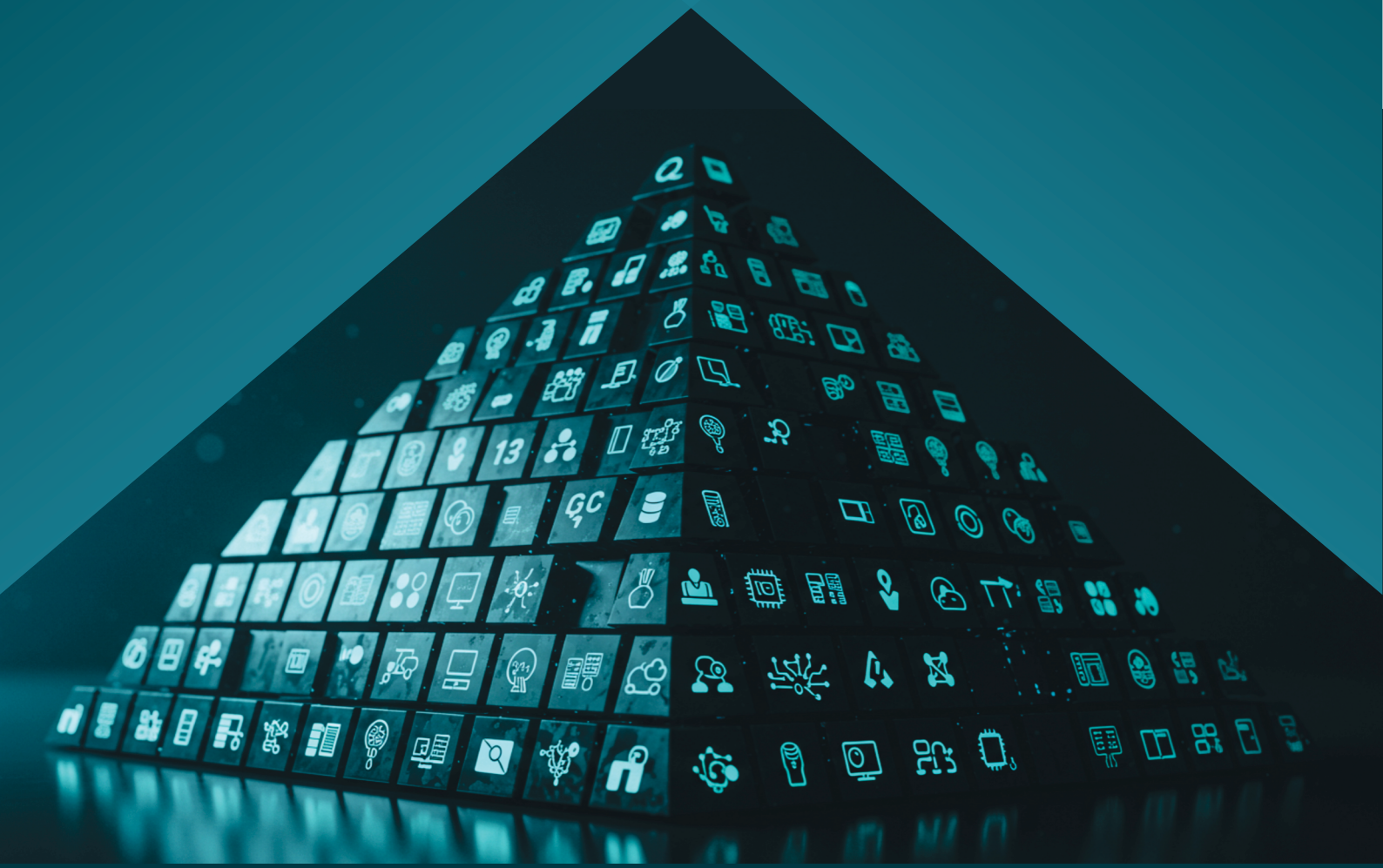
А. П. Клишин, Ф. Д. Пираков

МЕТОДЫ И СРЕДСТВА ПРОЕКТИРОВАНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ И ТЕХНОЛОГИЙ



Лабораторный практикум

МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Томский государственный педагогический университет»
(ТГПУ)

А. П. Клишин, Ф. Д. Пираков

МЕТОДЫ И СРЕДСТВА ПРОЕКТИРОВАНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ И ТЕХНОЛОГИЙ

Лабораторный практикум

Томск
Издательство ТГПУ
2025

ISBN 978-5-907791-42-8

УДК 004.415.2: 378.147.88
ББК 16я73
К49

Рекомендовано к изданию
редакционно-издательским советом
Томского государственного
педагогического университета

Рецензент:
доктор физико-математических наук, профессор кафедры общей
и экспериментальной физики Томского государственного университета
Л. В. Горчаков

Клишин, А. П.

К49 Методы и средства проектирования информационных систем и технологий : лабораторный практикум: / А. П. Клишин, Ф. Д. Пираков [Электронный ресурс] / Электрон. текстовые дан. (9,1 Mb) – Томск : Издательство Томского государственного педагогического университета, 2025. – 115 с. – 1 электрон. опт. диск (CD-ROM). – Загл. с титул. экрана.
ISBN 978-5-907791-42-8

Лабораторный практикум разработан для практического освоения программного инструментария и основных этапов технологии проектирования информационных систем. Содержит методики структурного и объектно-ориентированного анализа бизнес-процессов, моделирование данных, основы языка UML, этапы разработки технического задания на информационную систему, анализ требований и создание спецификаций, а также даны рекомендации по формированию проектной документации.

Предназначен для студентов, обучающихся по направлению подготовки 09.03.02 Информационные системы и технологии. Представленные материалы могут быть использованы студентами и преподавателями других технических и естественно-научных специальностей, а также разработчиками и пользователями информационных систем.

УДК 004.415.2:378.147.88
ББК 16я73

ISBN978-5-907791-42-8

Список сокращений и условных обозначений

АС – автоматизированная система

АИС – автоматизированная информационная система

API – прикладной интерфейс программирования

БД – база данных

ГОСТ – государственный стандарт

DFD – диаграмма потоков данных

ЕСПД – Единая система программной документации

ЕСКД – Единая система конструкторской документации

ЖЦ – жизненный цикл

ИС – информационная система

IDEF0 – методология, используемая для создания функциональной модели

IDEF1 – методология, используемая для создания информационной модели

IDEF1X – методология (расширение IDEF1), в которой представлен язык моделирования данных для разработки семантических моделей данных

IDEF2 – методология, используемая для создания динамической модели

IDEF3 – методология моделирования бизнес-процессов, которая используется для описания сценария и последовательности операций для каждого процесса

КД – контекстная диаграмма

КТС – комплекс технических средств

НМО – нормативно-методическое обеспечение

ПО – программное обеспечение

ПС – программное средство, это программа или логически связанная совокупность программ на носителях данных, снабжённая программной документацией

OSTD – метод описания переходов состояний объектов с указанием того, какие существуют в IDEF3

OSTN – диаграмма сети трансформаций состояния объекта

PFD – метод описания технологических процессов с указанием того, что происходит на каждом этапе технологического процесса в IDEF3

PFDD – диаграмма описания последовательности этапов процесса

СУБД – система управления базой данных

SADT – методология структурного моделирования

SRS – спецификация требований, формализованный документ, в котором подробно изложены все требования к информационной системе

ТЗ – техническое задание

UML – унифицированный язык моделирования

UOB – элемент поведения на схеме

УМК – учебно-методический комплекс

Содержание

Предисловие

В современных условиях цифровизации и стремительного развития информационных технологий информация становится одним из ключевых ресурсов, что делает актуальным изучение программного инструментария для проектирования и создания информационных систем в профессиональной деятельности будущих специалистов в области развития информационных технологий.

Предлагаемый лабораторный практикум позволит организовать практическую и самостоятельную работу студентов бакалавриата, обучающихся по направлению подготовки 09.03.02 Информационные системы и технологии и изучающих дисциплину «Методы и средства проектирования информационных систем и технологий».

Цель данного лабораторного практикума заключается в формировании у обучающихся практических навыков и компетенций, овладении современными методами и средствами проектирования информационных систем и технологий, а также в содействии осознанного выбора программных инструментов для решения задач, связанных с обработкой информации в контексте частичной или комплексной автоматизации в разных областях деятельности.

Представленные в лабораторном практикуме материалы и задания помогут обучающимся углубить свои знания в области проектирования и разработке информационных систем, освоить основные методы и средства проектирования, закрепить теоретические знания путем их практического применения, развить аналитическое и критическое мышление, необходимое для успешной работы в области информационных технологий. Освещены необходимые мероприятия предпроектного и проектного управления. Описаны методы разработки информационных систем, анализ бизнес-процессов, моделирование данных, основы языка UML, этапы разработки технического задания, анализ требований и создание спецификаций, даны рекомендации по формированию проектной документации. Большое внимание в работе отводится государственным стандартам, регламентирующим документам, которые позволяют на современном уровне организовать проектную деятельность. Материал включает практические задания различного уровня сложности, что позволяет учитывать уровень подготовки обучающихся при освоении дисциплины.

Практикум выполнен при поддержке Томского государственного педагогического университета в рамках научно-исследовательского проекта Е.01/2024.

Лабораторная работа № 1

Разработка технического задания

Тема занятия: подготовка технического задания для создания информационной системы.

Цель занятия: изучить государственные стандарты и освоить навыки по формированию технического задания на информационную систему.

Материалы и программное обеспечение

1. ГОСТ 34.602-2020. Техническое задание на создание автоматизированной системы.
2. ГОСТ Р 2.105-2019. Единая система конструкторской документации. Общие требования к текстовым документам.
3. ГОСТ 34.601-90. Автоматизированные системы. Стадии создания.
4. ГОСТ 24.104-2023. Автоматизированные системы управления. Общие требования.

Краткие теоретические сведения

Разработка информационной системы, как правило, начинается с разработки технического задания (ТЗ). ТЗ является важным элементом при создании информационной системы (ИС) и регламентирует как требования к системе, так и требования к этапам ее проектирования, внедрения и эксплуатации. При создании ТЗ необходимо соблюдать ГОСТ 34.602. Техническое задание на создание автоматизированной системы (АС), которое является основным документом, определяющим требования и порядок создания автоматизированной системы. В настоящее время термин «автоматизированная система» используется как информационная система и в более широком смысле.

При оформлении и форматировании текста ТЗ необходимо соблюдать требования ГОСТ Р 2.105–2019.

Разработка технического задания является важным и ответственным шагом, поскольку от правильности составления этого документа в дальнейшем будет зависеть успешность работы и может отразиться на процессе проектирования, разработки и внедрения информационной системы, так как эти этапы будут осуществляться в строгом соответствии с данным документом. От содержания ТЗ могут зависеть также результаты работы большого количества работников, отделов, подразделений и т. п. Поскольку в ТЗ при подписании указываются атрибуты двух сторон (заказчика и исполнителя), то в дальнейшем заказчик будет контролировать исполнение требований основных разделов ТЗ, и от качества проведенных исполнителем работ в соответствии с ТЗ будет зависеть окончательная приемка информационной системы заказчиком.

ТЗ разрабатывается обычно одним или несколькими специалистами на информационную систему в целом, хотя допускается разработка ТЗ на отдельные модули (подсистемы) системы в случае необходимости.

ТЗ разрабатывается на основании исследований и проведенного анализа предметной области (объектов автоматизации) по результатам проведенного об-

следования. В результате формируется итоговый отчет, который содержит совокупность диаграмм, таблиц и описаний основных процессов, подлежащих автоматизации.

Техническое задание по ГОСТ 34.602-2020 на АС содержит следующие основные разделы:

1. Общие сведения.
2. Назначение и цели создания системы.
3. Характеристика объектов автоматизации.
4. Требования к системе.
5. Состав и содержание работ по созданию системы.
6. Порядок контроля и приемки системы.
7. Требования к составу и содержанию работ по подготовке объекта автоматизации и вводу системы в действие.
8. Требования к документированию.
9. Источники разработки.

Раздел 1. Общие сведения

В разделе «Общие сведения» требуется предоставить следующие сведения:

- полное наименование системы и ее условное обозначение;
- наименование предприятий заказчика и разработчика;
- перечень документов, на основании которых создается система;
- плановые сроки начала и окончания выполнения работ.

В ТЗ включаются, как правило, документы, на основании которых начинается разработка системы. Примерами таких документов могут быть представления, акты обследования, приказы, планы стратегического развития предприятия и т. д.

Раздел 2. Назначение и цели создания системы

Данный раздел является одним из наиболее важных и обязательных в любом ТЗ. Как правило, в разделе приводится название процесса, который автоматизируется и для автоматизации которого предназначается данная система, и области ее применения или объекты, где предполагается ее использование. Например, если создается автоматизированная система для школы, то следует отразить процесс, который будет автоматизирован, а также регистрацию и учет движения контингента учащихся.

Цель создания системы – достижение каких-либо положительных показателей (технических, технологических, финансовых и т. п.). В этом разделе можно привести список основных задач, которые нужно выполнить для успешного создания системы. Основное требование в данном разделе – указать критерий оценки достижения цели. Это требуется для возможного контроля эффективности создаваемой системы, в противном случае целевые показатели не могут быть адекватными индикаторами внедрения системы для заказчика.

Раздел 3. Характеристика объекта автоматизации

Приводится описание объекта автоматизации или автоматизируемого процесса. Все описывается так, как есть на текущий момент. Указываются инструкции, в соответствии с которыми выполняется данный процесс или работает объект автоматизации.

Приводятся сведения об условиях эксплуатации объекта. В этот же раздел можно включить схему информационных потоков, связанных с данной системой, и организационную структуру. Для более качественного описания объекта автоматизации следует включить в ТЗ описание предметной области посредством методологии IDEF0.

Раздел 4. Требования к системе

Наиболее нагруженный и требующий знаний предметной области раздел ТЗ. При установке требований к системе рекомендуется принимать во внимание указания ГОСТ 24.104–2023.

Раздел «Требования к системе» состоит из трех подразделов:

– требования к системе в целом;

– требования к функциям (задачам), выполняемым системой;

– требования к видам обеспечения.

Подраздел 4.1. Требования к системе в целом.

В подразделе следует обратить внимание на следующие моменты:

1. Архитектура автоматизированной системы: определить тип архитектуры и указать, какие компоненты будут разрабатываться.

2. Способ взаимодействия компонент между собой: определить тип взаимодействия (проводной, беспроводной), стек протоколов передачи данных, стандартные интерфейсы (электрические, программные).

3. Необходимость взаимодействия с другими системами: при необходимости указать способ взаимодействия, протокол передачи данных, стандартные интерфейсы (электрические, программные).

4. Требования к надежности: указываются конкретные параметры и их величины.

5. Требования к персоналу: указывается количество и квалификация.

6. Требования к режимам функционирования: указывают вариант(ты) использования системы (круглосуточный, сменный, эпизодический), а также, если необходимо, специальные режимы (диагностика, минимальный функциональный набор).

7. Требования по защите информации: указывают ссылки на действующую нормативно-правовую базу и критерии, необходимые для достижения требуемых показателей.

Подраздел 4.2. Требования к функциям(задачам), выполняемым системой.

Необходимо четко прописать состав функций будущей системы, необходимый для достижения поставленной цели. Функции должны коррелировать с назначением и задачами, определенными во втором разделе технического задания.

В описание функции включают:

– временной регламент выполнения функции (например, время выполнения не более 1 с);

– требования к качеству реализации каждой функции (на пример, в поисковых системах релевантность);

– требования к форме представления выходной и входной информации (формат данных на входе файл MS Excel, интерфейс RS-485 на выходе, конкретные формы, нужные для пользователя);

– достоверность результатов (например, при передаче – коэффициент ошибок, количество знаков после запятой).

Подраздел 4.3. Требования к видам обеспечения.

Обязательно указывают требования к следующим видам обеспечения:

– информационному;

– программному;

– техническому;

– эргономическому;

– методическому.

В требованиях к информационному обеспечению определяют:

– состав, структуру и способы организации данных в системе (структура базы данных – ER модель);

– информационный обмен между компонентами системы (метод доступа к данным, который связан с архитектурой АС);

– информационную совместимость со смежными системами (совокупность интерфейсов и стек протоколов, который поддерживает система);

– унификацию и стандартизацию системы (ISO, ГОСТ, кодовые таблицы);

– структуру процесса сбора, обработки и передачи данных в системе и представление данных;

– защиту данных в системе (обеспечение целостности на уровне СУБД, бесперебойное энергоснабжение);

– контроль, хранение и восстановление данных (резервное копирование и репликация данных).

В требованиях к программному обеспечению определяют:

– требования к системному программному обеспечению (версии операционной системы, СУБД, утилиты, драйверы, BIOS);

– требования к прикладному программному обеспечению (приложения пользователя);

– общие требования – функциональная полнота, модульность, представлены на языках высокого уровня;

– дополнительно требования в соответствии ГОСТ 24.104–2023.

В требованиях к техническому обеспечению указывают характеристики аппаратных средств и вычислительной техники, которые будут использоваться в системе (покупные) или создаваться в рамках проекта. К техническим средствам обеспечения относятся: серверы, рабочие станции, терминалы ввода-вывода, различные адаптеры, периферийное оборудование, элементы систем видеоконтроля, контроля доступа, оборудование компьютерных и телекоммуникационных сетей. Под характеристиками подразумевают пропускную способность, количество портов, производительность, объем памяти, тактовую частоту, потребляемую мощность, напряжение питания и т. д.

В требованиях к эргономическому обеспечению называются ссылки на нормативные документы в области охраны труда и санитарных норм, выполнение которых обязательно при проектировании системы.

В требованиях к методическому обеспечению отмечается совокупность необходимых инструкций по эксплуатации, использованию, диагностике и ремонту автоматизированной системы.

Дополнительно указываются требования к обеспечению:

– метрологическое – целесообразно указывать только для информационно-измерительных систем;

– лингвистическое – требования к языкам программирования, применяемым для кодирования ПО;

– математическое – необходимо, если система выполняет расчеты параметров по определенным алгоритмам (электронная подпись, шифрование).

Раздел 5. Состав и содержание работ по созданию систем

Представлены календарные сроки и виды работ, выполнение которых осуществляется в рамках определенных временных интервалов, что крайне важно для контроля исполнения проекта и степени готовности АС. Раздел обычно оформляется в виде таблицы, где указываются сроки и работы.

Раздел 6. Порядок контроля и приемки системы

Определяются способы проверки соответствия внедряемой системы требованиям технического задания. Раздел содержит описание видов испытаний, необходимых для контроля соответствия. При написании раздела руководствуются требованиями ГОСТ Р 59792-2021. Виды испытаний АС.

Для АС устанавливают следующие основные виды испытаний:

– предварительные (автономные или комплексные);

– опытная эксплуатация;

– приемочные испытания.

При необходимости допускается проведение дополнительных видов испытаний АС и их частей. Для планирования проведения всех видов испытаний разрабатывают документ «Программа и методика испытаний».

Предварительные испытания могут быть автономные или комплексные. *Автономные испытания* охватывают части АС. Их проводят по мере готовности частей АС к сдаче в опытную эксплуатацию.

Комплексные испытания проводят для групп, взаимосвязанных частей АС или для АС в целом. Испытания проводят для определения ее работоспособности и решения вопроса о возможности приемки АС в опытную эксплуатацию.

Опытную эксплуатацию АС проводят с целью определения фактических значений количественных и качественных характеристик АС и готовности персонала к работе в условиях функционирования АС, определения фактической эффективности АС, корректировки (при необходимости) документации.

Приемочные испытания АС проводят для определения соответствия АС техническому заданию и решения вопроса о возможности приемки АС в постоянную эксплуатацию.

Все испытания закрываются актами.

Раздел 7. Требования к составу и содержанию работ по подготовке объекта автоматизации к вводу в действие

Раздел необходим для грамотной организации работы как исполнителя, так и заказчика на стадии ввода в действие. В разделе указывают:

– необходимые строительно-монтажные и пусконаладочные работы на объекте автоматизации (инженерные сети, дополнительные перегородки, монтаж оборудования);

– необходимые работы по управлению персоналом заказчика, обучение, введение новых рабочих должностей;

– работы по вводу данных в автоматизированную систему и адаптации программного обеспечения.

Важно в техническом задании четко прописать ответственность сторон за выполнение вышеперечисленных работ.

Раздел 8. Требования к документированию

Указывается, какой комплект документации следует исполнителю передать заказчику.

Раздел 9. Источники разработки

Перечисляются все документы, отчеты, научно-исследовательские работы, используемые для разработки технического задания и применяемые для разработки автоматизируемой системы.

Пример. ТЗ на разработку обучающего сайта «EngLang». Рисунки, схемы, и таблицы представлены в документе, находящемся в приложенном файле, который находится в папке учебно-методического комплекса (УМК).

Техническое задание

1. Общие требования.

1.1. Наименование системы: обучающий веб-сайт «EngLang».

1.2. Разработчик: Иванова Полина Геннадьевна, студентка 403 группы, четвёртого курса, физико-математического факультета, Томского государственного педагогического университета.

1.3. Основание для разработки: изучение применения веб-технологий для обучения английскому языку и получение навыка разработки обучающего веб-сайта, а также закрепление теоретических знаний и расширение кругозора в указанной области.

1.4. Сроки выполнения работ: 01.01.2025–01.04.2025.

2. Назначение и цели создания системы.

2.1. Назначение: предоставление пользователю доступа к различным курсам и материалам для обучения.

2.2. Цель создания системы: сокращение времени поиска теоретических и практических материалов для самостоятельного обучения.

2.3. Задачи и функции:

– разделы избранного для пользователя;

– наличие заданий для практики;

– возможность посмотреть результаты прохождения тестов;

– обратная связь.

3. Характеристики программного продукта.

3.1. Условия эксплуатации: работа в диапазоне комфортной для человека температуры, от 10 °C до 28 °C. Рекомендуемая влажность составляет от 40% до 60%; непрерывное электропитание и защита от скачков напряжения/перебоев;

использование антивирусов, брандмауэров и регулярное обновление программного обеспечения; проведение регулярных резервных копий данных.

3.2. Схема информационных потоков. Концептуально условный пользователь взаимодействует с веб-сайтом, давая ему свои данные, которые заносятся в хранилище, а взамен получает учебные материалы, хранящиеся в том же хранилище данных. Более подробный поток данных изображен на рисунках 5–6.

3.3. Описание предметной области. Процесс взаимодействия с веб-сайтом на входе имеет персональные данные пользователя, включающие в себя данные профиля, информацию об уровне навыков студентов. Процесс управляется инструкциями и правилами, которые обучающиеся должны соблюдать при обучении, например, сроки выполнения заданий, правила поведения на платформе обучения. Механизмом, с помощью которого реализуется процесс, является устройство для работы, например телефон или ПК, и браузер. На выходе получаем оценки и результаты обучения.

4. Требования к системе:

4.1. Требования к системе в целом:

4.1.1. Архитектура автоматизированной системы: веб-сайт состоит из восьми страниц: Главная, Профиль, Темы, Уроки, Учебники, Задания, Слова, Авторизация.

4.1.2. Способ взаимодействия компонент между собой: тип взаимодействия беспроводной; стек протоколов: HTTP, TCP, IP.

4.1.3. Требования к надёжности:

– время доступности (uptime) – высокая степень доступности веб-сайта, не менее 99,9%;

– своевременное обновление программного обеспечения и техническая поддержка веб-сайта для обеспечения его надежной работы;

– регулярное тестирование веб-сайта на предмет надежности и мониторинг его работы для своевременного выявления проблем.

4.1.4. Требования по защите информации: Федеральный закон «О персональных данных» от 27.07.2006 N 152-ФЗ (последняя редакция).

4.2. Требования к функциям:

4.2.1. Авторизация и регистрация. Время выполнения не более 1 с. Данные форм авторизации и регистрации должны проходить валидацию. Данные хранятся в таблице, пароль должен быть записан в хешированной форме.

4.2.2. Представление учебного контента. Время выполнения не более 1 с. Представление учебных материалов и контента должно быть структурировано и организовано в логическом порядке, чтобы пользователи могли легко найти нужную информацию.

4.2.3. Упражнения и тесты. Время выполнения не более 1 с. Интерактивные упражнения и тесты должны правильно отображаться и иметь пояснения к заданию и ответу. Результаты хранятся в БД.

4.2.4. Доступ к результатам. Время выполнения не более 1 с. Пользователь должен иметь возможность посмотреть результаты выполнения тестов, причём только свои, у себя в профиле.

4.2.5. Комментирование. Время выполнения не более 1 с. Пользователи должны иметь возможность оставлять комментарии, задавать вопросы и получать

обратную связь от преподавателей или других участников курса. Комментарии хранятся в БД.

4.3. Требования к видам обеспечения:

4.3.1. Информационное обеспечение:

– Состав и структура данных в системе. База данных веб-сайта “EngLang” состоит из 11 таблиц: User, Word, Book, My Dictionary, My Bookshelf, Comment, Quiz, Question, Answer, Choise, Result.

– Структура процесса сбора, обработки и передачи данных в системе:

Сбор данных: на этом этапе система собирает данные из БД и форм обратной связи. Данные представлены в текстовом формате.

Обработка данных: собранные данные проходят через процесс обработки, который включает в себя очистку данных, структурирование для подготовки данных к представлению.

Хранение данных: обработанные данные хранятся в базе данных для последующего доступа и использования.

Представление данных: система представляет обработанные и хранимые данные на веб-сайте в удобном для пользователя виде, например, в виде таблиц, графиков, диаграмм, отчетов и т. д.

Передача данных: для обеспечения доступа к данным на веб-сайте, данные передаются через сеть Интернет пользователям с помощью вышеуказанных протоколов и технологий.

4.3.2. Программное обеспечение:

– Требования к системному ПО: рекомендуется использование операционной системы Windows версии не ниже 7. СУБД: SQLite. Браузер: Google Chrome, Mozilla Firefox, Opera, Safari и другие.

4.3.3. Техническое обеспечение:

– Персональный компьютер: ноутбук или стационарный компьютер с достаточными ресурсами (процессор (8 ядер (16 логических потоков), частота – 3,5 ГГц и больше), оперативная память (16 Гб и больше), хранилище (HDD – 128 Гб, SSD – 256 Гб)) для работы с графическими программами, текстовыми редакторами и средами разработки.

4.3.4. Эргономическое обеспечение:

– Федеральный закон от 28 декабря 2013 г. № 426-ФЗ «О специальной оценке условий труда»;

– ГОСТ 12.2.003-91 «Системы обеспечения безопасности труда. Организация обучения. Общие требования».

4.3.5. Методическое обеспечение: программная документация:

5. Состав и содержание работ:

5.1. Планирование и анализ. Планирование и анализ веб-сайта включают в себя определение целей и требований, предъявляемых к веб-сайту, проектирование структуры веб-сайта и его функционала, изучение целевой аудитории и её потребностей.

5.2. Дизайн. На этапе дизайна происходит создание визуального концепта и макетов (wireframes) веб-сайта. Осуществляется выбор цветовой гаммы, шрифтов, лого и других визуальных элементов.

5.3. Разработка. Разработка несёт в себе создание HTML-разметки и CSS-стилей для оформления веб-сайта и проработку клиентской и серверной логики с использованием языков программирования, таких как JavaScript, PHP, Python и др., а также интеграцию базы данных веб-сайта.

5.4. Тестирование. Тестирование является проверкой функциональности и совместимости веб-сайта на различных платформах и браузерах. На данном этапе происходит выявление и исправление ошибок и недочетов.

5.5. Развёртывание и оптимизация. Развертывание и оптимизация подразумевают выбор и настройку хостинга для размещения веб-сайта, его публикацию на сервере и оптимизацию производительности, а также SEO-оптимизацию веб-сайта.

5.6. Поддержка и обновление. Последний этап поддержки и обновления означает регулярное обновление контента и функциональности веб-сайта, мониторинг работы веб-сайта, резервное копирование данных, поддержку пользователей и решение проблем.

Программа разрабатывается в течение учебного семестра в индивидуальном режиме работы обучающегося.

6. Порядок контроля и приёмки программного изделия:

6.1. Тестирование: проверка взаимодействия различных модулей и компонентов приложения;

6.2. Приёмка: написание и защита отчёта по проделанной работе.

7. Требования к документированию:

7.1. Пояснительная записка – подробное описание всех аспектов проведённых работ;

7.2. Приложения – используемые схемы и иллюстрации кода;

7.3. Инструкция по эксплуатации – руководство для пользователей, объясняющее основные функции, возможности и способы использования приложения.

Задание к работе

Разработать ТЗ по теме своей курсовой работы или выпускной квалификационной работы. Ознакомиться с этапами проекта по разработке и внедрению информационной системы (табл. 1.1). Тема должна быть связана с разработкой/модернизацией информационной системы (целиком или частично). Вместо информационной системы можно использовать разработку компьютерных программ, нейронных сетей, алгоритмов/протоколов, программное обеспечение роботов и т. д. Если ваши темы не связаны с разработкой информационной системы, то следует обратиться к преподавателю и выбрать наименование информационной системы из предложенного списка.

Методика выполнения работы

Перед выполнением задания необходимо изучить следующие стандарты: ГОСТ 34.602, ГОСТ 34.601-90, ГОСТ 24.104-2023. Ознакомиться с примерами технических заданий из УМК. Разработать ТЗ, руководствуясь приведенными рекомендациями.

Рекомендации по обработке и оформлению полученных результатов

Сформировать текстовый документ «Техническое задание» в электронной форме и представить преподавателю. Текст необходимо отформатировать в соответствии с ГОСТ Р 2.105-2019 ЕСКД. Общие требования к текстовым документам.

В таблице 1.1 представлены основные стадии и этапы создания информационной системы (АС) согласно государственному стандарту ГОСТ 34.601-90.

Таблица 1.1

Стадии и этапы работ по разработке и информационной системы

Стадия	Этап работ
1. Формирование требований к ИС (предпроектное обследование)	1.1. Формирование требований пользователя к ИС. Анкетирование работников, будущих пользователей системы
	1.2. Сбор данных об особенностях хозяйственной деятельности предприятия. Определение необходимого количества автоматизированных рабочих мест
2. Разработка концепции ИС	2.1. Изучение объекта автоматизации (информатизации). Моделирование предметной области
	2.2. Проведение необходимых научно-исследовательских работ. Моделирование ИС и ПО
	2.3. Разработка вариантов концепции ИС, удовлетворяющего требованиям пользователя
3. Техническое задание	3.1. Написание ТЗ
	3.2. Согласование, утверждение ТЗ
4. Эскизный проект	4.1. Разработка предварительных проектных решений по системе и её частям. Моделирование ИС и ПО
	5.2. Разработка документации на ИС и её части
5. Технический проект	5.1. Разработка проектных решений по системе и её частям. Моделирование ИС и ПО
	5.2. Разработка документации на ИС и её части
	5.3. Разработка и оформление документации на поставку компонент для комплектования ИС и (или) технических требований (технических заданий) на их разработку
6. Рабочая документация	6.1. Разработка рабочей документации на систему и её части
7. Внедрение системы	7.1. Подготовка к вводу в действие ИС
	7.2. Обучение пользователей работе с системой, их аттестация. Помощь пользователям в работе с системой
	7.3. Комплектация АС поставляемыми компонентами (программными и техническими средствами, программно-техническими комплексами, информационными изделиями), по необходимости
	7.4. Тестирование. Проверка функциональности системы, выявление замечаний. Устранение выявленных замечаний
	7.5. Проведение опытной эксплуатации
	7.6. Проведение приёмочных испытаний
8. Сопровождение	8.1. Сопровождение системы и промышленная эксплуатация

Вопросы для самоконтроля

1. Перечислить основные разделы технического задания.
2. Что указывается в разделе ТЗ – Требования к системе в целом?
3. Требуется ли в разделе ТЗ «Общие требования» указывать перечень документов, на основании которых создается система?
4. В каком разделе ТЗ описываются календарные сроки и виды работ?

Рекомендуемая литература

1. Грекул, В. И. Проектирование информационных систем : практикум: учебное пособие / В. И. Грекул, Н. Л. Коровкина, Ю. В. Куприянов. – Москва : Национальный Открытый Университет «ИНТУИТ.ru», 2012. – 187 с. – ISBN 978-5-9556-0133-5.
2. Паршин, К. А. Методы и средства проектирования информационных систем и технологий: учебное пособие / К. А. Паршин. – Екатеринбург : УрГУПС, 2018. – 129 с.

Лабораторная работа № 2
Моделирование бизнес-процессов с использованием программы RAMUS

Тема занятия: моделирование бизнес-процессов.

Цель занятия: освоить интерфейс программы RAMUS для моделирования бизнес-процессов в нотации IDEF0.

Материалы и программное обеспечение:

1. Р 50.1.028-2001. Информационные технологии поддержки жизненного цикла продукции. Методология функционального моделирования.
2. Программное обеспечение RAMUS.

Краткие теоретические сведения

Предпроектное обследование предметной области является важной фазой при разработке информационной системы. Допускаемые различные ошибки на этой стадии имеют, как правило, значительные последствия для разработки проекта, поскольку неверно подготовленные исходные данные могут существенно увеличить сроки разработки и приемки проекта.

Предметная область – часть реального мира, рассматриваемая в рамках определённой деятельности. Например, можно рассматривать такие предметные области, как университет, склад, кинотеатр, школа и т. д.

Предметная область информационной системы – совокупность реальных процессов и объектов (сущностей) в некоторой области деятельности для организации управления и в конечном счёте автоматизации.

Объект предметной области – факт, событие, предмет, человек, о котором могут быть собраны данные.

Информационный объект или *сущность* – описание некоторого класса реальных объектов в виде совокупности свойств.

Модель предметной области – система, имитирующая структуру или функционирование исследуемой предметной области и отвечающая основному требованию – быть адекватной этой области.

Предварительное моделирование предметной области позволяет сократить время и сроки проведения проектировочных работ и получить более эффективный и качественный проект.

К моделям предметных областей предъявляются следующие требования:

– формализация, обеспечивающая однозначное описание структуры предметной области;

– согласование требований заказчиков и разработчиков на основе применения графических средств отображения модели;

– реализуемость, подразумевающая наличие средств физической реализации модели предметной области в ИС;

– обеспечение оценки эффективности реализации модели предметной области на основе определенных методов и вычисляемых показателей.

Для реализации перечисленных требований, как правило, строится *система моделей*, которая отражает структурный и оценочный аспекты функционирования предметной области.

Структурный аспект предполагает построение:

– *объектной структуры*, отражающей состав взаимодействующих в процессах материальных и информационных объектов предметной области;

– *функциональной структуры*, отражающей взаимосвязь функций (действий) по преобразованию объектов в процессах;

– *структуры управления*, отражающей события и бизнес-правила, которые воздействуют на выполнение процессов;

– *организационной структуры*, отражающей взаимодействие организационных единиц предприятия и персонала в процессах;

– *технической структуры*, описывающей топологию расположения и способы коммуникации комплекса технических средств.

С моделированием непосредственно связана проблема *выбора языка* представления проектных решений, позволяющего наиболее полно привлекать будущих пользователей системы к ее разработке.

Язык моделирования – нотация, в основном графическая, которая используется для описания проектов. *Нотация* представляет собой совокупность графических объектов, используемых в *модели*, и является синтаксисом языка моделирования. Язык моделирования, с одной стороны, должен делать решения проектировщиков понятными пользователю, с другой стороны, предоставлять проектировщикам средства достаточно формализованного и однозначного определения проектных решений, подлежащих реализации в виде программных комплексов, образующих целостную систему программного обеспечения.

Методология проектирования информационной системы заключается в организации процесса её построения и обеспечении управления этим процессом. Она позволяет гарантировать выполнение требований как к самой системе, так и к характеристикам процесса разработки. Под построением понимают основные этапы ЖЦ ИС: проектирование, разработка и внедрение ИС. Построение ИС осуществляется с помощью применения последовательности методов. Поэтому можно говорить, что методология представляет собой совокупность методов применяемых в течении ЖЦ ИС.

Метод – последовательный процесс создания моделей, которые описывают вполне определенными средствами различные стороны разрабатываемой программной системы.

В основе различных методологий моделирования предметной области ИС лежат принципы последовательной детализации общих (абстрактных) представлений. Модели строятся на трех уровнях:

1. Внешний уровень (*определение/спецификация требований*). На внешнем уровне модель отвечает на вопрос, *что должна делать система*, то есть определяется состав основных компонентов системы: объектов, функций, событий, организационных единиц, технических средств.

2. Внутренний уровень *(реализации требований*). На внутреннем уровне модель отвечает на вопрос: *с помощью каких программно-технических средств реализуются требования к системе*?

3. Концептуальный уровень. На концептуальном уровне модель отвечает на вопрос, *как должна функционировать система*? В этом случае переделяется характер взаимодействия компонентов системы одного или разных типов.

В настоящем практикуме основное внимание при построении моделей будет обращено на структурный аспект, а именно на функциональную структуру (функциональный аспект) и объектные структуры.

Рассмотрим структурные методы анализа для создания ИС. Анализ состоит в исследовании системных требований и проблемы. Результат анализа выражается в *модели предметной области*, которая иллюстрируется в виде набора диаграмм с изображенными на них понятиями или объектами предметной области.

Методы структурного анализа и проектирования направлены на упрощение сложных систем путем применения нескольких основных идей по преобразованию этих систем и их представлению.

В основе методов структурного анализа лежат следующие основные принципы:

1. Разбиение системы на черные ящики (принцип «разделяй и властвуй»), которое должно удовлетворять следующим критериям:

– каждый черный ящик должен реализовывать одну единственную функцию системы;

– функция каждого черного ящика должна быть легко воспринимаема независимо от его сложности;

– связь между черными ящиками должна вводиться только при наличии связи между существующими реально функциями системы;

– связи между черными ящиками должны быть по возможности простыми для обеспечения независимости между ними.

2. Идея иерархии (принцип иерархического упорядочивания).

Помимо разбиения системы на части их необходимо определенным образом упорядочить в виде иерархических структур.

3. Структурные методы широко используют графические нотации, служащие для облегчения понимания сложных систем.

Структурный анализ – исследование системы, которое начинается с ее общего обзора и затем детализируется, приобретая иерархическую структуру со все большим числом уровней. Структурный подход не обеспечивает возможность создания предельно сложных систем (неэффективен в ООЯП).

Методология SADT – методология, содержащая правила и процедуры, предназначенные для функционального моделирования предметной области.

На базе методологии SADT разработан стандарт (нотация) IDEF0, являющийся частью семейства IDEF. В рамках данной методологии были разработаны несколько официальных стандартов анализа и проектирования.

Методология SADT может использоваться для широкого круга систем: для анализа функций, выполняемых системой, и механизмов, которые управляют этими функциями.

RAMUS – программа для построения визуальных диаграмм, используемых для наглядного отображения различных бизнес-процессов. Она поддерживает методологии моделирования бизнес-процессов IDEF0 и DFD.

Бизнес-процесс – совокупность взаимосвязанных действий, направленных на достижение определенной цели или результата.

IDEF0 – методологическая нотация для моделирования функциональных процессов, использующая блок-схемы с входами, выходами, управлениями и механизмами.

Декомпозиция – процесс разбивки сложного процесса на более простые элементы.

Основные элементы IDEF0

– *Функция (активность, работа)* – действие, представленное прямоугольником.

– *Входы* – ресурсы или информация, необходимые для выполнения функции (левый край блока).

– *Выходы* – результаты выполнения функции (правый край блока).

– *Управление* – условия или правила, регулирующие выполнение функции (верхняя сторона блока).

– *Механизмы (исполнители)* – ресурсы или инструменты, обеспечивающие выполнение функции (нижняя сторона блока).

В состав диаграммы входят *блоки* – функции (активности) и дуги, описывающие взаимоотношения между блоками (рис. 2.1). Блок описывает бизнес-процесс или любую функцию, возникающую на объекте автоматизации.

Блок –функция моделируемой системы. Имя блока – глагол или глагольный оборот.

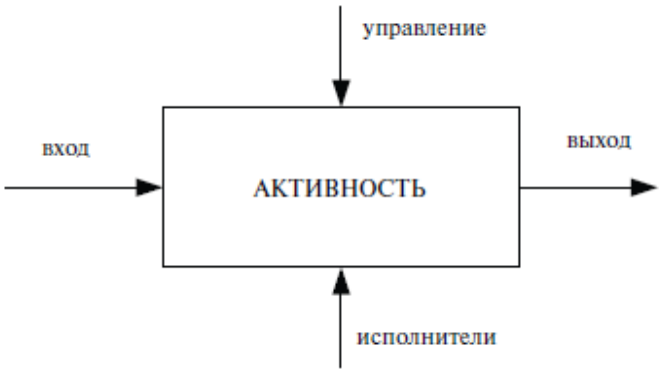


Рис. 2.1. Активность (работа), которая представляет функцию в нотации IDEF0

Для более подробного знакомства с методологией функционального моделирования необходимо ознакомиться со стандартом, который разработан в РФ –

Р 50.1.28–2001 Методология функционального моделирования IDEF0.2001 и находится в папке стандартов УМК на диске.

Задания к работе

Задание 1.

1. Изучить интерфейс программы RAMUS и основные возможности для работы с моделями в нотации IDEF0.
2. Построить модель бизнес-процесса по заданной теме (например, «Процесс обработки заказа»).
3. Провести декомпозицию основной функции на подпроцессы.
4. Для каждой стрелки (вход, выход, управление, механизм) оформить описание в виде таблицы с колонками: *Название:* идентификатор стрелки. *Описание:* что представляет данная стрелка.
5. Сохранить и экспортировать модель в формате PDF.

Задание 2.

1. Открыть программное обеспечение RAMUS.
2. Создать новый проект и задать его название (например, «Обработка заказа»).
3. Построить верхнеуровневую модель (контекстная диаграмма, A0).
4. Определить основную функцию.
5. Добавить входы, выходы, управления и механизмы.
6. Выполнить декомпозицию основной функции.
7. Добавить дочерние диаграммы для отдельных задач.
8. Указать связи между функциями.
9. Создать таблицы описания для всех стрелок диаграммы.
10. Проверить модель на соответствие правилам нотации IDEF0.
11. Сохранить проект и экспортировать его в файл.

Методика выполнения работы

Установить программное обеспечение RAMUS на свой компьютер. Ознакомиться с методологией функционального моделирования. В соответствии с полученным заданием от преподавателя разработать схему бизнес-процесса.

Пример 1. Построение таблицы с описанием дуг и диаграммы функциональной модели первого уровня.

Для моделирования процессов использовалась методология функционального моделирования IDEF0, которая применяется для создания функциональной модели, отображающей структуру и функции системы, а также потоки информации и материальных объектов, связывающих эти функции (табл. 2.1).

На рисунке 2.2 представлен результат построения контекстной диаграммы веб-приложения «Цифровой профиль».

Таблица 2.1

Дуги IDEF0

Название дуги	Смысловая нагрузка	Тип
Результаты тестирований, опросов	Получение данных психологических тестов и опросов, пройденных студентами	Вход
Е-деканат	Получение данных об успеваемости студента, освоенных дисциплинах	Вход
Электронное портфолио	Получение достижений студента в различных сферах деятельности	Вход
Стандарты	Норматив, разрабатываемый организацией с целью установления собственных требований по изготовлению веб-приложения	Управление
Нормативные документы	Совокупность документов, устанавливающих требования к методам изготовления, контроля, испытаний и применения материала	Управление
Стандарты (ГОСТы)	Общие стандарты производства однородной продукции, объединенной по схожим потребительским качествам, признакам, назначению и применению	Управление
Программное обеспечение	Программные средства, необходимые для запуска сайта	Механизм
Аппаратное обеспечение	Оборудование, необходимое для запуска сайта	Механизм
Веб-приложение «Цифровой профиль»	Готовый продукт	Выход

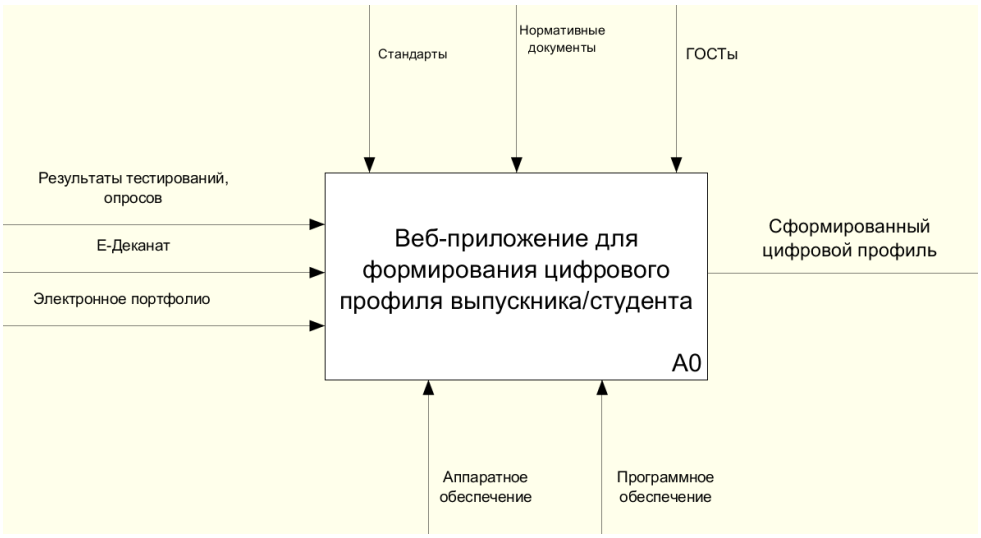


Рис. 2.2. Контекстная диаграмма веб-приложения «Цифровой профиль» в нотации IDEF0

Рекомендации по обработке и оформлению полученных результатов

Оформить описание стрелок в виде таблиц, как указано в примере 1. Подготовить пояснительный текст к модели. Представить результаты работы в виде отчета. Подготовить отчет в соответствии с рекомендациями, приведенными в разделе «Подготовка отчета по лабораторной работе».

Вопросы для самоконтроля

1. Что такое методология функционального моделирования и можно ли с ее помощью отобразить набор в изменениях модели?

2. К какому типу моделей относятся функциональные модели – к статическому или динамическому?

3. Каковы основные элементы нотации IDEF0?

4. Какие требования предъявляются к моделям в нотации IDEF0?

5. Что описывается стрелками на диаграммеIDEF0?

Варианты заданий

1. База данных *школа*.

Таблицы: учителя, должности, обучающиеся, дополнительная информация, классы, виды классов, предметы, расписание.

2. База данных *учебное расписание школы*.

Таблицы: учителя, обучающиеся, дополнительная информация, классы, виды классов, предметы.

3. База данных *курсы переподготовки*.

Таблицы: курсы, учителя, обучающиеся, дополнительная информация, предметы, расписание курсов.

4. База данных *тестовая система*.

Таблицы: курсы, учителя, обучающиеся, дополнительная информация, предметы, расписание курсов.

5. База данных *электронное портфолио*.

Таблицы: курсы, учителя, обучающиеся, дополнительная информация, предметы, расписание курсов.

6. База данных *электронная ведомость*

Таблицы: курсы, учителя, обучающиеся, дополнительная информация, предметы, расписание курсов.

7. База данных *библиотеки*.

Таблицы: преподаватели, обучающиеся, полка, книжный фонд, выданные книги, архив, выставка, дополнительная информация.

8. База данных *конкурсы (виды: общественная деятельность, научная работа, творческий, спортивной деятельности)*.

Таблицы: учителя, обучающиеся, описание конкурса, документы, результаты, дополнительная информация.

9. База данных *гостиницы*.

Таблицы: услуги, номера, сотрудники, должности, клиенты.

10. База данных *учебного кабинета*.

Таблицы: обучающиеся, учителя, издательства, жанры, книги, выданные книги, выданное оборудование.

11. База данных *заказ такси*.

Таблицы: сотрудники, должности, тип автомобиля, тарифы, дополнительные услуги, вызовы.

12. База данных *туристического агентства.*

Таблицы: сотрудники, должности, виды отдыха, отели, дополнительные услуги, клиенты, путёвки.

13. База данных *страховой компании*.

Таблицы: сотрудники, должности, риски, виды полисов, группы клиентов, клиенты, полисы.

14. База данных *автосервисного центра.*

Таблицы: сотрудники, должности, запчасти, ремонтируемые модели, виды неисправностей, обслуживаемые магазины, заказы.

15. База данных *склада.*

Таблицы: сотрудники, должности, товары, типы товаров, поставщики, заказчики, склад.

16. База данных *аптека.*

Таблицы: сотрудники, должности, товары, типы товаров, поставщики, заказчики, склад.

17. База данных *кинотеатра*.

Таблицы: сотрудники, должности, жанры, фильмы, текущий репертуар, места.

18. База данных *компьютерного магазина.*

Таблицы: сотрудники, должности, Виды комплектующих, комплектующие, заказчики, услуги, заказы.

19. База данных *строительной компании*.

Таблицы: сотрудники, должности, виды работ, код материала, материалы, бригады, заказчики, заказы.

20. База данных *транспортной компании*.

Таблицы: сотрудники, должности, виды автомобилей, марки автомобилей, виды грузов, грузы, автомобили, рейсы.

21. База данных *радиостанции*.

Таблицы: сотрудники, должности, исполнители, жанры, записи, график работы.

22. База данных *продажи билетов в аэропорту.*

Таблицы: сотрудники, должности, самолёты, типы самолётов, экипажи, рейсы, билеты.

Рекомендуемая литература

1.Грекул, В. И. Проектирование информационных систем : практикум: учебное пособие / В. И. Грекул, Н. Л. Коровкина, Ю. В. Куприянов. – Москва : Национальный Открытый Университет «ИНТУИТ.ru», 2012. – 187 с. – ISBN 978-5-9556-0133-5.

2. Коцюба, И. Ю. Основы проектирования информационных систем : учебное пособие / И. Ю. Коцюба, А. В. Чунаев, А. Н. Шиков. – Санкт-Петербург : Университет ИТМО, 2015. – 206 с.

3. Федотова, Д. Э. CASE-технологии: практикум / Д. Э. Федотова, Ю. Д. Семенов, К. Н. Чижик. – Москва : Горячая линия-Телеком, 2005. – 160 с. – ISBN 5-93517-121-X.

Лабораторная работа № 3
Создание диаграмм декомпозиций первого уровня

Тема занятия: моделирование бизнес-процессов.

Цель занятия: научиться создавать диаграммы декомпозиций первого уровня для бизнес-процессов.

Материалы и программное обеспечение

1. Р 50.1.028-2001. Информационные технологии поддержки жизненного цикла продукции. Методология функционального моделирования.
2. Исходные данные берутся из вариантов заданий, предложенных в лабораторной работе № 2.
3. Программное обеспечение RAMUS.

Краткие теоретические сведения

Модель IDEF0всегда начинается с представления системы как единого целого – одного функционального блока с интерфейсными дугами, простирающимися за пределы рассматриваемой области. Такая диаграмма с одним функциональным блоком называется *контекстной диаграммой* и обозначается идентификатором “A0”.

Декомпозиция – процесс разделения сложной функции на более мелкие, упрощенные подфункции (подпроцессы).

Принцип *декомпозиции* применяется при разбиении сложного процесса на составляющие его функции. При этом уровень детализации процесса определяется непосредственно разработчиком модели.

Декомпозиция позволяет последовательно и структурно представлять модель системы в виде иерархической структуры отдельных диаграмм, что делает ее менее перегруженной и легко усваиваемой.

В пояснительном тексте к контекстной диаграмме должна быть указана *цель* построения диаграммы в виде краткого описания и зафиксирована *точка зрения (View point).*

На диаграммах декомпозиций каждая функция верхнего уровня представляется как набор подпроцессов.

Определение и формализация цели разработки *IDEF0*-модели является крайне важным моментом. Фактически цель определяет соответствующие области в исследуемой системе, на которых необходимо фокусироваться в первую очередь. Например, если моделируется деятельность предприятия с целью построения в дальнейшем на базе этой модели информационной системы, то эта модель будет существенно отличаться от той, которая бы разрабатывалась для того же самого предприятия, но уже с целью оптимизации логистических цепочек.

Точка зрения определяет основное направление развития модели и уровень необходимой детализации. Четкое фиксирование точки зрения позволяет разгрузить модель, отказавшись от детализации и исследования отдельных элементов, не являющихся необходимыми, исходя из выбранной точки зрения на систему.

Например, функциональные модели одного и того же предприятия с точек зрения главного технолога и финансового директора будут существенно различаться по направленности их детализации. Это связано с тем, что в конечном итоге финансового директора не интересуют аспекты обработки сырья на производственных станках, а главному технологу ни к чему прорисованные схемы финансовых потоков. Правильный выбор *точки зрения существенно* сокращает временные затраты на построение конечной модели.

Пример. Декомпозиция первого уровня модели. Описание блоков представлено в таблице 3.1.

Таблица 3.1

Описание функциональных блоков декомпозици и первого уровня

Название блока	Описание блока
Авторизация	Авторизация пользователей (ввод логина и пароля)
Получение данных из ЭИОС	Получение данных о студентах/выпускниках из баз данных или информационных систем
Обработка данных	Обработка полученных данных
Генерация	Генерация веб-форм, графиков, моделей и т. д., отображение их на веб-странице

После определения требуемых функциональных блоков для декомпозиции первого уровня веб-приложения «Цифровой профиль» необходимо связать их с граничными стрелками (рис. 3.1).

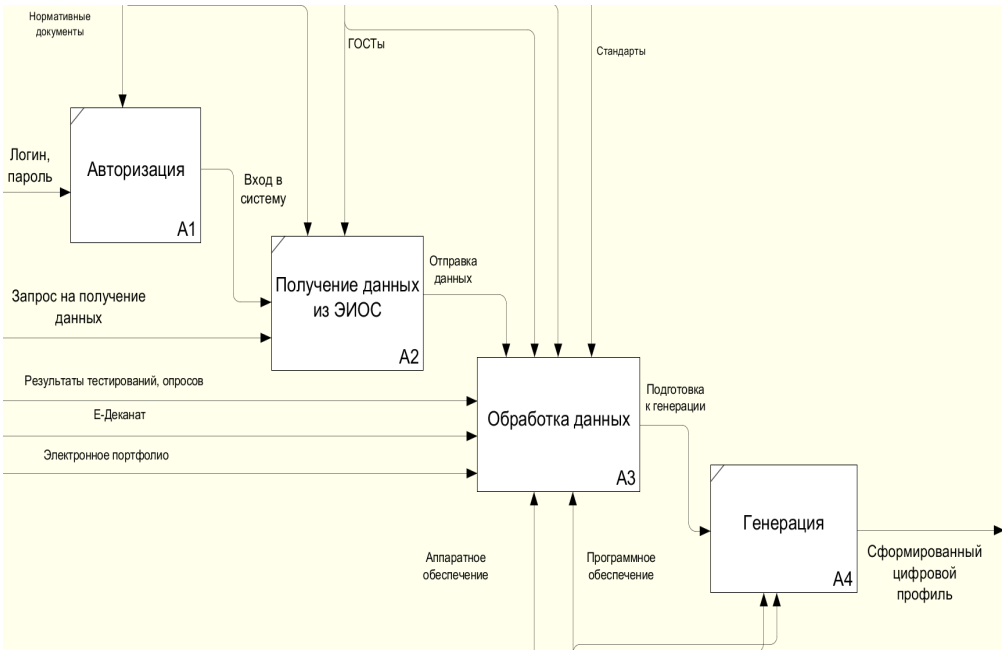


Рис. 3.1. Диаграмма декомпозиции первого уровня в нотации IDEF0

Задания к работе

1. Выполнить декомпозицию контекстной диаграммы (построенной в лабораторной работе № 1) на несколько функций.
2. Построить диаграмму декомпозиций первого уровня (L=1).
3. Описать стрелки диаграммы в таблицах (название/описание блока).
4. Сохранить диаграмму и экспортировать ее.

Методика выполнения работы

Открыть проект из лабораторной работы № 2 в программе RAMUS. Создать декомпозицию первого уровня модели для каждой функции. Добавить элементы управления, механизмы, входы и выходы. Проверить диаграмму на соответствие правилам IDEF0.

Рекомендации по обработке и оформлению полученных результатов

Убедиться в том, что декомпозиция отражает полный процесс. Подготовить пояснительный текст и таблицы описания стрелок.

Вопросы для самоконтроля

1. Что такое диаграмма декомпозиций и в чем ее отличие от контекстной диаграммы?
2. Какие элементы необходимо учитывать при построении диаграмм декомпозиций?
3. Каковы основные этапы декомпозиции бизнес-процесса?
4. Почему важно описывать стрелки на диаграммах IDEF0 и что должно быть указано в описании?
5. Какие правила следует соблюдать при построении диаграмм декомпозиций?
6. Какие функции диаграммы можно считать завершенными, а какие требуют дальнейших декомпозиции?

Рекомендуемая литература

1. Грекул, В. И. Проектирование информационных систем : практикум: учебное пособие / В. И. Грекул, Н. Л. Коровкина, Ю. В. Куприянов. – Москва : Национальный Открытый Университет «ИНТУИТ.ru», 2012. – 187 с. – ISBN 978-5-9556-0133-5.

2. Коцюба, И. Ю. Основы проектирования информационных систем : учебное пособие / И. Ю. Коцюба, А. В. Чунаев, А. Н. Шиков. – Санкт-Петербург : Университет ИТМО, 2015. – 206 с.

3. Федотова, Д. Э. CASE-технологии: практикум / Д. Э. Федотова, Ю. Д. Семенов, К. Н. Чижик. – Москва : Горячая линия-Телеком, 2005. – 160 с. – ISBN 5-93517-121-X.

Лабораторная работа № 4
Создание диаграммы декомпозиций второго уровня

Тема занятия: моделирование бизнес-процессов.

Цель занятия: освоить построение диаграмм второго уровня для детального описания бизнес-процессов.

Материалы и программное обеспечение

1. Р 50.1.028-2001. Информационные технологии поддержки жизненного цикла продукции. Методология функционального моделирования.
2. Исходные данные берутся из вариантов заданий, предложенных в лабораторной работе № 2.
3. Программное обеспечение RAMUS.

Краткие теоретические сведения

В процессе декомпозиции функциональный блок, который в контекстной диаграмме отображает систему как единое целое, подвергается детализации на следующей диаграмме, построенной в результате декомпозиции. В результате декомпозиции получим диаграмму второго уровня, которая содержит функциональные блоки, отображающие главные подфункции функционального блока контекстной диаграммы. Получившаяся диаграмма называется *дочерней*, а каждый из функциональных блоков, принадлежащих дочерней диаграмме, соответственно называется *дочерним блоком*. Функциональный блок, который является предком дочернего блока, называется *родительским блоком*, а диаграмма, которой он принадлежит, – *родительской диаграммой*. Каждая из подфункций дочерней диаграммы может быть далее детализирована путем аналогичной декомпозиции соответствующего ей функционального блока.

Важно отметить, что в каждом случае декомпозиции функционального блока все интерфейсные дуги, входящие в данный блок или исходящие из него, фиксируются на дочерней диаграмме. Этим достигается структурная целостность *IDEF0*-модели. Наглядно принцип декомпозиции представлен на рисунке 4.1.

Необходимо обратить внимание на взаимосвязь нумерации функциональных блоков и диаграмм. Каждый блок имеет уникальный порядковый номер на диаграмме (цифра в правом нижнем углу прямоугольника), а обозначение под правым углом указывает на номер дочерней для этого блока диаграммы. Отсутствие этого обозначения говорит о том, что декомпозиция для данного блока не выполнялась.

Контекстная диаграмма – модель, описывающая основную функцию системы и ее взаимодействие с внешней средой.

Декомпозиция – процесс последовательного разложения сложных функций на более простые элементы.

Диаграмма второго уровня – детализированное представление отдельных функций, входящих в диаграмму первого уровня, созданное в процессе декомпозиции.

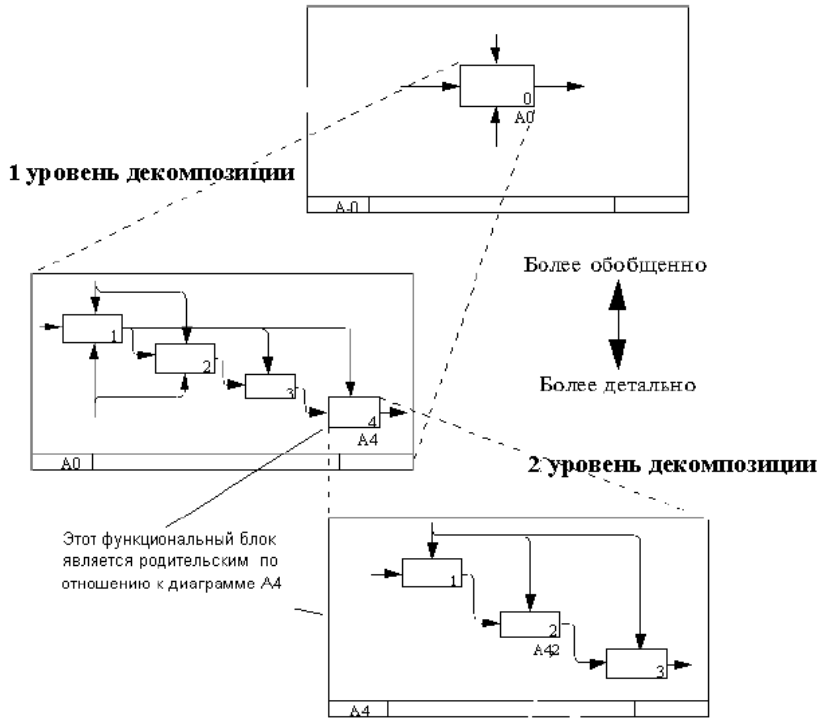


Рис. 4.1. Декомпозиция функциональных блоков первого и второго уровня

Принципы создания диаграммы второго уровня

Декомпозиция выполняется для функций, которые требуют дополнительного описания. Диаграммы второго уровня содержат больше деталей по сравнению с диаграммами верхнего уровня. Каждая функция должна быть связана с входами, выходами, управлениями и механизмами.

Элементы диаграммы второго уровня

1. *Функции* – представляют действия, происходящие внутри системы.
2. *Входы* – информация или ресурсы, поступающие в функцию.
3. *Выходы* – результаты выполнения функции.
4. *Управление* – правила, инструкции или условия, регулирующие выполнение функции.
5. *Механизмы* – ресурсы, обеспечивающие выполнение функции.

Пример 1. Декомпозиция модели второго уровня (табл. 4.1, 4.2).

Таблица 4.1

Описание функциональных блоков декомпозиции второго уровня

Название блока	Описание блока
Проверка данных	Валидация данных на соответствие требованиям и стандартам приложения
Нормализация данных	Приведение данных к единому формату и структуре для обеспечения единообразной обработки
Хранение данных	Сохранение обработанных данных в базе данных или другом хранилище
Отправка данных	Отправка обработанных данных приложению для дальнейшей генерации веб-форм

Таблица 4.2

Описание стрелок декомпозици и второго уровня

Название стрелки	Начало стрелки	Тип начала стрелки	Окончание стрелки	Тип окончания стрелки
Отправка данных	Граница диаграммы	Управляющее воздействие	Проверка данных	Управляющее воздействие
Стандарты (ГОСТы)	Граница диаграммы	Управляющее воздействие	Нормализация данных	Управляющее воздействие
			Хранение данных	
			Отправка данных	
Нормативные документы	Граница диаграммы	Управляющее воздействие	Нормализация данных	Управляющее воздействие
			Хранение данных	
			Отправка данных	
			Нормализация данных	
Стандарты	Граница диаграммы	Управляющее воздействие	Хранение данных	Управляющее воздействие
			Отправка данных	
			Проверка данных	
Результаты тестирований, опросов	Граница диаграммы	Вход	Проверка данных	Вход
Е-деканат	Граница диаграммы	Вход	Проверка данных	Вход
Электронное портфолио	Граница диаграммы	Вход	Проверка данных	Вход
Аппаратное обеспечение	Граница диаграммы	Механизм	Нормализация данных	Механизм
			Хранение данных	
			Отправка данных	
			Проверка данных	
Программное обеспечение	Граница диаграммы	Механизм	Нормализация данных	Механизм
			Хранение данных	
			Отправка данных	
			Нормализация данных	
Проверка перед нормализацией	Проверка данных	Выход	Хранение данных	Вход
Передача данных для хранения	Нормализация данных	Выход	Отправка данных	Вход
Отправка данных из хранилища	Хранение данных	Выход	Граница диаграммы	Вход
Подготовка к генерации	Отправка данных	Выход		Выход

На рисунке 4.2 представлена диаграмма декомпозиции второго уровня функционального блока «Обработка данных».

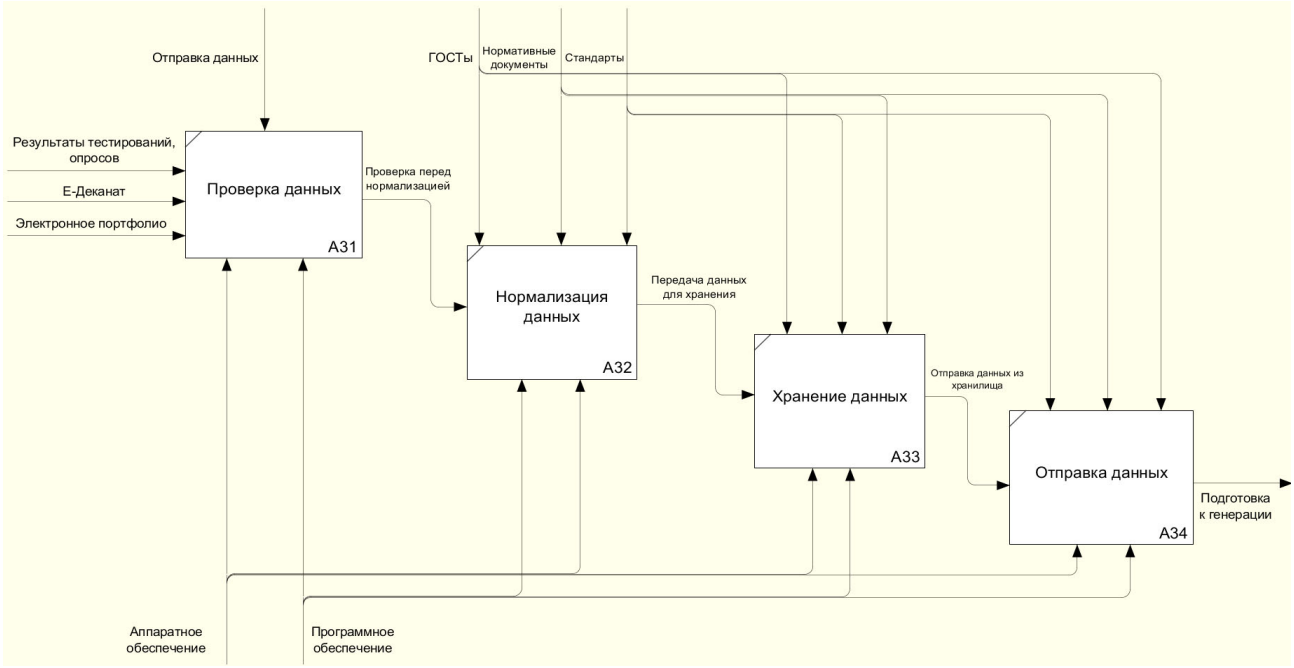


Рис. 4.2. Декомпозиция второго уровня моделивнотацииIDEF0

Задания к работе

1. Открыть проект, созданный в предыдущей лабораторной работе.
2. Выполнить декомпозицию одной из функций верхнего уровня, построив диаграмму второго уровня.
3. Добавить входы, выходы, управления и механизмы для каждой функции диаграммы второго уровня.
4. Описать все стрелки (входы, выходы, управления, механизмы) в виде таблицы с колонками:
5. Проверить модель на соответствие правилам нотации IDEF0.
6. Сохранить диаграмму и экспортировать ее в формат PDF-файла.

Методика выполнения работы

Запустите программное обеспечение RAMUS и загрузите проект из лабораторной работы № 2. Выбрать функцию первого уровня, которая требует детального описания, и выполнить ее декомпозицию. Добавить стрелки для входов, выходов, управлений и механизмов в соответствии с функциональными связями. Оформить описание стрелок в виде таблицы, предусмотренной заданием, и проверить корректность связей между элементами модели. Сохранить проект и экспортировать диаграмму второго уровня.

Рекомендации по обработке и оформлению полученных результатов

Убедиться, что каждая функция на диаграмме второго уровня полностью описана, а все стрелки правильно связаны с соответствующими функциями. Таблицы с описанием стрелок оформить в соответствии с требованиями задания. Отчет по работе должен содержать диаграмму второго уровня, таблицы с описанием стрелок и пояснительный текст, описывающий процесс декомпозиции.

Вопросы для самоконтроля

1. Что такое диаграмма второго уровня и каково ее отличие от диаграммы первого уровня?
2. Какие элементы диаграммы второго уровня наиболее важны?
3. Какие ошибки могут возникнуть при декомпозиции функций?
4. Каковы правила оформления входов, выходов, управлений и механизмов на диаграмме?
5. Почему важно детально описывать стрелки и их взаимосвязи?
6. Как проверить правильность построенной диаграммы второго уровня?
7. Какое значение имеют входы и управления для выполнения функций?

Рекомендуемая литература

1. Грекул, В. И. Проектирование информационных систем : практикум: учебное пособие / В. И. Грекул, Н. Л. Коровкина, Ю. В. Куприянов. – Москва : Национальный Открытый Университет «ИНТУИТ.ru», 2012. – 187 с. – ISBN 978-5-9556-0133-5.

2. Коцюба, И. Ю. Основы проектирования информационных систем : учебное пособие / И. Ю. Коцюба, А. В. Чунаев, А. Н. Шиков. – Санкт-Петербург : Университет ИТМО, 2015. – 206 с.

3. Федотова, Д. Э. CASE-технологии: практикум / Д. Э. Федотова, Ю. Д. Семенов, К. Н. Чижик. – Москва : Горячая линия-Телеком, 2005. – 160 с. – ISBN 5-93517-121-X.

Лабораторная работа № 5
Диаграммы потоков данных DFD

Тема занятия: моделирование потоков данных.

Цель занятия: освоить построение диаграмм потоков данных DFD и выполнить описание взаимосвязи между процессами при помощи IDEF3 (WorkFlow).

Материалы и программное обеспечение

1. Исходные данные берутся из вариантов заданий, предложенных в лабораторной работе № 2.
2. Программное обеспечение RAMUS.

Краткие теоретические сведения

Эскизное проектирование выполняется с целью получения предварительных решений по автоматизации технологических процессов без привязки к конкретным программным продуктам. На данной стадии осуществляется моделирование функций и базы данных будущей системы.

Диаграмма потоков данных (DFD) – основное средство моделирования *функциональных требований к системе*. Основное назначение DFD-диаграмм – показать, как каждый процесс системы преобразует свои входные данные в выходные, а также выявить отношения между процессами.

Основная идея подхода DFD заключается в следующем: источники информации (внешние сущности) порождают информационные потоки (потоки данных), переносящие информацию к подсистемам или процессам.

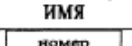
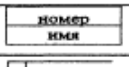
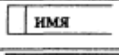
Подсистемы или процессы преобразуют информацию и порождают новые потоки, которые переносят информацию к другим процессам или подсистемам, накопителям данных или внешним сущностям (потребителям информации).

В лабораторной работе для построения диаграмм потоков данных используется нотация Гейна-Сорсона (табл. 5.1):

- внешние сущности;
- системы/подсистемы (для сложных систем);
- процессы;
- накопители данных;
- потоки данных.

Таблица 5.1

Нотация Гейна-Сорсона

Компонент	Обозначение
Поток данных	
Процесс	
Хранилище	
Внешняя сущность	

Внешние сущности

Внешняя сущность – материальный предмет или физическое лицо, представляющее собой источник или приемник информации. Внешняя сущность всегда находится за переделами границ анализируемой ИС

Системы и подсистемы

Декомпозиция системы на подсистемы осуществляется при необходимости, если предназначение системы трудно отразить в одном основном процессе на диаграмме самого верхнего уровня. При этом каждая подсистема – определенный процесс, выполняющий конкретную задачу в системе. Подсистемы выделяются для крупных ИС, территориально распределенных, включающих в себя несколько более мелких ИС. Номер подсистемы служит для ее идентификации.

Процессы

Процесс представляет собой преобразование входных потоков данных в выходные в соответствии с действием, задаваемым именем процесса. Номер процесса служит для его идентификации. В поле имени вводится наименование процесса в виде предложения с глаголом в неопределенной форме (вычислить, рассчитать, проверить, определить, создать, получить и т. д.). Физически процесс может быть реализован различными способами: это может быть подразделение организации (отдел), выполняющее обработку входных документов и выпуск отчетов, программа, аппаратно реализованное логическое устройство и т. д.

Хранилище (накопитель данных)

Накопитель данных представляет собой абстрактное устройство для хранения информации, которую можно в любой момент поместить в накопитель и через некоторое время извлечь, причем способы помещения и извлечения могут быть любыми. Накопитель данных может быть реализован физически в виде ящика в картотеке, таблицы в оперативной памяти, файла на магнитном носителе и т. д. Накопитель данных в общем случае является прообразом будущей базы данных, а описание хранящихся в нем данных должно быть увязано с информационной моделью.

Потоки данных

Поток данных определяет информацию, передаваемую через некоторое соединение от источника к приемнику. Реальный поток данных может быть информацией, передаваемой по кабелю между двумя устройствами, письмами, пересылаемыми по почте, магнитными лентами или дискетами, переносимыми с одного компьютера на другой, и т. д.

Поток данных на диаграмме изображается линией, оканчивающейся стрелкой, которая показывает направление потока. Каждый поток данных имеет имя, отражающее его содержание. При построении иерархии DFD переходить к детализации процессов следует только после определения содержания всех потоков и накопителей данных.

Построение модели с помощью DFD

Цель построения модели – создание ясных и понятных требований к системе на каждом уровне детализации. Декомпозиция DFD осуществляется на основе процессов, каждый процесс раскрывается на основе DFD нижнего уровня.

Для этого рекомендуется выполнять следующие требования:

1. На каждой диаграмме размещать от 3 до 6–7 процессов.
2. Не загромождать диаграммы несущественными деталями.
3. Выбирать ясные имена процессов потоков.
4. Декомпозицию потока данных проводить совместно с декомпозицией процесса.

Процесс построения модели:

1. Разбиение множества требований к системе на основные функциональные группы.
2. Идентификация внешних объектов.
3. Идентификация основных потоков информации, циркулирующих между внешними объектами и системой.
4. Разработка предварительной контекстной диаграммы, на которой основные функциональные группы представлены процессами, внешние объекты – внешними сущностями, основные потоки информации – потоками данных.

Важную роль при построении модели играет *контекстная диаграмма* (КД), моделирующая систему наиболее общим видом (рис. 5.1). КД отражает интерфейс системы с внешними сущностями, т. е. информационные потоки, которыми она связывается с внешним миром. На КД отображается, как правило, единственный основной процесс в системе, отражающий ее основную задачу и внешние сущности.

Рис. 5.1. Контекстная диаграмма потоков данных

Каждый проект имеет одну КД со звездообразной топологией.

Номер единственному основному процессу на КД первого уровня, как правило, не присваивается.

В центре КД находится главный процесс, соединенный с приемниками и источниками информации, посредством которых с системой взаимодействуют пользователи и другие внешние системы.

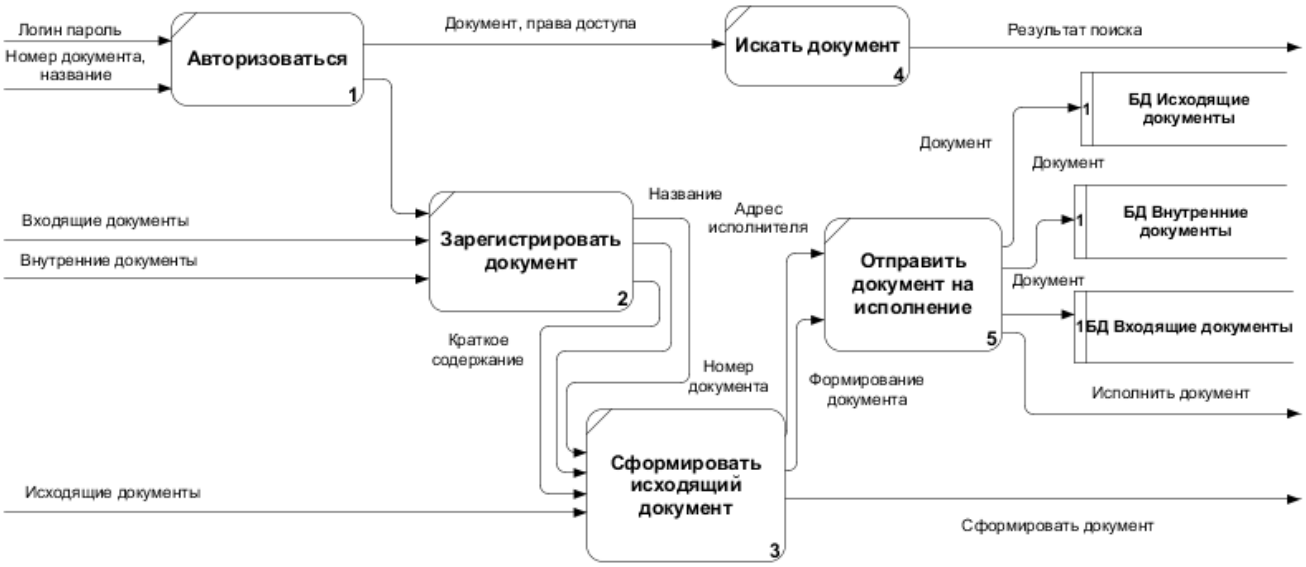


Рис. 5.2. Декомпозиция первого уровня

Для сложной ИС строится контекстная диаграмма, которая включает в себя подсистемы, соединенные между собой и внешними сущностями информационными потоками.

Признаками сложности ИС могут быть:

– наличие большого количества внешних сущностей (десять и более);

– распределенная природа системы;

– многофункциональность системы с уже сложившейся или выявленной группировкой функций в отдельные подсистемы.

Для каждой подсистемы, присутствующей на контекстных диаграммах, либо для одного главного процесса простой ИС выполняется ее детализация при помощи DFD. Каждый процесс на DFD, в свою очередь, может быть детализирован при помощи DFD или мини-спецификации – когда полученный процесс не имеет смысла в дальнейшей детализации. При детализации должны выполняться следующие правила:

– *правило балансировки* – означает, что при детализации подсистемы или процесса детализирующая диаграмма в качестве внешних источников/приемников данных может иметь только те компоненты (подсистемы, процессы, внешние сущности, накопители данных и т. д.), с которыми детализируемая подсистема или процесс имеют информационную связь на родительской диаграмме;

– *правило нумерации* – означает, что при детализации процессов должна поддерживаться их иерархическая нумерация.

После построения законченной модели системы ее необходимо верифицировать (проверить на полноту и согласованность). В согласованной модели для всех потоков данных и накопителей данных должно выполняться правило сохранения информации: все поступающие куда-либо данные должны быть считаны, а все считываемые данные должны быть кем-либо получены или куда-либо записаны.

Пример 1. Построение диаграммы DFD.

Диаграмма DFD (диаграмма потоков данных) представляет собой визуальное описание потоков данных и процессов, которые происходят в системе (рис. 5.3). В данной диаграмме можно выделить следующие блоки:

1. Авторизация.
2. Формирование карточки пользователя.
3. Прохождение опросов.
4. Обработка данных.
5. Построение графиков.
6. Формирование рекомендаций.
7. Отображение рекомендаций.
8. Формирование цифрового профиля.

Хранилища данных: база данных цифрового профиля; Е-деканат; Е-портфолио; база тестов; результаты тестирований.

Стрелки между блоками и хранилищами обозначают потоки данных и процессы, которые происходят между ними, например, стрелка «Логин, пароль» указывает на процесс аутентификации пользователя, а стрелка «Сохранение результатов» указывает на процесс сохранения результатов тестирования в хранилище данных.

Диаграмма DFD помогает понять, как данные перемещаются через систему и какие процессы обрабатывают эти данные, что важно для анализа и проектирования информационных систем. На рисунке 5.3 показана диаграмма потоков данных веб-приложения «Цифровой профиль».

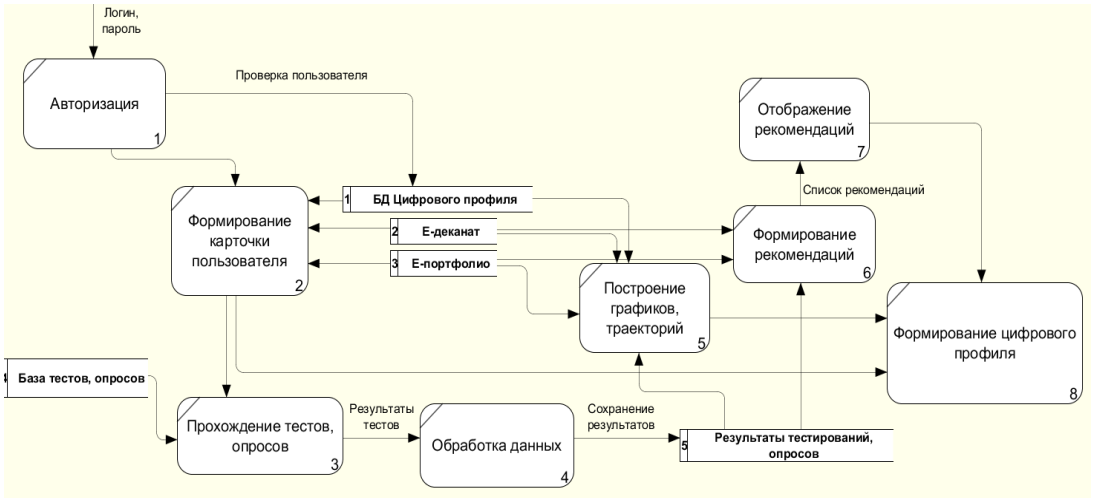


Рис.5.3. Диаграмма потоков данных «Цифровой профиль»

Задания к работе

1. Использовать проект, созданный в лабораторной работе № 2.
2. Создать контекстную диаграмму DFD первого уровня.
3. Выполнить декомпозицию одной из функций верхнего уровня, построив диаграмму второго уровня.
4. Проверить модель на соответствие правилам нотации DFD.
5. Сохранить диаграмму и экспортировать ее в формат PDF-файла.

Методика выполнения работы

Запустить программу RAMUS и создать проект DFD, используя задание из лабораторной работы № 2. Выбрать функцию первого уровня, которая требует детального описания, и выполнить ее декомпозицию. После этого сохранить проект и экспортировать контекстную диаграмму и диаграмму второго уровня.

Рекомендации по обработке и оформлению полученных результатов

Подготовить пояснительный текст к модели. Представить результаты работы в виде отчета. Подготовить отчет в соответствии с рекомендациями, приведенными в разделе «Подготовка отчета по лабораторной работе».

Вопросы для самоконтроля

1. Что такое DFD-диаграммы и для какой цели они используются?
2. К какому типу моделей относятся DFD-диаграммы – к статическому или динамическому?
3. Назвать основные элементы нотации DFD?
4. Какие требования предъявляются к моделям в нотации DFD?

Рекомендуемая литература

1. Грекул, В. И. Проектирование информационных систем : практикум: учебное пособие / В. И. Грекул, Н. Л. Коровкина, Ю. В. Куприянов. – Москва : Национальный Открытый Университет «ИНТУИТ.ru», 2012. – 187 с. – ISBN 978-5-9556-0133-5.

2. Федотова, Д. Э. CASE-технологии: практикум / Д. Э. Федотова, Ю. Д. Семенов, К. Н. Чижик. – Москва : Горячая линия-Телеком, 2005. – 160 с. – ISBN 5-93517-121-X.

Лабораторная работа № 6
Моделирование потоков работ с использованием нотации IDEF3

Тема занятия: моделирование потоков работ.

Цель занятия: освоить построение диаграмм потоков в нотации IDEF3 (Work Flow).

Материалы и программное обеспечение

1. Исходные данные берутся из вариантов заданий, предложенных в лабораторной работе № 2.

2. Программное обеспечение для моделирования потоков работ: Draw.io, Visual Paradigm, Concept Draw PRO.

Краткие теоретические сведения
Методология моделирования IDEF3

Нотация IDEF3 (Integrated DEFinition for Process Description Capture Method) предназначена для описания потоков работ бизнес-процесса (WFD). Используется совместно с нотацией IDEF0, но может применяться и независимо.

IDEF3 представляет собой методологию моделирования и стандарт документирования технологических процессов, происходящих в системе, а также механизм сбора информации о процессах. С использованием структурного метода данная методология демонстрирует причинно-следственные связи между ситуациями и событиями в понятной пользователю форме и описывает, как функционируют система, процесс или предприятие.

В течение длительного времени IDEF3 широко использовалась для создания моделей бизнес-процессов организации на нижнем уровне при описании работ, выполняемых в подразделениях и на рабочих местах. Эта нотация была взята за основу при создании методики описания процессов ARIS eEPC – «расширенной цепочки процесса, управляемого событиями».

Основные характеристики нотации IDEF3

1. Основная цель нотации – дать возможность пользователям описать ситуацию, когда процессы (действия) выполняются в определенной последовательности и взаимной зависимости, а также объекты, участвующие совместно в одном процессе.

2. Нотация IDEF3 чаще применяется для моделирования и анализа процессов *нижнего уровня* и, как правило, используется при *декомпозиции* блоков процесса модели IDEF0.

3. Нотация IDEF3 поддерживает возможность *декомпозиции* (каждый отдельный блок в модели IDEF3, в свою очередь, может быть представлен в виде отдельного подпроцесса).

4. Основной организационной единицей описания в IDEF3 является *диаграмма*. Очень важно придерживаться взаимной организации диаграмм внутри модели, так как они предназначены для чтения другими людьми.

5. Методика построения модели и рекомендации по построению диаграмм аналогичны тем, которые применяются при моделировании в нотации IDEF0.

6. Технологию работы с этой нотацией можно представить следующим образом:

– определяется цель моделирования – набор вопросов, на которые будет отвечать модель, точка зрения, границы моделирования с учетом глубины и широты – какие объекты войдут, а какие не будут отображены в модели;

– опрашиваются эксперты предметной области;

– составляются списки кандидатов на действия (работы), составляющих процесс, и кандидатов на объекты, участвующих в процессе, обозначающих результат выполнения работ.

7. В нотации IDEF3 нет ограничения на количество блоков на одной диаграмме и нет принципа «доминирования» блоков.

Варианты использования IDEF3

1. Сначала строится функциональная модель в нотации IDEF0 на основе исследования предметной области, затем, используя полученные знания о предметной области, строится отдельная модель в нотации IDEF3.

2. Создаётся смешанная модель, дополняя по мере необходимости функциональную модель в нотации IDEF0 диаграммами в нотации IDEF3.

3.Модель DFD дополняется диаграммами в нотации IDEF3.

В каждом конкретном случае при моделировании системы принимается решение о необходимости построения каждого вида модели.

Нотацию IDEF3 целесообразно применять в случае относительно простых процессов на нижнем уровне декомпозиции, то есть на уровне рабочих мест. В этом случае схема процесса может служить основой для создания документов, регламентирующих работу исполнителей. При помощи этой нотации достаточно сложно создавать комбинированные модели, в которых бы сочеталось описание потоков работ и процессы управления ими.

Методы и диаграммы IDEF3

Моделирование в нотации IDEF3 является частью структурного анализа систем (табл. 6.1). Информационная система описывается как упорядоченная последовательность событий с одновременным описанием объектов, имеющих отношение к моделируемому процессу. Моделирование IDEF3 может быть реализовано двумя методами:

1. *Process Flow Description* (PFD) – описание технологических процессов с указанием того, что происходит на каждом этапе технологического процесса.

2. *Object State Transition Description* (OSTD)– описание переходов состояний объектов с указанием существующих промежуточных состояний у объектов в моделируемой системе.

Основу методологии IDEF3 составляет графический язык описания процессов, поэтому модель в нотации IDEF3 может содержать два типа диаграмм:

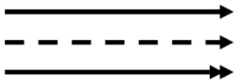
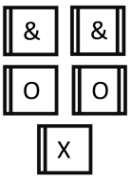
1. Диаграмма описания последовательности этапов процесса (Process Flow Description Diagrams, PFDD).

2. Диаграмма сети трансформаций состояния объекта (Object State Transition Network, OSTN).

Компоненты диаграммы описания процесса и их назначение

Таблица 6.1

Назначение основных компонент IDEF3

Обозначение	Наименование	Описание
	Блоки работы или действия (Boxes, Activities) или Unit of Behavior (UOB)	Объект служит для описания функций (процедур, работ), выполняемых подразделениями либо сотрудниками предприятия
	Стрелки или связи (Arrows, Links)	Показывают взаимоотношения работ. Бывают трех видов: стрелки предшествования, отношения, потоков объектов
	Перекрестки (Junctions) или логические операторы «И» (AND), «ИЛИ» (OR), исключающее «ИЛИ» (XOR) и их комбинации	Операторы, позволяющие описать ветвление или слияние процессов
	Объекты ссылок (Referent)	Объект, используемый для описания ссылок на другие диаграммы модели, циклические переходы в рамках одной модели, различные комментарии к функциям и перекресткам

Блоки работы или действия (boxes, activities)

Основные характеристики:

1. Блоки работы или действия представляют собой прямоугольники на диаграмме PFDD и называются *функциональными элементами* или *элементами поведения* (Unit of Behavior, UOB) и обозначают событие, стадию процесса или принятие решения, являются центральным компонентом модели.

2. Каждый UOB имеет свое имя, отображаемое в глагольном наклонении и уникальный номер.

3. Стрелки или линии являются отображением перемещения детали между UOB-блоками в ходе процесса.

4. Действия желательно именовать по тем же правилам, что и в модели IDEF0: глаголом (одиночным или в составе фразы с существительным), обычно отображающим основной результат работы.

5. Допускается использовать стиль именования через имя, выраженное отглагольным существительным.

6. Действия имеют входы и выходы, но не поддерживают управления и механизмы как функции в нотации IDEF0.

Например, «Применить сложение» (это действие) – глагол с существительным. Или другой пример: «Применение сложения» (это работа) – имя, выраженное отглагольным существительным.

Стрелки (линии) или связи (arrows, links)

Основные характеристики:

1. Стрелки (связи) показывают взаимоотношения между действиями (работами).

2. Стрелки всегда однонаправленные и могут быть направлены куда угодно, т. е. могут начинаться и заканчиваться на любой стороне блока.

3. Обычно диаграммы рисуют таким образом, чтобы стрелки были направлены слева направо и сверху вниз.

4. Стрелки бывают следующих видов:

– *старшая* (Precedence) – сплошная линия, связывающая UOB. Рисуется слева направо или сверху вниз;

– *отношения* (Relational Link) – пунктирная линия, используемая для изображения связей между UOB;

– *потоки объектов* (Object Flow) – стрелка с двумя наконечниками используется для описания того факта, что объект (деталь) используется в двух или более единицах работы, например, когда объект порождается в одной работе и используется в другой.

Связь «Старшая стрелка»

Основные характеристики:

1. Связь типа «Временное предшествование» (Precedence).

2. Обозначение – сплошная стрелка, связывающая единицы работ.

3. Показывает, что работа-источник должна быть закончена прежде, чем начнется работа-цель.

4. Во многих случаях завершение одного действия инициирует начало выполнения другого. Связь именуют так, чтобы была понятна причина ее появления.

Например, прежде чем приступить к лепке котлет, нужно порубить для них мясо. Временная шкала выполнения действий показана под иллюстрирующим рисунком. Вертикальными линиями показано начало и окончание действий. Время окончания 1.1 и время начала 1.2 может совпадать, может не совпадать.

Стрелка отношений

Основные характеристики:

1. Связь типа нечеткое отношение (Relational), которое изображается в виде пунктирной линии.

2. Используется для изображения связи между единицами работ, а также между единицами работ и объектами ссылок.

3. Используется, когда невозможно описать связи с использованием предшествующих или объектных связей. Значение такой связи должно быть четко

определено с помощью названия и описания стрелки, так как связи такого типа сами по себе не предполагают никаких ограничений.

4. Применение нечетких отношений: отображение задержки между действиями; отображение взаимоотношений между параллельно выполняющимися действиями.

5. Нечеткое отношение является альтернативой временному предшествованию и объектному потоку в смысле задания последовательности выполнения работ – работа-источник не обязательно должна закончиться, прежде чем работа-цель начнется. Более того, работа-цель может закончиться прежде, чем закончится работа-источник.

Поток объектов

Основные характеристики:

1. Стрелка, изображающая поток объектов (Object Flow).

2. Стрелка с двумя наконечниками.

3. Применяется для описания того, что объект используется в двух и более единицах работ, например, когда объект порождается в одной работе, а используется в другой.

4. Применяется для описания того, что результатом выполнения исходного действия является некоторый объект, который необходим для выполнения конечного действия.

Перекрестки (Junctions)

Основные характеристики:

1. Объект, обозначенный J1, – называется *перекрестком* (Junction).

2. Все перекрестки в PFDD-диаграмме нумеруются, каждый номер имеет префикс «J».

3. Перекрестки используются для отображения логики взаимодействия стрелок (потоков) при слиянии и разветвлении или для отображения множества событий, которые могут или должны быть завершены перед началом следующей работы.

4. Различают перекрестки для *слияния* (Fan-in Junction) и *разветвления* (Fan-out Junction) стрелок.

5. Перекресток не может использоваться одновременно для слияния и для разветвления стрелок.

6. При внесении перекрестка в диаграмму необходимо указать тип перекрестка.

7. Все соединения на диаграммах должны быть парными, т. е. любое разворачивающее соединение должно иметь парное себе сворачивающее соединение, типы соединений не обязательно должны совпадать.

8. Соединения могут комбинироваться для создания более сложных ветвлений.

9. В отличие от других методологий (IDEF0, DFD) стрелки могут сливаться или разветвляться только через перекрестки.

В таблице 6.2 приведены возможные типы перекрестков.

Таблица 6.2

Обозначение логических связок

№	Обозначение	Наименование	Слияние стрелок	Разветвление стрелок
1		AND асинхронное И	Все предшествующие процессы должны быть завершены	Все предшествующие процессы должны быть запущены
2		AND синхронное И	Все предшествующие процессы завершены одновременно	Все следующие процессы запускаются одновременно
3		OR асинхронное ИЛИ	Один или несколько предшествующих процессов должны быть завершены	Один или несколько следующих процессов должны быть запущены
4		ORсинхронное ИЛИ	Один или несколько предшествующих процессов завершаются одновременно	Один или несколько следующих процессов запускаются одновременно
5		XOR исключающее ИЛИ	Только один предшествующий процесс завершен	Только один следующий процесс запускается

Объекты ссылок

Основные характеристики:

1. Выражает идею, концепцию данных, которые нельзя связать со стрелкой, перекрестком, работой.
2. Используется при построении диаграммы для привлечения внимания пользователя к каким-либо важным аспектам модели.
3. Объекты ссылки должны быть связаны с единицами работ или перекрестками линиями.
4. Объект ссылки изображается в виде прямоугольника, похожего на прямоугольник работы.
5. В качестве имени можно использовать имя какой-либо стрелки, процесса, действия с других диаграмм или имя сущности из модели данных.
6. Кроме имени следует указывать тип объекта ссылки (см. таблицу 6.3).
7. Официальная спецификация IDEF3 различает три стиля объектов ссылок:

– безусловные (unconditional);
– синхронные (synchronous);
– асинхронные (asynchronous).

Таблица 6.3

Обозначение дополнительных элементов диаграмм

Элемент	Назначение
Объект (Object)	Используется для описания того, что в действии принимает участие какой-либо заслуживающий отдельного внимания объект
Ссылка (GOTO)	Используется для реализации цикличности выполнения действий. Этот объект также может относиться к перекрестку

Единица действия UOB (Unit Of Behavior)	Используется для многократного отражения на диаграмме одного и того же действия, но без цикла
Заметка (Note)	Используется для документирования какой-либо важной информации общего характера, относящейся к изображаемому на диаграммах. Служит альтернативой методу помещения текстовых заметок непосредственно на диаграммах
Уточнение (Elaboration, ELAB)	Для уточнения или более подробного описания, изображаемого на диаграмме. Обычно используется для детального описания разветвления или слияния стрелок на перекрестках

Декомпозиция

Основные характеристики:

1. Каждый функциональный блок (UOB) может иметь последовательность декомпозиций и, следовательно, может быть детализирован с любой необходимой точностью.

2. *Декомпозиция* – представление каждого функционального блока (UOB) с помощью отдельной IDEF3-диаграммы.

3. При этом диаграмма будет называться *дочерней* по отношению к *первичной*, а та, соответственно, *родительской*.

4. Применение принципа декомпозиции в IDEF3 позволяет структурированно описывать процессы с любым требуемым уровнем детализации.

Нумерация работ

Каждому действию присваивается уникальный номер (идентификатор), который никогда не меняется, в отличие от имени действия, которое в процессе уточнения и редактирования модели может меняться. Даже если действие будет удалено, его идентификатор не должен вновь использоваться для других действий. Обычно номер работы состоит из номера родительской работы, версии декомпозиции и собственного номера работы на текущей диаграмме.

Пример 1. Построение диаграммы данных IDEF3.

С помощью диаграмм IDEF3 обычно моделируют последовательности работ, имеющие технологические и временные связи. К таким моделям можно отнести проект разработки системы службы занятости, который и будет рассмотрен в данном примере.

1. Разработка технического задания.
1.1. Составление технического задания.
1.2. Утверждение технического задания.
2. Анализ.
2.1. Определение объектов системы и их атрибутов.
2.2. Определение категорий пользователей.
2.3. Создание запросов к системе.
3. Разработка модульной структуры.
3.1. Разработка модульной структуры всей системы.
3.2. Разработка модульной структуры подсистемы обработки запросов, определения категории пользователей.
3.3. Разработка модульной структуры подсистемы экспертных оценок.

3.4. Разработка модульной структуры подсистемы профессиональных и психологических тестов.

3.5. Разработка модульной структуры контроля успеваемости студентов.

4. Проектирование БД.

4.1. Проектирование логической структуры БД.

4.2. Проектирование физической структуры БД.

4.3. Определение взаимосвязей между БД.

4.4. Выбор СУБД.

В структуре работ определяются диаграммы, добавив в них взаимосвязи между работами (рис. 6.1, 6.2, 6.3).

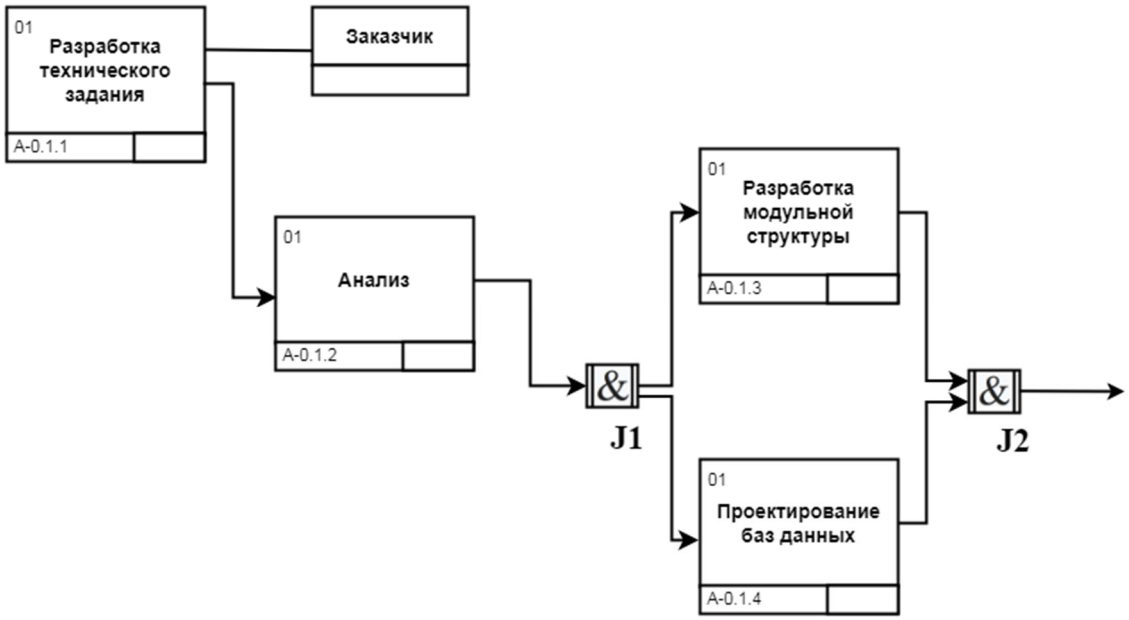


Рис. 6.1. Контекстная диаграмма «Разработка системы службы занятости»

Рис. 6.2. Диаграмма декомпозиции работы «Анализ»

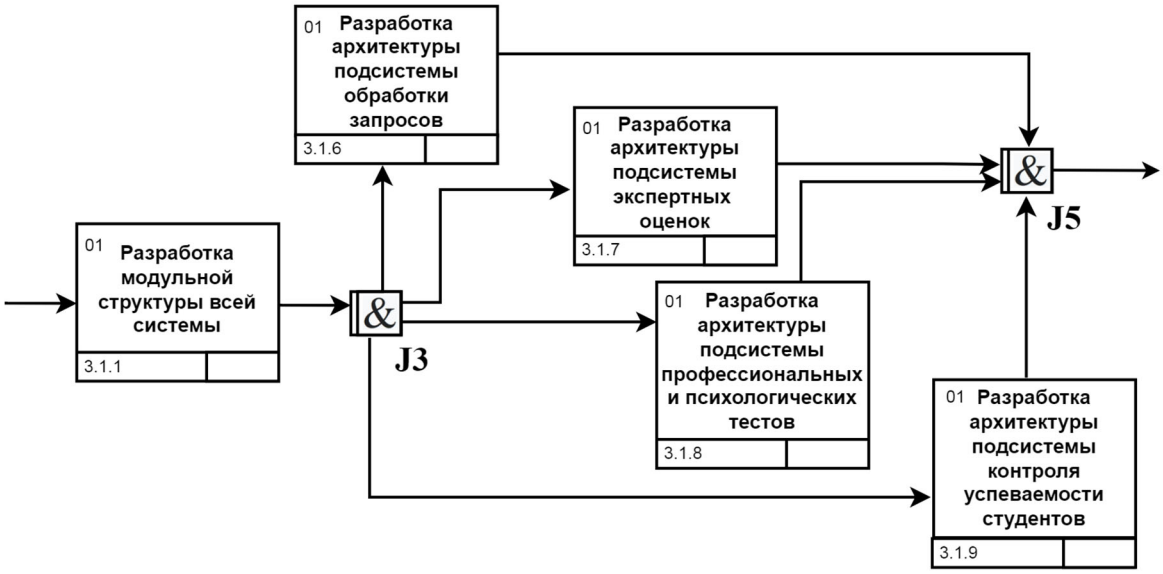


Рис. 6.3. Диаграмма декомпозиции работы «Разработка модульной структуры»

Задания к работе

1. В качестве исходных данных использовать проект, созданный в лабораторной работе № 2.

2. Создать контекстную диаграмму IDEF3.

3. Выполнить декомпозицию одной из функций верхнего уровня, построив диаграмму второго уровня.

4. Проверить модель на соответствие правилам нотации IDEF3.

5. Сохранить диаграмму и экспортировать ее в формат PNG-файла.

Методика выполнения работы

Запустить онлайн-редактор Draw.io и создать проект, используя задание из лабораторной работы № 2. Создать проект IDEF3. Выбрать функцию первого уровня, которая требует детального описания, и выполнить ее декомпозицию. Сохранить проект и экспортировать контекстную диаграмму и диаграмму второго уровня в PDF-файл.

Рекомендации по обработке и оформлению полученных результатов

Подготовить пояснительный текст к модели. Представить результаты работы в виде отчета.

Вопросы для самоконтроля

1. Что такое IDEF3-диаграммы и для какой цели они используются?

2. К какому типу моделей относятся IDEF3-диаграммы – к статическому или динамическому?

3. Назвать основные элементы нотации IDEF3?

4. Какие требования предъявляются к моделям в нотации IDEF3?

Рекомендуемая литература

1.Грекул, В. И. Проектирование информационных систем : практикум: учебное пособие / В. И. Грекул, Н. Л. Коровкина, Ю. В. Куприянов. – Москва : Национальный Открытый Университет «ИНТУИТ.ru», 2012. – 187 с. – ISBN 978-5-9556-0133-5.

3. Федотова, Д. Э. CASE-технологии: практикум / Д. Э. Федотова, Ю. Д. Семенов, К. Н. Чижик. – Москва : Горячая линия-Телеком, 2005. – 160 с. – ISBN 5-93517-121-X.

Лабораторная работа № 7
Моделирование информационных систем с использованием UML

Тема занятия: моделирование информационных систем с использованием UML.

Цель занятия: освоить основы работы с диаграммами UML для моделирования различных аспектов информационных систем.

Материалы и программное обеспечение

1. Исходные данные берутся из вариантов заданий, предложенных в лабораторной работе № 2.

2. Программное обеспечение для моделирования ИС: MS Visio, StarUML, Enterprise Architect, онлайн-сервисы: Draw.io, Visual Paradigm.

Краткие теоретические сведения

Процесс бизнес-моделирования может быть реализован в рамках различных методик, отличающихся прежде всего своим подходом к тому, что представляет собой моделируемая система. В соответствии с различными представлениями о системе методики принято делить на объектные и функциональные (структурные).

Объектные методики рассматривают моделируемую систему как набор взаимодействующих объектов – производственных единиц. Объект определяется как осязаемая реальность – предмет или явление, имеющие четко определяемое поведение. Целью применения данной методики является выделение объектов, составляющих систему, и распределение между ними ответственностей за выполняемые действия.

Принципиальное отличие между функциональным и объектным подходом заключается в способе декомпозиции системы. Объектно-ориентированный подход использует объектную декомпозицию, при этом статическая структура описывается в терминах *объектов и связей* между ними, а поведение системы описывается в терминах *обмена сообщениями* между объектами. Построение бизнес-модели организации, позволяющей перейти от модели сценариев использования к модели, определяющей отдельные объекты, участвующие в реализации бизнес-функций.

Концептуальной основой объектно-ориентированного подхода является объектная модель, которая строится с учетом следующих принципов:

- абстрагирование;
- инкапсуляция;
- модульность;
- иерархия;
- типизация;
- параллелизм;
- устойчивость.

Основными понятиями объектно-ориентированного подхода являются объект и класс.

Объект – предмет или явление, имеющий четко определенное поведение и обладающий *состоянием*, *поведением* и *индивидуальностью*. Структура и поведение схожих объектов определяют общий для них класс.

Класс – множество объектов, связанных общностью структуры и поведения. Следующую группу важных понятий объектного подхода составляют наследование и полиморфизм.

Полиморфизм – понятие, которое может быть интерпретировано как способность класса принадлежать более чем одному типу.

Наследование – построение новых классов на основе существующих с возможностью добавления или переопределения данных и методов.

Важным качеством объектного подхода является согласованность моделей деятельности организации и моделей проектируемой информационной системы от стадии формирования требований до стадии реализации. По объектным моделям может быть прослежено отображение реальных сущностей моделируемой предметной области (организации) в объекты и классы информационной системы.

В качестве языка моделирования объектного подхода используется унифицированный язык моделирования UML, который содержит стандартный набор диаграмм для моделирования.

Диаграмма (Diagram) – графическое представление множества элементов. Чаще всего она изображается в виде связного графа с вершинами (сущностями) и ребрами (отношениями) и представляет собой некоторую проекцию системы.

Преимущества объектно-ориентированного подхода.

– Объектная декомпозиция дает возможность создавать модели меньшего размера путем использования общих механизмов, обеспечивающих необходимую экономию выразительных средств. Использование объектного подхода существенно повышает уровень унификации разработки и пригодность для повторного использования, что ведет к созданию среды разработки и переходу к сборочному созданию моделей.

– Объектная декомпозиция позволяет избежать создания сложных моделей, так как она предполагает эволюционный путь развития модели на базе относительно небольших подсистем.

– Объектная модель естественна, поскольку ориентирована на человеческое восприятие мира.

К недостаткам объектно-ориентированного подхода относятся высокие начальные затраты. Этот подход не дает немедленной отдачи. Эффект от его применения сказывается после разработки двух-трех проектов и накопления повторно используемых компонентов. Диаграммы, отражающие специфику объектного подхода, менее наглядны.

УМЛ (англ. Unified Modeling Language, UML) – универсальный язык моделирования (визуального), который используется для описания, проектирования и документирования систем.

Объект – сущность, имеющая некоторое состояние (информацию) и предоставляющая программисту набор операций, с помощью которых можно изменять

или проверять это состояние. В программной среде объект является моделью или абстракцией реальной сущности.

Объекты в программе взаимодействуют между собой через свои *связи*.

Связь обозначает соединение, через которое объект может вызывать операции другого объекта или один объект перемещает данные в другой объект.

Инкапсуляция – принцип, согласно которому внутреннее содержимое сущностей скрывается от внешнего вмешательства.

Интерфейс – абстрактный тип данных, который содержит описание методов, как объект взаимодействует с окружающим миром.

Состояние объекта описывается значениями его свойств (атрибутов), которые хранятся в специальных переменных (*полях* объекта). Все объекты в программной среде отличаются друг от друга, т. е. обладают уникальностью. Некоторые свойства (атрибуты) объекта неизменны и их значения присущи только ему (уникальны).

Объект может состоять из других объектов, являющихся его атрибутами (включение одного объекта в другой: объект-контейнер, объект-атрибут).

В качестве атрибутов объект может использовать ссылки на другие объекты.

Объекты взаимодействуют между собой с помощью *сообщений*. Принимая сообщение, объект реагирует на него некоторым действием.

Метод объекта – действие, которое может выполнять объект. Говорят, что *метод* определяет некоторый *сервис* или *функцию* класса. В объектно-ориентированном программировании *метод* представляет собой процедуру или функцию, принадлежащую *классу* или *объекту*.

Совокупность *методов* и представляет собой *интерфейс* объекта. Некоторые атрибуты объекта могут быть доступны извне через его методы, другие предназначены только для внутреннего пользования. Однотипные объекты объединяются в классы.

Класс представляет собой *модель* для создания объектов определенного типа, описывающую их структуру (набор полей и их начальное состояние) и определяющую методы для работы с этими объектами, можно сказать, что он играет роль шаблона для создания объектов.

Все объекты одного и того же класса обладают одинаковым *интерфейсом* и реализуют объект одинаково. Их отличие может быть только в текущем состоянии. Индивидуальные объекты – *экземпляры класса*.

Экземпляр – конкретное представление объекта, принадлежащего некоторому классу. Объект представлен в программировании собирательным понятием, а экземпляром считается отдельный, конкретный объект, созданный в памяти. Свойствам экземпляра присваиваются определённые значения, которые будут отличать его от других экземпляров этого типа объекта.

Наследование представляет собой механизм, с помощью которого можно создавать новые классы на основе существующего (родительского). Свойства и функциональность родительского класса заимствуются новым классом. Родительский класс называется при этом *суперклассом*, а производный *подклассом* или *производным классом*.

Механизм наследования позволяет выделить общие части классов. Каким образом проводить наследование и создавать иерархию классов, какие классы выделять – решает сам программист.

Иерархия классов – описание отношений наследования между классами.

Некоторые классы, которые просто объединяют общие характеристики других классов, но по которым невозможно создать объекты, называются *абстрактными*.

Классы, экземпляры которых могут существовать в системе, называются *конкретными*. Если класс наследует методы и атрибуты одного класса – *простое наследование*, если нескольких – *множественное*. *Множественное наследование* позволяет объединять характеристики разных классов в одном.

Полиморфизм представляет собой механизм, позволяющий реализовывать обработку данных (некоторую функцию) для различных типов данных, то есть скрывать различные реализации за общим интерфейсом.

Существуют следующие основные отношения между классами: *ассоциация*, *наследование*, *использование*, *агрегация*.

Ассоциация описывает двухстороннюю связь между объектами разных классов, например: преподаватель и студент, работник и зарплата и т. д.

Студент может выбрать преподавателя в качестве научного руководителя, а преподаватель студента для выполнения учебных работ. Таким образом, между ними устанавливается связь-ассоциация «руководит – имеет руководителя».

Для ассоциаций характерно указание ее мощности: 1 – в точности одна связь, N – неограниченное число, 0..N – ноль или больше, 1..N – одна или больше, 2..6 – указанный интервал.

Наследование представляет собой отношение между классами (объектами), когда один объект наследует (структуру) все или часть атрибутов и методов (поведение) другого (*одиночное наследование*) или других объектов (*множественное наследование*).

Агрегация – представляет собой связь между классами типа «содержит» или «состоит из», когда один класс включает другой в качестве атрибута (объект-контейнер). Агрегация в отличие от других связей показывает отношение целого и его части. Агрегация может рассматриваться как специализированный случай ассоциации.

Основные типы агрегации приведены в таблице 7.1:

Таблица 7.1

Графическое изображение связей между классами

№	Отношения между классами	Обозначения UML
1	Ассоциация	
2	Наследование	
3	Агрегация по ссылке	
4	Агрегация по значению	
5	Использование	

1) агрегация по ссылке – в этом случае один объект (целое) содержит ссылку на другой объект (часть);

2) агрегация по значению (композиция) – в этом случае объект (целое) физически содержит другой объект (часть).

Объектно-ориентированный анализ и построение модели предметной области

Объектный анализ с точки зрения объектно-ориентированного анализа (ООА) сводится к исследованию объектов *предметной области*. Анализ включает написание технического задания и построение моделей предметной области (концептуальной модели). На этапе анализа определяются названия объектов, их логические атрибуты и связи между объектами. Каждый объект в анализируемой системе представляется в виде концептуального класса, изображаемого прямоугольником, внутри которого указывают имя объекта. Пример диаграммы класса и набора объектов, созданных на основе класса, представлен на рисунке 7.1.

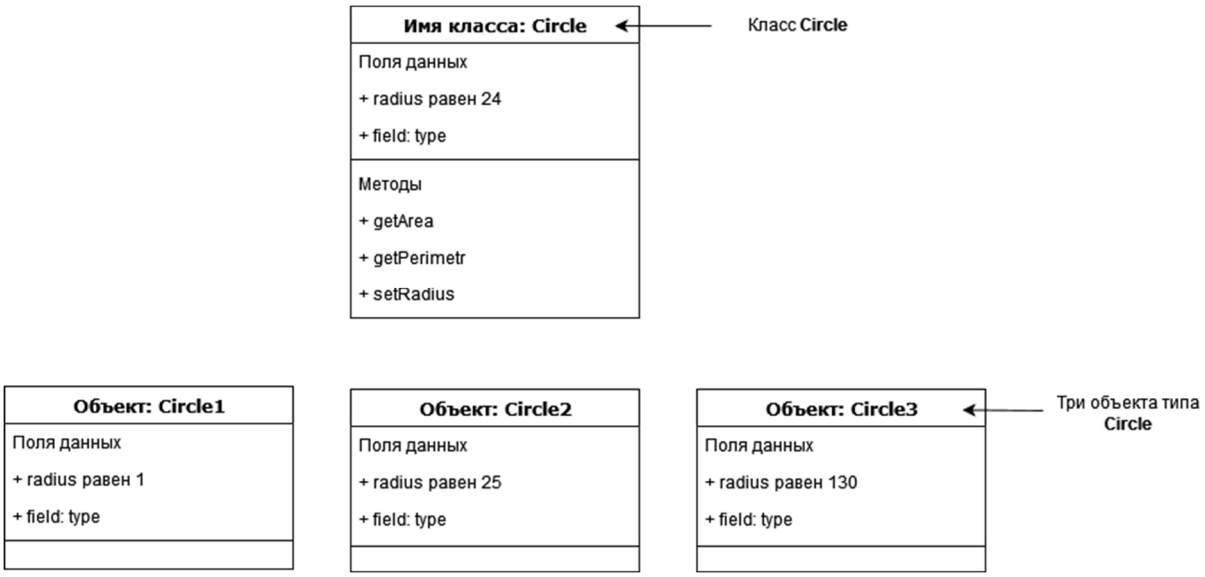


Рис. 7.1. Пример диаграммы класса Circle

На стадии проектирования, как правило, осуществляется выделение объектов, ответственных за управление и интерфейс. Объекты представляются в виде иерархии классов, которые в конечном итоге будут преобразованы в программный код. В процессе выполнения программы устанавливаются динамические отношения между объектами, сообщения, которыми они обмениваются, определяются программные объекты и способы их взаимодействия для выполнения системных требований (рис. 7.2).

Для графической поддержки перечисленных этапов в создании программного обеспечения каждый язык применяет набор диаграмм, выполняемых в определенной нотации. UML – это язык для определения, визуализации, конструирования и документирования артефактов программных систем.

Рис. 7.2. Стадии разработки проекта

Унифицированный язык моделирования UML

Основное назначение и принцип использования языка моделирования заключаются в совокупности подходов к описанию предметной области и моделированию структуры информационной системы. В фокусе рассмотрения при данном подходе находится работа информационной системы – с одной стороны (воздействия пользователей), а с другой – реакций системы. Основным понятием воздействия является вариант (прецедент). После описания вариантов (прецедентов) выполняется дальнейшее моделирование системы.

Основные типы диаграмм UML

1. Диаграмма вариантов (прецедентов) использования (Use case diagram). Описывает взаимодействие пользователя с системой.

Основные элементы: *акторы, варианты использования, связи*.

2. Диаграмма последовательности (Sequence diagram). Показывает взаимодействие объектов во времени.

Основные элементы*: объекты, сообщения, временная шкала*.

3. Диаграмма классов (Class diagram). Представляет структуру системы в виде классов, их атрибутов, методов и связей.

Основные элементы*: классы, ассоциации, наследование.*

4. Диаграмма состояний (State diagram). Моделирует поведение системы, изменения состояния во времени.

Основные элементы*: состояние, интерфейсы, объекты*.

5. Диаграмма кооперации или сотрудничества (Collaboration diagram). Описывает взаимодействия объектов.

Основные элементы*: сообщения, объекты, взаимодействие*.

6. Диаграмма деятельности (активности) (Activity diagram). Моделирует процессы и потоки управления.

Основные элементы*: действия, переходы, ветвления*.

7. Диаграмма компонентов (Component diagram). Описывает физические компоненты системы и их взаимодействие.

Основные элементы*: компоненты, интерфейсы, зависимости*.

8. Диаграмма развертывания (Deployment diagram).

Основные элементы*: компоненты, интерфейсы, зависимости*.

Разработка проекта с использованием UML

С точки зрения вариантов(прецедентов) процесс разработки *проекта информационной системы* охватывает варианты, которые описывают поведение системы, наблюдаемое конечными пользователями. Этот вид диаграммы описывает (специфицирует) неистинную организацию программной системы, а те аспекты воздействий на систему, от которых зависит формирование программной (системной) архитектуры. В языке UML статические аспекты этого вида передаются диаграммами вариантов, а динамические – диаграммами взаимодействия, состояний и деятельности.

С точки зрения проектирования процесс разработки охватывает классы, интерфейсы и кооперации, формирующие словарь задачи и ее решения. Этот вид поддерживает прежде всего функциональные требования, предъявляемые к системе, то есть те услуги, которые она должна предоставлять конечным пользователям. С помощью языка UML статические аспекты этого вида можно передавать диаграммами классов и объектов, а динамические – диаграммами взаимодействия, состояний и деятельности.

Процесс проектирования объектной системы может быть выполнен в следующей последовательности.

1. Используется Use case diagram для отображения списка операций, которые должна выполнять проектируемая система.

2. Каждый Use case (вариант) детализируется при помощи Sequence diagram. На этой диаграмме отображаются объекты из предметной области, таким образом, получаются экземпляры некоторых классов и их взаимодействие.

3. Создается статическая картина предметной области, используя Class diagram. Данная диаграмма – глобальный фундамент проектируемой системы.

4. Формируются компоненты информационной системы, которые отображаются на Component diagram, где показывается состав и зависимость компонентов между собой.

5. Моделируется размещение компонент информационной системы на инфраструктуре вычислительной системы, используя Deployment diagram.

6. Для описания динамики системы используются диаграммы кооперации (Collaboration diagram) и состояний (State diagram).

Таким образом, UML используется на всех этапах разработки информационной системы: от анализа требований до реализации и тестирования.

Диаграмма вариантов использования (Use case diagram)

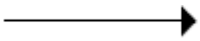
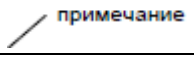
Диаграмма вариантов использования применяется для моделирования бизнес-процессов в организации и требований к информационной системе (таб. 7.2), пример, которой представлен на рисунке 7.3.

Вариант использования представляет собой последовательность действий (транзакций), выполняемых системой в ответ на событие, инициируемое некоторым внешним объектом (действующим лицом). При составлении *вариантов использования* необходимо рассмотреть все внешние по отношению к системе события, на которые система должна реагировать. По масштабу вариант использования может быть небольшим (системный прецедент) или крупным (бизнес-прецедент). На практике *вариант использования*, как правило, описывает некоторую очевидную *пользовательскую функцию*, которая содержится в общем сценарии поведения системы.

Актер, действующее лицо (actor) – роль, которую пользователь играет по отношению к системе. Действующие лица представляют собой роли, а не конкретных людей или наименования работ. Актер может создавать действующих лиц в системе, может взаимодействовать с системой или использовать систему. В UML при моделировании можно использовать нескольких актеров, но при этом нельзя забывать, что они являются внешними по отношению к системе.

Таблица 7.2

Основные компоненты диаграммы использования

№	Назначение	Обозначение
1	Замечание (note)	
2	Актер (actor)	
3	Однонаправленная связь	
4	Примечание	

Однонаправленная связь соединяет актеров и варианты использования.

Варианты использования могут быть связаны друг с другом двумя видами связей: связями *расширения* (extends) и *использования*.

Вариант использования B *расширяет* вариант использования A, если B определяет дополнительные действия, которые вставляются в некоторое место в сценарии работы A.

Вариант использования B *использует* (uses, или включает, includes) вариант использования A, если B включает полностью сценарий A где-то внутри своего сценария работы.

На диаграмме вариантов использования (рис. 7.3) показано взаимодействие между вариантами использования и действующими лицами. Она отражает требования к системе с точки зрения пользователя. Таким образом, варианты использования – функции, выполняемые системой, а действующие лица – заинтересованные лица (stakeholders) по отношению к создаваемой системе.

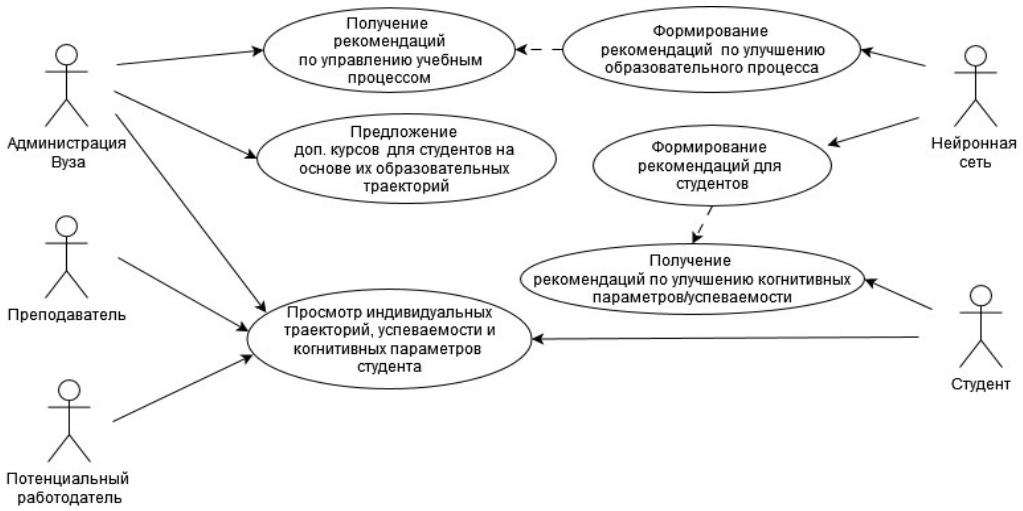


Рис. 7.3. Пример диаграммы вариантов использования

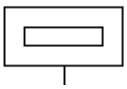
Сообщение (message) – средство, с помощью которого объект-отправитель запрашивает у объекта получателя выполнение одной из его операций.

Диаграмма последовательностей (sequence)

Взаимодействие объектов в системе происходит посредством приема и передачи сообщений между объектами (табл. 7.3). Существует два вида диаграмм взаимодействия: *диаграммы последовательностей* (sequence diagrams) и *кооперативные диаграммы* (collaboration diagrams). Диаграмма последовательностей отражает последовательность передачи сообщений между объектами (рис. 7.4).

Таблица 7.3

Основные компоненты диаграммы последовательности

№	Назначение	Обозначение
1	Объект	
2	Сообщение	
3	Возврат ответа	
4	Уничтожение объекта	
5	Сообщение самому себе	

Пунктирная линия под объектом называется линией жизни – фрагмент жизненного цикла в процессе взаимодействия. Каждое сообщение может иметь имя, аргументы или другую управляющую информацию. Прямоугольники на линии жизни – активизации показывают, когда метод становится активным.

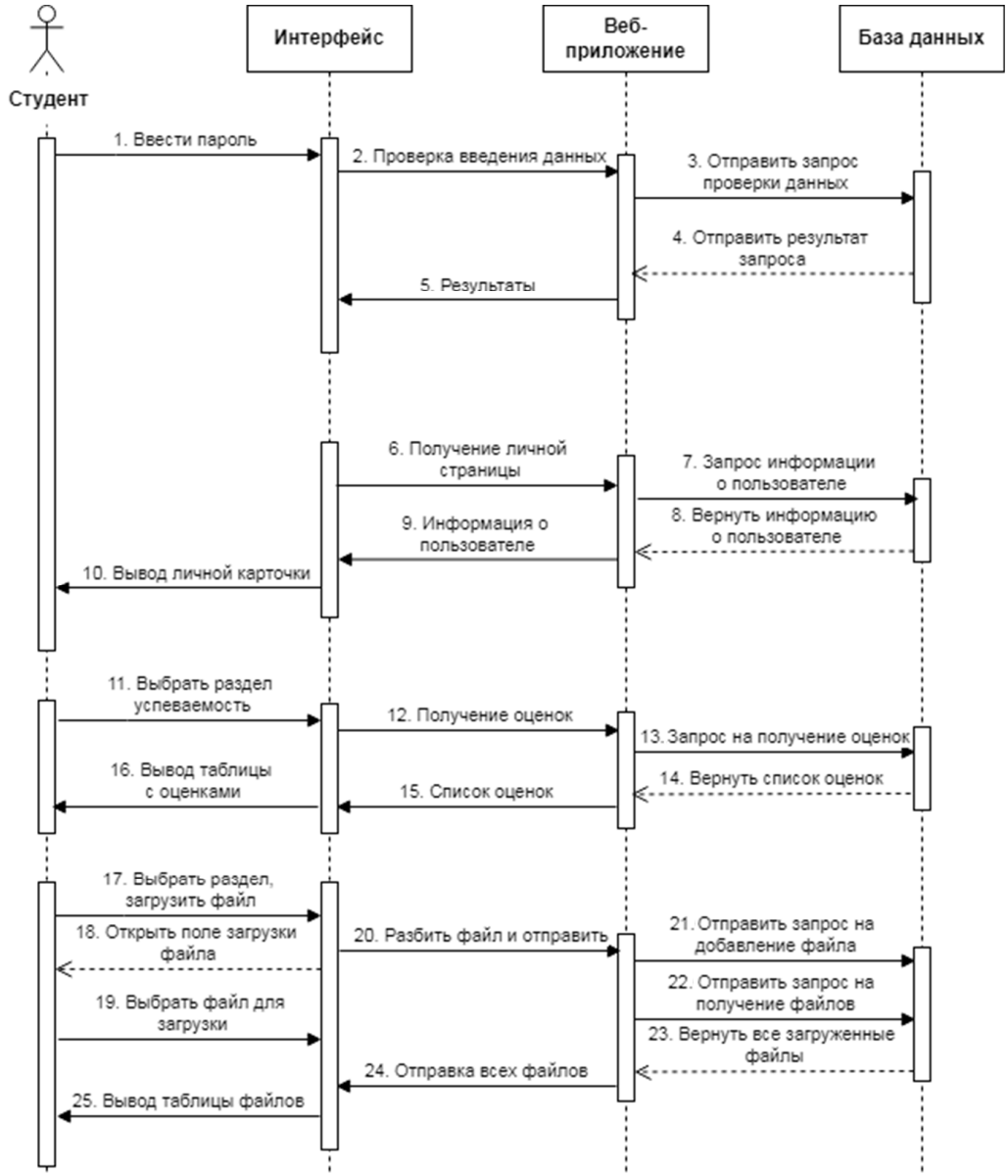


Рис. 7.4. Пример диаграммы последовательностей

Диаграмма классов (class)

Диаграмма классов является одной из важнейших диаграмм в методологии проектирования (рис. 7.5). На основе этих диаграмм строится код приложения.

Диаграмма классов – структурная диаграмма UML, демонстрирующая общую структуру *иерархии* классов системы, их кооперации, атрибутов (полей), методов, интерфейсов и взаимосвязей (отношений) между ними.

На стадии анализа диаграммы классов используются, чтобы выделить общие роли и обязанности сущностей, обеспечивающих требуемое поведение системы. На стадии проектирования диаграмма классов передает структуру классов, формирующих архитектуру системы. При этом система рассматривается со статической точки зрения.

В проекте все классы уникальны. Каждый класс – это самодостаточная структура, автономная и самоуправляемая, которая может существовать даже после завершения программы.

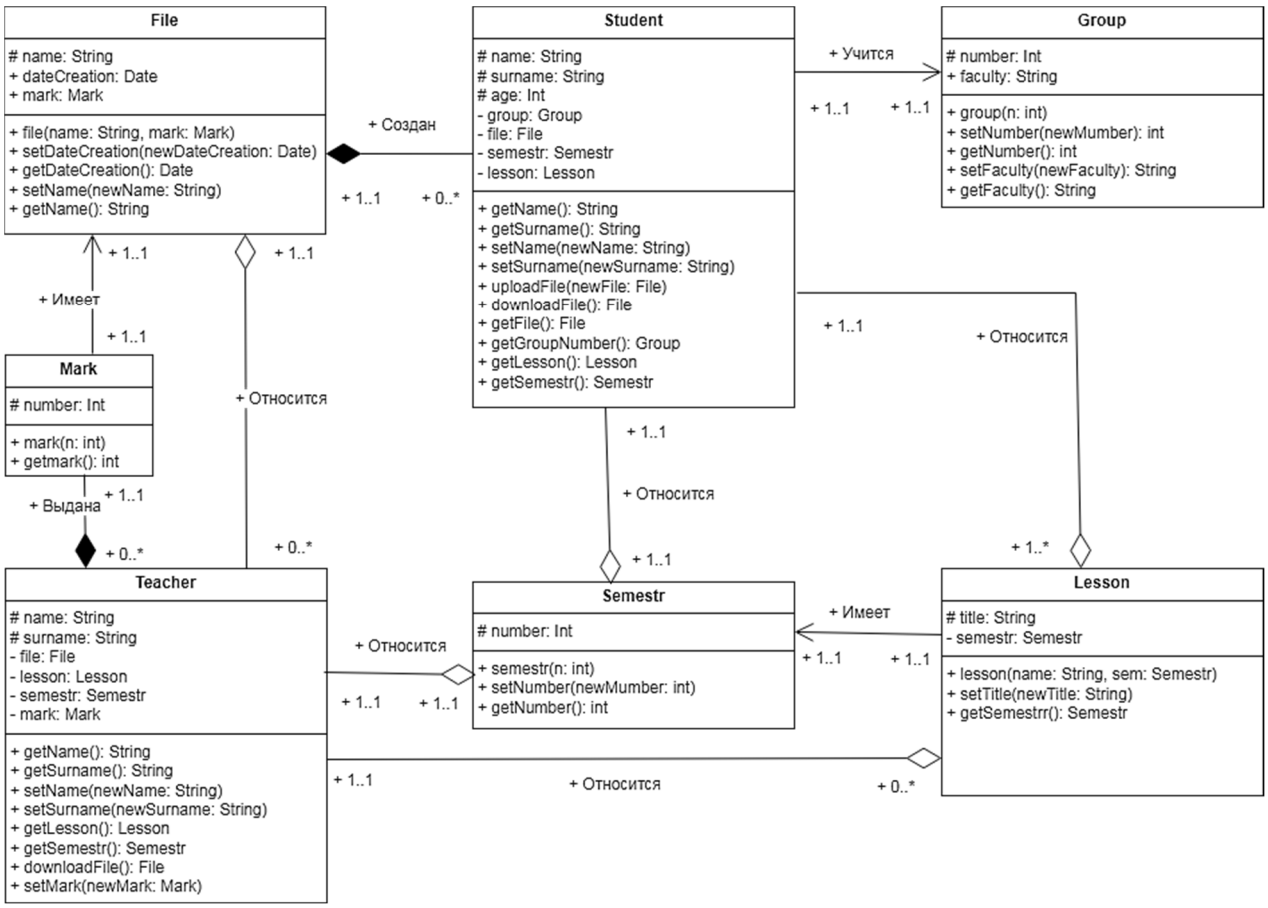


Рис. 7.5. Пример диаграммы классов

Для получения классов в программе надо разбить всю задачу на управляемые компоненты, выполняющие отдельные функции.

На диаграмме классов показываются также атрибуты классов и операции. Состояние класса определяется значениями его атрибутов.

Диаграмма сотрудничества (collaboration)

Диаграмма сотрудничества или *кооперативная диаграмма* относится ко второму типу диаграмм взаимодействия. На диаграмме представлены экземпляры объектов, соединенные линиями, обозначающими *сообщения*, обмен которыми происходит в рамках данного *варианта использования*. Все сообщения представляются в компактном виде. Данная диаграмма строится на базе *диаграммы последовательностей* и показывает то же самое, но в более компактном виде.

Диаграмма состояний (state)

Диаграмма состояний описывает поведение системы, определяет ее возможные состояния, а также условия смены состояний. Они отображают разрешённые состояния и переходы, а также события, которые влияют на эти переходы. *Диаграмма состояний* помогает визуализировать весь жизненный цикл объектов и лучше понять системы, основанные на состоянии (рис. 7.6).

Рекомендуется строить диаграммы состояний для классов пользовательского интерфейса и управляющих объектов. Основные элементы диаграммы представлены в таблице 7.4.

Таблица 7.4

Основные компоненты диаграммы состояний

№	Назначение	Обозначение
1	Вход	
2	Состояние	
3	Выход	
4	Переход в состояние	
5	Условие	

Пример диаграммы состояний представлен на рисунке 7.6.

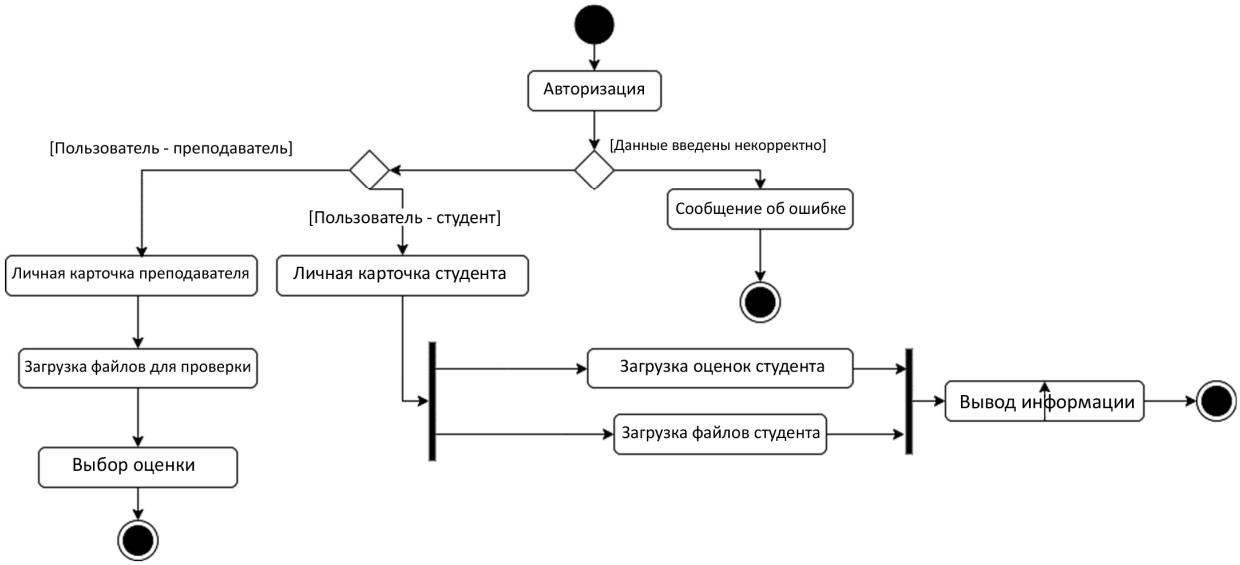


Рис. 7.6. Диаграмма состояний при работе пользователей
в системе электронного портфолио

Задания к работе

1. В качестве исходных данных использовать проект, созданный в лабораторной работе № 2.

2. Разработать UML-диаграммы для заданной предметной области: диаграмму вариантов использования, диаграмму классов и диаграмму последовательностей.

3. Оформить диаграммы с использованием профессионального инструмента.

4. Для диаграммы вариантов использования описать актеров и варианты в таблице.

5. Для диаграммы классов привести описание классов, атрибутов и методов в таблице.

Методика выполнения работы

Ознакомиться и провести анализ предметной области. Построить диаграмму вариантов использования, определив актёров (пользователей системы) и их взаимодействие с системой, а также основные функции. Создать диаграмму классов, определив ключевые сущности, их атрибуты и методы, а также связи между классами (ассоциация, наследование, агрегация). Построить диаграмму последовательностей, описав сценарий взаимодействия пользователя с системой и последовательность сообщений между объектами. Оформить диаграммы в соответствии с требованиями.

Рекомендации по обработке и оформлению полученных результатов

Каждая диаграмма должна быть понятной и содержать только необходимые элементы, при этом следует использовать единый стиль оформления, включая шрифты, цвета и размещение элементов. Все элементы диаграмм должны быть подписаны. Отчет должен включать краткое описание предметной области, построенные UML-диаграммы и таблицы с описанием элементов диаграмм.

Вопросы для самоконтроля

1. Назвать основные типы диаграмм UML.
2. Какую информацию можно получить из диаграммы вариантов использования?
3. В чем разница между ассоциацией, агрегацией и композицией в диаграмме классов?
4. Для чего используется диаграмма последовательности?
5. Какие преимущества дает использование UML на этапе проектирования систем?
6. В чем ключевые отличия между диаграммами активности и последовательности?

Рекомендуемая литература

1. Грекул, В. И. Проектирование информационных систем : учебное пособие / В. И. Грекул, Г. Н. Денищенко, Н. Л. Коровкина. – Москва : Национальный Открытый Университет «ИНТУИТ.ru», 2005. – 304 с. – ISBN 5-9556-0033-7.

2. Грекул, В. И. Проектирование информационных систем : практикум: учебное пособие / В. И. Грекул, Н. Л. Коровкина, Ю. В. Куприянов. – Москва : Национальный Открытый Университет «ИНТУИТ.ru», 2012. – 187 с. – ISBN 978-5-9556-0133-5.

3. Паршин, К. А. Методы и средства проектирования информационных систем и технологий : учебное пособие / К. А. Паршин. – Екатеринбург : УрГУПС, 2018. – 129 с.

4. OMG Unified Modeling Language. Superstructure. – URL: https://omg.org/spec/UML/2.2/Superstructure/PDF (дата обращения: 15.01.2025)

Лабораторная работа № 8
Проектирование и методологии разработки информационных систем

Тема занятия: знакомство с основными методологиями разработки информационных систем.

Цель занятия: изучить основные методологии разработки информационных систем, их преимущества и особенности применения.

Материалы и программное обеспечение

1. ГОСТ Р ИСО/МЕК 12207-2010. Процессы жизненного цикла программных средств.
2. Программное обеспечение для компьютерной презентации: PowerPoint или аналог, графический редактор.
3. Программное обеспечение для моделирования ИС: StarUML, Enterprise Architect или онлайн-сервисы: Draw.io, Visual Paradigm.

Краткие теоретические сведения

Основные методологии разработки информационных систем

Жизненный цикл разработки программного обеспечения – процесс, который используется для разработки, проектирования и тестирования программного обеспечения. Цикл состоит из нескольких этапов – планирование и анализ потребностей, определение требований, проектирование архитектуры продукта, создание или разработка продукта, тестирование продукта, развертывание на рынке и сопровождение.

1. Каскадная модель. Последовательный процесс разработки, при котором каждая стадия завершает предыдущую.

Этапы: анализ, проектирование, реализация, тестирование, эксплуатация.

Преимущества: ясная структура, простота управления проектом.

Недостатки: низкая гибкость, высокая стоимость внесения изменений.

2. V-модель. Расширение каскадной модели с акцентом на тестирование на каждом этапе.

Преимущества: строгий контроль качества, раннее выявление ошибок.
Недостатки: подходит только для проектов с четкими требованиями.

3. Инкрементальная разработка. Процесс разработки, где проект реализуется небольшими частями (инкрементами).

Преимущества: гибкость, возможность раннего использования частей системы.

Недостатки: сложность управления при увеличении числа инкрементов.

4. Спиральная модель. Итеративная модель, объединяющая элементы каскадной и инкрементальной моделей.

Преимущества: гибкость, возможность оценки рисков.

Недостатки: высокая стоимость, сложность управления.

Каскадная модель

Каскадная модель (англ. Waterfall model, модель водопада) – модель процесса разработки программного обеспечения, в которой процесс разработки вы-

глядит как поток, последовательно проходящий фазы анализа требований, проектирования, реализации, тестирования, интеграции и поддержки.

Она подразумевает строгую последовательность этапов: каждый следующий начинается только после полного завершения предыдущего (рис. 8.1).

Каскадная модель хорошо работает в условиях, где требования к проекту чётко определены с самого начала и вряд ли изменятся в процессе выполнения проекта. Это делает модель предсказуемой и упрощает планирование и оценку ресурсов.

Сейчас такая модель больше применима в сфере авиастроения, медицины и других отраслях, где нужны чётко выстроенные действия и сроки.

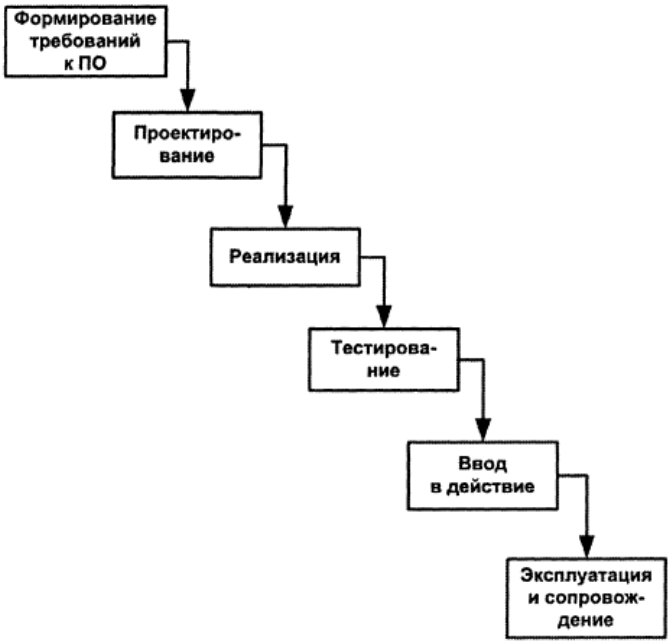


Рис. 8.1. Каскадная модель

V-модель

V-модель – методология разработки, которая описывает структурированный, последовательный процесс проектирования, создания и тестирования сложных систем, который выполняется последовательно в V-образной форме (рис. 8.2). Модель основана на объединении фазы тестирования с каждой соответствующей стадией разработки. Разработка каждого шага напрямую связана с этапом тестирования. Следующая фаза начинается только после завершения предыдущей. Каждый этап разработки напрямую связан с тестированием этого этапа.

Некоторые этапы V-модели:

1. *Анализ требований*. Определяются видение, объём и цели проекта, а также выявляются, анализируются и расставляются по приоритетам конкретные функциональные и нефункциональные требования.

2. *Проектирование системы*. Разрабатывается высокоуровневый архитектурный проект, который абстрагирует основные компоненты системы, их взаимосвязи, а также общую архитектуру программного и аппаратного обеспечения.

3. *Проектирование подсистемы (проектирование компонентов или архитектурный дизайн).* Этот этап включает детальное проектирование отдельных

функциональных компонентов или модулей, включая определение интерфейсов, структур данных, алгоритмов и рабочих процессов.

4. *Реализация.* Систему разбивают на небольшие модули. Определяется детальный дизайн модулей, Low-Level Design (LLD). Кодируется ПО, а другие артефакты разработки, такие как схемы баз данных, пользовательские интерфейсы и API, создаются в соответствии с детальным проектом.

5. *Модульное тестирование.* Отдельные компоненты или модули проверяются на функциональную правильность и соответствие подробным проектным спецификациям.

6. *Интеграционное тестирование.* Собранные подсистемы проверяются на предмет межкомпонентного взаимодействия, совместимости интерфейсов и общего поведения системы.

7. *Тестирование системы.* Полная интегрированная программная система подвергается серии тестов для проверки того, что она соответствует указанным требованиям, предполагаемому использованию и операционной среде.

8. *Приёмочное тестирование.* Заключительный этап V-модели, на котором ПО тестируется и проверяется в реальной среде предполагаемыми конечными пользователями, чтобы убедиться, что оно соответствует их требованиям.

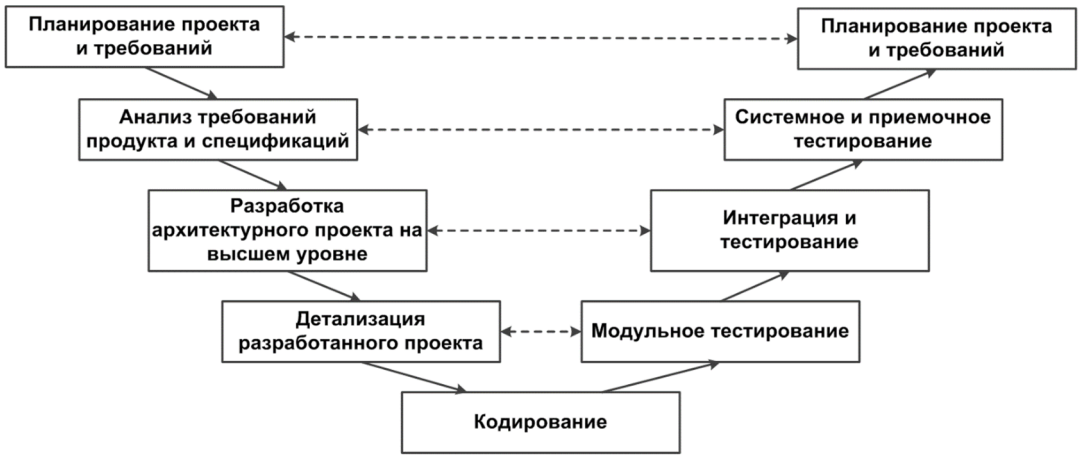


Рис. 8.2. V-модель процесса разработки программного обеспечения

Низкоуровневое проектирование (LLD) – процесс проектирования на уровне компонентов, который следует за пошаговым процессом уточнения. Представляет подробные сведения и определения фактической логики для каждого компонента системы.

Принципы V-модели

– *От большого к малому:* в V-модели тестирование выполняется в иерархической перспективе, например, требования, определенные командой проекта, создают этапы проекта высокого уровня и детального проектирования. По мере того как каждый из этих этапов завершается, требования, которые определяются, становятся все более и более уточненными и подробными.

– *Целостность данных/процессов*: принцип гласит, что успешное проектирование любого проекта требует интеграции и согласованности как данных, так

и процессов. Элементы процесса должны быть идентифицированы для каждого требования.

– *Масштабируемость*: согласно этому принципу концепция V-модели обладает гибкостью, позволяющей адаптировать произвольный ИТ-проект независимо от его сложности, продолжительности или масштаба.

– *Перекрестные ссылки*: прямая зависимость между требованиями и соответствующей деятельностью по тестированию называется перекрестными ссылками.

– *Материальная документация*: согласно этому принципу каждый новый проект должен вызывать как формирование нового документа, так и его изменения.

Инкрементальная разработка

Инкрементальная модель разработки – методология разработки программного обеспечения, основанная на последовательном выпуске функциональных блоков (модулей) продукта (рис. 8.3). Каждый блок (модуль) представляет собой работающее программное решение, которое может быть дополнено новыми функциями на последующих этапах разработки.

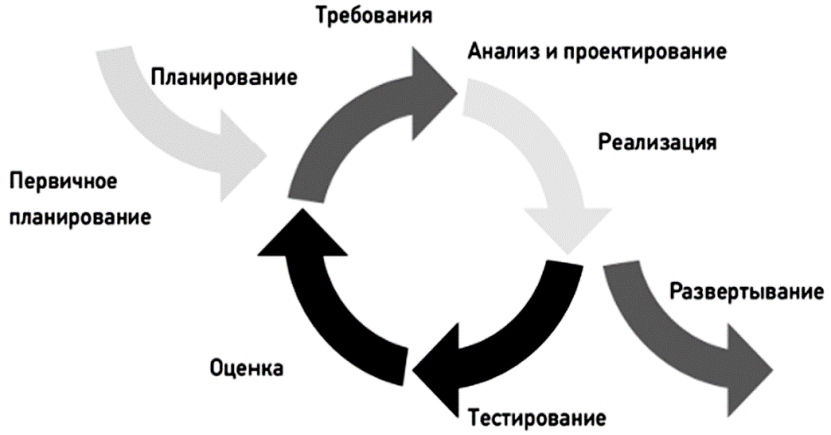


Рис. 8.3. Инкрементальная модель разработки программного обеспечения

Цикл разработки продукта разделен на более малые легко создаваемые блоки (модули). Каждый блок (модуль) проходит через фазы определения требований, проектирования, кодирования, внедрения и тестирования. Процедура разработки по инкрементальной модели предполагает выпуск на первом большом этапе продукта в базовой функциональности, а затем уже последовательное добавление новых функций, так называемых инкрементов. Процесс продолжается до тех пор, пока не будет создана полная система.

Область использования инкрементальных моделей ограничена и предполагает, что отдельные запросы на изменение продукта ясны, могут быть легко формализованы и реализованы.

Условия применения инкрементальной модели

Инкрементальную модель следует применять в случае, когда основные требования к системе четко определены и понятны, а некоторые детали могут дорабатываться с течением времени и требуется ранний вывод продукта на рынок.

Также имеет смысл применять данный подход, если имеется несколько целей у программного продукта.

Преимущества инкрементальной модели.

– *Небольшие начальные инвестиции.*

– *Быстрое получение обратной связи от пользователей* и возможность оперативно обновлять техническое задание.

– *Исправление ошибок приводит к небольшим финансовым затратам.* Если при разработке возникла ошибка, её исправление обойдётся дешевле по сравнению с другими моделями.

– Недостатки инкрементальной модели.

– Необходимо установить *единое понимание требований*, поскольку работа различных специалистов может приводить к значительному отличию в функциональности продукта.

– Возрастает роль менеджера проекта при осуществлении *контроля* над работой команды. Необходимо строго соблюдать графики выполнения важных модулей проекта.

Спиральная модель

Спиральная модель разработки – модель процесса разработки программного обеспечения, основанная на рисках, которая сочетает в себе элементы итеративного и водопадного подходов (рис. 8.4). В схематическом представлении модель выглядит как спираль со множеством витков. Точное количество витков спирали неизвестно, зависит от рисков и может варьироваться от проекта к проекту.

Рис. 8.4. Спиральная модель разработки программного обеспечения

Каждый *виток* спирали называется *фазой* или итерацией процесса разработки программного обеспечения и состоит из следующих этапов:

- определение условий и целей для итерации;
- проведение анализа рисков;
- разработка прототипа и тестирование;
- оценка результатов и планирование следующей итерации.

Спиральная модель лучше всего работает в проектах, где управление рисками имеет решающее значение. К ним в первую очередь относятся крупные и сложные проекты разработки программного обеспечения, а также проекты со значительной степенью неопределённости или высоким уровнем риска.

Преимущества спиральной модели: предоставление управления рисками, итеративное совершенствование и повышение удовлетворённости клиентов. Модель также обеспечивает гибкость при внесении изменений на любом этапе разработки и возможность обнаруживать и исправлять ошибки на ранних этапах процесса, что приводит к повышению качества и надёжности программного обеспечения.

Недостатки спиральной модели: высокая сложность и необходимость в наличии экспертных знаний по анализу рисков, что может увеличить ресурсы, необходимые для реализации. Отсутствие чётких временных рамок может затруднить мониторинг и контроль графика проекта, кроме того, этапы планирования, оценки и управления рисками также могут увеличить общую стоимость проекта.

Спиральная модель похожа на инкрементальную, но делает акцент на анализ рисков. Она хорошо работает для решения критически важных бизнес-задач, когда неудача несовместима с деятельностью компании, в условиях выпуска новых программ, при необходимости научных исследований и практической апробации.

Спиральная модель предполагает четыре этапа для каждого витка:

– планирование;
– анализ рисков;
– конструирование;
– оценка результата и при удовлетворительном качестве переход к новому витку.

Спиральная модель не подойдет для малых проектов, она подходит для сложных и дорогих, например, таких, как разработка информационной системы документооборота для банка, когда каждый следующий шаг требует большего анализа для оценки последствий, чем программирование бизнес-логики.

Применение спиральной методологии

Выбор методологии зависит от сложности проекта, уровня неопределенности требований, доступных ресурсов и времени.

Радиус спирали отражает расходы проекта, а угловое измерение демонстрирует достигнутый прогресс. Фазы спиральной модели аналогичны этапам в итеративной модели разработки программного обеспечения: в первой фазе создается программное обеспечение, а затем оно дорабатывается в следующих фазах.

– *Идентификация целей.* На этой фазе собираются требования от клиента, определяются и анализируются цели, а затем предлагаются альтернативные решения.

– *Управление рисками.* Фаза включает оценку всех потенциальных решений для выбора наилучшего. После этого выявляются и решаются риски, связанные с выбранным решением.

– *Разработка и тестирование.* В этой фазе происходит разработка программного обеспечения и его тестирование на качество. Компоненты разрабатываются

и интегрируются, образуя полный программный продукт или прототип. На ранних циклах программное обеспечение на этом этапе является лишь прототипом, в то время как на последующих – полностью функциональным продуктом.

– *Оценка.* Фаза определяет, достигнута ли цель, установленная на первом этапе. Фаза оценки также помогает команде разработки понять, сколько циклов потребуется для завершения всего проекта.

Задания к работе

1. Разделиться на команды по 2–4 человека. Выбрать одну из предложенных методологий разработки ИС (каскадная модель, V-модельная интеграция, инкрементальная разработка, спиральная модель).

2. Подготовить презентацию, содержащую описание выбранной методологии, основные этапы работы в рамках методологии, преимущества и недостатки.

3. Приведите примеры применения в реальных проектах.

4. Презентовать свою работу перед группой, объясняя ключевые аспекты методологии.

Методика выполнения работы

Собрать информацию по выбранной методологии из рекомендованных источников и разработать структуру презентации, включающую введение (описание методологии), основные этапы разработки, преимущества и недостатки, а также пример применения. Создать презентацию, соблюдая требования к оформлению, делая слайды лаконичными и сопровождая их визуальными элементами. Подготовиться к выступлению, распределив роли внутри команды, и презентовать свою работу перед группой, отвечая на вопросы по теме.

Рекомендации по обработке и оформлению полученных результатов

1. Представить презентацию, которая должна быть логически последовательной, структурированной и содержать не более 10–12 слайдов.

2. Использовать схемы и диаграммы для иллюстрации процессов.

3. Обеспечить единый стиль оформления (шрифты, цвета, разметка).

4. Включить краткое резюме в конце презентации.

5. Отчет по работе должен содержать текстовое описание выбранной методологии и презентацию в электронном виде.

Вопросы для самоконтроля

1. Какие основные этапы разработки включает выбранная методология?

2. В чем преимущество итеративного подхода по сравнению с каскадной моделью?

3. Какие риски можно минимизировать, используя спиральную модель?

4. Почему V-модель акцентирует внимание на тестировании?

5. Какая методология подходит для проектов с высокой степенью неопределенности?

6. Какие недостатки характерны для инкрементального подхода?

Рекомендуемая литература

1. Вендров, А. М. Проектирование программного обеспечения экономических информационных систем / А. М. Вендров. – Москва : Финансы и статистика, 2006. – 544 с. – ISBN 5-279-02937-8.

2. Коцюба, И. Ю. Основы проектирования информационных систем : учебное пособие / И. Ю. Коцюба, А. В. Чунаев, А. Н. Шиков. – Санкт-Петербург : Университет ИТМО, 2015. – 206 с.

3. AlvaT. Ещё раз про семь основных методологий разработки. – URL: https://habr.com/ru/companies/edison/articles/269789/ (дата обращения: 20.01.2024)

Лабораторная работа № 9
Проектирование архитектуры и физической модели информационной системы

Тема занятия: проектирование архитектуры и физической модели информационной системы на основе требований.

Цель занятия: научиться разрабатывать архитектурное решение для информационной системы, определять основные компоненты и строить UML-диаграммы компонентов и развертывания.

Материалы и программное обеспечение

1. ГОСТ Р 57100–2016. Описание архитектуры.
2. Исходные данные берутся из вариантов заданий, предложенных в лабораторной работе № 2.
3. Инструментальные средства проектирования UML-диаграмм: StarUML, Visual Paradigm, Draw.io.
4. Текстовый редактор: MSWord/LibreOffice, Google Docs.

Краткие теоретические сведения

Следующий этап проектирования информационной системы после логического проектирования включает в себя диаграммы размещения на технических средствах логических моделей, построенных на предыдущих этапах (модели приложения, СУБД, баз данных и др.). На данном этапе рассматриваются такие понятия UML, как:

– *компонент* как элемент физического представления системы, реализующий определенный набор интерфейсов;

– *зависимость* как связь между двумя элементами, обозначающая ситуацию, при которой изменение одного элемента модели влечет за собой изменение ее другого элемента;

– *процессор* и *устройство;*

– *узел*, выполняющий и не выполняющий обработку данных соответственно;

– *соединение* – это связь между процессорами и устройствами.

Архитектура информационной системы (ИС) – концепция, определяющая модель, структуру, выполняемые функции и взаимосвязь компонентов информационной системы, а также среды функционирования, обеспечивающие выполнение задач системы.

Компоненты информационной системы по выполняемым функциям можно разделить на три слоя:

1. *Слой представления* – всё, что связано взаимодействием с пользователем: нажатие кнопок, движение мыши, отрисовка изображения, вывод результатов поиска и т. д.
2. *Бизнес-логика* – правила, алгоритмы реакции приложения на действия пользователя или на внутренние события, правила обработки данных.
3. *Слой доступа к данным* – хранение, выборка, модификация и удаление данных, связанных с решаемой приложением прикладной задачей.

Основные уровни архитектуры

1. *Клиентский уровень* – обеспечивает взаимодействие пользователя с системой.

2. *Серверный уровень* – обрабатывает запросы клиентов, содержит бизнес-логику.

3. *Уровень базы данных* – отвечает за хранение и управление данными.

Проектирование архитектуры

Цели проектирования архитектуры системы:

– анализ взаимодействий между выделенными классами, выявление подсистем и интерфейсов;

– уточнение архитектуры с учетом возможностей повторного использования;

– идентификация архитектурных решений и механизмов, необходимых для проектирования системы.

Вводятся глобальные пакеты:

– базисные (foundation) классы (списки, очереди и т. д.);

– обработчики ошибок;

– математические библиотеки;

– утилиты;

– библиотеки других поставщиков (если необходимо).

Определяются проектные классы (дизайн классов):

– класс анализа отображается в *проектный класс*, если он простой или представляет единственную логическую абстракцию;

– сложный класс анализа может быть разбит на несколько классов, преобразован в пакет или в подсистему.

Примеры возможных подсистем:

– классы, обеспечивающие сложный комплекс услуг (например, обеспечение безопасности и защита);

– граничные классы, реализующие сложный пользовательский интерфейс или интерфейс с внешними системами;

– различные продукты: коммуникационное ПО, доступ к базам данных, типы и структуры данных (стеки, списки, очереди), общие утилиты (математические библиотеки), различные прикладные продукты.

Принятие решения о преобразовании класса в подсистему определяется опытом и знаниями архитектора проекта.

Соглашения по проектированию интерфейсов:

– имя интерфейса короткое (одно-два слова), отражающее его роль в системе;

– описание интерфейса должно отражать его обязанности (размер – небольшой абзац);

– описание операций – имя, отражающее результат операции, ключевые алгоритмы, возвращаемое значение, параметры с типами;

– документирование интерфейса описывает характер использования операций и порядок их выполнения (отражается с помощью диаграмм последовательностей), а также тестовые планы и сценарии.

Диаграмма компонентов (components diagram)

Диаграмма компонентов – статическая структурная диаграмма, которая показывает разбиение программной системы на структурные компоненты и связи (зависимости) между ними.

Диаграмма компонентов (components diagram) описывает физическую структуру системы, отображает иерархию подсистем, структурных компонентов и зависимостей между ними. Физическими компонентами выступают модули, базы данных, библиотеки, исполняемые файлы, приложения, интерфейсы и API.

Интерфейсы составного компонента делегируются в определенные интерфейсы внутренних компонентов, если требуется составить диаграммы компонентов для отображения внутренней структуры компонентов.

Основными целями построения диаграмм компонентов являются:

– определение архитектуры проектируемой системы;

– визуализация общей структуры исходного кода программной системы;

– построение концептуальной и физической моделей баз данных;

– представление структуры исходного и специфики исполняемого кода системы;

– обеспечение многократного использования определенных фрагментов программного кода.

Диаграмма компонентов помогает разработчикам и архитекторам лучше понять структуру и взаимосвязи системы, а также является инструментом для коммуникации и документирования архитектурных решений. Пример диаграммы компонентов представлен на рисунке 9.1.

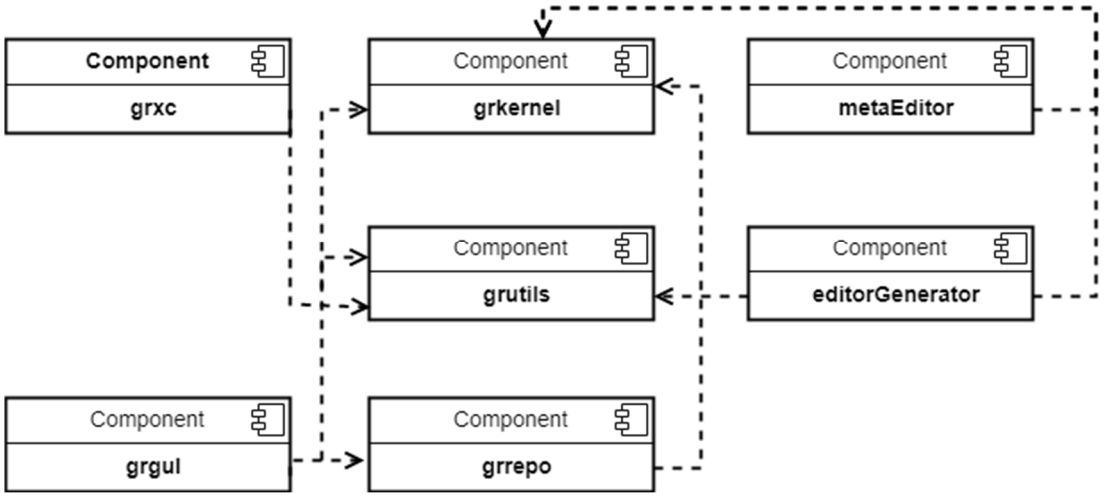
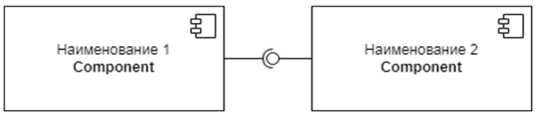
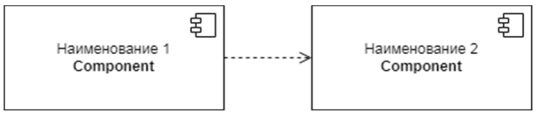


Рис. 9.1. Диаграмма компонентов программной системы

Диаграммы компонентов используются обычно специалистами, которые отвечают за компиляцию и сборку системы. На основе анализа диаграмм выявляется порядок, в котором необходимо компилировать компоненты, а также можно узнать, какие исполняемые компоненты будут созданы системой. На диаграмме компонентов можно видеть соответствие классов реализованным компонентам, что будет полезным перед генерацией кода (табл. 9.1).

Таблица 9.1

Основные компоненты диаграммы компонент

№	Название	Примечание	Обозначение
1	Компонент	Программный компонент	
2	Предоставляемый интерфейс и требуемый интерфейс	Позволяет соединить требуемый интерфейс компонента (представленный полукругом и сплошной линией) с предусмотренным интерфейсом (представленный окружностью и сплошной линией) другого компонента	
3	Порт	Обозначается квадратом в конце требуемого интерфейса или предоставляемого интерфейса, применяется, когда компонент делегирует интерфейсы внутреннему классу	
4	Зависимость	Отношения между двумя компонентами	

Этапы построения диаграммы компонентов

1. Выяснить назначение диаграммы и определить артефакты, такие как файлы, документы и т. д. в представленной системе для построения диаграммы.

2. По мере выяснения взаимосвязей между элементами, определенных ранее, создать начальный макет диаграммы.

3. По мере того как рисуется диаграмма, сначала добавить компоненты, группируя их внутри других компонентов, по вашему усмотрению.

4. Следующим шагом является добавление других элементов, таких как интерфейсы, классы, объекты, зависимости и т. д., в вашу компонентную диаграмму.

5. Добавить примечания к различным частям компонентной диаграммы, чтобы уточнить некоторые детали.

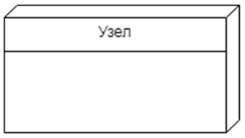
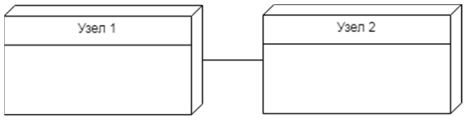
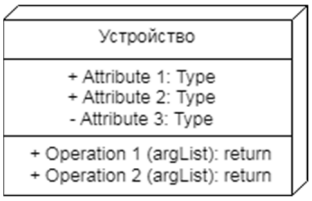
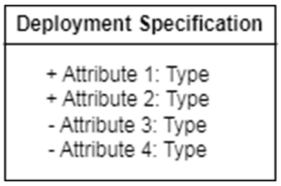
Диаграмма развертывания (deployment diagram)

Диаграмма развёртывания – структурная диаграмма, которая показывает физическое развёртывание программных компонентов на аппаратных узлах. Диаграмма визуализирует отображение программных компонентов на физические ресурсы системы, такие как серверы, процессоры, устройства хранения данных и сетевая инфраструктура.

Диаграмма развертывания (deployment diagram) отражает физические взаимосвязи между программными и аппаратными компонентами системы и является хорошим инструментом для того, чтобы показать маршруты перемещения объектов и компонентов в распределенной системе (табл. 9.2).

Таблица 9.2

Основные компоненты диаграммы развертывания

№	Название	Примечание	Обозначение
1	Узел	Представляет собой физическую сущность, которая выполняет одну или несколько компонентов, подсистем или исполняемых файлов. Узел может быть аппаратным или программным элементом	
2	Артефакт	Конкретные элементы, которые вызваны процессом разработки. Примерами артефактов являются библиотеки, архивы, конфигурационные файлы, исполняемые файлы и т. д.	
3	Коммуникационная ассоциация	Представлена сплошной линией между двумя узлами и показывает путь связи между узлами	
4	Устройство	Узел, который используется для представления физического вычислительного ресурса в системе. Примером устройства является сервер приложений	
5	Спецификация развертывания	Файл конфигурации, например, текстовый файл или XML-документ, в котором описывается, как артефакт развертывается на узле	

Каждый узел на диаграмме развертывания представляет собой некоторый тип вычислительного устройства, в большинстве случаев – часть аппаратуры, которая может быть простым цифровым устройством, а может быть и компьютером. Пример диаграммы развертывания представлен на рисунке 9.2.

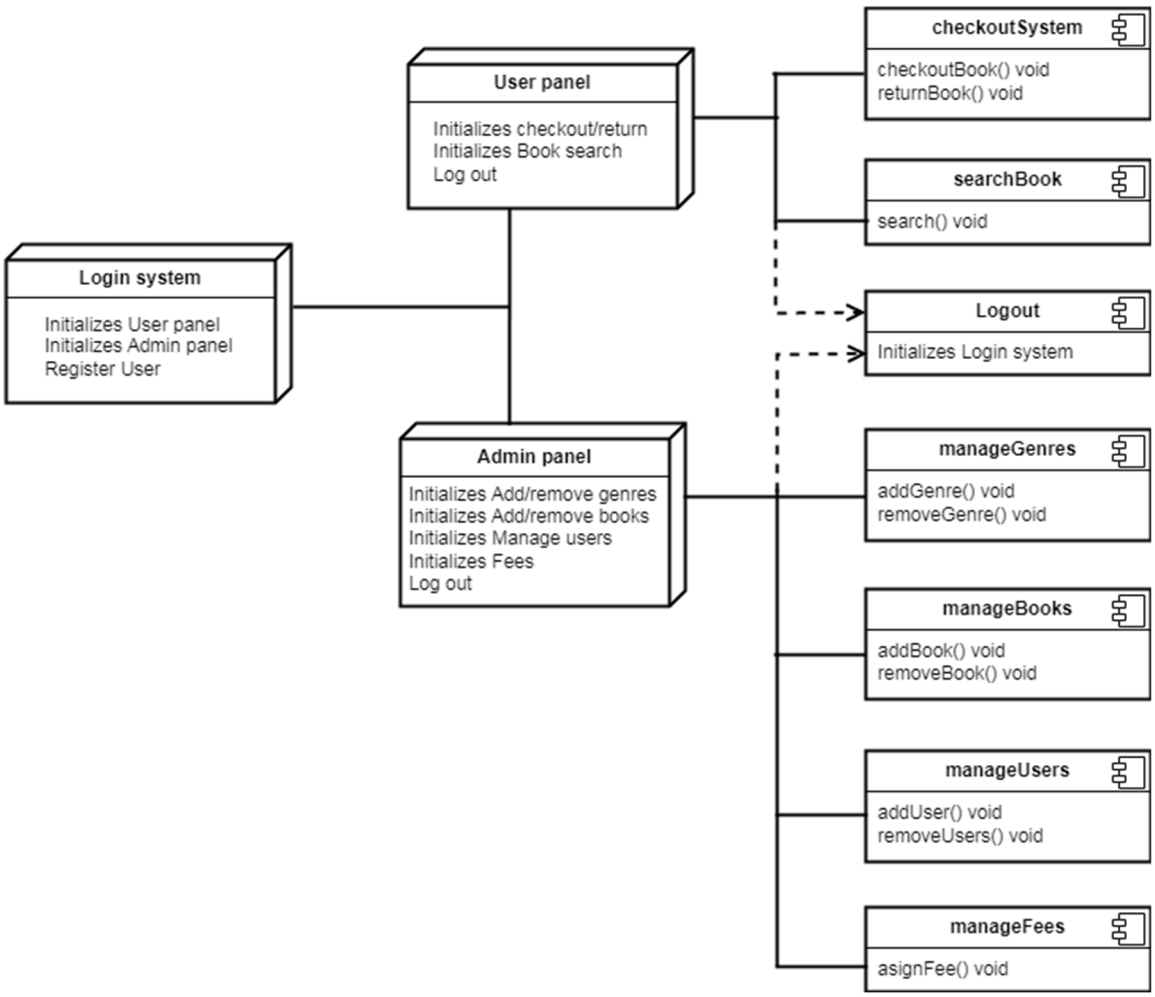


Рис. 9.2. Диаграмма развертывания для системы онлайн-покупок

Этапы построения диаграммы развертывания

1. Определить цель вашей схемы развертывания, для этого необходимо определить узлы и устройства в системе, которые будут представлены на диаграмме.

2. Требуется выяснить отношения между узлами и устройствами и далее перейти к добавлению коммуникационных ассоциаций на схеме.

3. Определить, какие еще другие элементы (компоненты, активные объекты) необходимо добавить для завершения диаграммы.

4. Добавить при необходимости зависимости между компонентами и объектами.

Задания к работе

1. Определить основные требования к информационной системе, которая была выбрана из вариантов заданий, предложенных в лабораторной работе № 2.

2. Разработать архитектуру системы (клиентский уровень, серверный уровень, уровень базы данных).

3. Построить UML-диаграммы компонентов и развертывания.

4. Описать архитектурное решение.

Методика выполнения работы

Ознакомиться с исходным материалом и провести анализ предметной области. Необходимо определить основные требования к информационной системе. Выделить ключевые компоненты архитектуры, основываясь на функциональных и нефункциональных требованиях. Используя программное обеспечение, создать UML-диаграмму компонентов, определить основные модули (клиентские приложения, серверы, базы данных) и описать их взаимодействия. Построить UML-диаграмму развертывания с указанием физических узлов (серверов, клиентских устройств) и связей между ними. Завершить работу текстовым описанием архитектурного решения.

Рекомендации по обработке и оформлению полученных результатов

Оформить диаграммы компонентов и развертывания в формате изображений или PDF. Краткое архитектурное описание должно включать обоснование выбранных компонентов и описание взаимодействий между уровнями системы. Составить отчет в текстовом редакторе, добавить в него диаграммы и текстовое описание архитектурного решения.

Вопросы для самоконтроля

1. Какие уровни включает архитектура информационной системы?
2. Какие задачи решает диаграмма компонентов?
3. В чем заключается цель диаграммы развертывания?
4. Назвать требования, которые необходимо учитывать при проектировании архитектуры ИС?
5. Перечислить, какие программные инструменты можно использовать для создания UML-диаграмм?

Рекомендуемая литература

1. Коцюба, И. Ю. Основы проектирования информационных систем : учебное пособие / И. Ю. Коцюба, А. В. Чунаев, А. Н. Шиков. – Санкт-Петербург : Университет ИТМО, 2015. – 206 с.

2. Вендров, А. М. Проектирование программного обеспечения экономических информационных систем / А. М. Вендров. – Москва : Финансы и статистика, 2006. – 544 с. – ISBN 5-279-02937-8.

3. Грекул, В. И. Проектирование информационных систем : практикум: учебное пособие / В. И. Грекул, Н. Л. Коровкина, Ю. В. Куприянов. – Москва : Национальный Открытый Университет «ИНТУИТ.ru», 2012. – 187 с. – ISBN 978-5-9556-0133-5.

Лабораторная работа № 10
Моделирование данных. Проектирование структуры базы данных

Тема занятия: моделирование данных, проектирование концептуальной и логической моделей базы данных.

Цель занятия: научиться проектировать структуру баз данных, создавать концептуальную и логическую модели, выполнять нормализацию данных и реализовывать базу данных с использованием SQL.

Материалы и программное обеспечение

1. Исходные данные берутся из вариантов заданий, предложенных в лабораторной работе № 2.
2. СУБД: MS SQL Server Express 2016, MySQL.
3. Программные сервисы (среды, программы) для работы с базами данных: My SQL Workbench, pgAdmin, SQL Server Management Studio.
4. CASE-средства для проектирования ER-диаграмм: Visual Paradigm, online.visual-paradigm.com, Draw.io, dbdiagram.io, MS Visio, ERwin Data Modeler Community Edition, ER/Studio Data Architect.
5. Текстовый редактор: MS Word/LibreOffice, Google Docs.

Краткие теоретические сведения

Моделирование данных – создание визуального представления об информационной системе либо ее части. Цель моделирования заключается в том, чтобы наглядно представить типы данных, которые используются и хранятся в системе, отношения между этими типами данных, способы группировки и организации данных, их форматы и атрибуты.

Модель данных обычно формируется на основе потребностей пользователей. Правила и требования к *модели данных* определяются заранее на основе требований заказчика.

Данные можно моделировать на различных уровнях абстракции. Процесс начинается со сбора требований от заинтересованных сторон и конечных пользователей – *стейкхолдеров*. Полученные правила и требования затем преобразуются в структуры данных. *Модель данных* сравнивают с дорожной картой, используя которую, можно лучше понять, что будет далее разрабатываться для хранения и управления в проекте.

Преимущества моделирования данных

Моделирование упрощает просмотр и понимание взаимосвязей между данными для разработчиков, архитекторов данных и других заинтересованных лиц. Моделирование данных позволяет:

- унифицировать документацию на предприятии;
- сократить количество ошибок при разработке баз данных;
- повысить производительность приложений и баз данных;
- упростить отображение данных;

– улучшить взаимодействие между стейкхолдерами;

– упростить и ускорить процесс проектирования базы данных на концептуальном, логическом и физическом уровнях.

Типы моделей данных

Разработка баз данных начинается со схемы высокого уровня абстракции, а далее происходит дальнейшая детализация и уточнение. По степени абстракции модели данных можно разделить на три категории: процесс начинается с концептуальной модели, затем переходит к логической модели и завершается физической моделью.

Концептуальные модели данных. Их еще называют *моделями предметной области*, которые описывают самую общую картину. Они показывают, что будет содержать система, как она будет организована и какие бизнес-правила будут задействованы. Концептуальные модели обычно создаются в начале, в процессе сбора исходных требований к проекту. Они включают классы сущностей, их характеристики и ограничения, отношения между сущностями, требования к безопасности и целостности данных (рис. 10.1).

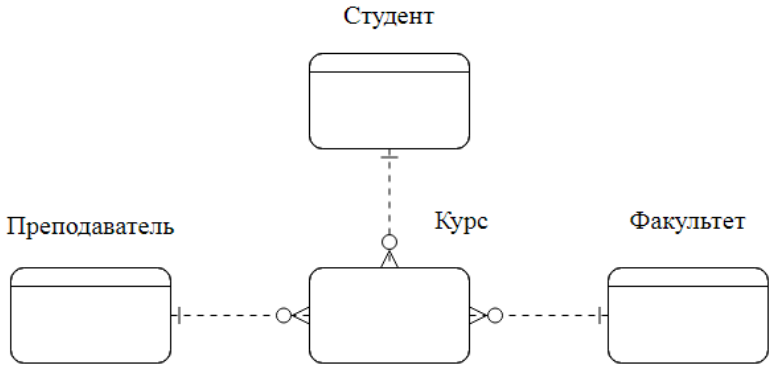


Рис. 10.1. Концептуальная модель данных

Логические модели данных уже менее абстрактны и предоставляют более подробную информацию о концепциях и взаимосвязях в рассматриваемой области. Они содержат атрибуты данных и показывают отношения между сущностями. Логические модели данных не определяют никаких технических требований к системе и могут быть полезны для проектов, ориентированных на данные. Например, для проектирования *хранилища данных* или разработки системы отчетности (рис. 10.2).

Физические модели данных представляют схему того, как данные будут храниться в базе данных. По сути, это наименее абстрактные из всех моделей. Они предлагают окончательный дизайн, который может быть реализован как *реляционная база данных*, включающая таблицы, которые демонстрируют отношения между сущностями, а также первичные и внешние ключи для связи данных (рис. 10.3).

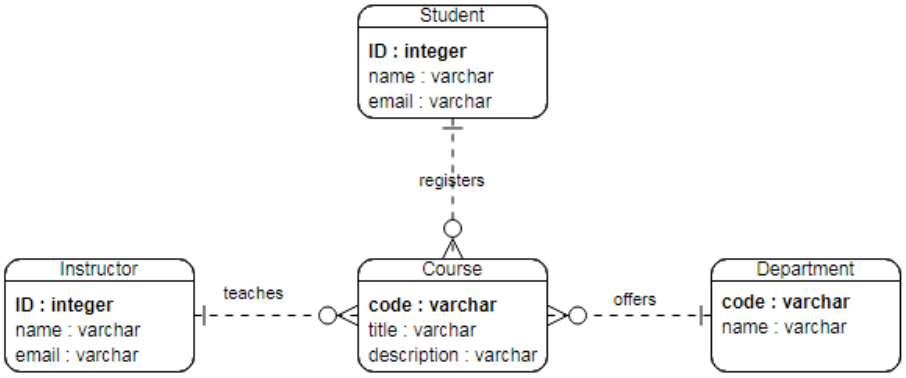


Рис. 10.2. Логическая модель данных

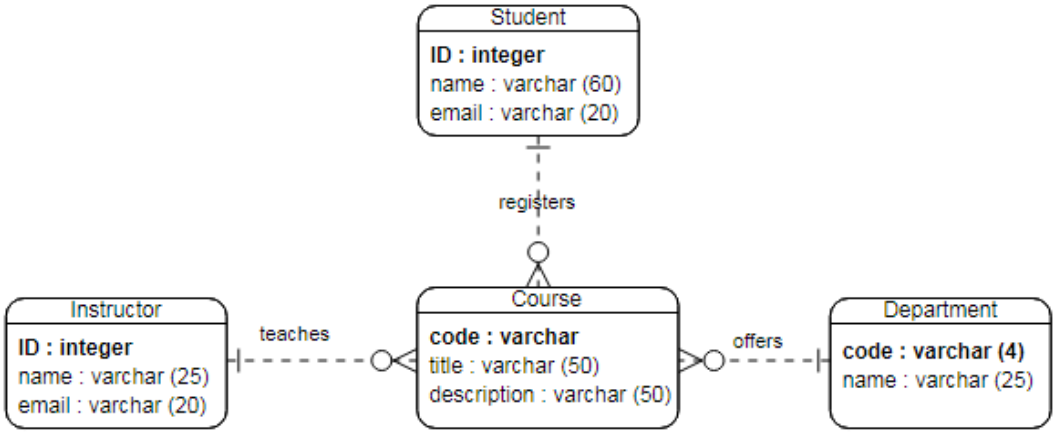


Рис. 10.3. Физическая модель данных

Процесс моделирования данных

Моделирование данных начинается с договоренности о том, какие символы используются для представления данных, как размещаются модели и как формулируются и реализуются требования. Это формализованный рабочий процесс, включающий ряд задач, которые должны выполняться итеративно. Сам процесс обычно выглядят так:

1. *Определить сущности*. На этом этапе идентифицируем объекты, события или концепции, представленные в наборе данных, который необходимо смоделировать. Каждая сущность должна быть целостной и логически отделенной от всех остальных.

2. *Определить ключевые свойства каждой сущности.* Каждый тип сущности можно отличить от всех остальных, поскольку он имеет одно или несколько уникальных свойств, называемых атрибутами.

3. *Определить связи между сущностями.* Самый ранний вид модели данных будет определять характер *отношений*, которые каждая сущность имеет с другими. Эти отношения обычно документируются с помощью унифицированного языка моделирования (UML).

4. *Полностью сопоставить атрибуты с сущностями.* Это гарантирует, что модель отражает то, как программа (логика работы) будет использовать данные.

Широко используются несколько формальных шаблонов (паттернов) моделирования данных. Объектно-ориентированные разработчики часто применяют шаблоны для анализа или шаблоны проектирования, в то время как заинтересованные стороны из других областей могут обратиться к другим паттернам.

5. *Назначить ключи по мере необходимости и определить степень нормализации.* Нормализация – метод организации моделей данных, в которых числовые идентификаторы (ключи) назначаются группам данных для установления связей между ними без повторения данных. *Нормализация* помогает уменьшить объем дискового пространства, необходимого для базы данных, но может сказываться на производительности запросов.

6. *Завершить и проверить модель данных*. Моделирование данных – это итеративный процесс, который следует повторять и совершенствовать под потребности выполняемой задачи.

Типы моделирования данных

Моделирование данных развивалось вместе с системами управления базами данных (СУБД), при этом типы моделей усложнялись по мере роста потребностей предприятий в хранении данных.

Иерархические модели данных представляют отношения «один ко многим» в древовидном формате. В модели этого типа каждая запись имеет единственный корень или родительский элемент, который сопоставляется с одной или несколькими дочерними таблицами. Эта модель была реализована в IBM Information Management System (IMS) в 1966 г. Хотя этот подход менее эффективен, чем новые, недавно разработанные модели баз данных, он все еще используется в системах расширяемого языка разметки (XML) и географических информационных системах (ГИС).

Реляционные модели данных были предложены Э.Ф. Коддом в 1970 г. Они широко применяются в реляционных базах данных. Реляционное моделирование не требует детального понимания физических свойств используемого хранилища данных. В нем сегменты данных объединяются с помощью таблиц, что упрощает базу данных.

Реляционные базы данных часто используют язык структурированных запросов (SQL) для управления данными. Эти базы подходят для поддержания целостности данных и минимизации избыточности.

ER-модель данных

В *ER-моделях данных* используют диаграммы для представления взаимосвязей между сущностями в базе данных. ER-модель представляет собой формальную конструкцию, которая не предписывает никаких графических средств её визуализации. В качестве стандартной графической нотации, с помощью которой можно визуализировать ER-модель, была предложена диаграмма «сущность-связь» (Entity-Relationship diagram, ER-диаграмма).

ER-диаграмма – разновидность блок-схемы, где показано, как разные «сущности» (люди, объекты, концепции и так далее) связаны между собой внутри

системы. ER-диаграммы чаще всего применяются для проектирования и отладки реляционных баз данных в сфере образования, исследования и разработки программного обеспечения и информационных систем для бизнеса. ER-диаграммы полагаются на стандартный набор символов, включая прямоугольники, ромбы, овалы и соединительные линии для отображения сущностей, их атрибутов и связей. Эти диаграммы устроены по тому же принципу, что и грамматические структуры: сущности выполняют роль существительных, а связи – глаголов.

Основные элементы ER-диаграммы.

– *Сущность* – физическое представление логической группировки данных (событий, предметов, состояний, людей). Сущности представляют собой объекты, данные о которых необходимо сохранять. Любой объект системы должен быть представлен только одной *сущностью*. Имя сущности должно отражать тип или класс объекта, а не его единичный экземпляр. Сущность изображается в виде прямоугольника, вверху которого располагается имя сущности. В прямоугольнике перечислены атрибуты сущности.

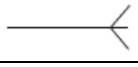
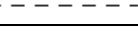
– *Атрибут* – любая характеристика сущности, значимая для рассматриваемой предметной области и предназначенная для квалификации, идентификации, классификации, количественной характеристики или выражения состояния сущности. В ER-модели атрибуты ассоциируются с конкретными сущностями. Таким образом, экземпляр сущности должен обладать единственным определенным значением для ассоциированного атрибута. Атрибут может быть либо обязательным, либо необязательным. Обязательность означает, что атрибут не может принимать неопределенных значений (null values). Атрибут может быть либо описательным (т. е. обычным дескриптором сущности), либо входить в состав уникального идентификатора (первичного ключа). Атрибуты сущностей приводятся к атрибутам отношений.

– *Уникальный идентификатор* – атрибут или совокупность атрибутов и/или связей, предназначенных для уникальной идентификации каждого экземпляра данного типа сущности. Уникальный идентификатор – это ключевой атрибут.

– *Связь* (relationship) или *отношение* – поименованная ассоциация между двумя сущностями, значимая для рассматриваемой предметной области. Связи может даваться имя, выражаемое грамматическим оборотом глагола и помещаемое возле линии связи. Все связи являются бинарными – это линии с двумя концами. Для каждой связи определяется степень множественности и обязательность (табл. 10.1).

Таблица 10.1

Обозначение основных типов связей на ER-диаграммах в нотации Ричарда Баркера

№	Назначение	Обозначение
1	Много	
2	Один	
3	Необязательная	
4	Обязательная	

Применяются следующие типы отношений:

1) 1 – 1 (один к одному);

2) 1 – *m* (один ко многим);

3) *n* – *m* (многие ко многим).

Основные этапы моделирования данных:

– идентификация сущностей, атрибутов и первичных ключей;

– идентификация отношений между сущностями и указание типов отношений;

– разрешение неспецифичных отношений.

Построение ER-диаграммы

ER-диаграмма является *абстрактной* моделью базы данных, поэтому для ее моделирования был разработан ряд правил, которые облегчают переход от диаграмм к реляционным отношениям:

– каждый правильный (сильный) тип сущности соответствует базовому реляционному отношению;

– каждая бинарная связь типа «многие-ко-многим» также соответствует отдельному отношению, которое должно включать в себя два внешних ключа, ссылающихся на потенциальные ключи отношений, соответствующих сущностям – участникам связи;

– связь типа «один-ко-многим» между сильными сущностями может быть представлена с помощью внешнего ключа и не требует отдельного отношения;

– связь слабого объекта с сильным, от которого он зависит, является связью типа «многие-к-одному» и может быть представлена внешним ключом. В некоторых случаях, когда и сильная, и подчиняющаяся ей слабая сущности представлены одним реляционным отношением, внешний ключ может ссылаться на первичный ключ своего же отношения.

Пример 1. ER-диаграммы системы планирования экзаменов в нотации IE (нотация К. Финкельштейна), которая представлена на рисунке 10.1.

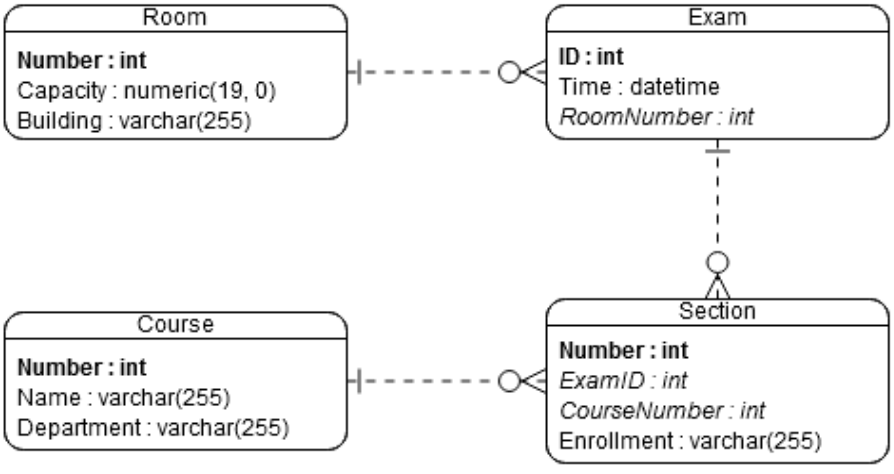


Рис. 10.1. ER-диаграмма системы планирования экзаменов

Варианты применения ER-диаграмм

1. Диаграммы применяются при концептуальном проектировании баз данных, когда выделяются основные сущности и определяются связи между этими сущностями.

2. При проектировании баз данных выполняется преобразование ER-диаграммы в конкретную схему базы данных.

Преимущества и недостатки ER-диаграмм

Основными преимуществами ER-диаграмм являются высокая наглядность и возможность представлять произвольное количество данных с любым количеством свойств, кроме того, диаграммы являются инструментом концептуального уровня. Данные диаграммы используются во многих системах проектирования баз данных.

К недостатком следует отнести статичность диаграмм при отражении ими сущностей и связей между ними в данной предметной области. При описании процессов информационного обмена между сущностями в предметной области необходимо использовать другие методики (UML, DFD и др.).

Основные этапы проектирования и разработки баз данных

1. *Проектирование базы данных*

Проектирование базы данных включает создание модели данных, описывающей объекты системы и связи между ними.

2. *Концептуальная модель данных (ER-диаграмма)*

ER-диаграмма (Entity-Relationship Diagram) представляет сущности, их атрибуты и связи:

– сущности (Entities) – основные объекты базы данных;
– атрибуты (Attributes) – характеристики сущностей;
– связи (Relationships) – описывают отношения между сущностями.

3.*Логическая модель данных*

Логическая модель преобразует концептуальную модель в структуру таблиц, полей и связей.

4. *Нормализация данных*

Процесс нормализации устраняет избыточность и аномалии данных. Основные нормальные формы:

– *первая нормальная форма (1НФ).* Все атрибуты атомарны;

– *вторая нормальная форма (2НФ).* Нет частичных зависимостей (сущности зависят от всего первичного ключа);

– *третья нормальная форма (3НФ).* Устранение транзитивных зависимостей (атрибуты зависят только от первичного ключа).

5. *Реализация базы данных в SQL*

SQL используется для создания таблиц и реализации связей между ними:

– *CREATE TABLE*. Создание таблиц;
– *ALTER TABLE*. Модификация структуры таблиц;
– *INSERT, SELECT, UPDATE, DELETE.* Манипуляции данными.

Пример 2. Описание базы данных «Цифровой профиль студента» в упрощенной формес использованием программы MSVisio.

В ходе проектирования базы данных для веб-приложения были выделены следующие объекты:

1. Пользователи.
2. Роли.
3. Факультеты.
4. Группы.
5. Студенты.
6. Оценки.
7. Файлы.
8. Компетенции.
9. Когнитивные параметры.

В соответствии с выделенными объектами были сформированы следующие таблицы базы данных.

Таблица 10. 2

Описание таблиц базы данных

Наименование таблицы	Назначение
User (Пользователь)	Учетные записи пользователей системы (администрация вуза, студенты)
Roles (Роли)	Дополнительная таблица для определения ролей системы (пользователь, администратор, менеджер, работодатель)
Faculties (Факультеты)	Информация о факультетах университета
Groups (Группы)	Информация о группах университета
Students (Студенты)	Информация о студенте (ФИО, группа, курс, факультет, специальность, почта, id пользователя)
Marks (Оценки)	Успеваемость студента за итоги сессий
Files (Файлы)	Загруженные файлы

Примеры описания таблиц базы данных (табл. 3 – табл. 10.7).

Таблица 10.3

Описание таблицы «User»

Атрибут		Признак ключа	Тип	Длина
Имя	Название			
id	Идентификатор	Первичный (Primary)	Целочисленный (Integer)	5
name	Имя пользователя		Строка (Varchar)	100
email	Почта (логин)		Строка (Varchar)	50
password	Пароль		Строка (Varchar)	50
role_id	Идентификатор роли пользователя	Внешний (Foreign)	Целочисленный (Integer)	5

Таблица 10.4

Описание таблицы «Roles»

Атрибут		Признак ключа	Тип	Длина
Имя	Название			
role_id	Идентификатор роли	Первичный (Primary)	Целочисленный (Integer)	5
name	Название роли		Строка (Varchar)	50

Таблица 10.5

Описание таблицы «Groups»

Атрибут		Признак ключа	Тип	Длина
Имя	Название			
fak_id	Идентификатор факультета	Первичный (Primary)	Целочисленный (Integer)	5
gruppa	Номер группы	Внешний (Foreign)	Строка (Varchar)	10
course	Курс		Целочисленный (Integer)	5

Таблица 10.6

Описание таблицы «Faculties»

Атрибут		Признак ключа	Тип	Длина
Имя	Название			
fak_id	Идентификатор факультета	Первичный (Primary)	Целочисленный (Integer)	5
name	Название факультета		Строка (Varchar)	25
IMS	Сокращенное название		Строка (Varchar)	15

Таблица 10.7

Описание таблицы «Students»

Атрибут		Признак ключа	Тип	Длина
Имя	Название			
id	Идентификатор студента	Первичный (Primary)	Целочисленный (Integer)	5
user_id	Идентификатор пользователя	Внешний (Foreign)	Строка (Varchar)	100
fio	ФИО студента		Строка (Varchar)	50
spec_napravl	Направление		Строка (Varchar)	50
facultet	Факультет		Строка (Varchar)	50
gruppa	Номер группы	Внешний (Foreign)	Строка (Varchar)	10
edecanat_id	Идентификатор в системе Е-деканат		Целочисленный (Integer)	5
course	Курс		Целочисленный (Integer)	5

На рисунке 10.2 построена структура базы данных, включающих в себя комплекс структурных компонентов базы данных, а также средств, обеспечивающих их взаимодействие как друг с другом, так и с конечным пользователем, системным персоналом. Для построения использовался графический редактор MS Visio.

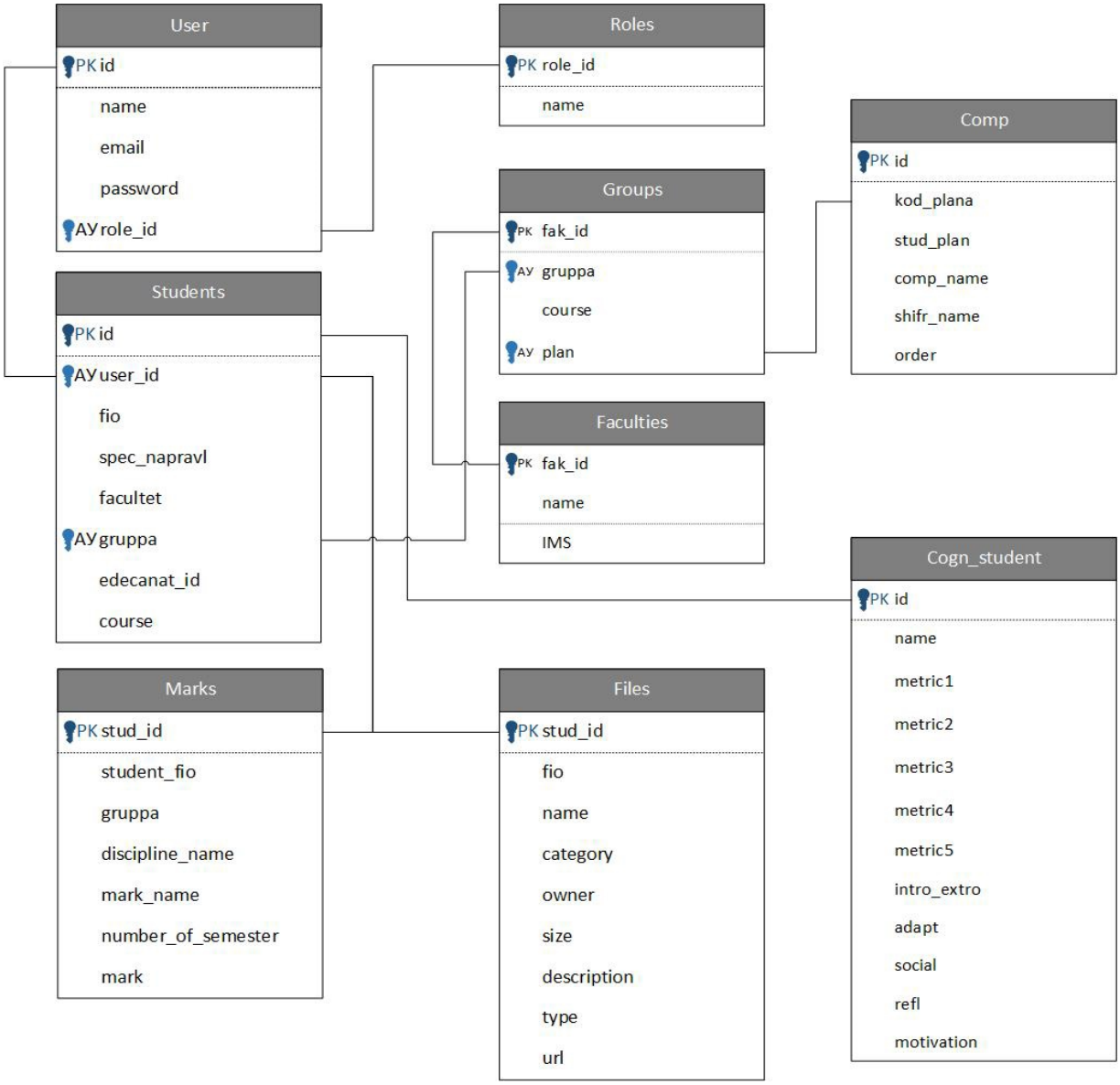


Рис. 10.2. Структура базы данных для веб-приложения «Цифровой профиль»

Задания к работе

1. Создать концептуальную модель базы данных (ER-диаграмму), которая была выбрана из вариантов заданий, предложенных в лабораторной работе № 2.

2. Разработать логическую модель данных (структуру таблиц с атрибутами и связями).

3. Выполнить нормализацию таблиц до третьей нормальной формы (3НФ).

4. Реализовать структуру базы данных с помощью SQL: создать таблицы; определить первичные и внешние ключи.

5. Заполнить таблицы тестовыми данными и проверить корректность реализации.

Методика выполнения работы

Ознакомиться с исходным материалом и провести анализ предметной области. Составить списки сущностей и атрибутов. Построить ER-диаграмму, преоб-

разовать ее в логическую модель, определив таблицы, их атрибуты и связи. Проверить модель на соответствие нормальным формам (1НФ, 2НФ, 3НФ). Используя выбранную СУБД, разработать SQL-скрипты для создания таблиц, связей и добавления данных. Выполнить тестовые запросы для проверки работы базы данных.

Рекомендации по обработке и оформлению полученных результатов

Оформите ER-диаграмму и логическую модель в формате изображений или PDF. Подготовьте отчет, включив в него описание предметной области, концептуальную и логическую модели, SQL-скрипты для создания и заполнения таблиц, а также примеры выполненных запросов. Оформите отчет в соответствии с установленными требованиями.

Вопросы для самоконтроля

1. Что такое ER-диаграмма и какие элементы она включает?
2. Каковы основные отличия концептуальной и логической моделей базы данных?
3. Что такое нормализация и зачем она нужна?
4. Какие типы связей существуют между сущностями?
5. Какие команды SQL используются для создания и модификации таблиц?

Рекомендуемая литература

1. Коцюба, И. Ю. Основы проектирования информационных систем : учебное пособие / И. Ю. Коцюба, А. В. Чунаев, А. Н. Шиков. – Санкт-Петербург : Университет ИТМО, 2015. – 206 с.

2. Вендров, А. М. Проектирование программного обеспечения экономических информационных систем / А. М. Вендров. – Москва : Финансы и статистика, 2006. – 544 с. – ISBN 5-279-02937-8.

Лабораторная работа № 11*
Реализация прототипа информационной системы

Тема занятия: разработка минимального работающего прототипа информационной системы с базовой функциональностью.

Цель занятия: освоить процесс разработки прототипа информационной системы, реализовать базовые функции (аутентификацию, CRUD-операции) и провести тестирование.

Материалы и программное обеспечение

1. Программное обеспечение. Фреймворк или платформа: Django, Laravel, Flask, Ruby.
2. Программное обеспечение для разработки клиентского ПО. Программные среды разработки: Visual Studio Code, PyCharm, PhpStorm.
3. Программное обеспечение для разработки серверного ПО. Серверная среда (Apache, Nginx, или встроенный сервер разработки фреймворка).
4. Программное обеспечение для поддержки баз данных. СУБД (MySQL, PostgreSQL, SQLite).
5. Программное обеспечение. Веб-браузер для тестирования.

Краткие теоретические сведения

1. Прототип информационной системы (ИС)

Прототип ИС – минимальная версия системы, которая позволяет проверить основные функции и архитектурные решения.

Основные функции прототипа.

– *Аутентификация* – проверка и управление доступом пользователей.

– *CRUD-операции* – операции создания (Create), чтения (Read), обновления (Update) и удаления (Delete) данных.

2. Выбор фреймворка

Фреймворки помогают ускорить процесс разработки, предоставляя готовые инструменты и структуры.

Примеры.

– *Django (Python)*– поддерживает быстрое создание веб-приложений, имеет встроенные средства аутентификации.

– *Laravel (PHP)*– популярный фреймворк для разработки приложений с оптимальным синтаксисом.

– *Flask (Python)* – легковесный фреймворк для небольших приложений.

3. Тестирование прототипа

Тестирование проводится для проверки функциональности и поиска ошибок.

Основные методы:

– мануальное тестирование (ручная проверка работы системы);

– написание юнит-тестов для проверки отдельных функций.

* Лабораторная работа повышенной сложности, обычно выполняется группой обучающихся.

Задания к работе

1. Выбрать фреймворк или платформу для реализации прототипа.
2. Разработать минимальную базовую функциональность: аутентификацию пользователей (регистрация, вход, выход); CRUD-операции для управления данными (например, управление списком задач или товаров).
3. Настроить у прототипа взаимодействие с базой данных.
4. Провести тестирование прототипа, проверить корректность работы функций.
5. Подготовить отчет с описанием выполненной работы.

Методика выполнения работы

Ознакомиться с документацией выбранного фреймворка, создать новый проект и настроить окружение разработки. Реализовать аутентификацию пользователей, настроив модель пользователя и создав формы для регистрации входа. Разработать CRUD-функции, определив модель данных и реализовав интерфейсы для создания, просмотра, обновления и удаления записей. Настроить маршруты и шаблоны для веб-интерфейса, а затем провести тестирование с помощью браузера или инструментов автоматического тестирования.

Рекомендации по обработке и оформлению полученных результатов

Оформить отчет, указав цель работы, использованные технологии, описание реализованной функциональности, а также включив кодовые примеры и скриншоты интерфейса. Проверить корректность работы системы на всех этапах тестирования. Подготовить результаты работы в установленном формате (PDF-отчет и ZIP-архив с проектом и отчетом) и отправить их преподавателю.

Вопросы для самоконтроля

1. Что такое прототип информационной системы?
2. Какие функции включены в базовую функциональность?
3. Какие фреймворки наиболее популярны для разработки веб-приложений?
4. Как реализовать аутентификацию пользователей в выбранном фреймворке?
5. Что такое CRUD-операции и как они реализуются?

Рекомендуемая литература

1. Bootstrap // Getbootstrap. – URL: https://getbootstrap.com/ (дата обращения: 01.04.2024).
2. Фреймворк Django. – URL: https://www.djangoproject.com/ (дата обращения: 30.12.2024).
3. Фреймворк Flask. – URL: https://flask.palletsprojects.com/ (дата обращения: 15.09.24).
4. Python // SkillFactory. – URL: https://blog.skillfactory.ru/glossary/ python/ (дата обращения: 15.09.24).
5. Фреймворк Laravel. – URL: https://laravel.com/ (дата обращения: 30.12.2024).

6. Язык программирования PHP. – URL: https://www.php.net/ (дата обращения: 30.12.2024).

7. Редактор кода Visual Studio Code. – URL: https://code.visualstudio.com/ (дата обращения: 30.12.2024).

8. PyCharm – интегрированная среда разработки для Python. – URL: https://www.jetbrains.com/pycharm/ (дата обращения: 30.12.2024).

9. PhpStotm – интегрированная среда разработки для PHP. – URL: https://www.jetbrains.com/ru-ru/phpstorm/ (дата обращения: 30.12.2024).

10. Apache HTTP-сервер. – URL: https://httpd.apache.org/ (дата обращения: 30.12.2024).

11. Nginx HTTP-сервер. – URL: https://nginx.org/ru/ (дата обращения: 30.12.2024).

12. Дюбуа, П. MySQL: сборник рецептов / П. Дюбуа. – Санкт-Петербург : Символ-Плюс, 2005. – 1056 с.–ISBN 978-5-8459-1185-8.

13. Дакетт, Д. JavaScriptijQuery. Интерактивная веб-разработка / Д. Дакетт. – Москва : Изд-во: «Э», 2017. – 640 с. – ISBN 978-5-699-80285-2.

14. jQuery. – URL: https://jquery.com/ (дата обращения: 14.04.2024).

15. Документация PostgreSQL. – URL: https://www.postgresql.org/ (дата обращения: 14.04.2024).

Лабораторная работа № 12
Анализ требований и создание спецификаций

Тема занятия: изучение процессов сбора, анализа и документирования требований к информационной системе.

Цель занятия: освоить процесс сбора, анализа и документирования требований, а также подготовить спецификацию требований к информационной системе, включая функциональные и нефункциональные требования.

Материалы и программное обеспечение

1. Стандарты: IEEE STD 830-1998, ISO/IEC/ IEEE 29148-2011.
2. Исходные данные берутся из вариантов заданий, предложенных в лабораторной работе № 2.
3. Средства моделирования: Microsoft Visio, Draw.io или аналогичные инструменты для построения диаграмм.
4. Текстовый редактор: MS Word/LibreOffice, Google Docs.
5. Программное обеспечение для создания и хранения спецификаций требований: Jira, Confluence или аналогичные инструменты.

Краткие теоретические сведения

Определение требований к системе и анализ

Определение требований к системе и анализ являются первым этапом создания ИС, на котором требования заказчика уточняются, согласуются, формализуются и документируются. Фактически на этом этапе дается ответ на вопрос: «Для чего предназначена и что должна делать информационная система?». С этой целью применяется системное рассмотрение проблемы формирования требований к информационной системе.

Целью системного анализа формирования требований к ИС является преобразование общих, расплывчатых знаний об исходной предметной области (требований заказчика) в точные определения и спецификации для разработчиков, а также генерация функционального описания системы. На этом этапе определяются и специфицируются:

- внешние и внутренние условия работы системы;
- функциональная структура системы;
- распределение функций между человеком и системой;
- интерфейсы;
- требования к техническим, информационным и программным компонентам системы;
- требования к качеству и безопасности;
- состав технической и пользовательской документации;
- условия внедрения и эксплуатации.

Разработка перечисленных выше спецификаций при создании ИС, предназначенной для автоматизации управленческих процессов, в общем случае проходит четыре стадии.

Первая стадия анализа – структурный анализ предприятия начинается с исследования того, как организована система управления предприятием, обсле-

дования функциональной и информационной структур системы управления, определения существующих и возможных потребителей информации.

По результатам обследования аналитик на первой стадии анализа выстраивает обобщенную логическую модель исходной предметной области, отображающую ее функциональную структуру, особенности основной деятельности и информационное пространство, в котором эта деятельность осуществляется. На этом материале аналитик строит функциональную модель «Как есть» (As Is).

Вторая стадия работы, к которой обязательно привлекаются *заинтересованные представители заказчика*, а при необходимости и независимые эксперты, состоит в анализе модели *«Как есть»*, выявлении ее недостатков и узких мест, определении путей совершенствования системы управления на основе выделенных критериев качества.

Третья стадия анализа, содержащая элементы проектирования, создание усовершенствованной обобщенной логической модели, отображающей реорганизованную предметную область или ее часть, которая подлежит автоматизации, – модель *«Как должно быть»* (As To Be).

Заканчивается процесс (*четвертая стадия*) разработкой «Карты автоматизации», представляющей собой модель реорганизованной предметной области, на которой *обязательно обозначены «границы автоматизации»*.

В большинстве случаев модель «*Как есть»* улучшается системным аналитиком за счет устранения очевидных несоответствий и узких мест, а полученный таким образом вариант модели рассматривается в дальнейшем в качестве предварительной модели *«Как должно быть»*, которая впоследствии дополняется в соответствии со стратегией развития предприятия.

На *стадии анализа требований* для проектируемой системы вводятся:

– классы пользователей и соответствующие диаграммы бизнес-процессов;

– модели (диаграммы) процессов прикладной деятельности и соответствующие перечни функциональных задач ИС;

– классы объектов предметной области и соответствующие диаграммы «сущность-связь», отражающие информационную модель этой предметной области;

– топология расположения подразделений и пользователей, обслуживаемых данной ИС;

– параметры защиты данных, информации и самой системы.

Основным документом, отражающим результаты работ первого этапа создания ИС, является *техническое задание* (ТЗ) *на проект (разработку)*, содержащее, кроме вышеперечисленных определений и спецификаций, также сведения об очередности создания системы, сведения о выделяемых ресурсах, директивных сроках проведения отдельных этапов работы, организационных процедурах и мероприятиях по приемке этапов, защите проектной информации и т. д.

Требования к информационной системе

Требования – описание функциональности и характеристик системы, которые должны быть выполнены для ее успешной реализации. Требования могут быть функциональными и нефункциональными.

Таким образом, можно сказать, что *требования* – это спецификация того, что и как должно быть реализовано в информационной системе.

Согласно международному глоссарию по терминологии требования включают описание:

– условий или возможностей, необходимых пользователю для решения поставленных проблем или достижения целей;

– функций и ограничений, которыми должна обладать система или системные компоненты, чтобы выполнить договор заказчика на разработку системы;

– положений и регламента используемых стандартов, отображаемых в спецификациях или других формальных документах на систему;

– документированное представление условий, возможностей ограничений среды на проектирование системы.

Основные виды требований к информационной системе

Требования к продукту – это совокупность утверждений относительно свойств или качеств программной системы, подлежащей реализации и определяющей согласованные условия по внешнему поведению системы.

Требования к программному обеспечению состоят из требований пользователей, функциональных, системных и нефункциональных требований.

Требования пользователей описывают цели и задачи, которые пользователи будут выполнять в информационной системе. Требования к интерфейсу и дизайну программы. Этот вид требований может быть описан с использованиемUML: варианты использования, сценарии, прецеденты и т. п.

Системные требования определяют технические условия, характеристики информационной системы, такие как её архитектура, структурные параметры конфигурации, требования к оборудованию, программным компонентам и интерфейсам.

Требования к качеству представляют собой некоторые ограничения к свойствам функций или к системе, важные для пользователей или разработчиков. Например, переносимость, целостность и устойчивость к сбоям системы.

Функциональные требования – это перечень функций или сервисов, которые должна выполнять система, а также ограничений на данные и поведение системы. Спецификация функциональных требований (software requirements specification) включает в себя описание функций, которые не должны быть противоречивыми и взаимоисключающими.

Нефункциональные требования определяют условия и среду выполнения функций (например, защита и доступ к БД, секретность и др.), они непосредственно не связаны с функциями, а отражают пользовательские потребности к выполнению функций.

Спецификация требований к ПО (SRS)

Спецификация требований (SRS) – формализованный документ, в котором подробно изложены все требования к информационной системе. В нем содержится описание всех функций, интерфейсов и нефункциональных требований. Спецификация служит основой для разработки системы, а также помогает гаранти-

ровать, что все стороны, включая заказчиков и разработчиков, имеют четкое представление о том, что должно быть реализовано в рамках проекта.

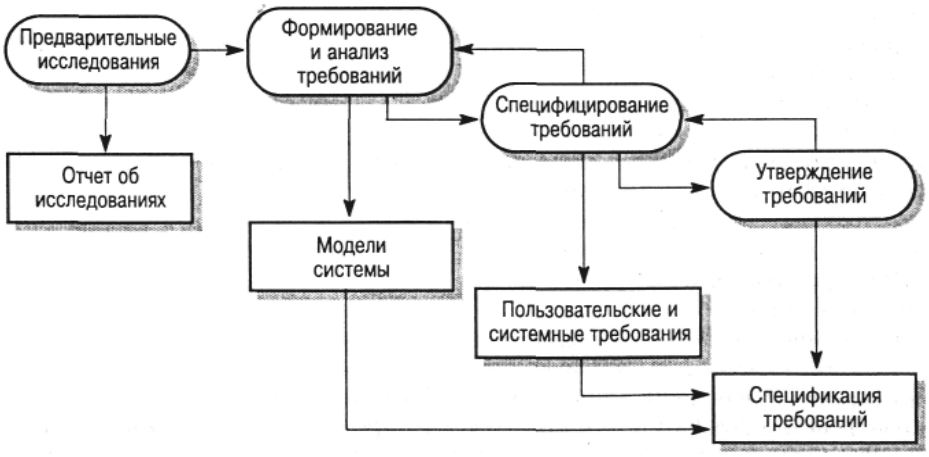


Рис.12.1. Порядок формирования требований к информационной системе

Процесс сбора требований

Процесс сбора требований включает взаимодействие с пользователями, аналитиками и заинтересованными сторонами для выявления, уточнения и документирования всех необходимых требований к системе. Этот процесс важен для того, чтобы обеспечить правильность и полноту требований, а также избежать недоразумений на этапах разработки и внедрения системы. В процессе сбора часто используются методы, такие как интервью, анкеты, анализ документации и прототипирование.

Процесс анализа требований

Анализ требований включает изучение собранных данных, выявление конфликтов, противоречий и неясностей, а также составление конечного списка требований для реализации в системе. Этот процесс помогает уточнить и детализировать требования, устранить противоречия и обеспечить их согласованность. На основе анализа формируется окончательная версия спецификации требований, которая становится основой для проектирования и разработки системы.

Задания к работе

1. Изучить предметную область и собрать требования к ИС, которая была выбрана из вариантов заданий, предложенных в лабораторной работе № 2.
2. Составить спецификацию требований к ИС в формате SRS.
3. Применить методы анализа требований, выявив возможные противоречия и неопределенности.
4. Подготовить отчет с описанием выполненной работы.

Методика выполнения работы

1. Ознакомиться с исходным материалом, провести анализ предметной области и выявить (собрать) потребности пользователей системы. Составить список функциональных и нефункциональных требований.

2. Использовать методы сбора требований (интервью, анкетирование, мозговой штурм и др.) для получения информации от заинтересованных сторон.

3. Разделить собранные требования на функциональные и нефункциональные. Для каждого требования указать критерии приемки. Провести анализ требований, выявив возможные проблемы, противоречия и неясности.

4. Создать спецификацию требований в формате SRS, следуя стандартам документирования (например, IEEE STD 830-1998).

Рекомендации по обработке и оформлению полученных результатов

Оформить отчет, указав цель работы, использованные методы сбора и анализа требований, описание всех функциональных и нефункциональных требований, а также спецификацию требований в формате SRS. Подготовить отчет в установленном формате (PDF-отчет и ZIP-архив с проектом и отчетом).

Вопросы для самоконтроля

1. Что такое функциональные и нефункциональные требования?
2. Как осуществляется сбор требований для информационной системы?
3. Какие методы используются для анализа требований?
4. Что такое спецификация требований SRS и как она оформляется?
5. Как можно проверять и управлять изменениями в требованиях?

Рекомендуемая литература

1. Коберн, А. Современные методы описания функциональных требований к системам / А. Коберн. – Москва : Лори, 2002. – 263 с.

2. Беляева, И. В. Архитектура информационных систем: учебное пособие / И. В. Беляева. – Ульяновск: УлГТУ, 2019. – 192 с. – ISBN 978-5-9795-1918-0.

3. Вигерс, К. Разработка требований к программному обеспечению / К. Вигерс, Дж. Битти. – Москва : Русская редакция, 2014. – 734 с. – ISBN 978-5-7502-0433-5.

Лабораторная работа № 13*
Тестирование и валидация информационной системы

Тема занятия: разработка тестовых сценариев и случаев для функционального и нефункционального тестирования.

Цель занятия: научиться разрабатывать тестовые сценарии для различных типов тестирования, провести тестирование прототипа информационной системы и составить отчет по результатам тестирования.

Материалы и программное обеспечение

1. Средства для написания и управления тестами: TestRail, Jira, Selenium или аналогичные инструменты.
2. Средства для автоматизированного тестирования: JUnit, TestNG, Selenium, Postman для тестирования API.
3. Программное обеспечение для отслеживания ошибок: Jira, Bugzilla.

Краткие теоретические сведения

Тестирование информационной системы

Тестирование – процесс проверки, соответствует ли система заявленным требованиям, работает ли она корректно в различных условиях и обеспечивается ли требуемая производительность.

Функциональное тестирование

Функциональное тестирование направлено на проверку выполнения всех функциональных требований системы. Оценка правильности выполнения функций, описанных в спецификации требований.

Нефункциональное тестирование

Нефункциональное тестирование проверяет характеристики системы, такие как производительность, безопасность, масштабируемость и другие нефункциональные аспекты.

Тестовые сценарии

Тестовые сценарии – детализированные инструкции для проведения тестов, которые описывают, что необходимо проверить в системе, в каком порядке и какие результаты должны быть получены.

Валидация – процесс подтверждения и проверки того, что система соответствует потребностям и ожиданиям пользователей.

Валидация фокусируется на вопросе: «Делаем ли мы правильный продукт?» и включает в себя следующие действия:

- тестирование функциональности;
- тестирование производительности;
- тестирование безопасности;
- тестирование совместимости;
- проведение пользовательского приема.

*Лабораторная работа повышенной сложности, обычно выполняется группой обучающихся.

Пример валидации.

Проверка того, что пользовательский интерфейс программного обеспечения удобен и соответствует ожиданиям пользователей.

Верификация – процесс проверки того, что продукт соответствует определенным требованиям и спецификациям на каждом этапе разработки. Верификация фокусируется на вопросе: «*Делаем ли мы программу правильно?*». Она включает в себя следующие действия:

- анализ требований;
- использование статических методов анализа кода;
- контроль проекта и процессов разработки;
- проведение код-ревью.

Пример верификации.

Проверка того, что требования к программному обеспечению ясны, полны и непротиворечивы.

Основные отличия между верификацией и валидацией.

Цель: верификация проверяет то, что продукт соответствует требованиям и спецификациям, в то время как валидация проверяет то, что продукт соответствует ожиданиям и потребностям пользователей.

Этапы разработки: верификация проводится на каждом этапе разработки, в то время как валидация проводится после завершения разработки.

Методы: верификация включает статические методы анализа (без исполнения кода), в то время как валидация включает динамические методы тестирования (с исполнением кода).

Задания к работе

1. Разработать тестовые сценарии для функционального тестирования прототипа информационной системы, которая была выбрана из вариантов заданий, предложенных в лабораторной работе № 2.
2. Разработать тестовые случаи для нефункционального тестирования (производительность, безопасность и т. д.).
3. Провести тестирование прототипа информационной системы, используя подготовленные тестовые сценарии.
4. Проанализировать результаты тестирования и составить отчет о выявленных ошибках и рекомендациях.

Методика выполнения работы

1. Разработать тестовые сценарии для функционального тестирования, описав действия пользователя и ожидаемые результаты.
2. Создать тестовые случаи для проверки нефункциональных требований, таких как производительность и безопасность.
3. Провести тестирование, используя соответствующие инструменты (например, вручную или с помощью автоматизированных средств).

4. Проанализировать результаты тестирования, выявить ошибки и предложить рекомендации по улучшению системы.

Рекомендации по обработке и оформлению полученных результатов

Оформить отчет, указав цель работы, методы тестирования, описать разработанные тестовые сценарии и случаи, а также результаты тестирования с анализом ошибок и рекомендациями по их исправлению. Подготовить отчет в установленном формате (PDF-отчет и ZIP-архив с проектом и отчетом).

Вопросы для самоконтроля

1. Что такое функциональное и нефункциональное тестирование?
2. Как разрабатываются тестовые сценарии?
3. Какие инструменты используются для автоматизированного тестирования?
4. Как анализировать результаты тестирования?
5. Какие методы используются для валидации информационной системы?

Рекомендуемая литература

1. TestRail – платформа для управления тестами. – URL: https://testrail.com/ (дата обращения: 01.02.2025).
2. Jira – программа управления проектами. – URL: https://atlassian.com/software/jira (дата обращения: 01.02.2025).
3. Selenium – инструмент для автоматизации действий веб-браузера. – URL: https://www.selenium.dev/ (дата обращения: 01.02.2025).
4. JUnit – фреймворк для модульного тестирования программного обеспечения. – URL: https://junit.org/ (дата обращения: 01.02.2025).
5. TestNG – фреймворк для автоматизированного тестирования программного обеспечения. – URL: https://testng.org/ (дата обращения: 01.02.2025).
6. Postman – платформа для разработки API. – URL: https://postman.com/ (дата обращения: 01.02.2025).
7. Sentry – платформа для отслеживания ошибок и мониторинга приложений. – URL: https://sentry.io/ (дата обращения: 01.02.2025).
8. Bugzilla – свободная система отслеживания ошибок. – URL: https://bugzilla.org/ (дата обращения: 01.02.2025).
9. Фундаментальная теория тестирования. – URL: https://habr.com/ru/articles/549054/ (дата обращения: 01.02.2025)
10. Лаврищева, Е. Методы и средства инженерии программного обеспечения : лекция 8. Методы проверки и тестирования программ и систем тестирования / Е. Лаврищева, В. Петрухин. URL: https://intuit.ru/studies/courses/2190/237/lecture/6130 (дата обращения: 01.02.2025)

Лабораторная работа № 14
Разработка и подготовка элементов документации на информационную систему

Тема занятия: разработка элементов документации для автоматизированной системы.

Цель занятия: научиться разрабатывать различные виды документации и ознакомление с основными правилами формирования документации.

Материалы и программное обеспечение

1. ГОСТ 19.101-2024. Единая система программной документации. Виды программ и программных документов.

ГОСТ 19.102-77. ЕСПД. Стадии разработки.

ГОСТ 19.503-79. ЕСПД. Руководство системного программиста.

ГОСТ 19.503-79. ЕСПД. Руководство оператора.

ГОСТ 19.401-78. ЕСПД. Текст программы.

ГОСТ 19.404-79. ЕСПД. Пояснительная записка.

ГОСТ 19.701-90 (ИСО 5807-85). ЕСПД. Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения.

2. ГОСТ Р ИСО/МЭК 15910-2002. ИТ. Процесс создания документации пользователя программного средства.

3. ГОСТ Р ИСО/МЭК 9294-93. Информационная технология. Руководство по управлению документированием программного обеспечения»

4. ГОСТ Р ИСО 9127-94. Системы обработки информации. Документация пользователя и информация на упаковке для потребительских программных пакетов.

5. ГОСТ Р 59795-2021. Информационные технологии. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Требования к содержанию документов.

6. РД 50–34.698-90. Методические указания. Автоматизированные системы, требование к содержанию документов.

7. ГОСТ 34.201-2020. Виды, комплектность и обозначения документов при создании автоматизированных систем.

8. Текстовый редактор: MSWord/LibreOffice, Google Docs.

Краткие теоретические сведения

Основу российской нормативной базы в области документирования программных средств (ПС) составляет комплекс стандартов Единой системы программной документации (ЕСПД). Стандарты ЕСПД в основном охватывают ту часть документации, которая создается в процессе разработки ПС, и связаны по большей части с документированием функциональных характеристик ПС.

Документ – уникально обозначенный блок информации для использования человеком, такой как отчет, спецификация, руководство или книга и т. д.

Документация – набор из одного или более связанных документов.

Программный документ – документ, содержащий сведения, необходимые для разработки, изготовления, эксплуатации и сопровождения программного изделия.

Номенклатуру программных документов определяет ГОСТ 19.101-2024 «Единая система программной документации. Виды программ и программных документов». В стандарте приводятся следующие определения.

Программа – совокупность команд и данных, обеспечивающая выполнение заданной последовательности действий средствами вычислительной техники.

В качестве основных видов программ стандартом определяются: *программный компонент*, *программный комплекс*, *комплекс программ*.

Программный компонент – программы, рассматриваемые как единое целое, выполняющие законченную функцию.

Программный комплекс – программа, состоящая из двух или более программных компонентов и/или программных комплексов, выполняющих взаимосвязанные функции.

Комплекс программ – совокупность программ, выполняющих невзаимосвязанные или непосредственно не связанные функции.

Основные виды программных и эксплуатационных документов их краткое содержание представлены соответственно в таблицах 14.1 и 14.2.

Таблица 14.1

Виды программных документов (ГОСТ 19.101-2024)

№	Вид программного документа	Содержание программного документа
1	Спецификация	Состав программы и документации на нее
2	Ведомость держателей подлинников	Перечень предприятий, на которых хранятся подлинники программных документов
3	Текст программы	Символическая запись программы и дополнительные описания и комментарии (при необходимости)
4	Описание программы	Сведения о функциональном назначении, логической структуре и функционировании программы
5	Программа и методика испытаний	Требования к программе и программной документации, подлежащие проверке при испытаниях, а также средства, порядок и методы испытаний
6	Техническое задание	Основания для разработки программы, назначение разработки, требования к программе, требования к программной документации, технико-экономические показатели, стадии и этапы разработки, порядок контроля и приемки
7	Пояснительная записка	Назначение и область применения программы, технические характеристики, ожидаемые технико-экономические показатели, источники, использованные при разработке
8	Эксплуатационные документы	Основные характеристики программы, сведения, необходимые для обеспечения функционирования, эксплуатации и выполнения программы

Таблица 14.2

Виды эксплуатационных документов (ГОСТ 19.101-2024)

№	Вид эксплуатационного документа	Содержание эксплуатационного документа
1	Ведомость эксплуатационных документов	Перечень эксплуатационных документов на программу
2	Формуляр	Основные характеристики программы, комплектность и сведения об эксплуатации программы
3	Описание применения	Сведения о назначении программы, области применения, классе решаемых задач, ограничениях для применения, минимальной конфигурации технических средств
4	Руководство системного программиста	Сведения для проверки, обеспечения функционирования и настройки программы на условия конкретного применения
5	Руководство программиста	Сведения для обеспечения эксплуатации программы
6	Руководство пользователя (оператора)	Сведения для обеспечения выполнения программы

Техническая документация является составляющей проекта по разработке, внедрению, сопровождению, и модернизации информационной системы на всем протяжении жизненного цикла.

Комплекс технических документов, который регламентирует деятельность разработчиков, называется нормативно-методическим обеспечением (НМО). В данный комплекс входят:

- стандарты;
- руководящие документы;
- методики и положения;
- инструкции и т. д.

НМО регламентирует порядок разработки, общие требования к составу и качеству программного обеспечения (ПО), связям между компонентами, а также определяет содержание проектной и программной документации.

Основным назначением *технической документации* является обеспечение эффективных процедур разработки и использование информационной системы как программного продукта, а также организация обмена между разработчиками и пользователями ИС.

Таким образом, можно выделить следующие функции *технической документации*:

1. Определяет условия функционирования ИС.
2. Предоставляет описание функциональных возможностей системы.
3. Фиксирует информацию о принятых и реализованных решениях.
4. Определяет условие функционирования ИС.
4. Содержит информацию по эксплуатации ИС.
5. Обеспечивает и регулирует права различных групп пользователей.
6. Содержит информацию о модернизации системы.

Перед тем как создается *документация,* необходимо выяснить следующие вопросы: предмет и объект документирования, цель, задачи, для кого проводится документирование, условия использования документации, сроки подготовки и др.

Ответы на эти вопросы, как правило, поступают в начале этапа разработки информационной системы.

Требования к технической документации

1. Информация, используемая в документации, должна обладать свойствами, точности, адекватности, четкости, однозначности и представлена в лаконичной форме.

2. Создаваться одновременно с этапом разработки информационной системы.

3. В большинстве случаев документация создается разработчиком.

4. При создании документации необходимо учитывать НМО, и прежде всего государственные и международные стандарты.

По отношению к информационным системам наибольшее значение имеют стандарты проектирования, стандарты оформления проектной документации, стандарты пользовательского интерфейса.

Стандарт проектирования должен устанавливать:

– набор необходимых моделей (диаграмм) на каждой стадии проектирования и степень их детализации;

– правила наименования объектов, оформления диаграмм, включая требования к форме и размерам объектов и т. д.;

– требования к конфигурации рабочих мест разработчиков, включая настройки операционной системы;

– правила интеграции подсистем проекта, правила поддержания проекта в одинаковом для всех разработчиков состоянии, правила проверки проектных решений на непротиворечивость.

Стандарт оформления проектной документации должен устанавливать:

– комплектность, состав и структуру документации на каждой стадии проектирования;

– требования к оформлению документации, включая требования к содержанию разделов, подразделов, пунктов, таблиц и т. д.;

– правила подготовки, рассмотрения, согласования и утверждения документации с указанием предельных сроков для каждой стадии;

– требования к настройке издательской системы, используемой в качестве встроенного средства подготовки документации;

– требования к настройке CASE-средств для обеспечения подготовки документации в соответствии с установленными требованиями.

Стандарт интерфейса пользователя должен устанавливать:

– правила оформления экранов (шрифты и цветовая палитра), состав и расположение окон и элементов управления;

– правила использования клавиатуры и мыши;

– правила оформления текстов помощи;

– перечень стандартных сообщений;

– правила обработки реакции пользователя.

Основные государственные стандарты в области документирования

Отечественными стандартами являются стандарты ЕСПД серии ГОСТ 19.X и комплекс стандартов на автоматизированные системы серии ГОСТ 34.X, созданные в 80–90-е годы XX века. Кроме того, существуют более современные стандарты на программное обеспечение.

Перечень стандартов ГОСТ 19.X

Единая Система Программной Документации

1. ГОСТ 19.001-77. Общие положения.
2. ГОСТ 19.101-77. Виды программ и программных документов.
3. ГОСТ 19.102-77. Стадии разработки.
4. ГОСТ 19.103-77. Обозначения программ и программных документов.
5. ГОСТ 19.104-78. Основные надписи.
6. ГОСТ 19.105-78. Общие требования к программным документам.
7. ГОСТ 19.106-78. Требования к программным документам, выполненным печатным способом.
8. ГОСТ 19.201-78. Техническое задание, требования к содержанию и оформлению.
9. ГОСТ 19.202-78. Спецификация. Требования к содержанию и оформлению.
10. ГОСТ 19.301-79. Программа и методика испытаний. Требования к содержанию и оформлению.
11. ГОСТ 19.401-78. Текст программы. Требования к содержанию и оформлению.
12. ГОСТ 19.402-78. Описание программы.
13. ГОСТ 19.403-79. Ведомость держателей подлинников.
14. ГОСТ 19.404-79. Пояснительная записка. Требования к содержанию и оформлению.
15. ГОСТ 19.501-78. Формуляр. Требования к содержанию и оформлению.
16. ГОСТ 19.502-78. Описание применения. Требования к содержанию и оформлению.
17. ГОСТ 19.503-79. Руководство системного программиста. Требования к содержанию и оформлению.
18. ГОСТ 19.504-79 Руководство программиста. Требования к содержанию и оформлению.
19. ГОСТ 19.505-79. Руководство оператора. Требования к содержанию и оформлению.
20. ГОСТ 19.506-79. Описание языка. Требования к содержанию и оформлению.
21. ГОСТ 19.507-79. Ведомость эксплуатационных документов.
22. ГОСТ 19.508-79. Руководство по техническому обслуживанию. Требования к содержанию и оформлению.
23. ГОСТ 19.601-78. Общие правила дублирования, учета и хранения.
24. ГОСТ 19.602-78. Правила дублирования, учета и хранения программных документов, выполненных печатным способом.
25. ГОСТ 19.603-78. Общие правила внесения изменений.
26. ГОСТ 19.604-78. Правила внесения изменений в программные документы, выполненных печатным способом.

Стандарты ЕСПД практически не имеют содержательной составляющей и дают формальные требования к составу, содержанию и оформлению документов, описывающих программу на разных стадиях ее жизненного цикла.

Комплекс ГОСТ 34.Х задумывался как всеобъемлющий комплекс взаимоувязанных межотраслевых документов и рассчитанный на взаимодействие заказчика и разработчика. Он должен был разрешить проблему «вавилонской башни», при которой в различных отраслях и областях деятельности использовалась плохо согласованная или несогласованная нормативно-техническая документация. Объектами стандартизации являются автоматизированные системы различных видов и все виды их компонентов, а не только программное обеспечение и базы данных. Комплекс рассчитан на взаимодействие заказчика и разработчика, при этом в нем предусмотрено, что заказчик может разрабатывать систему для себя сам.

Перечень стандартов ГОСТ 34.Х

1. ГОСТ 34.003-90 Информационная технология (ИТ). Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Термины и определения.

2. ГОСТ 34.201-89 Информационная технология (ИТ). Комплекс стандартов на автоматизированные системы. Виды, комплектность и обозначения документов при создании автоматизированных систем.

3. ГОСТ 34.320-96 Информационные технологии (ИТ). Система стандартов по базам данных. Концепции и терминология для концептуальной схемы и информационной базы.

4. ГОСТ 34.321-96 Информационные технологии (ИТ). Система стандартов по базам данных. Эталонная модель управления данными.

5. ГОСТ 34.601-90 Информационная технология (ИТ). Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания.

6. ГОСТ 34.602-89 Информационная технология (ИТ). Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы.

7. ГОСТ 34.603-92 Информационная технология (ИТ). Виды испытаний автоматизированных систем.

8. РД 50-34.698-90 Методические указания. Информационная технология. Комплекс стандартов и руководящих документов на автоматизированные системы. Автоматизированные системы. Требования к содержанию документов.

Если разрабатывается документация на программу (систему), созданную под конкретную организацию, следует воспользоваться требованиями ГОСТов 34. Если разрабатывается документация на программу массового применения, то следует использовать ГОСТы серии 19.

Международные стандарты применяются для разработки документации международного уровня. Как правило, они небесплатные, так как разрабатываются не государственными организациями, но в отличие от отечественных разработаны недавно.

В основе практически всех современных промышленных технологий создания программных средств лежит международный стандарт ISO/IEC 12207 Information technology. System and software engineering. Software life cycle processes

(ГОСТРИСО/МЭК 12207-2010 Информационная технология. Системная и программная инженерия. Процессы жизненного цикла программных средств). Первая редакция стандарта ISO/IEC 12207 была опубликована в августе 1995 г. и явилась первым международным стандартом, содержавшим представительный набор процессов жизненного цикла в отношении программного обеспечения, которое рассматривалось как часть большой системы, а также применительно к программным продуктам и услугам. Стандарт определяет процессы, виды деятельности и задачи, которые используются при приобретении программного продукта или услуги, а также при поставке, разработке, применении по назначению, сопровождении и прекращении применения программных продуктов.

Основными характеристиками данного стандарта являются:

1. Динамичность: один процесс при необходимости вызывает другой или его часть, что позволяет реализовать любую модель жизненного цикла.

2. Адаптивность: стандарт предусматривает исключение процессов, видов деятельности и задач, неприменимых в конкретном проекте.

Кроме того, существуют международные стандарты (на английском языке), которые направлены на написание документации:

1. IEEE Std 1063-2001 «IEEE Standard for Software User Documentation» – стандарт для написания руководства пользователя. В документе обозначены требования к структуре, содержимому и формату инструкций пользователя.

2. IEEE Std 1016-1998 «IEEE Recommended Practice for Software Design Descriptions» – стандарт для написания технического описания программы. Представлены рекомендации к документам, описывающим архитектуру программного обеспечения.

3. ISO/IECFDIS 18019:2004 «Guidelines for the design and preparation of user documentation for application software» – стандарт для написания руководства пользователя. В документе есть большое количество примеров. В приложениях есть чек-листы «что не забыть сделать в процессе разработки документации» и «что должно быть».

4. ISO/IEC 26514:2008 «Requirements for designers and developers of user documentation» – стандарт для дизайнеров и разработчиков пользователей документации.

Документация локальных приложений и небольших информационных систем

На все программные компоненты, которые создаются в процессе разработки ИС и всю ИС в целом, разрабатывается программная документация. Виды программной документации описываются в ГОСТ 19.101–77. Виды программ и программных документов.

В общем случае программная документация для небольших локальных ИС и программных приложений состоит из следующих документов:

1. Описание программы. Документ разрабатывается в соответствии с ГОСТ 19.402–78 «Описание программы» на каждый программный компонент, входящий в ИС. В документе описываются назначение компонента, его логическая структура, входные и выходные данные, алгоритмы его работы.

2. Руководство системного программиста. Документ разрабатывается в соответствии с ГОСТ 19.503–79 «Руководство системного программиста. Требования к содержанию и оформлению» для всей ИС в целом.

3. Руководство оператора. Документ разрабатывается в соответствии с ГОСТ 19.505–79 «Руководство оператора. Требования к содержанию и оформлению» для всей ИС в целом.

Документация распределенных информационных систем

Документация для распределенных и корпоративных информационных систем разрабатывается на основе группы стандартов 34.Х и международных стандартов.

Технический проект автоматизированной системы (АС)

Стадия технического проекта подразумевает окончательное формирование решения по автоматизации. Описания совокупности информационного, технического, программного и других видов обеспечения приводятся в соответствующих разделах пояснительной записки к проекту. Состав пояснительной записки согласно РД 50-34-695 представлен ниже.

Содержание пояснительной записки технического проекта АС

Документ содержит разделы:

1) общие положения;

2) описание процесса деятельности;

3) основные технические решения;

4) мероприятия по подготовке объекта автоматизации к вводу системы в действие.

В разделе «Общие положения» приводят:

– наименование проектируемой АС и наименования необходимых документов, их номера и дату утверждения, на основании которых ведут проектирование АС;

– перечень организаций, участвующих в разработке системы, сроки выполнения стадий;

– цели, назначение и области использования АС;

– подтверждение соответствия проектных решений действующим нормам и правилам техники безопасности, пожаро- и взрывобезопасности и т. п.;

– сведения об использованных при проектировании нормативно-технических документах;

– сведения о НИР, передовом опыте, изобретениях, использованных при разработке проекта;

– очередность создания системы и объем каждой очереди.

В разделе «Описание процесса деятельности» отражают состав процедур (операций) с учетом обеспечения взаимосвязи и совместимости процессов автоматизируемой деятельности, формируют требования к организации работ в условиях функционирования АС.

В разделе «Основные технические решения» приводят:

– решения по структуре системы, подсистем, средствам и способам связи для информационного обмена между компонентами системы, подсистем;

– решения по взаимосвязям АС со смежными системами, обеспечению ее совместимости;

– решения по режимам функционирования и диагностированию работы системы;

– решения по численности, квалификации и функциям персонала АС, режимам его работы и порядку взаимодействия;

– сведения об обеспечении заданных в техническом задании (ТЗ) потребительских характеристик системы (подсистем), определяющих ее качество;

– состав функций, комплексов задач (задач), реализуемых системой (подсистемой);

– решения по комплексу технических средств, его размещению на объекте;

– решения по составу информации, объему, способам ее организации, видам машинных носителей, входным и выходным документам и сообщениям, последовательности обработки информации и другим компонентам;

– решения по составу программных средств, языкам деятельности, алгоритмам процедур и операций и методам их реализации.

Графическая часть проекта представлена в виде набором схем и спецификаций, позволяющих получить полное представление о проекте. Конкретные виды рабочей документации проекта представлены в разделе ГОСТ 34.201-89.

Руководство пользователя

1. Документ «Руководство пользователя» должен содержать разделы:

– введение;

– назначение и условия применения;

– подготовка к работе;

– описание операций;

– аварийные ситуации;

– рекомендации по освоению.

2. В разделе «Введение» приводят:

– область применения средства автоматизации;

– краткое описание возможностей средства автоматизации;

– уровень подготовки пользователя;

– перечень эксплуатационной документации, с которой необходимо ознакомиться пользователю.

3. В разделе «Назначение и условия применения» приводят:

– виды деятельности, функции, для автоматизации которых предназначено данное средство автоматизации;

– условия применения средства автоматизации в соответствии с назначением (например, характеристики и конфигурация технических средств, операционная среда и общесистемные программные средства, входная информация, требования к подготовке специалистов и т. п.).

4. В разделе «Подготовка к работе» приводят:

– состав и содержание носителя данных, содержащего загружаемые программы и данные;

– порядок загрузки программ и данных;

– порядок проверки работоспособности.

5. В разделе «Описание операций» приводят:

– описание всех выполняемых функций, задач (комплексов задач), процедур;

– описание операций технологического процесса обработки данных, необходимых для выполнения функций, задач (комплексов задач), процедур.

6. Для каждой операции приводят:

– наименование;

– условия, при соблюдении которых возможно выполнение операции;

– подготовительные действия;

– основные действия в требуемой последовательности;

– заключительные действия;

– ресурсы, расходуемые на операцию.

7. В разделе «Аварийные ситуации» приводят:

– действия в случае несоблюдения условий выполнения технологического процесса, в том числе при длительных отказах технических средств;

– действия по восстановлению программ и/или данных при отказе носителей данных или обнаружении ошибок в данных;

– действия в случаях обнаружения несанкционированного доступа к данным;

– действия в других аварийных ситуациях.

8. В разделе «Рекомендации по освоению» приводят рекомендации по освоению и эксплуатации средства автоматизации, включая описание контрольного примера, правила его запуска и выполнения.

Методика выполнения работы

Ознакомиться с предлагаемыми государственными стандартами на разработку и подготовку документации. Изучить стандарты в части документирования программного средства. Ответить на контрольные вопросы, подготовить руководство пользователя и пояснительную записку к проекту.

Рекомендации по обработке и оформлению полученных результатов

Подготовить отчет с ответами на контрольные вопросы. Оформить отчет в соответствии с установленными требованиями. В отдельных файлах подготовить руководство пользователя и пояснительную записку к проекту.

Вопросы для самоконтроля

1. Какой нормативный документ определяет участников работ по стандартизации, правила разработки стандартов и их взаимосвязь с техническими регламентами?

2. Какие положения устанавливают основополагающие стандарты?

3. Что такое качество документации ПС?

4. Что такое профиль стандартов?

Рекомендуемая литература

1. Липаев, В. В. Документирование сложных программных средств / В. В. Липаев. – Москва : Синтег, 2005 – 123 с.

2. Глаголев, В. А. Разработка технической документации: руководство для технических писателей и локализаторов ПО / В. А. Глаголев. – Санкт-Петербург : Питер, 2008 – 190 с. – ISBN 978-5-388-00101-6.

3. Вендров, А. М. Проектирование программного обеспечения экономических информационных систем / А. М. Вендров. – Москва : Финансы и статистика, 2006. – 544 с. – ISBN 5-279-02937-8.

Подготовка отчета по лабораторной работе

Представить отчет по лабораторной работе в конце занятия или к следующему занятию по согласованию с преподавателем. Отчет принимается только в электронном виде. При подготовке отчета обязательно еще раз обратить внимание на используемые стандарты.

Требования к оформлению отчета

Текст отчета по лабораторной работе оформляется по стандартным требованиям (файлы, подготовленные в редакторе Word/LibreOffice/OpenOffice, которые предусмотрены для реферата или курсовой работы):

1. Все материалы оформляются на листах стандартного формата А4 на одной стороне, все страницы должны быть пронумерованы, на титульном листе номер не проставляется.

2. Рекомендуется использовать шрифт для основного текста отчета: *Times New Roman*14 пт. с межстрочным одинарным интервалом, поля: левое – 2,5 см, правое – 1,5 см, верхнее – 2 см, нижнее – 2 см.

3. Таблицы, схемы и прочий графический материал должны иметь название и соответствующий номер.

4. Рисунки, должны быть пронумерованы, отформатированы по общим требованиям и подписаны.

5. В расчетных выражениях (формулах) должны быть использованы стандартные математические обозначения, все формулы должны быть пронумерованы.

6. При формировании текста отчета необходимо использовать рекомендации ГОСТ Р 2.105-2019 ЕСКД. Общие требования к текстовым документам.

Структура отчета по лабораторной работе

1. Титульный лист (название работы).
2. Цель и основные задачи лабораторной работы.
3. Программы и аппаратное обеспечение, используемые в работе.
4. Ход выполнение лабораторной работы согласно основному заданию.
5. Ответы на контрольные вопросы.
6. Выводы по лабораторной работе.

Глоссарий[1]

Автоматический процесс – процесс, осуществляемый без участия человека.

Автоматизированная система (АС) – система, состоящая из персонала и комплекса средств автоматизации его деятельности, реализующая информационную технологию выполнения установленных функций.

Автоматизированный процесс – процесс, осуществляемый при совместном участии человека и средств автоматизации.

Автоматизированный производственный комплекс – автоматизированный комплекс, согласованно осуществляющий автоматизированную подготовку производства, само производство и управление им.

Алгоритм – конечный набор предписаний для получения решения задачи посредством конечного количества операций.

Алгоритм проектирования – совокупность предписаний, необходимых для выполнения проектирования.

Алгоритм функционирования автоматизированной системы – алгоритм, задающий условия и последовательность действий компонентов.

Взаимодействие автоматизированных систем – обмен данными, командами и сигналами между функционирующими АС.

Документация на автоматизированную систему – комплект взаимоувязанных документов, полностью определяющих технические требования к АС, проектные и организационные решения по созданию и функционированию АС.

Задача автоматизированной системы – функция или часть функции АС, представляющая собой формализованную совокупность автоматических действий, выполнение которых приводит к результату заданного вида.

Задание на проектирование – первичное описание объекта проектирования в заданной форме.

Информационное средство – комплекс упорядоченной относительно постоянной информации на носителе данных, описывающей параметры и характеристики заданной области применения, и соответствующей документации, предназначенный для поставки пользователю.

Информационное обеспечение автоматизированной системы – совокупность форм документов, классификаторов, нормативной базы и реализованных решений по объемам, размещению и формам существования информации, применяемой в АС при ее функционировании.

Информационная технология – приемы, способы и методы применения средств вычислительной техники при выполнении функций сбора, хранения, обработки, передачи и использования данных.

Информационная модель – модель объекта, представленная в виде информации, описывающей существенные для данного рассмотрения параметры и переменные величины объекта, связи между ними, входы и выходы объекта, и позволяющая путем подачи на модель информации об изменениях входных величин моделировать возможные состояния объекта.

Компонент автоматизированной системы – часть АС, выделенная по определенному признаку или совокупности признаков и рассматриваемая как единое целое.

---

[1](по ГОСТ 34.003-90. Автоматизированные системы. Термины и определения)

Критерий эффективности деятельности – соотношение, характеризующее степень достижения цели деятельности и принимающее различные числовые значения в зависимости от используемых воздействий на объект деятельности или конкретных результатов деятельности.

Лингвистическое обеспечение автоматизированной системы – совокупность средств и правил для формализации естественного языка, используемых при общении пользователей и эксплуатационного персонала АС с комплексом средств автоматизации при функционировании АС.

Методическое обеспечение автоматизированной системы – совокупность документов, описывающих технологию функционирования АС, методы выбора и применения пользователями технологических приемов для получения конкретных результатов при функционировании АС.

Математическое обеспечение автоматизированной системы – совокупность математических методов, моделей и алгоритмов, примененных в АС.

Надежность автоматизированной системы – комплексное свойство АС сохранять во времени в установленных пределах значения всех параметров, характеризующих способность АС выполнять.

Организационное обеспечение автоматизированной системы – совокупность документов, устанавливающих организационную структуру, права и обязанности пользователей и эксплуатационного персонала АС в условиях функционирования, проверки и обеспечения работоспособности АС.

Объект деятельности – объект (процесс), состояние которого определяется поступающими на него воздействиями человека (коллектива) и, возможно, внешней среды.

Пользователь автоматизированной системы – лицо, участвующее в функционировании АС или использующее результаты ее функционирования.

Правовое обеспечение автоматизированной – совокупность правовых норм, регламентирующих правовые отношения при функционировании АС и юридический статус результатов ее функционирования.

Программное обеспечение автоматизированной системы – совокупность программ на носителях данных и программных документов, предназначенная для отладки, функционирования и проверки работоспособности АС.

Программно-методический комплекс системы автоматизированного проектирования, ПМК САПР – взаимосвязанная совокупность компонентов программного, информационного и методического обеспечения системы автоматизированного проектирования, включая, при необходимости, компоненты математического и лингвистического обеспечения, необходимая для получения законченного проектного решения по объекту проектирования или выполнения унифицированной процедуры.

Проектный документ – документ, выполненный по заданной форме, в котором представлено одно или несколько проектных решений.

Проектное решение – описание в заданной форме объекта проектирования или его части, необходимое и достаточное для определения дальнейшего направления проектирования.

Приемочная документация на автоматизированную систему – документация, фиксирующая сведения, подтверждающие готовность АС к приемке ее в эксплуатацию, соответствие АС требованиям нормативных документов.

Рабочая документация на автоматизированную систему – комплект проектных документов на АС, разрабатываемый на стадии «Рабочая документация», содержащий согласованные решения по системе в целом, ее функциям, всем видам обеспечения АС,

достаточные для комплектации, монтажа, наладки и функционирования АС, ее проверки и обеспечения работоспособности.

Развитие автоматизированной системы – целенаправленное улучшение характеристик или расширение функций АС.

Результат проектирования – проектное решение (совокупность проектных решений), удовлетворяющее заданным требованиям, необходимое для создания объекта проектирования.

Система – совокупность элементов, объединенная связями между ними и обладающая определенной целостностью.

Сопровождение автоматизированной системы – деятельность по оказанию услуг, необходимых для обеспечения устойчивого функционирования или развития АС.

Техническое задание на автоматизированную систему (ТЗ на АС) – документ, оформленный в установленном порядке и определяющий цели создания АС, требования к АС и основные исходные данные, необходимые для ее разработки, а также план-график создания АС.

Техническое обеспечение автоматизированной системы – совокупность всех технических средств, используемых при функционировании АС.

Технический проект автоматизированной системы – комплект проектных документов на АС, разрабатываемый на стадии «Технический проект», утвержденный в установленном порядке, содержащий основные проектные решения по системе в целом, ее функциям и всем видам обеспечения АС и достаточный для разработки рабочей документации на АС.

Технорабочий проект автоматизированной системы – комплект проектных документов АС, утвержденный в установленном порядке и содержащий решения в объеме технического проекта и рабочей документации на АС.

Типовое проектное решение – проектное решение, предназначенное для повторного использования при проектировании.

Управление – совокупность целенаправленных действий, включающая оценку ситуации и состояния объекта управления, выбор управляющих воздействий и их реализацию.

Функция автоматизированной системы – совокупность действий АС, направленная на достижение определенной цели.

Цель деятельности – желаемый результат процесса деятельности.

Эксплуатационная документация на автоматизированную систему – часть рабочей документации на АС, предназначенная для использования при эксплуатации системы, определяющая правила действия персонала и пользователей системы при ее функционировании, проверке и обеспечении ее работоспособности.

Язык проектирования – язык, используемый в системе автоматизированного проектирования и предназначенный для представления и преобразования описаний при проектировании.

Приложение
Образец оформления титульного листа отчета

МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Томский государственный педагогический университет»
(ТГПУ)

Физико-математический факультет

Кафедра информатики

ОТЧЕТ
по лабораторной работе № ___
Наименование работы

Выполнил(а):
студент(ка) 423 гр., ФМФ
Иванов И. И.
Проверил:
к. ф.-м. н., доцент кафедры информатики
Петров И. В.

Томск 20____

Учебное электронное издание

КЛИШИН Андрей Петрович
ПИРАКОВ Фаррух Джамшедович

МЕТОДЫ И СРЕДСТВА ПРОЕКТИРОВАНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ И ТЕХНОЛОГИЙ

Лабораторный практикум

Ответственный за выпуск: *Ю. Ю. Афанасьева*
Корректор: *Н. В. Богданова*
Технический редактор: *Ю. А. Ворошилова*

Подписано к использованию: 30.06.2025
Гарнитура Times. Объем издания: 9,1 Мб
Заказ № 073/ЭУ

На обложке используются изображения,
сгенерированные при помощи нейросети ImageFX

Издательство Томского государственного педагогического университета
634061, г. Томск, ул. Киевская, 60
тел. 8(3822)311-484
E-mail: izdatel@tspu.edu.ru