тестирование вручную. В данном контексте тест состоит из двух этапов: стимулирование кода и проверка результатов его работы. Автоматический тест выполняется иначе: вместо программиста стимулированием кода и проверкой результатов занимается компьютер, который отображает на экране результат проведения теста: код работоспособен или неработоспособен. При этом происходит «инверсия ответственности»: от логики тестов и их качества зависит, будет ли код соответствовать техническому заданию. Методика разработки через тестирование заключается главным образом именно в организации автоматических тестов и выработке определённых навыков тестирования.

В данном разделе необходимо описать тесты для программной системы, их разработку и листинги получившихся тестов.

3 Варианты заданий к курсовой работе

Задание к курсовой работе содержит тему работы и исходные данные (тип операционной системы, среду программирования и т. п.).

Примеры тем курсовых работ приведены в таблице 2.

Таблица 2 − Темы вариантов курсовых работ

Вариант	Тема работы
1	Сервис для быстрого бронирования столиков онлайн
2	Программная система «Экспертная система для оптимизации настроек светофоров «Зеленая волна»
3	Программная система для анализа и распознавания примитивных образов
4	Система информирования работников образовательного учреждения
5	Программная система «Учет и планирование поверок приборов измерения»
6	Программная система для тестирования знаний на категорию В
7	Комплексная тема «Программная система для тестирования знаний на категорию В»
8	Система контроля влажности изолирующего слоя теплотрассы
9	Экспертная система для музыкальных рекомендаций
10	Программная система «Гитарный тюнер со встроенным метрономом»
11	Программная система Timemanagement
12	Программная система сбора статистических данных по сессиям
13	Модуль «Опрос. Панель администратора» для CMS
14	Мобильный агрегатор новостей Belarus Online

Тема работы должна быть согласована с руководителем курсовой работы.

4 Оформление курсовой работы

4.1 Оформление графической части

Графическая часть проекта выполняется на одном листе формата А1; схемы алгоритмов и диаграммы – в соответствии с требованиями ГОСТ 19.701–90 в среде инженерной графики, согласованной с руководителем работы. При этом наименьшая величина высоты отдельного символа процесса и данных – 10 мм, а наименьшее расстояние между блоками – 5 мм. Все блоки структурной схемы рекомендуется изображать одинаковой высоты и ширины. При этом допускается такое отклонение в соотношениях геометрической формы изображений, которое не затрудняет определение назначения блока при чтении схемы.

В графе основной надписи листа записывается обозначение, состоящее не менее чем из двух букв, сокращенно означающих наименование заданного метода, и семизначного цифрового кода вида 00.00.000, заканчивающегося обо-

значением кода документа чертежа Д1, означающего документы прочие.

Пример оформления шифра для численного явного метода интегрирования Эйлера: ЯМЭ 00.00.000 Д1.

4.2 Оформление пояснительной записки

Пояснительная записка оформляется на листах формата А4, ее текст – с соблюдением всех требований ГОСТ 2.105–95 *Общие требования к текстовым документам*, а содержание соответствовать требованиям раздела 2. Используемые в пояснительной записке термины должны отвечать ГОСТ ИСО/МЭК 2382–99, а все страницы иметь основную надпись согласно ГОСТ 2.104–2006. Каждый раздел, кроме содержания, введения, заключения и списка литературных источников, должен начинаться с новой страницы с основной надписью формы 2 высотой 40 мм. Остальные страницы должны иметь основную надпись формы 2а высотой 15 мм. Обозначение документа, записываемое в основной надписи листов пояснительной записки, должно совпадать с обозначением основной надписи листа графической части и заканчиваться шифром ПЗ вместо Д1.

4.2.1 Оформление содержания.

Содержание пояснительной записки размещается сразу после бланка задания к курсовой работе. Оно включает наименования разделов и подразделов пояснительной записки с указанием номера страницы. Сквозная нумерация ее листов выполняется в правом верхнем углу, начиная с титульного листа (на нем номер не ставится). Содержание фактически расположено на третьей странице пояснительной записки. При оформлении электронного варианта текста рекомендуется использовать иерархическую структуру заголовков, автоматическую расстановку номеров страниц и вставку автосодержания.

4.2.2 Оформление введения.

Введение оформляется на отдельной странице. Объем его не должен превышать четыре процента от общего состава пояснительной записки. В тексте введения описывается, к какой области относится данная работа и на основе какого документа производится ее выполнение.

4.2.3 Оформление и состав заключения.

В заключении необходимо проанализировать полученные результаты в ходе выполнения курсовой работы. Отражаются сильные и слабые стороны разработанной программы, дается рекомендация по ее дальнейшему применению и развитию. Объем заключения не должен превышать 4 % от общего состава пояснительной записки.

4.2.4 Оформление списка используемых источников.

Оформление списка литературных источников, используемых при выполнении курсовой работы, производится в соответствии с требованиями

ГОСТ 7.1–2003 *Библиографическое описание документа. Общие требования и правила составления*.

Список используемых источников составляется согласно порядку упоминания в тексте пояснительной записки, а ссылки на список источников выполняются в прямоугольных скобках. Примером оформления являются данные методические рекомендации.

Пояснительная записка должна помещаться в жесткую обложку для курсовой работы с титульным листом без основной надписи.

5 Оценивание курсовой работы

Для расчета итоговой оценки по курсовой работе используется методика на основании Положения о проведении рейтинг-контролей (таблица 3).

Таблица 3 – Начисление баллов при выполнении курсовой работы

Этап	Минимальный балл	Максимальный балл
Выполнение раздела 1	7	12
Выполнение раздела 2	7	12
Выполнение раздела 3	8	12
Выполнение раздела 4	7	12
Выполнение раздела 5	7	12
Итого	36	60

Максимальные баллы начисляются за выполненные досрочно (точно в срок) разделы; минимальные – при несвоевременном выполнении разделов (без соблюдения календарных сроков выполнения курсовой работы).

По результатам защиты курсовой работы начисляется дополнительно от 15 до 40 баллов. Итоговая оценка вычисляется как сумма баллов и переводится в пятибалльную шкалу по таблице 4.

Таблица 4 – Итоговая оценка по курсовой работе

Оценка	Отлично	Хорошо	Удовлетворительно	Неудовлетворительно
Балл	85–100	68–84	51–67	0–50

Список литературы

1 **Dennis, A.** System Analysis & Design. An Object-Oriented Approach with UML=Системный анализ и проектирование на универсальном языке моделирования / A. Dennis, B. Wixom, D. Tegarden. – 5th ed. – New York: John Wiley & Sons, 2015.

2 **Арлоу, Д.** UML 2 и Унифицированный процесс. Практический объектно-ориентированный анализ и проектирование / Д. Арлоу. – Москва: Символ-Плюс, 2015. – 624 с.

3 **Гагарина, Л. Г.** Технология разработки программного обеспечения [Электронный ресурс]: учебное пособие / Л. Г. Гагарина, Е. В. Кокорева, Б. Д. Сидорова-Виснадул; под ред. Л. Г. Гагариной. – Москва: ФОРУМ; ИНФРА-М, 2022. – 400 с. – Режим доступа: https://znanium.com/catalog/product/1699927. – Дата доступа: 20.09.2022.

4 **Гэртнер, М.** ATDD-разработка программного обеспечения через приемочные тесты: пер. с англ. / М. Гэртнер. – Москва: ДМК Пресс, 2013. – 232 с.: ил.

5 **Макаровских, Т. А.** Документирование программного обеспечения. В помощь техническому писателю : учебное пособие / Т. А. Макаровских. – 2-е изд. – Москва : ЛЕНАНД, 2015. – 266 с.

6 **Макконнелл, С.** Совершенный код. Мастер-класс=Code Complete. Second Edition: пер. с англ. / С. Макконнелл. – Санкт-Петербург: БХВ, 2020. – 896 с. : ил.

7 **Орлов, С. А.** Программная инженерия : учебник для вузов / С. А. Орлов. – Санкт-Петербург : Питер, 2016. – 640 с.

8 Паттерны проектирования=Head First Design Patterns: пер. с англ. / Э. Фримен [и др.]. – Санкт-Петербург : Питер, 2016. – 656 с. : ил.

9 **Рихтер, Д.** CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C# : пер. с англ. / Д. Рихтер. – 4-е изд. – Санкт-Петербург : Питер, 2016. – 896 с. : ил.

10 **Ройс, У.** Управление проектами по созданию программного обеспечения : унифицированный подход / У. Ройс. – Москва : Лори, 2014. – 424 с.