

Министерство науки и образования
Российской Федерации
Институт математики, экономики и информатики
ФБУН «Иркутский государственный университет»

Черкашин Е. А.

**Как спроектировать текст
выпускной квалификационной работы**

учебное пособие

Иркутск—2017

Оглавление

Введение	3
1. Логическая структура ВКР	4
1.1. Первый этап – заключение	4
1.2. Второй этап — Введение	5
1.3. Собственные эксперименты	
Реализация программной системы	8
1.4. Алгоритмы и программные модули	22
2. Оформление текста	23
2.1. Текстовые объекты	23
3. Формальные части	26
4. Презентация ВКР	27
4.1. Список литературы	28
Литература	29

Введение

Что-то о логической структуре ВКР, а также о логических связях в структуре информации.

Структура ВКР, в целом, соответствует структуре жизненного цикла программного обеспечения (ЖЦ ПО) [1].

1. Логическая структура ВКР

Текст ВКР представляет собой отчет о проделанной работе с дополнительными комментариями. Для того, чтобы заполнить ВКР текстом необходимо знать, прежде всего, что во время дипломной практики было сделано. И тогда самое простое – это написать раздел «Заключение».

1.1. Первый этап – заключение

Заключение представляет собой несколько абзацев текста, подобного следующему

В данной выпускной квалификационной работе/дипломной работе/диссертации/статье рассмотрена задача...Получены следующие результаты:

1. Исследована предметная область ...;
2. Выделен набор функций, которые необходимо было реализовать;
3. Спроектирована архитектура [программной системы];
4. При помощи ... метода разработана информационная модель ...;
5. Произведена реализация [программной системы] на языке программирования ...;
6. Проведено тестирование

Далее делаем вывод о том, достигнута ли цель ВКР или нет и помещаем его после списка задач.

В заключении так же имеет смысл изложить в одном абзаце критические замечания по полученному результату: какие требования удалось реализовать, а какие нет; оценить качество полученного результата; показать каким образом можно удовлетворить требования и улучшить качество.

Кроме того, один абзац, последний, можно посвятить направлениям дальнейшего продвижения в разработке: что еще можно улучшить, какие функции добавить к программной системе, а какие убрать; оценить возможности программного продукта как инструмента решения более широкого класса задач.

Что можно улучшить в тексте заключения Откомментировать каждую задачу, только очень кратко. Например, в п. 3 на стр 4, который посвящен архитектуре, можно указать ИЛИ название архитектуры (например, клиент–сервер), ИЛИ добавить фразу о том, как было получено решение (например, «В результате анализа организационной структуры предприятия спроектирована архитектура ...»).

Теперь имеем текст заключения, который уже практически готов, к нему мы вернемся уже в самом конце для проверки его непротиворечивости основному тексту и, возможно, дополнению еще парой–тройкой фраз.

1.2. Второй этап — Введение

Введение представляет собой сокращенный вариант основного текста с добавлением общей информации. Поэтому писать введение удобно [для мозга] с середины его текста: копируем первый, третий и второй абзацы заключения (задача ВКР, список задач и вывод о достижимости цели).

Сначала изменим список задач. Для этого выкинем из предложений все комментарии.

...Для этого решены следующие задачи:

1. Исследована предметная область ...;
2. Выделен набор функций, которые необходимо было реализовать;
3. Спроектирована архитектура [программной системы];
4. ~~При помощи ... метода~~ Разработана информационная модель ...;
5. Произведена реализация [программной системы] ~~на языке программирования ...~~;
6. Проведено тестирование

Обращаю внимание, что текст списка задач представлен в том же времени и форме, что и в заключении, так как ВКР — это отчет о проделанной работе, а не план работ на будущее.

На следующем этапе формулируется *цель*¹ ВКР – это либо проведение исследования (научная цель), разработка программы (инженерная).

¹ предсказуемый результат некоторой деятельности

Научные исследования ведутся с целью обогащения наших знаний новыми знаниями об исследуемом *объекте* и его *свойствах*. Научные исследования делятся на две категории – *фундаментальные* и *прикладные*. Фундаментальные исследования направлены на решения принципиальных вопросов, например, есть ли жизнь на Марсе, или $P = NP$. Признаком фундаментальной проблемы является ее постановка в виде вопроса, на который должен быть получен ответ – “Да” или “Нет”. Известна метафора о прикладных исследованиях: “Из лука стреляем в стену и вокруг места попадания рисуем мишень”.

Прикладные исследования направлены на выработку подходов, методов и методик решения конкретных технических задач. Примерами выступают все задачи Теории изобретательства и рационализации (ТРИЗ) [?].

/метод, методика подход.../

Таким образом исследовательская цель записывается в следующем виде: “Целью [ВКР] является исследование [такого-то] объекта и поиск в нем [таких-то] свойств/зависимостей”.

Разработка программ направлена на автоматизацию обработки информации, что на самом деле значит, что происходит процесс частичной реализации функций (задач, последовательности действий), выполняемых вручную, при помощи методов обработки информации. Таким образом цель программистской ВКР записывается как фраза о проектировании и разработке программы или программной системы, автоматизирующей какие-либо рутинные процессы.

Записываем цель ВКР перед списком задач. Следующий этап – перечисление требований к используемым методам исследования и технологиям. Требования возникают как ограничения, накладываемые на любую деятельность. Например, зарабатывать состояние необходимо в условиях соблюдения законодательства страны, или, если программа разрабатывается под заказ, то всегда эту программу необходимо интегрировать с существующими у заказчика программными системами и форматами данных. Вот эти ограничения с краткими комментариями необходимо перечислить сразу после списка задач.

Пример списка требований для программной системы

К программной системе предъявлены следующие требования:

- База данных должна быть реализована при помощи Microsoft SQL Server версии 2012;
- Обеспечить обмен данными с существующими программными системами при помощи технологии RESTful;

- Интерфейс пользователя реализовать в виде Интернет-приложения;
- Обеспечить доступ к приложению из любой точки мира.

Перечисленные требования отражают сетевую инфраструктуру предприятия–заказчика.

Для исследования, например, список выглядит следующим образом:

В процессе исследования рекомендован к использованию следующий перечень подходов и методов:

- Использовать численные методы для решения линейных дифференциальных уравнений, реализуемых на вычислительных кластерах;
- ...

[комментарий, объясняющий]

Если объем ВКР предполагается большим, то принято помещать пару абзацев, излагающих общую структуру ВКР, в конце введения.

Квалификационная работа состоит из Глав 1, где в п. 1.1 приводятся определения теории ..., в п. 1.2 представлен литературный обзор, и приводятся основные свойства [объекта исследования]. В п. 1.3 представлен предмет исследования в виде гипотезы [о наличии в системе интересующего нас свойства]. ... В главе 2 представлены собственные эксперименты, а именно в п. 2.1. ...Третья глава посвящена вопросам реализации разработанного подхода на ЭВМ (в виде программной системы).

...

Про введение в структуре ничего писать не надо, так как оно уже прочитано. А мы ценим время читателя, которым также является и ваш рецензент.

Если к этому моменту или в процессе написания разделов появляются новые мысли, то желательно их записывать на бумажки. Как только вы заканчиваете мысль, абзац или раздел, надо быстро перемещаться в тот раздел, куда записанные мысли должны быть помещены, ну и, собственно, записать их туда в содержательном смысле. Форме изложения не надо много внимания уделять, так как основная задача – схватить мысль (grasp) и зафиксировать ее¹. В результате, например, во введении появится пара абзацев, касающихся какому-либо аспекту разработки, но носящие достаточно общий вид.

¹ Этот абзац как раз появился благодаря такой мысли и реализации этого принципа

1.2.1. Объект и предмет исследования

К этому моменту у вас в мозгу уже должно появиться некоторое понимание того, с каким объектом исследования или автоматизации вы имеете дело. *Объект исследования* – часть предметной области, на который или на часть которого направлена ваше деятельность. Например, в задаче определения наличие жизни на Марсе объектом исследования является Марс; в задаче автоматизации бухгалтерской деятельности – автоматизируемое предприятие. Как правило, в качестве объекта исследования/автоматизации выступает физический или идеальный объект, физический или идеальный процесс. Пример объекта – процесса – это автоматизация химического производства без привязки к конкретному предприятию.

Предмет исследования – часть объекта исследования, модель которой строится¹. В задаче с Марсом предмет исследования – наличие различных признаков жизни, например, аминокислот в почве, наличие воды и т.д., т.е. необходимых условий существования жизни. В задаче бухгалтерского учета предмет автоматизации – учет материальных и денежных средств предприятия.

Объект и предмет исследования/автоматизации представляются в тексте одним абзацем каждый и помещаются последовательно где-то перед целью.

На этом с разделом введения можно на время закончить и перейти к разделу, текст которого вам написать проще всего. Обычно – это то, как программа пелизована, какие именно исследования были проведенный и какие результаты былиполучены, т.е. это то, что есть уже на самом деле.

1.3. Собственные эксперименты

Реализация программной системы

Проще всего данную главу² начать с описания результата: какие свойства объекта и предмета были получены, какая программная систма реализована, какие общие выводы можно сделать из результатов.

[Описание РЕЗУЛЬТАТА исследований]

Что можно сказать о программе? Да много чего. Общая мысль – про-рекламировать пользователям свою программу. Пишите текст, отвечая на следующие вопросы:

1. Для кого предназначена программа? Каким (конкретный класс) физическим или юридическим лицам использование вашей программ поз-

¹ Программа – это реализация модели вычислительного процесса, процесса обработки информации, что тоже модель.

воляет избавляться от рутинных операций, экономить материальные средства, зарабатывать больше денег, совершенствовать организационную структуру и т.п.

2. В чем состоит основная автоматизируемая функция? Сначала дается определение. Затем кратко перечисляются этапы автоматизируемого процесса, снабженные краткими комментариями о том, как эти этапы реализованы в программе.
3. Какие особенности у вашей программы по сравнению с программами-аналогами? Здесь имеет смысл выделить одну-три выдающейся характеристик, которые значительно лучше, чем у остальных¹.
4. ...

Полученный текст с перечислением достоинств программы (результатов исследований) помещаем в **конец главы ???**.

Общая идея представления результата – программной системы – состоит в том, чтобы представлять материал от общего к частному. Т.е. по тексту главы сначала описываются модели программы, например, Функциональная модель, архитектура, ER-диаграмма (модель Чена), UML-модель классов (Диаграмма классов), ..., затем переходим к программному коду компонент, и потом уже к результатам тестирования, и т.д. В принципе, такой порядок отражает жизненный цикл программного обеспечения [?]: дизайн, проектирование, реализация, тестирование, ввод в эксплуатацию, сопровождение и снятие с эксплуатации. Все эти разделы могут быть представлены в ВКР.

Общая стратегия изложения текста ВКР в главе, посвященной реализации, – от общего к частному. Сначала излагаются общие вопросы, как вышеописанный перечень характеристик, затем переходим к описанию моделей программы, затем к программному коду, тестированию, развертыванию и т.д. Такой стиль изложения удобен рецензенту: рецензент является специалистом в предметной области, основных методах решения проблем, он также знаком с информационными технологиями. Чтобы оценить вас как специалиста, он должен понять задачу и то, как вы справились с ее решением. Построение текста от общего к частному позволяет при чтении постепенно детализировать внимание рецензента. Рецензент встраивает ваш ВКР в свою систему знаний. А это удобно делать именно в этом порядке.

В начале главы надо представить читателю общую концепцию проекта вашей разработки – в двух словах что и как вы собираетесь делать. Например, «Программная система представляет собой АРМ на основе реляционной базы данных, при этом, взаимодействие строится на основе клиент-

¹ У гиков этот раздел называется features

серверных технологий.» Из примера сразу становится видно, что собой представляет программный комплекс и каким классом программных технологий будет дальнейшее изложение ограничено.

В общих вопросах рассматривается также концепция взаимодействия пользователя с программной системой, и то, какую стратегию вы выбираете для удовлетворения требований.

Интерфейс пользователя АРМа представляет собой динамическую WEB-страницу. Это позволяет решить проблему зависимости программного обеспечения от операционной системы. Взаимодействие клиентской (веб-браузер) и серверной части реализовано на основе двустороннего обмена информацией, что позволяет обеспечить обмен сообщениями между пользователями в режиме, близком к реальному времени. Для обеспечения доступа к серверной части по протоколу RESTful предусмотрен специальный модуль к WEB-серверу. ...

Стиль фраз В примере появилась важная особенность стиля изложения текста – *доказательный*. Важно не просто описать вашу программу – ее возможности и то, как она устроена – но и обосновать **каждое** вами принятое **решение**. Чтобы проверить, удалось ли выдержать такой стиль достаточно к каждой фразе, где излагается решение задать вопросы «Почему это так?» и «Зачем это нужно?». Если в тексте есть фраза отвечающая на один из этих вопросов, значит все хорошо¹.

1.3.1. Место программы в организационной структуре

Каждая программа где-то и кем-то используется для решения какой-то задачи. В рамках ВКР чаще всего в качестве задачи рассматривается автоматизация обработки информации в учреждении или предприятии. Согласно системному подходу [?] получается, что программа является функциональным блоком некоторой организационной структуры или системы бизнес-процессов [?]. Вот эту структуру и бизнес-процессы необходимо описать сразу после раздела общих вопросов.

1.3.1.1. Организационная структура

Организационная структура современных учреждений представляет собой, как правило, древовидный граф иерархического подчинения, либо

¹ Вообще чаще всего задается первый вопрос. Вопрос «Зачем?» надо задавать, чтобы оценивать уровень рационализма принятых решений.

комитетную матрицу. Чтобы представить предприятие с иерархическим подчинением в ВКР надо нарисовать граф, отражающий а) верхний уровень подчинения, б) места отделов, где находится в эксплуатации ваша программа. На диаграмме можно учесть также элементы организационной структуры связанные с вашей программой косвенно, например, в этих отделах готовят исходные данные для обработки или анализируют отчеты. то есть диаграмма должна отражать ту часть организационной структуры, которая связана с функционированием вашей программы.

Комитетную организационную структуру, которая встречается «в природе» несколько реже, удобно отображать в виде матрицы. По горизонтали располагаются, например, виды деятельности, а по вертикали – специалисты или отделы. Комитетная структура удобна, если предприятие занимается несколькими проектами одновременно. Каждый проект в матрице отражается при помощи выделенных цветами клеточек. На самом деле чисто комитетной структуры не бывает, всегда есть лица, ответственные за функционирование всего предприятия/учреждения. Внутри проектов сотрудники также выстраиваются в иерархии подчинения и ответственности. ...В матрице выделяются проекты и клетки, в которых используется разработанная программа.

Полученная диаграмма сопровождается текстом пояснения, ... В тексте в общем виде указывается, какие функции выполняет (решает задачи) программа в каждом выделенном блоке диаграммы, а также информация какого вида там преобразуется. При чтении текста к текущему моменту становится понятным, зачем и кому нужна программа. А вы в результате вашей творческой деятельности получили перечень функций, это перечень – первая абстрактная модель вашей программы – *функциональная модель*.

1.3.1.2. Функциональная модель

Функциональная модель представляет программную систему как механизм выполняющий (исполняющий) некоторый набор функций. Модель показывает **что** делает программа в более формальном виде, чем просто перечень свойств (фич). Модель может учитывать связи между входными и выходными данными функций или нет. Можно объединять функции в классы и функциональные блоки. Функциональное моделирование выполняется при помощи различных *нотаций*¹.

Перечень функций – самый простой метод моделирования, где функции

¹ Нотациями в программировании называют методы моделирования, включающие язык представления элементов моделей, правила построения корректных моделей, семантику и интерпретацию элементов моделей и композиций.

просто перечисляются одна за одной, например в виде обобщенной таблицы спецификаций.

IDEF0 на основе SADT. IDEF0 – это нотация иерархического представления организационной структуры сложного объекта (предприятия, прибора и т.п.), преобразующего некоторый «вход» в «выход» при помощи «механизма» в условиях некоторых «ограничений» и под воздействием «управления» (см. рис. ??). Популярной методикой анализа сложного объекта и построения иерархии IDEF0-диаграмм является *SADT* (Structural Analysis and Design Technique), «методика структурного анализа и дизайна».

Диаграмма UML UseCase используется для моделирования программного обеспечения как граф, отражающий взаимоотношения между функциями системы и агентами, которые пользуются этими функциями. В диаграмме можно представлять наследование свойств объектов и декомпозицию функций (рис. ??).

ARIS Европейский проект по расширению нотации IDEF0 в направлении поддержки состояний (а не только структур). Методики проектирования, основанные на ARIS, применяются, например, в SAP/R3-системах автоматизации производства.

BPMN2.0 Нотация называется Business Process Modelling Notation, нотация для моделирования бизнес-процессов. В настоящее время едва ли не самый выразительный подход к моделированию. Позволяет представлять как декомпозицию структуры автоматизируемого объекта, агентов, документы, потоки информации, события и состояния, но также и альтернативные пути реализации функций, задание и обработку исключений, порождение подпроцессов и т.д. Существуют даже системы интерпретации таких моделей в виде приложений.

В данном пособии используем классический подход к функциональному моделированию – методику SADT. Согласно этой методике автоматизация – это процесс преобразования организационной структуры предприятия. Это процесс называется *реинжиниринг бизнес-процессов*. Модель IDEF0 представляет собой иерархию блоков, каждый из которых представляет процесс (функцию преобразования входа в выход). Реинжиниринг состоит из двух этапов: анализ структуры предприятия и выявление тех его процессов, автоматизация которых позволит улучшить качественные (упрощение организационной структуры) или количественные характеристики (производительность труда). В результате первого этапа создается модель

(иерархический набор диаграмм), называемая «As Is», т.е. «Как есть сейчас». Второй этап – преобразование результата предыдущего этапа в модель «To Be». Декомпозиция проводится при помощи программ BpWin [?] или **/как ее там, которая на яве ;-)[?].../**.

При построении модели «As is» сначала выбирается сотрудник организации, с точки зрения которого будет производиться декомпозиция бизнес-процессов. Таким сотрудником может быть директор, бухгалтер, прораб и др. Далее рассматривается основная функция предприятия, что она входе этой функции (деньги, люди, документы, заказы), что на выходе (деньги, люди, документы, отчеты), какие механизмы реализуют эту функцию (сотрудники, приборы, программное обеспечение), какие ограничения накладываются на реализацию функции (законодательство Российской Федерации, стандарты отрасли). Далее производится рекурсивная декомпозиция основной функции предприятия до уровня, где автоматизируемая вами функция представляет собой блок со входом, выходом, механизмами и ограничениями. В результате выполнения декомпозиции вы получаете

- модель предприятия с различными уровнями абстракции представления ее деятельности;
- понимание того, какие объекты на входе и какие на выходе, т.е. информацию об интерфейсах вашей программы;
- перечень агентов (сотрудников), которые будут работать с вашей программной системой;
- перечень программ, вовлеченных в реализацию функции;
- информацию об источниках стандартизованных процедур, используемых в реализации функций программы.

Согласитесь, стоит потратить один день – провести декомпозицию – для того, чтобы всю эту информацию получить.

Модель «To be» – это декомпозиция автоматизируемого блока на один или два дополнительных уровня с учетом реализации автоматизации. Для построения этой модели может быть придется декомпонировать блок в модели «As is», чтобы выявить автоматизируемые функции и информационные связи между ними.

Теперь остается все это оформить и описать в виде текста. Файл с декомпозицией поместить на сервер (Dropbox или Github), чтобы специалисты могли получить доступ к вашим результатам, а в текст ВКР обучаемо помещается в виде рисунка верхний уровень (функция организации) «As

is», модель A0, уровень, на котором выявлена автоматизируемая функция, а также декомпозиции «As is» и «To be» этой функции. К каждому рисунку пишете по одному абзацу комментариев, где кратко описывается бизнес процесс, например, как процесс преобразования входа (в основной блок) разными функциями в разные выходы, и, в конечном счете, в выход основного блока. Затем комментируете использование механизмов и ограничений. Интересные конструкции, например, если выход одного блока превращается в управление/ограничение другого, или вообще в механизм, то этот процесс хорошо рассмотреть отдельно и подробно.

1.3.1.3. Диаграмма прецедентов

На основе полученной IDEF0- функциональной модели уже можно очень быстро построить UML диаграмму прецедентов. Что она может содержательно¹ дополнить – так это указать как агенты, процессы и ресурсы наследуют свойства между собой. Например, в нотариальной конторе сам нотариус может выполнять функции секретаря, т.е. готовить тексты документов. Поэтому нотариус – это частный случай секретаря, наследует все его свойства. Этот факт фиксируется стрелкой с пустым треугольником на конце (рис. ??). Аналогично можно классифицировать ресурсы.

С диаграммой прецедентов долго возиться не надо. Просто рисуем агентов, ресурсы, функции, ставим стрелки. Надо только внимательно относиться к функциям – их следует организовать в цепочки или деревья, выделяя общие части в отдельный эллипс. Задача Use Case – диаграммы – классифицировать повторяющиеся функции разных бизнес-процессов, выявить общие части, назначить роли агентам.

После рисунка диаграммы пишем комментирующий абзац, где в вербальной форме, например, описываются роли агентов (пользователей).

1.3.2. Архитектура программной системы

Итак, вы описали функциональные модели вашей программной системы, теперь понятно **что делает ваша система**, и это организационный аспект моделирования. Теперь надо переходить к тому, как она устроена, т.е. к инженерно-организационному аспекту моделирования.

На Википедии дано следующее определение архитектуре [?] *Архитектура программного обеспечения* (англ. *software architecture*) – совокупность важнейших решений об организации программной системы. Архитектура включает:

¹ Добавить конкретной новой (неповторяющейся) осмысленной информации.

- выбор структурных элементов и их интерфейсов, при помощи которых сконструирована система, а также задание их поведения в рамках сотрудничества друг с другом;
- соединение выбранных элементов структуры и поведения во более крупные системы;
- архитектурный стиль, который задает принципиально всю организацию программной системы – все элементы, их интерфейсы и их соединение и взаимодействие.

Документирование архитектуры программного обеспечения упрощает процесс коммуникации между разработчиками, позволяет зафиксировать принятые проектные решения и предоставить информацию о них эксплуатационному персоналу системы, повторно использовать компоненты и шаблоны проекта в других задачах.

Общепринятого определения «архитектуры программного обеспечения» не существует. Так, сайт Software Engineering Institute приводит более 150 определений этого понятия.

На каком-то этапе разработки вашей программной системы вы приняли концептуальное решение, например, реализовать базу данных на сервере PostgreSQL. Что это означает? Это означает, что взаимодействие ядра системы с БД будет вестись по правилам «Клиент-сервер», где в качестве клиента выступает ядро, а сервера – сервер БД PostgreSQL. Ядро с целью получения кортежей данных делает запросы к серверу, а он, в свою очередь, реализует загрузку кортежей с дисковой памяти, их фильтрацию, а также другие преобразования кортежей, заканчивающиеся пересылкой данных ядру. В качестве сервера и клиента, например, выступают веб-сервер Apache и Google Chrome, соответственно.

В тексте ВКР под описание архитектуры выделяется раздел в главе «Реализация ...», либо в конце главы «Теоретические основы ...». Сначала идет абзац с общей идеей, вложенной в архитектуру, например, что она «клиент–серверная». Затем следует рисунок архитектуры. На рисунке при помощи прямоугольников и стрелок рисуется система взаимодействия между функциональными блоками, агентами и ресурсами.

Клиент и сервер взаимодействуют друг с другом на основе некоторых заданных разработчиком правил. Эти правила называются *протоколом*. Нарисовав архитектуру, описываем ее функциональные блоки и протоколы взаимодействия.

Предположим, что мы разрабатываем транслятор языка управления каким-нибудь АРМом, использующем базу данных, хранящуюся на сервере.

...Архитектура системы состоит из ... функциональных блоков. Пользователь при помощи «Интерфейса пользователя» (функциональный блок) вводит запросы в систему. Запросы в виде строки (протокол) передаются на вход [функционального] блока «Лексический анализатор», где входная строка преобразуется в поток лексем с соответствующими семантическими атрибутами. Лексемы подаются на вход Синтаксического анализатора (можно писать и не с большой буквы), который создает дерево синтаксического разбора, интерпретируемое в ...

Затем пишем, что там дальше происходит, как осуществляется запрос к БД: какие блоки в этом процессе участвуют, какие данные передают и в какой форме. Отслеживаем цепочку обратно. Читателю ведь интересно, как результат запроса к БД преобразуется и выдается пользователю.

Далее добавляем пару предложений, обосновывающих эту модель, что она то, что нужно и придумать что-то лучше можно, но не особо нужно (ничего особенного новая модель не добавит).

1.3.3. Обоснование выбора технологий реализации

Следующим решением, от которого зависит то, как реализована программная система, является выбор программных технологий, при помощи которых все функциональные блоки создаются и обеспечивается их взаимодействие. В качестве информационных технологий выступают:

- аппаратное обеспечение рабочих станций, серверов, сетей, маршрутизаторов и т.д.;
- операционные системы;
- прошивки ПЗУ устройств;
- структуры данных – форматы данных, теоретико-множественные представления структур данных программы (графы, категории);
- библиотеки и готовые системы (frameworks);
- Сервера баз данных, сторонние программы и их интерфейсы, а также отдельные функциональные блоки и динамические библиотеки;
- программные платформы (Microsoft, Oracle, Linux/Apache/Mysql/Perl);
- языки программирования;

- трансляторы – компиляторы, интерпретаторы, виртуальные машины;
- инструментальные средства разработки (IDE).

Все перечисленное является кирпичами и инструментами, при помощи которых создается программная система, реализуется ее архитектура и функции в виде конкретного приложения.

В начале раздела используемые технологии просто можно перечислить с небольшими комментариями.

Программная система реализована при помощи следующих информационных технологий:

1. база данных создана на основе сервера PostgreSQL версии 9.1;
2. ...
3. интерфейс пользователя реализован в HTML с использованием фреймворка /**AngularJS...**/;
4. сервер приложения разработан на основе платформы Node.js и библиотеки ExpressJS для реализации интерфейса и передачи данных по протоколу RESTful на сервер;
5. ...

Добавляем абзац с общим обоснованием и самокритикой этого совокупного набора технологий. Сопоставляем с задачами и требованиями из раздела ВКР «Введение». Только подробно по пунктам это делать не надо, иначе читатель подумает, что издеваетесь над ним. Достаточно в общем охарактеризовать, обратив внимание на что-то действительно важное, например на основную задачу и самое «занятное и замысловатое» требование.

Далее в разделе под подзаголовками третьего уровня просто описываем основные функции, архитектурные особенности, протоколы, инструментарий каждой (или основных) из перечисленных технологий, обращая внимание в тексте на то как замечательно данная технология будет решать ваши задачи и удовлетворять требования.

1.3.4. Информационная модель

Н. Вирт дал такое определение программе:

Алгоритм + Структуры данных = Программа

При чтении чужой программы сначала удобно изучить ее структуры данных, а потом уже алгоритмы и их реализации. Думаю не ошибусь, если скажу, что именно структуры данных определяют общий вид программы. Поэтому качество моделирования структур данных значительно влияют на качество всего программного проекта.

В ВКР подлежат описанию структуры данных следующих классов:

1. Внутренние структуры данных, например, представления прафов, деревьев, списков, множеств и т.д.;
2. То же самое для внешних структур данных, данных, сохраняемых на жестких носителях;
3. Протоколы обмена между функциональными блоками по интерфейсам.

Внутренние структуры данных, как правило, делятся на два класса – простые, которые не разделяются на другие структуры данных, и, соответственно, сложные. К простым относятся строки массивы, матрицы и т.п. Может случиться так, что внутри простой структуры данных представлена сложная, например, в матрице можно закодировать граф. В этом случае простая структура описывается так, как будто бы она сложная.

Самым простым методом описания простых структур данных является таблица спецификаций. Таблица состоит из столбцов, выражающих расположение структуры в модулях программы, ее название, тип, размер, краткое описание назначения, диапазон допустимых значений, комментариев. Таблица ?? представляет собой пример таблицы спецификаций для модуля нахождения корней системы линейных уравнений.

Таблица 1.1

Таблица спецификаций модуля решения системы линейных уравнений

Сложные структуры данных удобно описывать с примера конкретного типичного примера (рис. 1.1). На рисунке при помощи разных фигур обозначаются разные конструктивные элементы: экземпляры классов (class), структур (struct), объединений (union). Затем, в тексте даем этим конструк-

Рис. 1.1. Структура представления формулы (пример представления структуры данных).

тивным элементам вербальное (словесное) описание, либо при помощи вставок программного кода. Графическими нотациями вида UML – Диаграмма

классов удобно представлять взаимодействие на уровне классов, структур и экземпляров классов, имеющих определенную специфику. На этой диаграмме экземплярами, например, представляются общие ресурсы – соединения с базами данных, мьютексы, разделяемая оперативная память и т.д. Одним из удобных моментов использования Class Diagram является возможность задания стереотипов (например, «interface») классам, что дает дополнительные выразительные возможности, позволяющие представлять блоками не только классы, но и структуры, объединения, перечисления, экземпляры, динамические отображения. Можно использовать различные типы линий для отрисовки элементов диаграммы. Заливку и поллутона лучше не использовать, т.к. при печати и отображении на проекторе стиль наверняка будет искажен. Можно использовать стандартную штриховку, штриховку по контуру.

Скорей всего ваши структуры данных похожи на общеизвестные (деревья, графы, хэш-таблицы), для ясности можно сравнивать с одной–двумя такими структурами, выделить основные отличия. Показать, что дает это отличие.

Внешние структуры данных представляются аналогично внутренним, только надо выделить особенности, связанные с сохранением их на носителе информации. Например, из-за того, что минимальной единицей хранения на винчестере является блок в 512 байт, размер узла индекса (сбалансированного дерева) должен быть кратен размеру блока или размеру кластера (минимальной выделяемой единице в файловой системе). Это ограничение, как известно, приводит к тому, что приходится отходить от бинарных деревьев и переходить к n -арным. Узел дерева становится уже не двойкой а динамическим массивом. Это, в свою очередь, влияет на используемые алгоритмы преобразования (добавления и удаления узлов) дерева, а также на реализацию этих алгоритмов в виде программы.

В ВКР, связанных с разработкой информационных систем (ИС) и автоматизированных рабочих мест (АРМ), центральным звеном является хранилище данных, называемое базой данных. Вскорей всего ваша ИС использует стандартные подходы к организации этого хранилища – использует сервер реляционных баз данных. Для этого конкретно случая методика проектирования и описания внешних структур данных специальным образом технологически уже стандартизована. Рассмотрим подробнее эту методику.

1.3.5. Структура базы данных

Проектирование структуры реляционной базы данных (БД) начинается с выделения сущностей (что мы храним в БД) и атрибутов (как мы эти сущности характеризуем). Наиболее популярен метод «Сущность–Связь», в результате которого строится модель Чена в некоторой нотации (рис.). Процесс проектирования можно описать по шагам, по одному отношению

Рис. 1.2. Разные нотации представления модели «Сущность–Связь»

(связи). Если структура базы данных объемная, можно выделить только основные отношения, на основе которых строится большинство процедур обработки. Например, в 1С Бухгалтерии представляет основной интерес структура баз данных проводок, документов и формат данных конфигурации системы.

Следующим шагом согласно методике является построение логической и физической модели таблиц и отношений в проектируемой БД. Здесь пишем об основных принципах объединения отношений в таблицы и выбора типов данных (доменов) для атрибутов. Рисунки логической и физической моделей можно привести как оба, либо только физическую, т.к. она – содержательный результат применения метода проектирования структуры БД. В принципе можно привести программный код DDL–инструкций для создания одной–двух желательных связанных отношений таблиц.

Физическая модель не показывает всего перечня индексов полей, их уникальность, поэтому пару фраз и примеров можно вставить в текст, обосновав необходимость их явного задания.

Если проектируется программная система, использующая существующую базу данных, то ее так же можно описать как всю, так и основную часть, через которую происходит взаимодействие. Процесс аналогичен вышеописанному за исключением того факта, что БД проектировали не вы. Т.е. не надо вкладывать усилия в обоснование вами не принятых проектных решений. Бывает так, что для реализации вашей ИС необходимо достроить существующую БД. В этом случае опять все тоже самое, но необходимо явным образом выделить новую часть БД (таблицы и отношения) и существовавшие ранее таблицы, с которыми происходит взаимодействие.

Методика проектирования БД включает в себя вопросы проверки качества получаемого результата. Для этого введены три нормальные формы: первая, вторая и третья, а также форма «Бойса–Кодда». Профессиональный инженер, понимая необходимость получения объективизированной оценки качества разрабатываемого продукта, стремиться его измерить. Результаты измерения записываются в одном–двух абзацах текста. Можно

выделить даже подраздел.

Если по каким-то причинам, например, требование производительности, масштабируемости, специфики хранимой информации, не удастся выполнить все требования, то так и пишем в тексте

В базе данных при проектировании не удалось выполнить такое-то стандартное требование по такой-то причине. В результате этого от хранилища данных следует ожидать такое-то аномальное поведение. Алгоритмы такие-то будут иметь такую-то специфику и реализовываться как-то особым образом.

Типичным примером аномалии в БД является таблица с пользователями (ролями), которая, как правило, не входит во взаимодействие с другими таблицами. Так же как в примере с матрицей, внутри которой зашит граф, реляционную базу данных можно использовать не по ее типичному назначению. Бывли проекты, в которых XML-БД представлялась в виде таблиц реляционной БД. Еще один пример – хранение объектно-ориентированной структуры в реляционной БД. В этом случае приходится моделировать наследование и объединять несколько таблиц по одному общему ключевому полю.

Главное то, что не следует считать, что отклонение от постулатов метода при проектировании – это что-то плохое. Вероятно ваша задача выходит за рамки класса решаемых задач этим методом. Но, естественно, вам как будущему специалисту необходимо отдавать себе отчет о том так это или нет на самом деле. Для сложных случаев у вас есть руководитель ВКР, с которым следует обсудить все ключевые решения до защиты.

1.3.6. Протоколы обмена данными

Много проектов ВКР связаны с взаимодействием функциональных блоков через сеть в явном и неявном виде. Примерами выступают ИС, построенные по трехзвенной клиент-серверной архитектуре, сложные ИС, включающие аналитические модули из других систем. В первом случае имеем явное сетевое взаимодействие веб-браузера и сервера приложений, во втором – неявное, если все взаимодействующие модули находятся на одной рабочей станции, или даже в одном адресном пространстве процесса. Так вот протоколы обмена информации для читателей вашего ВКР представляют тоже определенный интерес. Рассмотрим пример описания протокола, используемого для доступа к системе кэширования memcached....

Система memcached является вспомогательным сервером к серверу данных, в частности, веб-серверу. Задача memcached – сохранять и обеспечивать доступ к данным, которые меняются очень редко, но не могут быть

представлены в виде статического файла. Примером таких данных является домашняя страница сайта, которая подгружает на сервере информацию о погоде или курсах валют центробанка. Эта страниц не может быть статической, но и данные для разных пользователей почти всегда не отличаются, так как подгружаемая информация меняется очень медленно. Проблему повышения производительности веб-сервера – не заставлять его на каждое соединение генерировать новую страницу – решается при помощи промежуточного слоя, который принимает соединение от веб-браузера и решает, генерировать ли страницу, или отправить уже готовую. Хранением в оперативной памяти образа сгенерированной страницы и занимается демон memcached.

Разработчики memcached приняли решение максимально упростить взаимодействие с сервером со стороны клиентов. Взаимодействие происходит через сокет (socket) при помощи байт-ориентированного протокола (см. 1.3

Рис. 1.3. *протокол взаимодействия с сервером memcached.*

/memcached example itself shining like the Sun.../

1.3.7. Стандартные форматы

В некоторых задачах решение представляется в виде программы, обрабатывающей входные данные в существующем стандартном формате. Получается, что вы как инженер приняли решение о том, что этот стандарт следует использовать, естественно на основе глубоких рассуждений о...

1.4. Алгоритмы и программные модули

/подпорки!!!!.../

2. Оформление текста

Структура ВКР состоит из основных и дополнительных разделов. К основным разделам относятся:

1. Титульный лист, на котором представлена информация формального характера:
 - вуз, в котором выполнена работа,
 - /кафедра.../,
 - название ВКР,
 - автор и научный руководитель;
2. Введение, где изл....;
3. Глава, посвященная изложению теоретических аспектов работы;
4. Глава, где рассмотрены аспекты реализации программного продукта;
5. Заключение, в котором кратко перечисляются полученные результаты, /самокритика.../ и /направления дальнейшего совершенствования.../.
6. Список используемых источников (список литературы).

2.1. Текстовые объекты

2.1.1. Формулы

Пример правильно оформленной формулы:

$$\vec{F} = -G \frac{Mm}{|r^3|} \vec{r}, \quad (2.1)$$

где \vec{F} – вектор силы тяжести, действующий на материальную точку массой m со стороны тела (материальной точки) массой M , находящегося в начале координат. Радиус-вектор \vec{r} направлен из начала координат в центр другого тела. Уравнение Ньютона (2.1) в скалярной форме выглядит в следующем виде:

$$F = G \frac{Mm}{r^2}, \quad r = |\vec{r}|, F = |\vec{F}|.$$

2.1.2. Рисунки

Рисунки в тексте отображаются в разрыве текста, по центру с нумерацией и подписью.

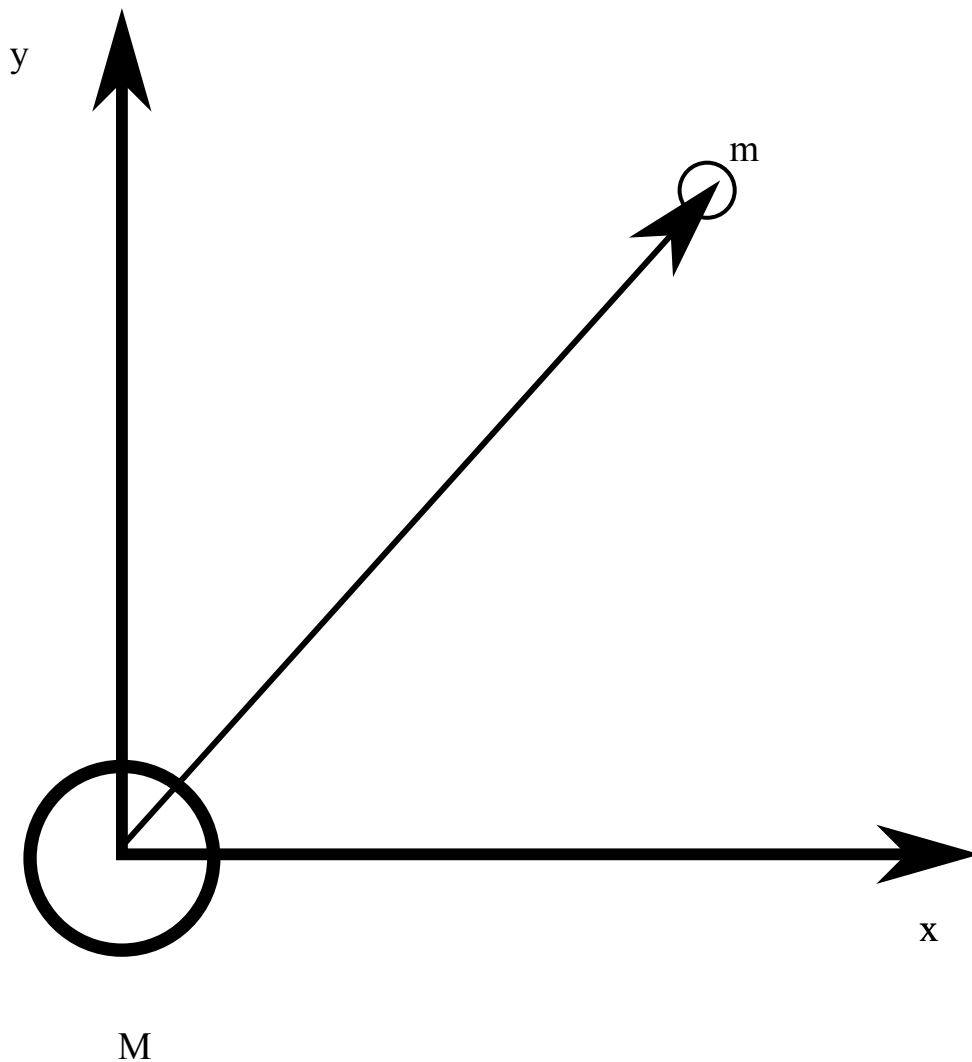


Рис. 2.1. Ньютоновская система двух тел

2.1.3. Алгоритмы

2.1.4. Тексты программ

```
/* HelloWorld.java
 */

public class HelloWorld
{
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```



```
    }  
}  
  
// Hello3.cs  
// arguments: A B C D  
using System;  
  
public class Hello3  
{  
    public static void Main(string[] args)  
    {  
        Console.WriteLine("Hello, World!");  
        Console.WriteLine("You entered the following {0} command line  
↪ arguments:",  
            args.Length );  
        for (int i=0; i < args.Length; i++)  
        {  
            Console.WriteLine("{0}", args[i]);  
        }  
    }  
}
```

3. Формальные части

4. Презентация ВКР

Презентация используется в качестве наглядного материала, сопровождающего доклад. Обычно доклад по содержанию ВКР занимает 7–10 минут. Опыт показывает, что количество слайдов равняется количеству минут доклада, или на 50% больше. Поэтому, делать презентацию размером более 15 слайдов особого смысла не имеет.

Презентация включает следующие слайды:

1. **Титульный лист**, на котором представлена тема ВКР, автор и руководитель. У руководителя указывается его преподавательская должность (доцент или проф.) или звание (доцент, проф., академик и т.д.). У автора указывается группа (ЭВМ-45 или 2344).
2. **Актуальность темы ВКР**. Актуальность представляется, например, виде текста о проблемной области: существующие нерешенные проблемы, требующие решения.
3. **Цели и задачи ВКР**. Здесь кратко на одном слайде представляется цель и задачи. Их можно скопировать из текста диплома.
4. **Организационная структура предприятия**. Данный слайд может быть помещен перед актуальностью или слайдом «Цели и задачи». На слайде (или группе слайдов) представляется результат реинжиниринга бизнес-процессов.
5. **Функциональная модель**. IDEF0/SADT, Use Case - диаграммы, BPMN2.0 и т.д.
6. **Архитектура программной системы**.
7. **Информационная модель**. Здесь, как правило, представляют внешние структуры данных, например, структуру базы данных.
8. **Структуры внутренних данных**: объекты ООП, деревья, матрицы и т.п. – типы данных, которые используются при преобразовании информации.
9. **Модели процессов**: диаграммы последовательностей (UML), блок-схемы алгоритмов, автоматы.
10. **Реализация программной системы**.
 - (a) **Интерфейсы API**.

- (b) **Иерархии классов.**
 - (c) **Куски кода**, демонстрирующие реализации алгоритмов.
 - (d) **Примеры исполнения API.**
11. **Примеры работы программы или тестирование.** Задача этого слайда – показать, что разработанная программа действительно работает.
 12. **Результаты внедрения программы** в организационную структуру предприятия или результаты научных исследований, улучшение заранее заданных характеристик “чего-либо”.
 13. **Требования к аппаратному и программному обеспечению.** На этом слайде обычно в табличном виде представляют требования минимальные и рекомендуемые.
 14. **Организационно–экономическая часть.** В технических вузах, как правило, эта часть выносится в виде результатов итоговых расчетов, включая заключения, с базовыми начальными данными.
 15. **Безопасность жизнедеятельности.** Слайд оформляется аналогично предыдущему.
 16. **Заключение.** На этом слайде представлены результаты. В конце доклада на защите ВКР этот слайд можно оставлять на экране, чтобы комиссии было удобно задавать вам вопросы.

Кроме перечисленных слайдов, в презентацию перед заключением можно вставить сканы отзывов заказчиков, документов о внедрении. Можно заготовить несколько слайдов для демонстрации ответов на вопросы комиссии (предусмотреть). Эти слайды помещаются после заключения.

4.1. Список литературы

Литература

1. Жизненный цикл программного обеспечения — Википедия. [электронный ресурс] URL: https://ru.wikipedia.org/wiki/%D0%96%D0%B8%D0%B7%D0%BD%D0%B5%D0%BD%D0%BD%D1%8B%D0%B9_%D1%86%D0%B8%D0%BA%D0%BB_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F (дата обращения: 28.04.2016)