МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Программное обеспечение информационных технологий»

АРХИТЕКТУРА ПРОГРАММНЫХ СИСТЕМ

Методические рекомендации к курсовому проектированию для студентов направления подготовки 09.03.04 «Программная инженерия» очной формы обучения

Могилев 2023

УДК 004.4
ББК 32.973.26-018.2
А87

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой «Программное обеспечение информационных технологий» «29» ноября 2022 г., протокол № 5

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 «Архитектура программных систем» для студентов направления подготовки 09.03.04 «Программная инженерия» очной формы обучения.

Учебно-методическое издание

АРХИТЕКТУРА ПРОГРАММНЫХ СИСТЕМ

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 16 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 языке программирования, использования пакетов программ инженерной графики и математических систем.

Задачи курсовой работы:

– разработка архитектуры проектируемой программной системы;

– 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.

1 Общие положения

1.1 Организация выполнения курсовой работы

Курсовая работа выполняется в соответствии с выданным заданием, подписанным руководителем проекта и утвержденным заведующим кафедрой. В задании указываются тема курсовой работы, исходные данные, содержание проекта и календарный план выполнения работы.

После каждого этапа в соответствии с контрольными датами по календарному плану все материалы курсовой работы должны быть предоставлены руководителю для контроля. При выполнении курсовой работы рекомендуется использовать [1–9].

Выполненная курсовая работа подписывается студентом и сдается руководителю на проверку. В ходе проверки оценивается степень соответствия проекта выданному заданию, отмечаются положительные стороны и недостатки работы, а в случае необходимости указывается, что следует доработать. По итогам проверки формулируется вывод о допуске работы к защите. Если курсовая работа не допускается к защите, то необходимо устранить замечания руководителя и снова сдать работу на проверку.

В ходе выполнения курсовой работы рекомендуется использовать литературные источники [1–8].

1.2 Содержание пояснительной записки

Рекомендуемый объем пояснительной записки – 25–30 страниц.

В состав пояснительной записки к курсовой работе входит следующее.

Введение.

Описание предметной области.

Обзор литературы.

Постановка задачи.

Обоснование актуальности задачи.

Обоснование используемых технологий, принципов, методик решения задачи.

Проектирование архитектуры программно-информационной системы.

Установление требований.

Составление спецификации требований.

Реализация программно-информационной системы.

Результаты.

Заключение.

Приложение 1 – Код программных модулей.

1.3 Содержание графической части работы

Графическая часть работы выполняется на листах формата А1, где изображаются основные диаграммы системы.

Изменения размера формата и числа листов графической части допускаются только по согласованию с руководителем курсовой работы.

1.4 Критерии оценки курсовой работы

Курсовая работа включает шесть разделов, которые входят по три в каждый модуль. Каждый раздел оценивается количеством баллов от 6 до 10.

При этом:

– максимальное количество баллов по разделу начисляется в том случае, если студент выполнил раздел в полном объеме и в соответствии с методическими рекомендациями, проявил элементы творчества, использовал достаточное количество литературных и нормативных источников, аккуратно и правильно оформил пояснительную записку, вовремя или досрочно представил материалы раздела руководителю;

– минимальное количество баллов по разделу начисляется в том случае, если студент выполнил раздел в соответствии с методическими рекомендациями, не проявил творчества, использовал явно недостаточное количество источников, допустил ошибки в расчетах или графических материалах, но устранил их, представил материалы раздела без соблюдения календарных сроков выполнения курсовой работы;

– промежуточные значения баллов начисляются в зависимости от уровня творчества студента, наполнения раздела, качества оформления, сроков представления материалов.

По результатам защиты курсовой работы начисляется дополнительно от 15 до 40 баллов. При оценке работы учитываются следующие пункты.

1 Полнота решения всех задач проекта и качество содержания проекта.

2 Самостоятельность решения поставленных задач.

3 Наличие элементов научных исследований (теоретических и экспериментальных).

4 Наличие элементов творчества студента.

5 Оформление графической части.

6 Оформление пояснительной записки.

7 Четкость и грамотность изложения.

8 Качество и глубина ответов на вопросы.

Каждый из приведенных пунктов оценивается максимальным количеством баллов 5.

Итоговая оценка курсового проекта (работы) представляет собой сумму баллов за его выполнение и защиту и переводится в пятибалльную шкалу в соответствии с таблицей 1.

Таблица 1 – Итоговая оценка по курсовой работе

Оценка	Отлично	Хорошо	Удовлетворительно	Неудовлетворительно
Балл	87–100	65–86	51–64	0–50

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 [1]. В этом документе кратко описывается 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 путем указания названия источника, режима и даты доступа.

В подразделе 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. При необходимости можно самостоятельно вывести расчетные формулы. В этом случае указывается метод вывода полученных выражений.

В подразделе не приводятся 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 [2]. 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 Обоснование актуальности задачи.

При обосновании актуальности задачи в разделе курсовой работы необходимо решить, почему именно эту проблему нужно в настоящее время изучать.

Актуальность исследования – это степень его важности на данный момент и в данной ситуации для решения определенной проблемы или задачи [1].

Актуальность проблемы исследования – это востребованность изучения и решения данной проблемы в обществе.

Обоснование актуальности исследования – это объяснение необходимости изучения данной темы и проведения исследования в процессе общего познания.

Актуальность темы исследования обусловлена следующими факторами:

– восполнение каких-либо пробелов в науке;

– дальнейшее развитие проблемы в современных условиях;

– своя точка зрения в вопросе, по которому нет единого мнения;

– обобщение накопленного опыта;

– суммирование и продвижение знаний по основному вопросу;

– постановка новых проблем для привлечения внимания общественности.

Актуальность курсовой работы может состоять в необходимости получения новых данных, проверки совсем новых методов и т. п.

Часто в исследовательском проекте вместе со словом «актуальность» используют слово «новизна» исследования.

2.3 Обоснование используемых технологий, принципов, методик решения задачи.

Современные крупные проекты характеризуются, как правило, следующими особенностями:

– сложность описания (достаточно большое количество функций, процессов, элементов данных и сложные взаимосвязи между ними), требующая тщательного моделирования и анализа данных и процессов;

– наличие совокупности тесно взаимодействующих компонентов (подсистем), имеющих свои локальные задачи и цели функционирования (например, традиционных приложений, связанных с обработкой транзакций и решением регламентных задач, и приложений аналитической обработки (поддержки принятия решений), использующих нерегламентированные запросы к данным большого объема);

– отсутствие прямых аналогов, ограничивающее возможность использования каких-либо типовых проектных решений и прикладных систем;

– необходимость интеграции существующих и вновь разрабатываемых приложений;

– функционирование в неоднородной среде на нескольких аппаратных платформах;

– разобщенность и разнородность отдельных групп разработчиков по уровню квалификации и сложившимся традициям использования тех или иных инструментальных средств;

– существенная временная протяженность проекта, обусловленная, с одной стороны, ограниченными возможностями коллектива разработчиков, и, с другой стороны, масштабами организации-заказчика и различной степенью готовности отдельных ее подразделений к внедрению информационной системы (ИС).

Для успешной реализации проекта объект проектирования (ИС) должен быть прежде всего адекватно описан, должны быть построены полные и непротиворечивые функциональные и информационные модели ИС [3]. Накопленный к настоящему времени опыт проектирования ИС показывает, что это логически сложная, трудоемкая и длительная по времени работа, требующая высокой квалификации участвующих в ней специалистов. Однако до недавнего времени проектирование ИС выполнялось в основном на интуитивном уровне с применением неформализованных методов, основанных на искусстве, практическом опыте, экспертных оценках и дорогостоящих экспериментальных проверках качества функционирования ИС. Кроме того, в процессе создания и функционирования ИС информационные потребности пользователей могут изменяться или уточняться, что еще более усложняет разработку и сопровождение таких систем.

Перечисленные факторы способствовали развитию исследований в области методологии программирования. Программирование обрело черты системного подхода с разработкой и внедрением языков высокого уровня, методов структурного и модульного программирования, языков проектирования и средств их поддержки, формальных и неформальных языков описаний системных требований

и спецификаций и т. д.

В курсовой работе необходимо подробно описать (и обосновать) выбор языка программирования, фреймворка или библиотек, которые будут использоваться в проекте. Выбор обязательно обосновать (например, методом экспертных оценок).

Методы экспертных оценок являются частью обширной области теории принятия решений, а само экспертное оценивание – процедура получения оценки проблемы на основе мнения специалистов (экспертов) с целью последующего принятия решения (выбора) [4].

В случаях чрезвычайной сложности проблемы, ее новизны, недостаточности имеющейся информации, невозможности математической формализации процесса решения приходится обращаться к рекомендациям компетентных специалистов, прекрасно знающих проблему, – к экспертам. Их решение задачи, аргументация, формирование количественных оценок, обработка последних формальными методами получили название метода экспертных оценок.

Существуют две группы экспертных оценок:

1) индивидуальные оценки, основанные на использовании мнения отдельных экспертов, независимых друг от друга;

2) коллективные оценки, основанные на использовании коллективного мнения экспертов.

К первой группе относится такая оценка, когда каждый эксперт принимает решение самостоятельно.

Ранжирование – это расположение объектов в порядке возрастания или убывания какого-либо присущего им свойства. Ранжирование позволяет выбрать из исследуемой совокупности факторов наиболее существенный.

Непосредственная оценка. Часто бывает желательным не только упорядочить (ранжировать объекты анализа), но и определить, на сколько один фактор более значим, чем другие. В этом случае диапазон изменения характеристик объекта разбивается на отдельные интервалы, каждому из которых приписывается определенная оценка (балл), например, от 0 до 10. Именно поэтому метод непосредственной оценки иногда именуют также балльным методом.

Метод ранжировки заключается в том, что выбирается m критериев, каждый из которых оценивается экспертом по степени важности k (от 1 до 10), и для каждого сравниваемого продукта Π_i ($i = \overline{1..n}$, n – количество продуктов) записывается оценка a_{ij} ($j = \overline{1..m}$) по каждому критерию (таблица 1).

Таблица 1 – Матрица экспертных предпочтений

	1	…	j	…	m
Π_1	a_{11}	…	a_{1j}	…	a_{1m}
…	…	…	…	…	…
Π_i	a_{i1}	…	a_{ij}	…	a_{im}
	…	…	…		…
Π_n	a_{n1}	…	a_{nj}	…	a_{nm}

Затем для каждого продукта Пi подсчитывается оценка $S_i = \sum ka_{ij}$.

Максимальная оценка S_i будет у наилучшего варианта.

2.4 Проектирование архитектуры программно-информационной системы.

В разделе рассматриваются вопросы проектирования архитектуры программно-информационной системы.

2.4.1 Установление требований.

Документ, описывающий требования, является практическим результатом этапа установления требований и должен быть представлен в разделе проектирования архитектуры.

Ядро документа описания требований состоит из формулировок (изложения) требований. Требования могут быть сгруппированы в виде формулировок сервисов (зачастую называемых функциональными требованиями) и формулировок ограничений. Формулировки сути сервисов могут быть затем разделены на требования к функциям (function requirements) и требования к данным (data requirements) [1, 9].

Не говоря уже о самих требованиях, документ описания требований должен обращаться к проектным вопросам. Обычно проектные вопросы рассматриваются в начале документа, а затем в конце документа.

Во вводной части документа рассматривается бизнес-контекст проекта, включая цель проекта, участников проекта, а также основные ограничения. Ближе к заключительной части документа поднимаются другие проектные вопросы, включая план-график выполнения проектных работ, бюджет, риски, документацию и т. д.

Шаблоны документов описания требований широко доступны. Их можно найти в учебниках, стандартах, выпускаемых такими организациями как ISO, IEEE и т. д., на Web-страницах консалтинговых фирм, программных средствах разработки и т. д.

Шаблон документа описания требований определяет структуру документа и содержит подробные указания о содержании каждого из разделов документа. Указания могут включать содержание вопросов, мотивацию, примеры и дополнительные соображения.

Документ описания требований [9] имеет следующее содержание (рисунок 1).

Далее охарактеризуем основные разделы документа описания требований.

Предварительные замечания к проекту дают ориентиры руководителям и участникам проекта, ответственным за принятие решений. В начале документа необходимо ясно обозначить цели и рамки проекта, а затем описать деловой контекст системы.

Документ описания требований должен создать прецедент для системы. В частности, необходимо упомянуть обо всех усилиях, приложенных для обоснования необходимости системы на этапе планирования системы. Этот документ также должен прояснить вопрос о том, каким образом предлагаемая система может способствовать достижению деловых целей и решению задач организацией.

Необходимо указать конкретные имена участников проекта системы.

1. Предварительные замечания к проекту.
1.1. Цели и рамки проекта.
1.2. Деловой контекст.
1.3. Участники проекта.
1.4. Идеи в отношении решений.
1.5. Обзор документа.
2. Системные сервисы.
2.1. Рамки системы.
2.2. Функциональные требования.
2.3. Требования к данным.
3. Системные ограничения.
3.1. Требования к интерфейсу.
3.2. Требования к производительности.
3.3. Требования к безопасности.
3.4. Эксплуатационные требования.
3.5. Политические и юридические требования.
3.6. Другие ограничения.
4. Проектные вопросы.
4.1. Открытые вопросы.
4.2. Предварительный план-график.
4.3. Предварительный бюджет.
Приложения.
Глоссарий.
Деловые документы и формы.
Ссылки.

Рисунок 1 – Структура документа описания требований

Документ описания требований должен предоставлять перечень существующих программных пакетов и компонент, которые должны быть в дальнейшем изучены в качестве вариантов возможных решений.

Системные сервисы. Основная часть документа описания требований посвящена определению системных сервисов. Эта часть может занимать до половины всего объема документа. Это, пожалуй, единственная часть документа, которая может содержать обобщенные модели – модели бизнес-требований.

Рамки системы можно моделировать с помощью диаграммы контекста. В пояснениях к диаграмме контекста должны быть определены рамки системы.

Функциональные требования можно моделировать с помощью диаграммы бизнес-прецедентов. Однако диаграмма охватывает перечень функциональных требований только в самом общем виде. Все требования необходимо обозначить, классифицировать и определить.

Требования к данным можно моделировать с помощью диаграммы бизнес-классов. Так же, как и в случае функциональных требований, диаграмма бизнес-классов не дает полного определения структур данных для бизнес-процессов.

Каждый бизнес-класс требует дальнейших пояснений. Необходимо описать атрибутное наполнение классов и определить идентифицирующие атрибуты классов. В противном случае невозможно правильно представить ассоциации.

Системные ограничения. Системные сервисы определяют, что должна делать система. Системные ограничения определяют, насколько система ограничена при выполнении обслуживания. Системные ограничения связаны со следующими видами требований:

- требования к интерфейсу;
- требования к производительности;
- требования к безопасности;
- эксплуатационные требования;
- политические и юридические требования.

Возможны и другие виды ограничений. Например, в отношении некоторых систем могут предъявляться повышенные требования к легкости их использования (требования в отношении пригодности к использованию) или легкости их сопровождения (требования в отношении пригодности к сопровождению).

Проектные вопросы. Заключительная часть документа описания требований обращается к другим проектным вопросам. Один из важных разделов этой части называется «Открытые вопросы». Здесь поднимаются все вопросы, которые могут сказаться на успехе проекта и которые не рассматривались в других разделах документа. Сюда относится ожидаемое возрастание значения некоторых требований, которые в текущий момент выходят за рамки проекта. Сюда можно отнести также любые потенциальные проблемы и отклонения в поведении системы, которые могут начаться в связи с развертыванием системы.

В этой же части необходимо представить предварительный план-график выполнения основных проектных заданий. Сюда же относится предварительное распределение людских и других ресурсов. Для выработки стандартных плановых графиков можно использовать программные средства управления проектами, например, такие как система PERT (program evaluation and review technique – метод оценки и пересмотра планов) или карты Ганта [1, 2].

Приложения. Приложения содержат остальную, полезную для понимания требований информацию.

Раздел ссылок содержит перечень документов, которые упоминаются или используются при подготовке документа описания требований. К ним могут относиться книги и другие опубликованные источники информации, но – что, пожалуй, даже более важно – необходимо также упомянуть протоколы совещаний, служебные записки и внутренние документы.

2.4.2 Составление спецификации требований.

Требования необходимо специфицировать (т. е. задать) графически или каким-либо иным формальным способом. Всесторонняя спецификация системы может потребовать использования многих типов моделей. Язык UML изобилует интегрированными методами моделирования, способными помочь бизнес-аналитику справиться с этой задачей. Спецификация – подобно процессу разработки ПО в целом – итеративный процесс с пошаговым наращиванием уровня детали-

зации моделей. Немаловажную роль в успешном моделировании играет использование CASE-средств.

В результате спецификации требований вырабатываются три категории моделей: модели состояний, модели поведения и модели изменения состояния. Для каждой из категорий существует несколько методов работы с ними. Далее объясняются и иллюстрируются на примерах все основные методы моделирования языка UML.

Принципы спецификации требований. Спецификация требований связана с детальным моделированием требований заказчиков, определенных в процессе установления требований. При этом рассматриваются только услуги, которые стремятся получить от системы заказчики (формулировки сервисов). На этапе спецификации требований формулировки ограничений не подлежат дальнейшей проработке, хотя и могут претерпеть изменения как результат обычного цикла итерации [4].

В качестве входной информации процесса спецификации требований выступают неформальные требования заказчиков, а результатом этого процесса являются модели спецификации проектных конструкций. Эти модели дают более формальное определение различных сторон (представлений) системы. Обычно требования пользователей в процессе спецификации подразделяются на две основные категории: функциональные требования и требования к данным.

В качестве результата этапа спецификации выступает расширенный («детально проработанный») документ описания требований. Новый документ часто называют документом спецификации требований (или просто «спецификацией» на жаргоне разработчиков). Структура исходного документа не изменяется, однако содержание значительно расширяется за счет глав, которые определяют требования заказчиков. Постепенно для целей проектирования и реализации документ спецификации требований заменяет документ описания требований (на практике расширенный документ может по-прежнему называться документом описания требований).

Модели спецификаций можно разделить на три группы [1]:

1) модели состояний;
2) модели поведения;
3) модели изменения состояний.

Модели состояний «детализируют» требования к данным. Модели поведения обеспечивают детализированные спецификации для функциональных требований. Модели изменения состояний охватывают два вида требований и объясняют, каким образом действие функций приводит к изменению данных.

Модели представляются в виде диаграмм на языке визуального моделирования (Visual Modeling Language) – в нашем случае это язык UML [6–8]. Обычно диаграмма служит целям моделирования одной из сторон системы – состояний, поведения или изменения состояний. Заметное исключение составляет диаграмма классов, которая определяет все три аспекта – состояние и поведение объектов, и, косвенно, изменения состояний объектов.

Каждая диаграмма дает представление об определенной стороне системы. Взятые вместе диаграммы дают возможность разработчикам и пользователям

взглянуть на предлагаемое решение с разных точек зрения, выделяя одни его стороны и игнорируя другие. Ни одна из диаграмм в отдельности не дает полного определения системы. Систему можно понять только через взаимосвязанный набор диаграмм.

Аналогично случаю интерпретации завершенных моделей конструирование диаграмм – это не последовательный процесс построения одной диаграммы за другой. Диаграммы разрабатываются параллельно, и в результате каждой последующей итерации к ним добавляются новые детали. В то время, как разработчики должны следовать строго определенному процессу разработки, решение о том, какая из моделей должна играть роль «движущей силы» разработки, в значительной мере зависит от личных предпочтений аналитика. Обычно диаграммы прецедентов и модели классов – как наиболее важные типы моделей – конструируются параллельно, взаимно «обогащая» друг друга идеями [2].

С каждой новой итерацией разработки глубина и степень детализации спецификации возрастает. Многие более глубокие свойства объектов модели выражаются скорее в текстовом, нежели графическом виде. Некоторые свойства определяют замысел объекта модели, а не результат анализа. Некоторые другие свойства могут отражать особенности CASE-средств.

Спецификации состояний. Состояние объекта определяется значениями его атрибутов и ассоциаций. Например, объект BankAccount (Банковский счет) может находиться в состоянии «превышение кредитного лимита», если значение атрибута balance (баланс) отрицательно. Поскольку состояния объекта определяются структурам данных, модели структур данных называются спецификациями состояний. Спецификация состояний дает статический взгляд на систему (поэтому моделирование состояний часто называют статическим моделированием). Здесь основной задачей является определение классов проблемной области, их атрибутов и отношений с другими классами. Вначале операции классов обычно не рассматриваются. Они выводятся из моделей спецификации поведения.

В типичной ситуации сначала определяются классы-сущности, т. е. классы, которые определяют проблемную область и характеризуются постоянным присутствием в базе данных системы. Подобные классы иногда называются «бизнес-классами». Классы, которые обслуживают системные события (управляющие классы) и классы, которые представляют GUI-интерфейс (классы представления или пограничные классы), не устанавливаются до тех пор, пока не станут известны поведенческие характеристики системы.

2.5 Реализация программно-информационной системы.

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.

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 в пояснительную записку 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 ПО 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.) [3].

Выбор средств проектирования и разработки по возможности необходимо аргументировать, сравнивая их с аналогичными средствами, существующими на рынке.

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 соот-

ветствующим 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 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.

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) [4]. 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.

2.6 Результаты. В разделе приводятся основные результаты проектирования и разработки программной системы. Приводится пример использования основного функционала системы.

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.

Тема работы должна быть согласована с руководителем курсовой работы.

Таблица 2 − Темы курсовых работ

Вариант	Тема работы
1	Архитектура программной системы «Сервис для быстрого бронирования столиков онлайн»
2	Архитектура программной системы «Экспертная система для оптимизации настроек светофоров «Зеленая волна»
3	Архитектура программной системы для анализа и распознавания примитивных образов
4	Архитектура программной системы «Система информирования работников образовательного учреждения»
5	Архитектура программной системы «Учет и планирование поверок приборов измерения»
6	Архитектура программной системы для тестирования знаний на категорию В
7	Архитектура программной системы «Система контроля влажности изолирующего слоя теплотрассы»
8	Архитектура программной системы «Экспертная система для музыкальных рекомендаций»
9	Архитектура программной системы «Гитарный тюнер со встроенным метрономом»
10	Архитектура программной системы Timemanagment
11	Архитектура программной системы «Сбор статистических данных по сессиям»
12	Архитектура программной системы Мобильный агрегатор новостей Belarus Online

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. 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 ме-

тода, и семизначного цифрового кода вида 00.00.000, заканчивающегося обозначением 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 2382–1, 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 упомина-

ния в тексте пояснительной записки, а ссылки на список источников выполняются в прямоугольных скобках. Примером оформления являются данные методические рекомендации.

Список литературы

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

2 **Лионг, Б.** Практическая программная инженерия на основе учебного примера / Б. Лионг, Л. Мацяшек. – Москва : Бином, 2013. – 960 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: 05.12.2022.

4 **Назаров, С. В.** Архитектура и проектирование программных систем [Электронный ресурс] : монография / С. В. Назаров. – 2-е изд., перераб. и доп. – Москва: ИНФРА-М, 2020. – 374 с. – Режим доступа: https://znanium.com/catalog/product/1093643. – Дата доступа: 05.12.2022.

5 **Рыбальченко, М. В.** Архитектура информационных систем : учебное пособие для вузов / М. В. Рыбальченко. – Москва : Юрайт, 2016. – 91 с.

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

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

8 **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. – 5 th ed. – New York: John Wiley & Sons, 2015. – 544 p.

9 Проектирование архитектур информационных систем: методические указания к лабораторным работам / Сост. К. С. Беляев. – Ульяновск : УлГТУ, 2010. – 48 с.