Министерство науки и высшего образования Российской Федерации

Томский государственный университет
систем управления и радиоэлектроники

В.Г. Резник

ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННЫХ СИСТЕМ

Учебное пособие

Томск
2022

УДК 004.8 (004.9)
ББК 65.2-5-05
Р-344

Резник, Виталий Григорьевич

Р-344 Проектирование информационных систем. Учебное пособие / В.Г. Резник. – Томск : Томск. гос. ун-т систем упр. и радиоэлектроники, 2022. – 180 с.

Учебное пособие предназначено для обучения дисциплине «Проектирование информационных систем» для студентов направлений подготовки бакалавриата: 09.03.01 — «Информатика и вычислительная техника».

УДК 004.8 (004.9)
ББК 65.2-5-05

ВВЕДЕНИЕ

Данное учебное пособие содержит материал предназначенный для обучения студентов уровня бакалавриата по дисциплине «Проектирование информационных систем» (ПИС) в рамках направлений подготовки 09.03.01 — «Информатика и вычислительная техника».

Предметная область изучаемой дисциплины является частью более обширного научно-технического направления — «Проектирование автоматизированных систем (АС)». В этом отношении «Информационная система» (ИС) понимается как подсистема АС, обеспечивающая функционирование других её подсистем. В более широком смысле ИС понимается «Интегрированная информационная среда» (ИИС), в которой осуществляется не только управление предприятием, но и вся её производственная деятельность.

Целью изучаемой дисциплины ПИС является теоретическая и практическая подготовка студентов к их последующей производственной деятельности на уровне квалификации бакалавриата.

Основной задачей изучения дисциплины является профессиональная подготовка студента, обеспечивающая последующее успешное прохождение производственной практики, написание и защиту выпускной квалификационной работы (ВКР).

В процессе обучения студент использует литературные источники, рекомендованные программой обучения по данному курсу, а по результатам обучения должен:

а) знать содержание и границы своей профессиональной предметной области, в пределах которой осуществляется его проектная деятельность; стадии и этапы проектирования автоматизированных систем; роль и место назначения информационной подсистемы в процессах управления организациями и их производственной деятельностью; теоретические основы концептуального и объектного проектирования информационных систем предприятий; технологии структурного анализа и моделирования информационных потребностей организаций; инструментальные средства проектирования информационных систем; программные средства реализации информационных систем и сопутствующего им программного обеспечения; правила оформления проектной документации.

б) уметь проводить анализ требований к информационной системе; осуществлять концептуальное проектирование информационной системы на основе моделей IDEF0, IDEF3 и DFD; осуществлять объектное проектирование информационных систем на основе распределенной сервис-ориентированной архитектуры, диаграмм классов, пакетов и модели «Сущность-связь»; выполнять проект концептуальной модели базы данных информационной системы; разрабатывать экранные формы и отчёты для обеспечения решения задач информационной системы; разрабатывать архитектуру программного обеспечения информационных систем; выявлять готовый программный продукт и задачи, требующие разработку нового программного обеспечения; выполнять постановку задач на новое программное обеспечение и формировать техническое задание на их реализацию.

в) владеть навыками создания информационных систем в интегрированных средах разработки программного обеспечения, СУБД и серверах приложений.

Общее содержание дисциплины по главам отражает следующую познавательную тематику:

а) **Тема 1**. Введение в предметную область дисциплины.

б) **Тема 2**. Формирование требований к ИС.

в) **Тема 3**. Концептуальное проектирование ИС.

г) **Тема 4**. Объектное проектирование ИС.

Раздел 1 даёт краткое описание изучаемой предметной области и базовый обзор современных ИТ-технологий, применяемых в проектировании информационных систем. Основная задача этого раздела — обеспечить студента необходимым набором терминов и их определений, существующих в изучаемой предметной области дисциплины, и дать им максимально точное семантическое толкование, опирающееся на имеющиеся стандарты ИТ-технологий.

Раздел 2 описывает начальную стадию процесса проектирования ИС, ориентированную на построение наиболее абстрактной и неопределённой её модели в рамках ещё более абстрактной макромодели предприятия. Основная задача этого раздела — обеспечить студента рядом общих методик, формально помогающих студенту сформировать описание среды, в которой должна существовать проектируемая ИС.

Раздел 3 описывает методики концептуального проектирования ИС на основе функционального моделирования бизнес-процессов, которые ИС должна обслуживать. Приводится достаточно подробное описание трёх графических структурных моделей IDEF0, IDEF3 (WFD) и DFD, достаточных для всех стадий проектирования ИС. Даются также проекции этих моделей на стадии проектирования автоматизированной системы предприятия.

Раздел 4 описывает ряд современных технологий объектного проектирования ИС, которые естественным образом должны быть привязаны к функциональной модели ИС. Здесь предполагается, что целевая ИС может быть распределённой и требовать адекватных инструментальных средств для последующей реализации.

СПИСОК УСЛОВНЫХ ОБОЗНАЧЕНИЙ И СОКРАЩЕНИЙ

АС — автоматизированная система

АСУ — автоматизированная система управления

БД — база данных

ВМ — вычислительная машина

ВС — вычислительная система

ЖЦИ — жизненный цикл изделия

ИИС — интегрированная информационная среда

ИО — информационный объект; информационное обеспечение

ИС — информационная система

ИТ — информационная технология

ООП — объектно-ориентированное программирование

ОС — операционная система

ПО — программное обеспечение

СОД — система обработки данных

СОИ — система обработки информации

СУБД — система управления базами данных

ЭВМ — электронная вычислительная машина

1 ВВЕДЕНИЕ В ПРЕДМЕТНУЮ ОБЛАСТЬ ДИСЦИПЛИНЫ

В процессе обучения по выбранному направлению подготовки и её специализации (09.03.01 или 09.03.03) студент изучал множество дисциплин, каждая из которых имеет свою предметную область, целевые установки, методы и методики применения теории в будущей практической деятельности. На заключительном этапе обучения, студент должен успешно пройти производственную практику, написать и защитить выпускную квалификационную работу (ВКР). В указанном целевом аспекте дисциплина «Проектирование информационных систем» (ПИС) суммирует в себе знания, практическое применение которых будет характеризовать и оценивать степень профессиональной подготовки бакалавра.

В заявленном целевом аспекте дисциплины, первый раздел данного учебного пособия содержит не только наиболее важные термины и определения, касающиеся изучаемой предметной области, но также обсуждает различные их толкования, обусловленные соответствующими целевыми представлениями о предмете проектирования.

В идеале считается, что студент, проходя производственную практику, выбирает тему будущей ВКР, изучает предметную область объекта проектирования и проводит проектные работы, опираясь на знания, изложенные в данном учебном пособии. Свои проектные решения студент согласовывает с требованиями руководителя практики, а итоговые результаты производственной практики использует как исходный материал для будущей ВКР. Такой подход считается оптимальной целевой установкой студента в условиях строгого ограничения по времени периода написания и защиты ВКР.

В методическом плане, данный раздел разбит на шесть частей, последовательно излагающих заявленный выше учебный материал.

Первые четыре подраздела с различных позиций тематики раздела раскрывают ключевое базовое понятие «Информационная система» (ИС):

а) *как начальную парадигму* развития вычислительных систем;

б) *как новую парадигму* проектирования сложных программных систем;

в) *как часть общего системного понятия* «Автоматизированная система управления» (АСУ);

г) *как парадигму* CALS-технологий.

Пятый подраздел обсуждает непосредственные вопросы проектирования ИС, привязывая её предметную область к объектному пространству АСУ.

Шестой подраздел описывает конкретную учебную инфраструктуру проектировщика ИС, привязывая её к требованиям изучаемой дисциплины.

1.1 ИС как результат развития вычислительных систем

Словарь по информационным технологиям, оформленный как документ **ГОСТ 33707-2016** «Информационные технологии (ИТ). Словарь» на базе стандартах ISO/IEC 2382:2015, введённый в действие с 01 сентября 2017 года, содержит следующий набор определений [1]: « ... 4.451 **информационная база**: Набор экземпляров типов, соответствующих друг другу и информационной модели, принадлежащей экземпляру рассматриваемой предметной области.

П р и м е ч а н и е — Информационная база может либо не может быть пригодной для компьютерной обработки. Например, её не следует считать пригодной для компьютерной обработки, если она имеет форму рукописного документа. С другой стороны, если она задана в виде базы данных или компьютерного файла, то её следует считать пригодной для компьютерной обработки и, следовательно, её можно также называть объектной базой.

4.452 **информационная система**: Система, организующая обработку информации о предметной области и её хранение.

4.453 **информационный анализ**: Изучение документов и определение объёма формируемой и используемой информации, а также разработка схемы документооборота и модели информационных связей.

4.454 **информационный бит**: Бит, используемый для представления данных пользователя, отличных от целей управления.

4.455 **информационный объект**; ИО: Совокупность данных и программного кода, обладающая свойствами (атрибутами) и методами, позволяющими определённым образом обрабатывать данные. Самостоятельная единица применения и хранения в ИИС.

4.456 **информационный поиск**: Процесс нахождения и выбора (выдачи) требуемой (т. е. определённой заранее заданными признаками) информации из отдельного текста, документа, совокупности документов или вообще из запоминающего устройства любой физической природы.

4.457 **информация (в области обработки информации)**: Любые данные, представленные в электронной форме, написанные на бумаге, высказанные на совещании или находящиеся на любом другом носителе, используемые финансовым учреждением для принятия решений, перемещения денежных средств, установления ставок, предоставления ссуд, обработки операций и т. п., включая компоненты программного обеспечения системы обработки».

В дальнейшем мы будем использовать эти определения как базовые, дополняя их более конкретным семантическим содержанием. Но главное, что должен помнить студент, приведённые определения являются нормативными требованиями для специалистов его направления подготовки и могут быть необязательными или непонятными для специалистов других направлений и профилей обучения.

В процессе прохождения производственной практики студенту будет необходимо общаться с сотрудниками предприятия, которые проходили обучение в разные периоды времени, а их мышление и терминология тесно связаны с их непосредственной производственной деятельностью. Студенту будет необходимо не только собирать нужные сведения о своём объекте проектирования, но и объяснять свою позицию и точку зрения по этому вопросу. А чтобы успешно осуществлять указанную деятельность, студент должен уверенно разбираться в истории вопроса, который мы далее рассмотрим в двух аспектах:

а) *проблемы становления* информационных технологий;

б) *проблемы концепции* «*Программа-массив*».

1.1.1 Становление информационных технологий

На ранних этапах развития вычислительной техники, она использовалась исключительно для организации вычислений [2]. Вместо термина «*Информация*» повсеместно использовался термин «*Данные*», которые с точки зрения разработчика и пользователя программного обеспечения интерпретировались как входные (исходные) и выходные (результат вычислений) данные программы.

Сами данные типизировались (форматировались) как в аспектах аппаратного обеспечения вычислительных машин (ВМ), так и в аспектах языков программирования.

С появлением операционных систем (ОС) стали формироваться объекты, называемые «*Файловые системы*» (ФС). Первоначально они были сильно привязаны к конкретной реализации ОС, но со временем стали стандартизироваться и превратились в «очевидное» простейшее хранилище данных. ОС UNIX даже имеет свою базовую парадигму - «*Все есть файл*», подчёркивая фундаментальное значение этих систем.

С появлением цифровых средств связи, стало возможным как прямое соединение компьютеров, так и объединение их в компьютерные сети. Появилась возможность распределённой обработки данных, включая сетевые файловые системы. Появилось обобщённое понятие «*Системы обработки данных*» (СОД), которое в словаре [1] определяется как:

«4.1270 **система обработки данных:** Система, выполняющая автоматизированную обработку данных и включающая аппаратные средства, программное обеспечение и соответствующий персонал».

В учебном пособии [3] анализируется классификация СОД, предложенная А.М. Ларионовым, С.А. Майоровым и Г.И. Новиковым. Она делит СОД на две группы: *сосредоточенные* и *распределенные* системы.

В пределах изучаемой дисциплины, студент должен хорошо представлять себе классификацию *сосредоточенных систем*, например, в следующей интерпретации:

а) **ЭВМ** — согласованный в функциональном назначении аппаратный конструктив, дополненный базовым ПО ОС до конкретной виртуальной машины; в последующем, такая машина рассматривается как элемент более сложных систем;

б) **Вычислительные системы** — все прикладные аспекты программного обеспечения, реализованные поверх виртуальной машины ОС; в последующем, такие системы также могут рассматривается как элементы более сложных систем;

в) **Вычислительные комплексы** — системы аппаратных средств, ориентированные на повышении вычислительной мощности и надёжности всей конструкции, а также — соответствующей виртуальной машине ОС; в последующем, такой комплекс также может рассматриваться как отдельная ЭВМ или составляющая часть вычислительной системы.

Как альтернатива, дополняющая группу сосредоточенных систем, в структуре СОД имеется *класс распределенных систем*, включающий в себя «*Системы телеобработки*» и «*Вычислительные сети*».

Система телеобработки — *вычислительная система*, которая дополнена аппаратными и программными *средствами независимого удалённого доступа* к ней; каждый пользователь подключается к такой системе по отдельному каналу связи.

Вычислительная сеть — *удалённые друг от друга вычислительные системы*, объединённые между собой программными и аппаратными средствами сетевой связи. В словаре [1, п. 223] этот термин определяется как: «Сеть, узлы которой состоят из компьютеров и аппаратуры передачи данных, а ветви которой являются линиями передачи данных».

Примечание — Обратим внимание, что на этапе становления информационных технологий, термин «*Информация*» понимался как некоторое свойство документов, объектов искусства и литературы доступное для восприятия и обработки только человеком. Поэтому, студенту в общении с сотрудниками предприятий, не имеющими должной подготовки в области информационных технологий, не следует использовать термин «Информация» в силу неверной трактовки его собеседниками. Следует использовать общедоступные термины дисциплин «Вычислительная математика», «Программирование», «Операционные системы», «Сети и телекоммуникации».

1.1.2 Парадигма «Программа-массив»

Начальный этап становления информационных технологий характеризуется:

а) *централизованной организацией* вычислительных процессов на ВМ мейнфреймов, которая ограничивает доступ разработчиков ПО к необходимым им вычислительным ресурсам;

б) *слабыми вычислительными ресурсами* вычислительных систем, включая быстродействие процессоров, размеры оперативной и внешней памяти, состава системного и инструментального программного обеспечения.

В условиях указанных ограничений, программное обеспечение разрабатывалось и отлаживалось достаточно долго, а насущная потребность в таких вычислительных приложениях вынуждало программистов использовать парадигму «Программа-массив».

Суть парадигмы «Программа-массив» — *сосредоточение основного внимания* алгоритму решения задачи, *оставляя второстепенное внимание* исходным данным. Более того, формат исходных данных и вывод результата выбирался под реализацию самого алгоритма, что делало их форматы непригодными для автоматической обработки другими программами.

Последствием такого подхода стал кризис в индустрии разработки ПО вызванный выбросом на рынок множества несовместимых между собой программ, решающих достаточно много узкоспециализированных задач, попытки объединения которых порождали потребность создания ещё большего количества программ, преобразующих форматы входных и выходных данных других программ.

С целью более качественного обоснования изложенных выше утверждений, проведём наглядную демонстрацию указанной ситуации двумя примерами.

Пример 1.1. На рисунке 1.1 показаны две программы **A** и **B**, которые реализуют два совместимых алгоритма общей задачи, но не являются совместимыми между собой по структуре и форматам входных и выходных данных. Такая ситуация требует разработки ещё двух программ **AB** и **BA**, обеспечивающих совместимость входных и выходных данных.

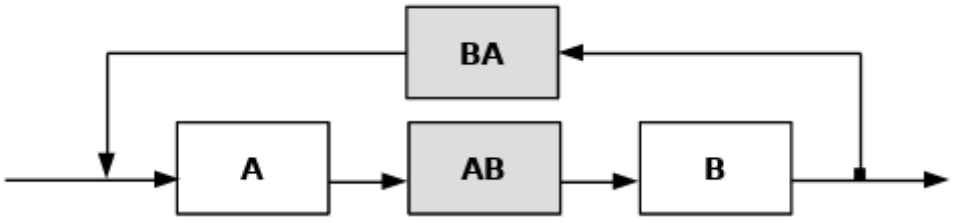


Рисунок 1.1 — Связь программ для Примера 1.1

Пример 1.2. На рисунке 1.2 показаны пять программ, имеющих между собой двунаправленные (дуплексные) связи с попарно несовместимыми интерфейсами.

Вопрос: Сколько дуплексных связей будет, если таких программ — **N**, и сколько нужно написать новых программ, чтобы эта система была программно совместима?

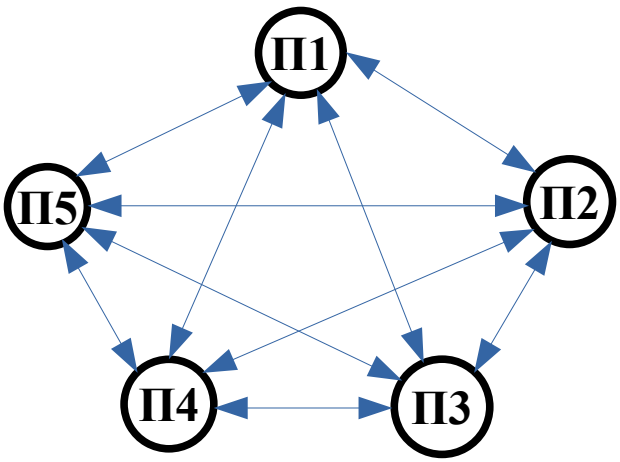


Рисунок 1.2 — Связь программ для Примера 1.2

Решение данного примера является достаточно простым. Действительно, если обозначить через $N_{св}$ и $N_{пр}$ *количество связей* и *новых программ*, то получаем расчётные формулы:

$$N_{св} = N * (N-1)/2; \quad (1.1)$$

$$N_{пр} = N * (N-1) = 2 * N_{св}. \quad (1.2)$$

Таким образом, историческое развитие вычислительных систем, порождает проблему парадигмы «Программа-массив», что приводит к «взрывному» росту количества отдельных программных элементов и требует новых подходов, обеспечивающих более широкое использование средств вычислительной техники.

Как альтернатива чисто вычислительного подхода, выдвигающего на первое место реализацию алгоритмов и порождающего проблему парадигмы «Программа-массив», выступает ***информационный подход,*** породивший парадигмы информационного проектирования.

Цель информационного подхода — устранение недостатков парадигмы «Программа-массив» за счёт проектирования модели предметной области, что должно устранить необходимость разработки алгоритмов и реализацию дополнительных программ, занимающихся исключительно индивидуальным преобразованием данных для базовых программ, обеспечивающих общую целевую функциональность создаваемых систем.

Весь последующий учебный материал данного раздела фактически и посвящён различным аспектам указанной альтернативе — *информационному подходу проектирования ИС*.

1.2 ИС как парадигма информационного подхода

Суммируя итоги предыдущего подраздела, можно смело утверждать, что, с увеличением вычислительных ресурсов ВМ и сложности решаемых задач, парадигма «*Программа-массив*» породила три проблемы:

а) ***сложное взаимодействие*** большого количества программ;

б) ***сложность подготовки и ввода*** большого количества исходных данных;

в) ***сложность сохранения*** результатов расчётов и последующую их обработку.

Указанные три проблемы стимулировали бурное развитие парадигмы информационного подхода, которая стала распространяться как «*Технология проектирования предметной области*», явно выделяя фактор преобладания значимости типизации и хранения данных над значимостью алгоритма.

Парадигма информационного подхода активно пропагандировала централизованное хранение данных в универсальных форматах, что обеспечивалось ресурсами научных, университетских и производственных вычислительных центров, обещая разработчикам ПО единые интерфейсы предоставления исходных данных и результатов вычислений.

Новая парадигма стимулировала развитие нового научного направления — «*Информатика*» (*Informatique* или *Computer science*). В университетах открываются соответствующие направления подготовки студентов, появляются специалисты ИТ-технологий, термин «Информация» получает указанный в предыдущем подразделе смысл.

Обработка информации (данных) — [1]: «Совокупность операций, связанных с хранением, поиском, анализом, оценкой, воспроизведением информации с целью представления её в виде данных, удобных для использования потребителями».

Система обработки информации (СОИ) — [1]: «Совокупность технических средств и программного обеспечения, а также методов обработки информации и действий персонала, обеспечивающая выполнение *автоматизированной обработки информации*».

> Примечание — Студент, изучающий данную дисциплину, должен понимать, что сам термин «*Информация*» или его производные не должен заменять или подменять терминологию конкретных предметных областей, иначе это нанесёт серьёзный ущерб его проектной и производственной деятельности.

Со временем, парадигма информационного подхода, сформулированная как «*Технология проектирования предметной области*», утратила свой первоначальный смысл и разделилась на два направления: «*Технология описания предметной области*» и «*Технология универсального управления данными*».

1.2.1 Технологии описания предметной области

Технологии описания предметной области предполагают теоретическое изучение сложных социальных или производственных систем и последующее документирование этих описаний. Подобную деятельность и стали называть «*Проектированием*».

Очевидно, что такое проектирование сильно зависит от свойств самой предметной области, поэтому, рассматривая проект как систему, можно выделить только обобщённые его части, показанные на рисунке 1.3.

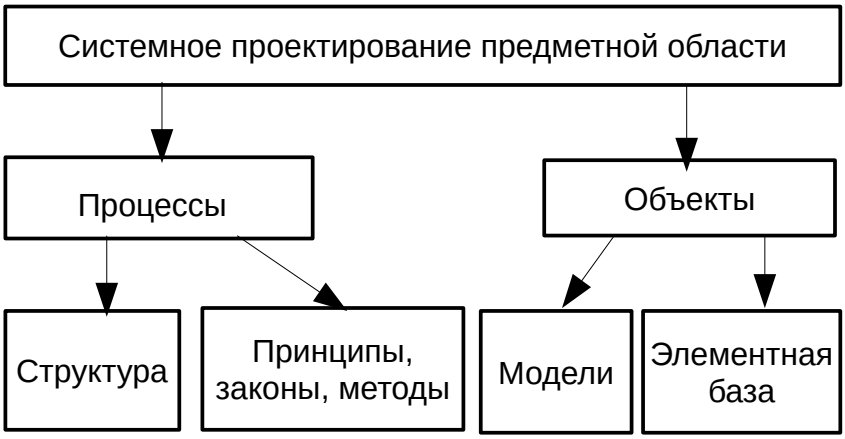


Рисунок 1.3 — Основные части системного проектирования

В общем случае, системное проектирование достаточно чётко выделяет *функциональный* и *объектный* аспекты описания предметной области.

В начале 1970-х годов, Дуглас Т. Росс стал пропагандировать методику «*Структурный анализ и проектирование*» (SADT, Structured Analysis and Design Technique), в которой предлагалось проводить функциональное описание систем в виде набора графических диаграмм.

В конце 1970-х годов, департамент военно-воздушных сил США реализовал идею SADT посредством стандартов **IDEF** (*ICAM Definitions*):

1) **IDEF0** — Function Modeling (Функциональное моделирование);
2) **IDEF1** — Information Modeling (Моделирование информационных потоков внутри системы);
3) **IDEF1X** — Data Modeling (Моделирование баз данных на основе модели «Сущность-связь»);
4) **IDEF2** — Simulation Model Design (Динамическое моделирование развития систем);
5) **IDEF3** — Process Description Capture (Документирование технологических процессов);
6) **IDEF4** — Object-Oriented Design (Проектирование объектных систем);
7) **IDEF5** — Ontology Description Captue (Проектирование онтологии систем);
8) **IDEF6** — Design Rational Capture (Проектирование обоснования проектных действий);
9) **IDEF7** — Information Auditing (Аудит информационных систем);
10) **IDEF8** — User Interface Modeling (Разработка интерфейсов пользователя);
11) **IDEF9** — Business Constraint Discovery method (Метод исследования бизнес-ограничений);
12) **IDEF10** — Implementation Architecture Modeling (Моделирование архитектуры выполнения);
13) **IDEF11** — Information Artifact Modeling (Информационное моделирование артефактов);
14) **IDEF12** — Organization Modeling (Организационное моделирование);
15) **IDEF13** — Three Schema Mapping Design (Трёхсхемное проектирование преобразования данных);
16) **IDEF14** — Network Design (Проектирование компьютерных сетей).

Не все из перечисленных стандартов получили должное признание и применение. Наиболее популярными из них являются: IDEF0, IDEF1X и IDEF3. Дополнительно, широкое распространение получила модель ***DFD*** или ***Data Flow Diagrams***.

> Студент должен хорошо осознавать, что *функциональный* и *объектный* аспекты описания предметной области понимаются в обобщённом смысле, а не в терминах языков программирования.

В середине 1990-х годов, группа сотрудников компании Rational Software (Гради Буч, Джеймс Рамбо и Ивар Якобсон) стали разрабатывать структурные методы описания предметной области программного обеспечения для языков ООП. В дальнейшем общее описание этих методов получило название **UML** (*Unified Modelling Language*) или «*Унифицированный язык моделирования*».

В 1996 году, прототип языка UML, был передан консорциуму **OMG** (*Object Management Group*), который ***в 1997 году*** выпустил спецификацию языка **UML 1.0**.

Далее:

а) *в июле 2005 года* была опубликована спецификация **UML 2.0**;

б) *в апреле 2012 года* были опубликованы стандарты ISO/IEC 19505-1:2012 и ISO/IEC 19505-2:2012.

Отметим, что все диаграммы языка UML сгруппированы в трёх частях, отражающих их следующее прикладное назначение.

Структурные диаграммы (*Structure diagrams*):

а) Диаграммы классов — *Class Diagrams*.

б) Диаграммы компонентов — *Component Diagrams*.

в) Диаграммы составной структуры — *Composite structure Diagrams*.

г) Диаграммы развёртывания — *Deployment Diagrams*.

д) Диаграммы объектов — *Object Diagrams*.

е) Диаграммы пакетов — *Package Diagrams*.

ж) Диаграммы профилей — *Profile Diagrams*.

Диаграммы поведения (*Behavior Diagrams*):

а) Диаграммы деятельности — *Activity Diagrams*.

б) Диаграммы состояний — *State Machine Diagrams*.

в) Диаграммы вариантов использования — *Use case Diagrams* (*диаграммы прецедентов*).

Диаграммы взаимодействия (*Interaction Diagrams*):

а) Диаграммы коммуникаций — *Communication Diagrams*.

б) Диаграммы обзора взаимодействия — *Interaction Diagrams*.

в) Диаграммы последовательности — *Sequence Diagrams*.

г) Диаграммы синхронизации — *Timing Diagrams*.

> Примечание — **Важное свойство языка UML** — его диаграммы ориентированы на поддержку методологии «*Разработка, управляемая моделью*» (*Model Driven Development* или *MDD*). Суть этой методологии — генерация программного кода на основе построенных диаграмм. В частности, имеется свободный фреймворк ***Eclipse Modeling Framework***, который генерирует такой код, инструменты и прочие приложения на основе структурированной метамодели данных.

Таким образом, в настоящее время имеется достаточно большой набор средств для полного и качественного описания предметной области ИС.

1.2.2 Технологии универсального представления данных

Технологии универсального представления данных предполагают разработку универсальных средств хранения и манипулирования этими данными, а также разработку универсальных языков, обеспечивающих доступ и управление этими данными. Со временем, эти технологии сформировали своё собственное технологическое направление и стали называться **системами управления базами данных** (СУБД).

Сами данные размещались в специальных хранилищах, называемых **базами данных** (БД), а доступ к ним осуществлялся согласно *базовой модели структурного представления данных*:

а) **иерархическая модель** — модель, аналогичная архитектуре файловых систем ОС и удобная для отображения классов языков ООП;

б) **сетевая модель** — модель, ориентированная на отображение множественных семантических связей объектов;

в) **реляционная модель** — модель, хорошо ассоциируемая с таблицами представления данных.

Наибольшую популярность получила реляционная модель, поскольку она имеет достаточно простую интерпретацию в виде таблиц данных, достаточно развитый математический аппарат обработки данных и достаточно простой язык манипулирования данными — **SQL** (*Structured Query Language* — «Язык Структурированных Запросов»).

В случае централизованного хранения всех данных, проектирование простейших ИС сводится к следующим основным проектным решениям:

а) с помощью **ER-модели** (*ERM, Entity-Relationship Model*, *модель «сущность — связь»*) *описывается концептуальная схема предметной области*;

б) *проводится логическое проектирование*, которое заканчивается построением **даталогической модели**;

в) *выбирается СУБД* и *проводится* **физическое проектирование**, которое заканчивается построением схемы базы данных (*схемы БД*).

> Примечание — Студент достаточно хорошо знаком со всеми аспектами работы с реляционными СУБД, пройдя обучение по дисциплине «Базы данных», поэтому более подробно эти вопросы в данной дисциплине не рассматриваются.

Уже отмеченный успех реляционной модели вполне обоснован, например, благодаря мощным возможностям языка SQL, практически отпала необходимость создания таких систем как «*Генераторы отчётов*», которые необходимы для ИС построенных на других архитектурах данных. Тем не менее, реляционные модели обладают рядом существенных недостатков, необходимых для понимания проектировщику ИС. И основным их недостатком является ***чувствительность к стабильности схемы БД***.

Формально, схема БД описывается *сценарием на DDL* (*Data Definition Language*) — языке описания данных конкретной СУБД. После запуска сценария или взаимодействия с СУБД в интерактивном режиме, схема БД создаётся и не может меняться в процессе эксплуатации БД.

Поскольку даталогическая модель БД проектируется на основе ERM-описания предметной области, то любые изменения в отношениях сущностей (*Unity*) предметной области приводят и к изменению схемы БД. Вариантами такого изменения отношений предметной области, например, могут быть:

а) *замена централизованной схемы хранения* и обработки данных на распределённую, предполагающую использование нескольких СУБД;

б) *внедрение готовых ИС* или обрабатывающих центров, имеющих свои оригинальные форматы представления данных.

Обратим внимание, что технологии универсального представления данных разрабатываются не только в рамках технологического направления СУБД. Все достаточно сложные программные системы создают свои специализированные представления данных, многие из которых становятся достаточно популярными и используются в ИС. В качестве примера приведём наборы офисных приложений, ставшие привычным дополнением к ПО ОС.

Типовой набор офисных приложений включает: *текстовый процессор*, *электронную таблицу*, *программу подготовки презентаций*, *систему управления базами данных*, *графическую программу* и *редактор формул*.

Первоначально, все эти программы имели оригинальные способы представления данных, сохраняемых в корпоративных форматах файлов. Последние версии этих программ используют стандартизированные способы представления данных, основанные на языке **XML** (*e**X**tensible **M**arkup **L**anguage*). Так:

а) корпорация Microsoft использует представления «*Microsoft Office Open XML*», соответствующие проекту ISO/IEC IS 29500:2008;

б) LibreOffice ориентируется на международный формат «*OpenDocument Format*», утверждённый стандартом ISO/IEC 26300:2006.

> Примечание — Приступая к прохождению производственной практики, студент должен хорошо понимать, что проектируемая им ИС во многом будет определяться ***структурой предметной области***, а важнейшим свойством этой структуры является «***Управление***».

Таким образом, парадигма информационного подхода ориентирована на создание инструментальных средств для тематики проектирования информационных систем.

1.3 ИС как подсистема АСУ

С конца 70-х годов (ещё во времена СССР), с целью внедрения передовых технологий в управление производством, создаётся серия стандартов **ГОСТ 24.xxx** — Система технической документации на АСУ (Единая система стандартов автоматизированных систем управления). Эти стандарты охватывали различные аспекты управления производством, а также определяли различные требования к самой системе управления:

а) **ГОСТ 24.101-80** — Виды и комплектность документов;

б) **ГОСТ 24.102-80** — Обозначение документов;

в) **ГОСТ 24.103-84** — Автоматизированные системы управления. Общие положения;

г) **ГОСТ 24.104-85** — Автоматизированные системы управления;

д) **ГОСТ 24.202-80** — Требования к содержанию документа «Технико-экономическое обоснование»;

е) **ГОСТ 24.203-80** — Требования к содержанию общесистемных документов;

ж) **ГОСТ 24.204-80** — Требования к содержанию документа «Описание постановки задачи»;

з) **ГОСТ 24.205-80** — Требования к содержанию документов по информационному обеспечению;

и) **ГОСТ 24.206-80** — Требования к содержанию документов по техническому обеспечению;

к) **ГОСТ 24.207-80** — Требования к содержанию документов по программному обеспечению;

л) **ГОСТ 24.208-80** — Требования к содержанию документов стадии «Ввод в эксплуатацию»;

м) **ГОСТ 24.209-80** — Требования к содержанию документов по организационному обеспечению;

н) **ГОСТ 24.210-82** — Требования к содержанию документов по функциональной части;

о) **ГОСТ 24.211-82** — Требования к содержанию документа «Описание алгоритма»;

п) **ГОСТ 24.301-80** — Общие требования к выполнению текстовых документов;

р) **ГОСТ 24.302-80** — Общие требования к выполнению схем;

с) **ГОСТ 24.304-82** — Требования к выполнению чертежей;

т) **ГОСТ 24.602-86** — Состав и содержание работ по стадиям;

у) **ГОСТ 24.703-85** — Типовые проектные решения. Основные положения.

В 90-е годы (после смены экономической модели государства), многие аспекты стандартов серии **24.xxx** стали неприемлемы для новых условий. Прежде всего расширилось само понятие термина «*Управление*», большую популярность стали приобретать стандарты серии **34.xxx** - «*Автоматизированные системы*» (АС). Многие авторы учебных пособий, например, Коцюба И.Ю., Чунаев А.В., Шиков А.Н. Основы проектирования информационных систем. Учебное пособие. – СПб: Университет ИТМО, 2015. – 206 с. [4] стали интерпретировать ИС как АС, необоснованно ссылаясь на жизненные циклы (ЖЦ) АС и этапы его проектирования.

Сложившаяся ситуация не только запутывает и без того сложные вопросы проектирования ИС, но и явно искажает концептуальную суть создаваемых систем. Поэтому, важней-

шая задача данного и следующего подразделов — формирование у студента правильных теоретических представлений о *предметной области изучаемой дисциплины*.

> Примечание — Студентам, изучающим автоматизированное проектирование, настоятельно рекомендуются использовать учебник [5]: Норенков, И.П. Основы автоматизированного проектирования: Учеб. для вузов. 4-е изд., перераб. и доп. - М.: Изд-во МГТУ им. Н.Э. Баумана, 2009. - 430 с.: ил. - (Сер. «Информатика в техническом университете»).

Чтобы более подробно раскрыть заявленную выше тему и устранить концептуальные противоречия во взглядах различных специалистов ИТ-технологий, рассмотрим понятия терминов АСУ и АС в трёх аспектах:

а) ***трёхуровневая архитектура АСУ*** — как базовая управляющая концепция автоматизации деятельности предприятия;

б) ***стандарты и компоненты АС*** — как нормативная база для проектирования автоматизированных систем;

в) ***стадии и этапы проектирования АС*** — как шаблон, в пределах которого осуществляется проектирование ИС.

Остальные уточняющие аспекты предметной области проектирования, касающиеся *жизненного цикла изделий* (ЖЦИ), будут рассмотрены в следующем подразделе.

1.3.1 Трёхуровневая архитектура АСУ

Плановая парадигма хозяйственной деятельности СССР (до 1991 года) естественным образом выводила *аспект управления* на главное место. В условиях такой парадигмы, результат внедрения средств вычислительной техники в промышленное или другое производство (организацию, предприятие) рассматривался как «*Автоматизированная система управления*» (АСУ), имеющая трёхуровневую архитектуру показанную ниже таблицей 1.1.

В плане развития такой архитектуры, уровни *стратегического управления* и *оперативного исполнения* считаются главными, постепенно поглощающими уровень *тактического управления*, поэтому сначала дадим характеристику именно им.

Уровень стратегического управления осуществляет административно-финансовое планирование, называемое АСУ производством или АСУП. Само управление рассматривается как разработка и последующая актуализация соответствующих планов, которые распространяются на нижние уровни АСУ.

В свою очередь, эти планы ограничиваются (управляются) *внешними требованиями государственного управления для юридических лиц*. Поскольку ограничения (условия) на административно-финансовую деятельность юридических лиц — достаточно формализованы, то разрабатываются и внедряются программные продукты известные как ERP, MRP и MRP2.

MRP (*Material Requirements Planning*) — система планирования потребностей в материалах, являющаяся наиболее популярной логистической концепцией.

MRP2 (*Manufacturing Resource Planning*) — система планирования производственных ресурсов, обеспечивающая как операционное, так и финансовое планирование производства.

ERP (*Enterprise Resource Planning*) — система планирования ресурсов предприятия, обеспечивающая организационную стратегию интеграции производства и операций, управление трудовыми ресурсами, финансового менеджмента и управления активами.

Таблица 1.1 — Трёхуровневая архитектура АСУ

Уровень управления	***Уровень АСУ***
Уровень стратегического управления	**АСУП** — «*АСУ производством*» или **ERP**-системы. Стратегическое административно-финансовое планирование и управление, решающее задачи: *что произвести*, *в каких объемах*, *к каким срокам*, *из чего* и прочее. **ERP** (*Enterprise Resource Planning*) — планирование ресурсов предприятия.
Уровень тактического управления	**АСУПП** — «*АСУ производственными процессами*» или «*АСУ подготовки производства*» или **MES**-системы**.** **Уровень** начальников производств, цеховых технологов, диспетчеров, мастеров, решающих задачу: *как произвести заданное*, *по каким технологиям*, *на каких станках*, *в каком порядке выполнять заказы*, чтобы минимизировать издержки и максимально эффективно использовать ресурсы. **MES** (*Manufacturing Execution System*) — система управления производственными процессами.
Уровень оперативного исполнения	**АСУТП** — «*АСУ технологическими процессами».* **Уровень** контроллерного управления, HMI с человеком исполнителем, **SCADA**-системы — решает задачи *поддержания заданных технологических параметров производственных процессов*. **SCADA** (*Supervisory Control And Data Acquisition*) — диспетчерское управление и сбор данных.

В настоящее время считается, что системы ERP полностью покрывают задачи, решаемые системами MRP и MRP2. Особо отметим, что к классу ERP-систем относится и продукция известной российской компании «**1С**».

Уровень оперативного исполнения – система технологических процессов или АСУТП, которая реализует планы, поступившие сверху иерархической цепочки управления. Считается, что поступающие планы преобразуются в программы для станков с ЧПУ (станков с числовым программным управлением) или в программы для роботизированных участков производства.

Высокая степень формализации среды данного уровня, обусловленная отсутствием человеческого фактора, создаёт условия и простор для реализации различных алгоритмов обработки информации. В перспективе — это системы SCADA.

SCADA (*Supervisory Control And Data Acquisition*) — системы диспетчерского управления и сбора данных, которые обеспечивают также в реальном масштабе времени *обработку*, *отображение* и *архивацию информации* об объекте мониторинга или управления.

В настоящее время диспетчерское управление характерно для *воздушного* и *железнодорожного транспорта*, в *химической*, *атомной* и других видах промышленности, где требуется критическое по времени и согласованное по взаимодействиям вмешательство, контроль и ответственность человека. В будущем предполагается, что SCADA-системы будут обычным явлением всех АСУТП, а человеку будет отведена только роль пассивного наблюдателя.

Уровень тактического управления включает в себя задачи, которые по тем или иным причинам оказались *недостаточно автоматизированы*, чтобы их можно было перенести на уровень АСУП или АСУТП.

Для этого уровня используются сокращения **АСУПП** (АСУ производственными процессами) или **MES** (Manufacturing Execution System), а некоторые авторы дают интерпретацию как «АСУ подготовки производства» или используют сокращение **АСТПП** (Автоматизированная система технологической подготовки производства).

Как отмечено выше, уровень АСУПП был намечен к последующему поглощению его другими уровнями, но действительность оказалась другой — задачи тактического управления плохо формализуются по причине большого многообразия производственных процессов, различных критериев их качественной оценки и экономической целесообразности их реализации.

Чтобы преодолеть указанные трудности, ***в 1992 году*** была создана международная организация **MESA** (*Manufacturing Execution System Association*) как ассоциация поставщиков и пользователей MES-систем. ***Цель ассоциации*** — информирование производителей о системах отслеживания выполнения заказов на производстве.

В 1997 году, MESA опубликовала «*Функциональную модель MES*», включающую одиннадцать функций:

1) Контроль состояния и распределение ресурсов (*RAS*, *Resource Allocation and Status*);
2) *Оперативное/Детальное планирование* (*ODS*, *Operations/Detail Scheduling*) — *удалена в модели 2004 года*;
3) Диспетчеризация производства (*DPU*, *Dispatching Production Units*) ;
4) *Управление документами* (*DOC, Document Control*) — *удалена в модели 2004 года*;
5) Сбор и хранение данных (*DCA*, *Data Collection/Acquisition*);
6) Управление персоналом (*LM*, *Labor Management*);
7) Управление качеством продукции (*QM*, *Quality Management*);
8) Управление производственными процессами (*PM*, *Process Management*);
9) *Управление производственными фондами (техобслуживание)* (*MM, Maintenance Management*) — *удалена в модели 2004 года*;
10) Отслеживание истории продукта (*PTG*, *Product Tracking and Genealogy*);
11) Анализ производительности (*PA*, *Performance Analysis*).

В 2001 году, ассоциация сменила своё название на ***Manufacturing Enterprise Solutions Association***, чтобы показать, что область интересов ассоциации включает всё программное обеспечение, используемое на производстве, обмен передовым опытом и инновационными идеями для распространения знаний о решениях в области оперативного управления производственными предприятиями.

В 2004 году, MESA представила новую модель из восьми функций — ***Collaborative Manufacturing Execution System*** (*c-MES*), удалив функции ODS, DOC и MM. Тем не менее и до настоящего времени, уровень MES-систем продолжает существовать и содержать задачи, требующие своей автоматизации.

> Примечание — Завершая теоретические рассуждения по данному пункту подраздела, отметим, что трёхуровневая модель АСУ является одной из важнейших основ классификации предметной области проектирования для всех организаций. На её основе производится и классификация создаваемых и уже используемых систем автоматизации предприятий.

Студенту следует хорошо усвоить назначение каждого из уровней управления АСУ, а также хорошо ориентироваться в используемой терминологии, сокращениях и типах задач, решаемых на каждом из них.

1.3.2 Стандарты и виды обеспечения АС

После 1991 года, смена политического курса России привела и к смене её экономического курса, в смысле отказа от планового управления хозяйственной деятельностью страны. Жёсткие плановые требования к предприятиям сменились конкурентной инициативой и внутреннее управление производственной деятельностью потеряло свой прежний смысл. На первое место стали выходить международные стандарты, а в российских ГОСТ стала широко применяться серия стандартов ***34.xxx***, относящаяся к научно-техническому направлению «Информационные технологии», где термин «Управление» удалён из названия типа системы, подразумевая достаточно широкое его толкование.

Познавательная цель данного пункта — дать максимально точное определение понятию ***АС*** («Автоматизированная система») и уточнить отличие этого термина от термина ***ИС*** («Информационная система»).

Достижение указанной цели осуществим ссылками на стандарты серии ГОСТ 34.xxx, понимая их наследственную связь с ГОСТ 24.xxx, что прямо указано в тексте многих из этих документов:

а) **ГОСТ 34.003-90** — Автоматизированные системы. Термины и определения [6];

б) **ГОСТ 34.201-89** — Виды, комплектность и обозначения документов, при создании автоматизированных систем;

в) **ГОСТ 34.320-96** — Концепции и терминология для концептуальной схемы и информационной базы;

г) **ГОСТ 34.321-96** — Система стандартов по базам данных. Эталонная модель управления [7];

д) **ГОСТ 34.601-90** — Автоматизированные системы. Стадии создания [8];

е) **ГОСТ 34.602-89** — Техническое задание на создание автоматизированной системы [9];

ж) **ГОСТ 34.603-92** — Виды испытаний автоматизированных систем.

Начнём с ГОСТ 34.003-90 [6], где в разделе «1. АВТОМАТИЗИРОВАННЫЕ СИСТЕМЫ. ОБЩИЕ ПОНЯТИЯ» находим два определения.

Автоматизированная система (АС) — [6]: «Система, состоящая из персонала и комплекса средств автоматизации его деятельности, реализующая информационную технологию выполнения установленных функций.

Примечания:

1. В зависимости от вида деятельности выделяют, например следующие виды АС: автоматизированные системы управления (АСУ), системы автоматизированного проектирования (САПР), автоматизированные системы научных исследований (АСНИ) и др.

2. В зависимости от вида управляемого объекта (процесса) АСУ делят, например, на АСУ технологическими процессами (АСУТП). АСУ предприятиями (АСУП) и т. д.».

Интегрированная автоматизированная система (ИАС) — [6]: «Совокупность двух или более взаимоувязанных АС, в которой функционирование одной из них зависит от результатов функционирования другой (других) так, что эту совокупность можно рассматривать как единую АС».

Примечание — Приведённые выдержки из источника [6] наглядно показывают наследственную преемственность ГОСТ 34.xxx от трёхуровневой архитектуры АСУ.

Примечание — Обратите внимание, что в явном виде упоминаются только два уровня управления: АСУП и АСУТП. Более того, для понятия АСУТП выделен даже целый раздел: «*7. АВТОМАТИЗИРОВАННЫЕ СИСТЕМЫ УПРАВЛЕНИЯ ТЕХНОЛОГИЧЕСКИМИ ПРОЦЕССАМИ. ОСНОВНЫЕ ПОНЯТИЯ*».

К сожалению источник [6] не даёт прямое определение ИС, но в межгосударственном стандарте ГОСТ 34.321-96 [7], в разделе 4 «Требования к управлению данными», находим такое толкование.

Информационная система — [7]: «... это система, которая организует *процессы сбора*, *хранения* и *обработки информации о проблемной области*. Она может быть размещена на одной или нескольких компьютерных системах. Если информационная система размещена на нескольких компьютерных системах, то она будет рассматриваться как *распределенная информационная система*.

Данные поступают в информационную систему и исключаются из неё, и эти взаимодействия могут осуществляться или людьми, или процессами.

Управление данными в настоящем стандарте будет касаться организации и управления постоянными данными. Постоянные данные — это данные, которые хранятся в информационной системе в течение определённого периода времени. Система, которая выполняет функцию организации и управления постоянными данными, называется *системой управления данными*».

По мнению автора данного пособия, источник [7] несколько сужает понятие ИС, не раскрывая взаимодействие ИС с другими подсистемами АС. Чтобы устранить этот недостаток, обратимся к разделу «*2. ОСНОВНЫЕ КОМПОНЕНТЫ АВТОМАТИЗИРОВАННЫХ СИСТЕМ*» источника [6] (ГОСТ 34.003-90).

Пользователь АС — [6]: «Лицо, участвующее в функционировании АС или использующее результаты её функционирования».

Эксплуатационный персонал АС — Лица, поддерживающие функционирование АС.

Организационное обеспечение АС (**ОО** АС) — [6]: «Совокупность документов, устанавливающих организационную структуру, права и обязанности пользователей и эксплуатационного персонала АС в условиях функционирования, проверки и обеспечения работоспособности АС».

Методическое обеспечение АС (**МетО** АС) — [6]: «Совокупность документов, описывающих технологию функционирования АС, методы выбора и применения пользователями технологических приёмов для получения конкретных результатов при функционировании АС».

Техническое обеспечение АС (**ТО** АС) — [6]: «Совокупность всех технических средств, используемых при функционировании АС».

Математическое обеспечение АС (**МО** АС) — [6]: «Совокупность математических методов, моделей и алгоритмов, применённых в АС».

Программное обеспечение АС (**ПО** АС) — [6]: «Совокупность программ на носителях данных и программных документов, предназначенная для отладки, функционирования и проверки работоспособности АС».

Информационное обеспечение АС (**ИО** АС) — [6]: «Совокупность форм документов, классификаторов, нормативной базы и реализованных решений по объёмам, размещению и формам существования информации, применяемой в АС при её функционировании».

Лингвистическое обеспечение АС (**ЛО** АС) — [6]: «Совокупность средств и правил для формализации естественного языка, используемых при общении пользователей и эксплуатационного персонала АС с комплексом средств автоматизации при функционировании АС».

Правовое обеспечение АС (**ПравО** АС) — [6]: «Совокупность правовых норм, регламентирующих правовые отношения при функционировании АС и юридический статус результатов ее функционирования.

П р и м е ч а н и е. Правовое обеспечение реализуют в организационном обеспечении АС».

Эргономическое обеспечение АС (**ЭО** АС) — [6]: «Совокупность реализованных решений в АС по согласованию психологических, психофизиологических, антропометрических, физиологических характеристик и возможностей пользователей АС с техническими характеристиками комплекса средств автоматизации АС и параметрами рабочей среды на рабочих местах персонала АС».

Таким образом, мы с уверенностью можем утверждать, что **ИС** — это **ИО** АС, представляющая *одну из девяти обеспечивающих подсистем АС*, и в таком аспекте она и должна проектироваться.

Студент должен хорошо знать состав всех обеспечивающих подсистем АС, поскольку их перечень и требования к ним обязательно отражаются в документе «*Техническое задание*» (ТЗ) на проектируемую АС, а значит они должны быть отражены и в описании предметной области ИС.

Если учесть, что **ПравО** АС реализуется в **ОО** АС, то взаимодействие между этими подсистемами можно представить показанным ниже рисунком 1.4.

Если учесть, что все обеспечивающие подсистемы АС взаимодействуют между собой, то проектировщик ИС, учитывая каждое ограничение, которое АС накладывает на **ИО** АС, должен учитывать ещё семь ограничений, накладываемых другими обеспечивающими подсистемами.

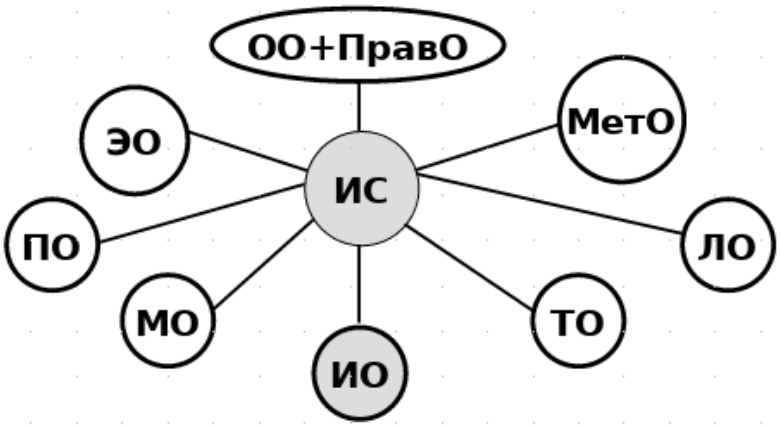


Рисунок 1.4 — Взаимодействие подсистем АС, отображаемых ИС

Примечание — Согласно стандартам АС, студент должен рассматривать и проектировать **ИС** как обеспечивающую подсистему АС (**ИО** АС), которая должна предоставлять всю необходимую информацию остальным обслуживающим подсистема АС.

С такой точки зрения, проектирование ИС напрямую связано со стадиями и этапами создания АС. И здесь уместно сделать три важных замечания, касающиеся обеспечивающих подсистем, проявляющихся в плане их приоритетных отношений:

а) ***в вычислительных системах***, основанных на парадигме «Программа-массив», *подсистема ИО* фактически не существовала как самостоятельное понятие, являясь вспомогательной частью подсистемы ПО;

б) ***в парадигме информационного подхода***, *подсистема ИО* приобретает своё самостоятельное смысловое значение и системные свойства; в совокупности с инструментальными средствами СУБД эти системные свойства формируют понятие информационной системы (ИС);

в) ***в парадигме АСУ***, *существенная часть подсистемы ИО* начинает обслуживать компоненту управления, которая сосредоточена в подсистемах *ОО АС* и *ПравО АС*.

1.3.3 Стадии и этапы создания АС

Познавательная цель данного пункта — дать общее представление о проектировании АС.

> Примечание — Поскольку ИС является обеспечивающей подсистемой АС, то она проектируется на основании требований, предъявляемых к ней АС, а значит — привязана к стадиям и этапам создания АС.

Все стадии и этапы создания АС определены ГОСТ 34.601-90 [8] и в обобщённом виде перечисляют все необходимые для выполнения работы и результирующий перечень документов. Вся эта информация о стадиях и этапах приведена ниже в таблице 1.2.

Стадия создания АС — [6]: «Одна из частей процесса создания АС, установленная нормативными документами и заканчивающаяся выпуском документации на АС, содержащей описание полной, в рамках заданных требований, модели АС на заданном для данной стадии уровне, или изготовлением несерийных компонентов АС, или приёмкой АС в промышленную эксплуатацию».

Этап создания АС — [6]: «Часть стадии создания АС, выделенная по соображениям единства характера работ и (или) завершающего результата или специализации исполнителей».

Очередь АС — [6]: «Часть АС, для которой в техническом задании на создание АС в целом установлены отдельные сроки ввода и набор реализуемых функций».

Безусловно, проектировщик ИС должен досконально знать все стадии и этапы создания АС, но его профессиональный интерес сосредоточен не на всех стадиях одинаково. Поэтому в таблице 1.2 выделим группы стадий, как это показано в таблице 1.3, а затем дадим качественную характеристику каждой группе.

Группа стадий «До ТЗ» — комплекс работ, по завершению которых принимается проектное решение — «*Будет создаваться АС или нет*».

Весь комплекс работ выполняется двумя группами заинтересованных лиц:

а) ***Заказчик*** — организация (предприятие), для которой предполагается создание АС;

б) ***Потенциальные исполнители*** — проектные организации, которые конкурируют за право создания АС, предлагая свои «Технико-коммерческие предложения» (ТКП).

На основании конкурса ТКП и принимается основное проектное решение о создании АС. Если такое решение является положительным, то определяются:

а) головной исполнитель;

б) список частных исполнителей (субподрядчиков);

в) общая стоимость создания АС.

Таблица 1.2 — Стадии и этапы создания АС [8]

Стадии	***Этапы работ***
1. Формирование требований к АС.	1.1. Обследование объекта и обоснование необходимости создания АС. 1.2. Формирование требований пользователя к АС. 1.3. Оформление отчёта о выполненной работе и заявки на разработку АС (тактико-технического задания).
2. Разработка концепции АС.	2.1. Изучение объекта. 2.2. Проведение необходимых научно-исследовательских работ. 2.3. Разработка вариантов концепции АС, удовлетворяющего требованиям пользователя. 2.4. Оформление отчёта о выполненной работе.
3. Техническое задание.	Разработка и утверждение технического задания на создание АС.
4. Эскизный проект.	4.1. Разработка предварительных проектных решений по системе и ее частям. 4.2. Разработка документации на АС и её части.
5. Технический проект.	5.1. Разработка проектных решений по системе и её частям. 5.2. Разработка документации на АС и её части. 5.3. Разработка и оформление документации на поставку изделий для комплектования АС и (или) технических требований (технических заданий) на их разработку. 5.4. Разработка заданий на проектирование в смежных частях проекта объекта автоматизации.
6. Рабочая документация.	6.1. Разработка рабочей документации на систему и её части. 6.2. Разработка или адаптация программ.
7. Ввод в действие.	7.1. Подготовка объекта автоматизации к вводу АС в действие. 7.2. Подготовка персонала. 7.3. Комплектация АС поставляемыми изделиями (программными и техническими средствами, программно-техническими комплексами, информационными изделиями). 7.4. Строительно-монтажные работы. 7.5. Пусконаладочные работы. 7.6. Проведение предварительных испытаний. 7.7. Проведение опытной эксплуатации. 7.8. Проведение приёмочных испытаний.
8. Сопровождение АС.	8.1. Выполнение работ в соответствии с гарантийными обязательствами. 8.2. Послегарантийное обслуживание.

Таблица 1.3 — Группы стадий создания АС

Группа стадий	***Список стадий создания АС***
До ТЗ	1-Формирование требований к АС 2-Разработка концепции АС
Подписание ТЗ	3-Техническое задание
Исполнение системы	4-Эскизный проект 5-Технический проект 6-Рабочая документация (Рабочий проект)
Завершение работ	7-Ввод в действие 8-Сопровождение АС

Примечание — Для студента знание стадий и этапов работ, входящих в группу стадий «До ТЗ», является *важным в методологическом плане знанием*, поскольку ему придётся фактически повторять эти работы во время проектирования ИС.

Группа стадий «Подписание ТЗ» — соответствует стадии «*Техническое задание*» (**ТЗ**), в процессе которой создаются, согласовываются и утверждаются два документа:

а) ***ТЗ на АС*** согласно ГОСТ 34.602-89 [9], где описываются все требования ко всем обслуживающим подсистемам АС, а также перечисляются последующие стадии создания АС;

б) ***Договор на создание АС***, в котором перечислены стадии работ по созданию АС, их стоимость, условия оплаты выполненных работ и порядок разрешения конфликтов сторон.

На стадии «*Техническое задание*» разрешается:

а) ***исключить*** стадию «*Эскизный проект*»;

б) ***объединить*** стадии «*Технический проект*» и «*Рабочая документация*» в одну стадию «*Технорабочий проект*».

После завершения стадии «*Подписание ТЗ*», ТЗ является тем документом, на основании которого оценивается результат проектирования ИС, поэтому, для проектировщика ИС, — это основной источник информации.

Группа стадий «Исполнение системы» — комплекс работ, которые должен выполнить головной исполнитель (*Исполнитель*).

Как отмечено выше, ГОСТ 34.602.89 допускает различное количество стадий этой группы, но они закреплены в требованиях ТЗ и Договоре на создание АС.

Исполнитель, без согласования с Заказчиком, но на основании внутренних распоряжений, может:

а) ***делить*** стадии и этапы работ в пределах ограничений основного ТЗ;

б) ***формировать*** «*Частные технические задания*» (ЧТЗ) на подсистемы АС и программное обеспечение.

Проектировщик ИС должен:

а) определить все подсистемы АС, выделенные **Исполнителем**;

б) подготовить и утвердить у **Исполнителя** необходимые ЧТЗ на ИС;

в) разработать и передать **Исполнителю** Рабочую Документацию на ИС.

После завершения группы стадий «Исполнение системы», Исполнитель передаёт Заказчику:

а) технические и программные средства АС;

б) рабочую документацию на АС.

Группа стадий «Завершение работ» — комплекс работ, которые Исполнитель должен выполнить в соответствии с требованиями ТЗ и условиями Договора.

Примечание — Обычно, на группе стадий «*Завершение работ*», Исполнитель выступает в качестве «Наблюдателя» или «Консультанта».

Завершая тематику данного подраздела, отметим, что стандарты ГОСТ серий 24.ххх и 34.ххх описывают предприятия и организации, продукция которых не меняется или мало меняется со временем. В таких условиях, все проектные решения, касающиеся как АС, так и ИС, будут долгое время сохранять свою актуальность.

С другой стороны в данном подразделе не рассматривалась сама продукция, ради создания которой предприятие и функционирует. Продукция предприятия, даже если потребность в её производстве вполне обоснована, требует проектирования, чтобы удовлетворять

современным условиям. К тому же современный динамичный мир предъявляет к проектным решениям дополнительные требования, которые усиливаются условиями конкурентной борьбы.

Все сказанное выше также требует рассмотрения и анализа деятельности предприятия с позиции создания самой продукции, подчинив этой цели управленческую составляющую его функционирования.

Примечание — Следует заметить, что как АС, так и ИС, являются результатом соответствующей производственной деятельности, поэтому они могут рассматриваться как изделия субъекта, названного **Исполнителем**.

Таким образом, в современных представлениях, ***ИС*** должна рассматриваться как изделие, входящее составной частью в изделие ***АС***, на которые распространяются общие требования теории и практики создания изделий промышленности.

1.4 ИС как парадигма CALS-технологий

В предыдущем подразделе ИС рассматривалась как подсистема АСУ (АС), отражая концептуальные представления, главенствующую роль в которых играет свойство — «*Управление*».

Познавательная цель данного подраздела — описать место ИС в структурном представлении предприятия (организации), когда в идею его архитектуры положен принцип «Жизненного цикла изделия» (ЖЦИ).

Инициатором появления и развития CALS-технологий считается Министерство обороны США, когда оно в 1985 году объявило о создании глобальной АС для электронного описания всех этапов проектирования, производства и эксплуатации продуктов военного назначения. В процессе развития идея CALS-технологий распространились на структуры НАТО, а затем — и на другие отрасли промышленности, где используется большой номинал различной продукции (изделий).

CALS-технологии (*Continuous Acquisition and Life cycle Support*) — зонтичный термин, обозначающий непрерывную информационную поддержку поставок и жизненного цикла изделий.

ИПИ (*Информационная Поддержка Изделий*) — русскоязычный термин, эквивалентный понятию CALS, где информационная поддержка процессов ЖЦИ применяется к проектированию и производству высокотехнологичной и наукоемкой продукции.

Достаточно подробно CALS-технологии описаны в уже рекомендованном ранее учебнике Норенкова И.П. «*Основы автоматизированного проектирования*» [5], где этой теме посвящена отдельная глава «6. ИНФОРМАЦИОННАЯ ПОДДЕРЖКА ЭТАПОВ ЖИЗНЕННОГО ЦИКЛА ИЗДЕЛИЙ — CALS-ТЕХНОЛОГИИ». Мы будем также придерживаться другого достаточно авторитетного источника [10] «Концепция развития CALS-технологий в промышленности России. НИЦ CALS-технологий «Прикладная логистика», который более тесно связан со стандартами Технического комитета по стандартизации ТК 431 «*CALS-технологии*».

Как и архитектура АСУ, концепция CALS-технологий или ИПИ имеет свою вертикальную архитектуру:

а) Инвариантные понятия ИПИ;

б) Стадии ЖЦ изделия (ЖЦИ);

в) Интегрированная информационная среда (ИИС) или Единое информационное пространство (ЕИП);

г) Инструментарий: CAE/CAD/CAM, PDM и другие.

Рассмотрим эти уровни более подробно.

Инвариантные понятия ИПИ — *понятия, имеющие разные названия*, но один и тот же смысл, которые условно делятся на две группы [10]: «*Основные ИПИ принципы*» и «*Базовые ИПИ технологии*».

Основные ИПИ принципы — концептуальные намерения, которые ИПИ должна реализовывать. К ним относятся:

а) *анализ и реинжиниринг бизнес-процессов* (*Business-processes analysis and reengineering*) — фундаментальное переосмысление и радикальное перепроектирование бизнес-процессов;

б) *безбумажный обмен данными* (*Paperless data interchange*) — электронный цифровой обмен данными с использованием ЭЦП (электронной цифровой подписи);

в) *параллельный инжиниринг* (*Concurrent Engineering*) — параллельное выполнение различных стадий создания изделия;

г) *системная организация постпроизводственных процессов ЖЦИ* — интегрированная логистическая поддержка изделия.

Базовые ИПИ технологии — это конкретные технологии, по сути реализующие одни и те же решения. К ним относятся:

а) управление проектом (*Project Management*);

б) управление данными об изделии (*Product Data Management, PDM*);

в) управление конфигурацией изделия (*Configuration Management*);

г) управление ИИС, в том числе информационными потоками (*Information Management*);

д) управление качеством (*Quality Management*);

е) управление потоками работ (*Workflow Management*);

ж) управление изменениями производственных и организационных структур (*Change Management*).

Примечание — Важно отметить [10], что: «ИПИ-технологии реализуются силами многопрофильных рабочих групп, объединяющих в своём составе экспертов различных специальностей».

Стадии ЖЦ изделия (ЖЦИ) — качественные состояния, которые изделие проходит от момента концептуальной идеи его создания до полной утилизации. ***ГОСТ Р 50.1.031-2001*** «Терминологический словарь. ...» [11] выделяет следующие девять стадий ЖЦИ:

1) Маркетинговые исследования;
2) Составление технического задания;
3) Проектирование;
4) Технологическая подготовка производства;
5) Изготовление;
6) Поставка;
7) Эксплуатация;
8) Ремонт;
9) Утилизация.

Примечание — Более подробно стадии ЖЦИ обсуждаются в пункте 1.4.1 данного подраздела.

Интегрированная информационная среда (ИИС) или **ЕИП** — ***единое информационное пространство***) — *совокупность распределенных баз данных*, взаимодействующих по единой системе правил и доступных для всех участников стадий ЖЦИ.

Более подробно ИИС и ЕИП обсуждаются в пункте 1.4.2 данного подраздела.

Инструментарий — *набор программных средств* (программных систем), с помощью которых информация в ИИС *создаётся*, *преобразуется*, *хранится* и *передаётся* от одного участника ЖЦИ к другому. Источник [10] выделяет следующие шесть инструментальных средств:

а) автоматизированные системы конструкторского и технологического проектирования (CAE/CAD/CAM);

б) программные средства управления данными об изделии (изделиях) (PDM);

в) автоматизированные системы планирования и управления производством и предприятием (MRP/ERP);

г) программно-методические средства анализа логистический поддержки и ведения баз данных по результатам такого анализа (LSA/LSAR);

д) программные средства управления потоками работ (WF);

е) методология и программные средства моделирования и анализа бизнес-процессов (SADT) и другие.

Примечание — Более подробно инструментарий обсуждается в пунктах 1.4.3, 1.4.4.

На рисунке 1.5 наглядно показано иерархическое отношение указанных понятий.

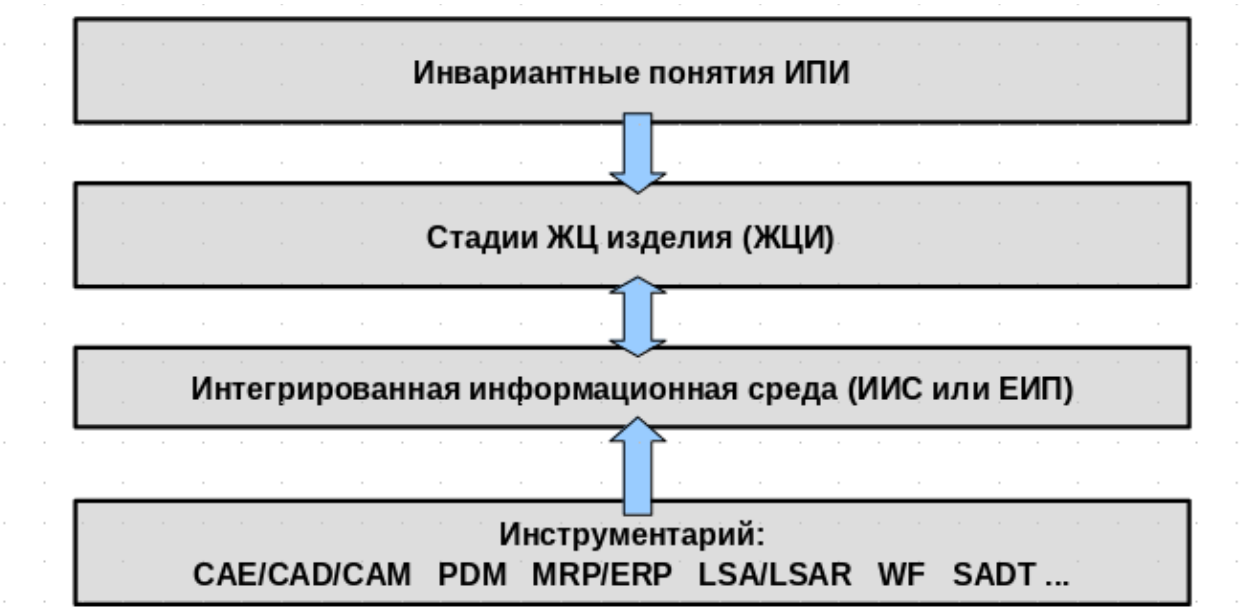


Рисунок 1.5 — Иерархическое представление концепции CALS-технологий

Что касается стандартизации CALS-технологий, то она достаточно запутана и основана на зарубежных источниках. Как отмечает источник [10]: «... приказом Госстандарта РФ №515 от 01.12.99 создан Технический комитет ТК431 «CALS-технологии», силами которого разработан ряд стандартов серии ГОСТ Р ИСО 10303, являющихся адекватными переводами соответствующих международных стандартов». Приведём их список, выделив группировку по общей тематике.

Стандарты группы «Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными»:

а) ГОСТ Р ИСО 10303-1-99 — Часть 1. Общие представления и основополагающие принципы;

б) ГОСТ Р ИСО 10303-11-2000 — Часть 11. Методы описания. Справочное руководство по языку EXPRESS;

в) ГОСТ Р ИСО 10303-12-2000 — Часть 12. Методы описания. Справочное руководство по языку EXPRESS-I;

г) ГОСТ Р ИСО 10303-21-99 — Часть 21. Методы реализации. Кодирование открытым текстом структуры обмена;

д) ГОСТ Р ИСО 10303-22-2001 — Часть 22. Методы реализации. Стандартный интерфейс доступа к данным;

е) ГОСТ Р ИСО 10303-41-99 — Часть 41. Интегрированные обобщённые ресурсы. Основы описания и поддержки изделий;

ж) ГОСТ Р ИСО 10303-45-2000 — Часть 45. Интегрированные обобщённые ресурсы. Материалы;

з) ГОСТ Р ИСО 10303-203-2003 — Часть 203. Протокол применения 203 – Проектирование изделия управляемой конфигурации.

В 2001 году Госстандарт РФ и выпустил ряд документов в статусе ***Рекомендаций по стандартизации*** (РС). Они относятся к стандартам группы «Информационные технологии поддержки жизненного цикла продукции»:

а) ГОСТ Р 5***.01-02 — Техническая документация в электронном виде. Основные положения и общие требования;

б) РС Р 50.1.027-2001 — Автоматизированный обмен технической информацией. Основные положения и общие требования;

в) РС Р 50.1.028-2001 — Методология функционального моделирования;

г) РС Р 50.1.029-2001 — Интерактивные электронные технические руководства. Общие требования к содержанию, стилю и оформлению;

д) РС Р 50.1.030-2001 — Интерактивные электронные технические руководства. Логическая структура базы данных;

е) РС Р 50.1.031-2001 — Терминологический словарь. Часть 1. Стадии жизненного цикла продукции [11];

ж) РС Р 50.1.032-2001 — Терминологический словарь. Часть 2. Применение стандартов серии ГОСТ Р ИСО 10303.

Примечание — Как отмечено в источнике [10]: «Термины, установленные в РС, обязательны для применения во всех видах документации и литературы по технологиям непрерывной информационной поддержки жизненного цикла продукции».

1.4.1 Концепция жизненного цикла изделия (ЖЦИ) применительно к ИС

Как отмечено ранее, концепция жизненного цикла изделия (ЖЦИ) зародилась в недрах министерства обороны США, а затем распространилась на различные отрасли производства и, прежде всего, — на отрасли машиностроения.

В данном пункте мы рассмотрим термин ЖЦИ в более широком смысле — как модель, в которой понятие изделия распространяется на все, что создаётся на предприятии. Например:

а) ***АСУ*** или ***АС*** — как изделие для управления предприятием;

б) ***ИС*** — как изделие для АС или ИИС.

Тогда любая модель ЖЦИ включает в себя следующие обобщённые понятия:

а) *Стадии* и *Этапы* работ;

б) *Результаты* выполнения работ на каждой стадии;

в) *Ключевые события* — точки завершения работ и принятия решений.

В пределах выделенных обобщённых понятий можно использовать одну из трёх разновидностей моделей ЖЦИ: *каскадную*, *итерационную* или *спиралевидную*.

Каскадная модель ЖЦИ — *описывает классический подход* к разработке систем в любых предметных областях. Она предусматривает последовательную организацию работ, которые разбиты на стадии и этапы. Следующая стадия начинается только после завершения предыдущей стадии. Каждая стадия завершается выпуском полного комплекта документации.

Такую модель ЖЦИ для создания АС требует ГОСТ 34.601-90 [8], выделяя наличие восьми последовательно выполняющихся стадий (см. таблицу 1.2, пункта 1.3.1).

Итерационная модель ЖЦИ — *не предполагает чёткого разбиения на стадии и этапы*: система проектируется сразу на нескольких уровнях и состоит из серии коротких циклов (шагов) по ***планированию***, ***реализации***, ***изучению*** и ***действию***. Здесь имеется множество проектных решений *«снизу-вверх»*, когда проектные решения по отдельным задачам объединяются в общие системные решения.

Такую модель ЖЦИ предлагают нам CALS-технологии в плане непрерывной информационной поддержки изделий.

Спиралевидная модель ЖЦИ — *предполагает наличие стадий (этапов)*: ***разработка требований***, ***проектирование***, ***реализация***, ***тестирование***, ***ввод в действие***. Но, в отличие от каскадной модели, эта модель *предполагает итерационный процесс разработки изделия*. Каждая итерация представляет собой законченный цикл разработки, приводящий к выпуску внутренней или внешней версии изделия. На каждой итерации продукция совершенствуется, чтобы стать законченным и более совершенным изделием.

Такая модель ЖЦИ характерна для разработки программного обеспечения больших и достаточно автономных систем, например, ПО офисных систем, когда изделие постоянно совершенствуется, но поступает на рынок как продукт, который объявлен и сопровождается как конкретная версия.

Примечание — В пределах нашей дисциплины, проектирование ИС привязано к жизненного циклу АС, поэтому мы будем опираться на ГОСТ 34.601-90 [8].

1.4.2 Концепции ИИС и ЕИП

В формальном плане термин ИИС определяется в ***Рекомендациях по стандартизации*** (***РС***) ГОСТ Р 50.1.031-2001 [11] в подразделе «*3.2 Интегрированная информационная среда и информационное взаимодействие*». Одновременно там даются понятия, раскрывающие эти термины.

Интегрированная информационная среда (***ИИС***, *Integrated Information Environment*, *IIE*) — [11]: «Совокупность распределенных баз данных, содержащих сведения об изделиях, производственной среде, ресурсах и процессах предприятия, обеспечивающая корректность, актуальность, сохранность и доступность данных тем субъектам производственно-хозяйственной деятельности (ПХД), участвующим в осуществлении ЖЦИ (далее — субъекты ПХД), кому это необходимо и разрешено. Все сведения (данные) в ИИС хранятся в виде информационных объектов».

В тех случаях, когда речь идёт о «виртуальном предприятии», используется термин «Единое информационное пространство» или ЕИП.

Информационный объект (***ИО***, *Information Object*) — [11]: «Совокупность данных и программного кода, обладающая свойствами (атрибутами) и методами, позволяющими определенным образом обрабатывать данные. Самостоятельная единица применения и хранения в ИИС».

Класс информационных объектов (***КИО***, *Entity, Class of Information Objects*) — [11]: «ИО, свойства которого объявлены, но им не присвоены конкретные значения. Класс может порождать экземпляры, наследующие его свойства и методы».

Экземпляр класса (***Instans***) — [11]: «ИО, получающийся из КИО присвоением свойствам конкретных значений».

Коллекция объектов (***КО***, *Information Objects Collection*) — [11]: «Совокупность ИО, относящихся к одному или нескольким классам. КО позволяет добавлять и удалять ИО и обращаться к любому из них. КО может использоваться для создания нового ИО».

Информационное взаимодействие (*Information Interaction*) — [11]: «Совместное использование данных, находящихся в ИИС, и обмен данными, осуществляемые субъектами ПХД, в соответствии с установленными правилами».

Совместное использование данных (*Join data using*) — [11]: «Независимое обращение субъектов ПХД к ИО, находящимся в ИИС, с целью их использования в приложениях или модификации в соответствии с установленными правилами».

Правила информационного взаимодействия и обмена данными — [11]: «Правила, регламентирующие для субъектов ПХД:

а) доступ к ИО;

б) право модификации ИО;

в) право на помещение нового ИО в ИИС;

г) протоколы передачи данных по каналам связи;

д) условия защиты информации в ИИС;

е) структуру и форму обменного файла и т. д.».

Общая база данных об изделиях (***ОБДИ***, *Common product data base, CPDB*) — [11]: «Часть ИИС — хранилище ИО, содержащих в произвольном формате информацию, требуемую для выпуска и поддержки технической документации, необходимой на всех стадиях ЖЦИ, для всех изделий, выпускаемых предприятием. Каждый ИО в ОБДИ идентифицируется уникальным кодом и может быть извлечён из ОБДИ для выполнения действий с ним. ОБДИ обеспечивает информационное обслуживание и поддержку деятельности:

а) *заказчиков* (владельцев) изделия;

б) *разработчиков* (конструкторов), технологов, управленческого и производственного персонала предприятия-изготовителя;

в) *эксплуатационного и ремонтного персонала заказчика*. ОБДИ может состоять из нескольких разделов:

1) нормативно-справочного;

2) долговременного;

3) актуального».

Общая база данных о предприятии (***ОБДП***, *Common enterprise data base*) — [11]: «Часть ИИС — хранилище ИО, содержащих в произвольном формате данные о финансово-экономическом состоянии предприятия, его внешних связях, производственно-технологической среде, действующей на предприятии системе качества и т. д.».

Анализируя уже приведённые определения, становится ясно, что ИИС представляет из себя систему, интегрирующую другие системы, элементами которых являются *информационные объекты* (**ИО**).

Примечание — Учитывая этот факт, сокращение **ИО** (в смысле **ИО АС** — «*Информационное Обеспечение*» по ГОСТ 34.003-90 [6]) далее использоваться не будет.

Следует также заметить, что, в рамках общей БД предприятия, в ИИС выделяются:

а) *база данных по экономике и финансам*;

б) *база данных о внешних связях предприятия*;

в) *база данных о производственно-технологической среде предприятия*;

г) *база данных о системе качества*.

Дополнительно, в том же подразделе, ГОСТ Р 50.1.031-2001 [11] раскрывает понятия *электронного хранилища* и *электронного документа*.

Электронное хранилище (*Vault*) — [11]: «Область хранения ИИС. В хранилище находятся либо ИО, либо информация о путях доступа к ним. Информация в электронных хранилищах контролируется на основе специальных правил и порождаемых ими процессов».

Документ электронный (***ДЭ***, *Electronic Document*) — [11]: «Информационный объект, состоящий из двух частей:

а) реквизитной, содержащей идентифицирующие атрибуты (имя, время и место создания, данные об авторе и т. д.) и электронную цифровую подпись ...;

б) содержательной, включающей текстовую, числовую и/или графическую информацию, которая обрабатывается в качестве единого целого.

При необходимости ДЭ может приобретать различные формы визуального отображения: на экране или бумаге (см. ГОСТ Р 51141)».

Документ технический электронный (***ДТЭ***, *Technical Electronic Document*) — [11]: «Электронный документ, содержательная часть которого включает технические данные».

По отношению к ДЭ и ДТЕ применяются варианты: ***оригинал***, ***подлинник*** и ***копия***.

Документация электронная (*Electronic Documentation, ED*) — [11]: «Комплект ДЭ и/или ДТЭ, объединённых по тематическому или предметному принципу, например техническая документация в электронной форме: чертежи, ведомости, спецификации, сборочные таблицы, сопроводительные документы, производственные требования и стандарты; прочая информация, подготовленная в процессе проектирования и относящаяся к конструкции, производству, контролю, управлению, снабжению и сбыту, испытаниям и контролю изделий. Документация электронная представляет собой коллекцию ИО ..., формируемую в ИИС».

Электронная цифровая подпись (***ЭЦП***, *Digital signature*) — [11]: «Специальное криптографическое средство обеспечения подлинности, целостности и авторства ДЭ или ДТЭ. ЭЦП связывает содержание документа и идентификатор подписывающего лица и делает невозможным изменение документа без нарушения подлинности подписи. Формирование ЭЦП электронного документа или пакета документов (файла или файлов) при их подготовке и передаче, а также проверка наличия и неискажённости подписи обеспечиваются специальными программными средствами (см. ГОСТ Р 34.10)».

Завершим перечень определений, характеризующих ИИС, описанием понятия изделие, которое даётся в подразделе «*3.3 Изделие и предприятие*» рассматриваемого стандарта.

Изделие (Product, Item) — [11]: «По 3.2.26 ГОСТ Р ИСО 10303-1.

Примечание — Изделие может представлять собой *материальный предмет*, *вещество*, *услугу*, *программный продукт*, *систему*, состоящую из материальных предметов и

программных средств, взаимодействующих между собой, являющихся результатом деятельности предприятия».

Таким образом, **концепция ИИС** требует создания единой информационной проекции предприятия.

> Примечание — *Организационно*, эта проекция представляет собой единое корпоративное **Электронное хранилище** (*Vault*), а *логически* — распределенную неоднородную объектную систему, подсистемы которой сосредоточены в базах данных различного назначения, содержащих и предоставляющих доступ к **Информационным Объектам** (**ИО**).

Представленная интерпретация концепции ИИС — весьма привлекательна для базового представления тематики изучаемой дисциплины, поскольку позволяет интерпретировать **ИС** как **ИИС** или её достаточно самостоятельную часть. Тем более, что такая интерпретация ИС является более значимой по сравнению с её проекцией — как обеспечивающей подсистемы **АС**.

С другой стороны, кажущаяся простота концепции ИИС — обманчива. Она переносит проблематику создания **Электронного хранилища** в две плоскости:

а) в проблематику наличия или создания ***Инструментария*** нужного назначения, обеспечивающего создание и манипулирование информационными объектами (ИО);

б) в проблематику наличия или создания ***Распределенной вычислительной сети*** (***РВ-сети***), обеспечивающей взаимодействие различных частей ИИС.

Первый аспект проблем обсуждается в последующих двух пунктах данного подраздела, а тематика второго аспекта проблем — в других темах пособия (по мере необходимости). Здесь лишь отметим, что РВ-сети, в плане своего развития (см. учебное пособие [3]), стали предоставлять сервис ***Виртуальных предприятий***. Если предприятие или организация используют подобный сервис для развёртывания своего ***Инструментария***, то вместо термина ***ИИС*** принято использовать термин ***ЕИП***, который расшифровывается как ***Единое Информационное Пространство***.

1.4.3 САПР и их классификации

Самым нижним уровнем, в иерархии представления концепции CALS-технологий показанном на рисунке 1.5, является **Инструментарий**, который обеспечивает среду ИИС в плане создания, модификации и перемещения ИО или более сложных объектов ДЭ и ДТЭ. Этот инструментарий может быть достаточно разнообразным, но тот, который предназначен для автоматизации проектирования изделий, в России называется **САПР** или **Система автоматизированного проектирования**. Более точное определение САПР даётся в ГОСТ 22487 — 77 «ПРОЕКТИРОВАНИЕ АВТОМАТИЗИРОВАННОЕ. Термины и определения» [12].

Система автоматизированного проектирования (***САПР***) — [12]: «Комплекс средств автоматизации проектирования, взаимосвязанных с необходимыми подразделениями проектной организации или коллективом специалистов (пользователем системы), выполняющий автоматизированное проектирование».

В целом, САПР стали интенсивно развиваться ***с середины 1960-х годов***, когда появились ЭВМ достаточной мощности, чтобы осуществлять подобную деятельность. В частности, в это время американская корпорация IBM выпустила свою систему «*IBM Drafting System*».

В англоязычной литературе термин САПР переводится различными способами. Например на рисунке 1.5 ему соответствует многозначное обозначение **CAE/CAD/CAM**, но

общепринятым обобщённым термином является **CAD** или **Computer-Aided Design**. Чтобы разобраться с указанной неопределённостью обозначений, учтём, что развитие САПР шло параллельно с развитием аппаратных средств вычислительной техники и его программного обеспечения. К тому же САПР — достаточно сложные и дорогие системы, которые классифицируются по двум признакам: *отраслевому* и *целевому* назначениям.

По отраслевому назначению, САПР делятся *на три большие группы* и обозначаются:

а) **MCAD** (*Mechanical Computer-Aided Design*) — автоматизированное проектирование механических устройств. Машиностроительные САПР, применяются в автомобилестроении, судостроении, авиакосмической промышленности, производстве товаров народного потребления. Они включают в себя разработку деталей и сборок (механизмов) с использованием параметрического проектирования на основе конструктивных элементов, технологий поверхностного и объёмного моделирования: SolidWorks, Autodesk Inventor, CATIA;

б) **EDA** (*Electronic Design Automation*) или **ECAD** (*Electronic Computer-Aided Design*) — САПР электронных устройств, радиоэлектронных средств, печатных плат и другие: Altium Designer, OrCAD;

в) **AEC CAD** (*Architecture, Engineering and Construction Computer-Aided Design*) или **CAAD** (*Computer-Aided Architectural Design*) — САПР в области архитектуры и строительства. Используются для проектирования зданий, промышленных объектов, дорог, мостов и другое: Autodesk Atchitectural Desktop, Piranesi, ArchCAD.

По целевому назначению, САПР делятся *на четыре группы* и обозначаются:

а) **CAD** (*Computer-Aided Design/Drafting*) — средства автоматизированного проектирования, предназначенные для автоматизации двумерное или трёхмерного геометрического проектирования, создания конструкторской или технологической документации, САПР общего назначения. Для обозначения данного класса средств САПР используется также термин **CADD** (*Computer-Aided Design and Drafting*) — автоматизированное проектирование и создание чертежей, а системы геометрического моделирования обозначают как **CAGD** (*Computer-Aided Geometric Design*);

б) **CAE** (*Computer-Aided Engineering*) — средства автоматизации инженерных расчётов, анализа и симуляции физических процессов, которые осуществляют динамическое моделирование, проверку и оптимизацию изделий. Подкласс средств CAE, используемый для компьютерного анализа, обозначается термином **CAA** (*Computer-Aided Analysis*);

в) **CAM** (*Computer-Aided Manufacturing*) — средства технологической подготовки производства изделий, которые обеспечивают автоматизацию программирования и управления оборудованием с ЧПУ или ГАПС (Гибких автоматизированных производственных систем). Русскоязычным аналогом термина является **АСТПП** — автоматизированная система технологической подготовки производства.

г) **CAPP** (*Computer-Aided Process Planning*) — средства автоматизации планирования технологических процессов, применяемые на стыке систем CAD и CAM.

Подводя итог анализу терминологии САПР, можно сделать следующие выводы:

а) отсутствуют САПР, которые бы предназначались для непосредственного проектирования ИС;

б) даётся прямое указание на то, что САПР сами являются автоматизированными системами (АС), которые уже были рассмотрены нами в предыдущем подразделе 1.3.

Чтобы более обосновано подтвердить сделанное заявление, снова обратимся к ГОСТ 22487— 77 [12], где перечисленны все обеспечивающие подсистемы САПР. Они полностью совпадают с перечисленными ранее обеспечивающими подсистемами АС. Единственное, что в САПР добавляется проектирующее программное обеспечение. В своей работе [13], автор учебника по автоматизированному проектированию И.П. Норенков представляет *общую архитектуру ПО САПР*, показанную на рисунке 1.6.

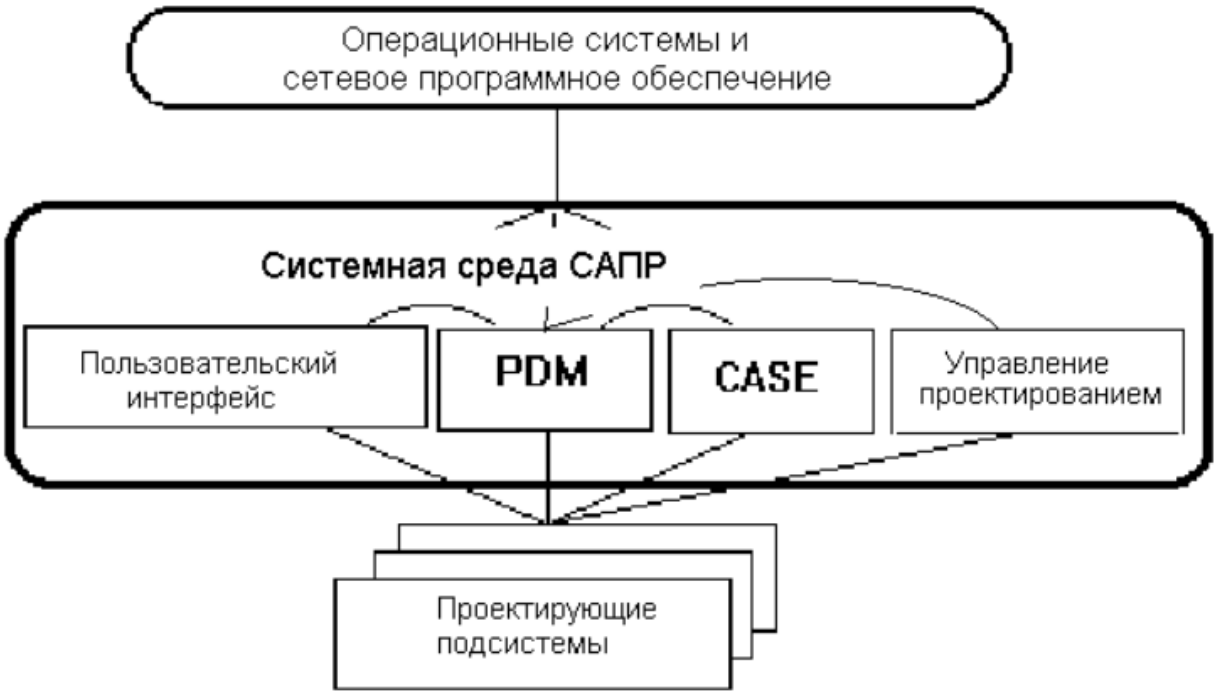


Рисунок 1.6 — Структура программного обеспечения САПР [13]

Хорошо видно, что структура программного обеспечения САПР состоит из трех частей: ОС и сетевого ПО, обеспечивающих подсистем, обозначенных на рисунке как «*Системная среда САПР*», и проектирующих подсистем.

Проектирующие подсистемы непосредственно выполняют проектные процедуры, примерами которых могут служить подсистемы геометрического трёхмерного моделирования механических объектов, изготовления конструкторской документации, схемотехнического анализа, трассировки соединений в печатных платах и другие процедуры.

Обслуживающие подсистемы предназначены для функционирования проектирующих подсистем САПР. Их обычно называют системной средой (или оболочкой) САПР. В качестве типичного состава обслуживающих подсистем выделяют:

а) **PDM** (*Product Data Management*) — подсистема управления проектными данными, являющаяся организационно-технической системой управления всей информацией о сложных изделиях, таких как корабли, самолёты, ракеты, компьютерные сети и другие;

б) **DesPM** (*Design Process Management*) — подсистема управления самим процессом проектирования;

в) подсистема управления пользовательскими интерфейсами для связи разработчиков с ЭВМ;

г) **CASE** (*Computer Aided Software Engineering*) — набор инструментальных средств для разработки и сопровождения программного обеспечения САПР.

Примечание — Практически все обслуживающие подсистемы САПР, кроме CASE-систем, являются достаточно специализированными инструментами, чтобы их можно было изучать в общем курсе по проектированию ИС.

1.4.4 CASE-средства CALS-технологий

Согласно своему названию CASE-средства стали развиваться как инструменты и методы программной инженерии для целей проектирования программного обеспечения. Собственно в таком качестве они и вошли в состав САПР. Но с момента появления стандарта ISO/IEC 14102:2008 «*Guide-line for thee valuation and selection of CASE tools*» — Руководство по оценке и выбору инструментов CASE, эти средства стали рассматриваться как поддержка жизненного цикла ПО, что вполне соответствует тематике данного подраздела.

Здесь совершенно нельзя обойти вниманием широкоизвестный учебник Вендрова А. М. «*Проектирование программного обеспечения экономических информационных систем*» [14], где автор систематически рассматривает ЖЦИ ПО, попутно интерпретируя все уже описанные нами технологии применительно к выбранной тематике. Дополнительно, изложенная в [14] теоретическая часть технологии проектирования ПО дополняется учебным пособием: Вендров А.М. «*Практикум по проектированию программного обеспечения экономических информационных систем*» [15], где автор, используя инструментарий компании ***Rational Software*** и её методологии ***RUP*** (***Rational Unified Process***), демонстрирует ряд проектных решений, включая применение языка UML.

Примечание — Следует обратить внимание, что прямое использование методологии и инструментария CASE-средств ещё не означает проектирование ИС, но представляет значительный интерес в плане её возможной реализации.

В плане тематики нашей дисциплины, CASE-средства интересны тем, что интенсивно используют структурный подход:

а) SADT (Structured Analysis and Design Technique);

б) DFD (Data Flow Diagrams);

в) ERD (Entity-Relationship Diagrams).

Кроме того, в общем случае и в плане функциональной ориентации, CASE-средства подразделяются:

а) средства анализа, предназначенные для построения и анализа модели предметной области;

б) средства проектирования баз данных;

в) средства разработки приложений;

г) средства реинжиниринга процессов;

д) средства планирования и управления проектами;

е) средства тестирования ПО систем;

ж) средства документирования проектов.

В указанных аспектах, кроме достаточно дорогих CASE-средств компании ***Rational Software***, которые ***с 2003 года*** принадлежат корпорации IBM, имеется большое количество свободного ПО, выполняющего аналогичные или подобные функции.

Начнём с того, что первоначально средства автоматизации разработки программного обеспечения стали формироваться как системы **IDE** (*Integrated Development Environment*). В частности, та же корпорация IBM, стремясь навести порядок в своих инструментальных

средствах основанных на языке Java, сформировала ***к 2001 году*** инструментальную систему (IDE) под названием Eclipse («*Затмение*»). Затратив (по сообщениям самой же IBM) порядка 40 миллионов долларов, эта система была передана, ***в феврале 2004 года***, в общественный фонд под названием ***Eclipse Foundation***. ***В июне 2004 года года***, этот фонд выпустил версию Eclipse 3.0, которая полностью соответствовала спецификациям сервисной платформы **OSGi** (*Open Services Gateway initiative*) и приобрела модульную расширяемую структуру. ***В сентябре 2017 года***, правопреемница технологий языка Java корпорация Oracle передала Eclipse Foundation права на использование Java EE (*Java Platform Enterptise Edition*). ***В апреле 2018 года***, Eclipse Foundation переименовала Java EE в новую платформу **Jakarta EE**, добавив в неё поддержку облачных технологий.

Интерес к IDE Eclipse не является случайным. Дело в том, что эта инструментальная платформа хорошо интегрируется с другими инструментальными средствами, имеющими к ИС прямое отношение: СУБД и сервера web-сервисов. На момент написания данного текста (март 2020 года), дистрибутивы Eclipse представлены тринадцатью специализированными системами для ОС MS Windows, Mac Cocoa и Linux. Среди них можно выделить:

а) Eclipse IDE for **Enterprise** Java Developers — среда для разработки приложений уровня предприятий;

б) Eclipse IDE for **Web** and **JavaScript** Developers — среда для разработки современных web-приложений;

в) Eclipse **Modeling** Tools — среда для моделирования приложений, включающая средства построения различных диаграмм.

Разработанная на языке Java, IDE Eclipse имеет средства доступа ко всем известным СУБД посредством ПО **JDBC** (*Java DataBase Connectivity*), но наиболее интересна её тесная связь с ПО организации-фонда **Apache Software Foundation** (ASF).

Фонд ASF был ***основан в 1999 году***, а первым его наиболее известным брендом стал **Apache HTTP-сервер** — свободный web-сервер, написанный на языке C в 1995 году, но поддерживающий модульную структуру, он один из первых реализовал поддержку языка **PHP** (*Hypertext Preprocessor*), открыв дорогу другим решениям основанным на технологии клиент-сервер и протоколе HTTP.

Для нас, с практической точки зрения, более интересен проект **Apache Tomcat**, реализованный ASF в 1999 году на языке Java, он представляет контейнер сервлетов с поддержкой технологий **JSP** (*JavaServer Pages*) и **JSF** (*JavaServer Faces*). В совокупности это позволило разрабатывать web-приложения уровня предприятия.

В последующие годы ASF реализовала или обеспечивала поддержку нескольким десяткам различных проектов, многие из которых содержали различные технологические новинки. Среди них можно отметить:

а) **Apache Derby** — реляционная СУБД, написанная на Java в 1997 году и обеспечивающая как сетевой, так и встроенный варианты подключения приложений;

б) **Apache OpenOffice** — свободный пакет офисных приложений, реализованный в апреле 2002 года и одним из первых поддерживающий новый открытый формат ISO/IEC 26300 (*OpenDocument*);

в) **Apache TomEE** — свободный сервер приложений, начатый разрабатываться в апреле 2012 года и полностью реализованный в сентябре 2017 года.

Современный уровень развития свободного программного обеспечения вполне реализует замену многих CASE-средств устаревших технологий.

1.4.5 Выводы по подразделу

Таким образом, парадигма CALS-технологий является наиболее современной концепцией построения ИС, поскольку она включает в себя достижения всех предыдущих парадигм, а также вносит свои собственные концептуальные представления, которые отсутствуют у рассмотренных ранее.

Примечание — Прежде всего отметим, что парадигма технологий и ИС, как система, — это совершенно разные, хотя и связанные, понятия.

Парадигма технологий — это набор концептуальных представлений, которые нам дают качественные представления о средствах реализации системы, например, ИС.

ИС, **как система**, — это фактически проект, который описывает как концептуальные представления должны быть реализованы и что из этого получится.

Чтобы наглядно показать различие заявленных понятий, обратим внимание, что парадигмы вычислительных систем, информационного подхода, концепции АСУ и CALS-технологий упорядочены по мере развития средств вычислительной техники и ее программного обеспечения.

Парадигма вычислительных систем основана на развитии языков программирования и алгоритмов быстрого вычисления различных функций. Сами вычисления сначала проводились в пакетном режиме запуска отдельных программ, а затем — в мультипрограммном режиме вычислительной среды различных ОС. При этом информация как таковая не являлась элементом вычислительных технологий. Она присутствовала в головах разработчиков алгоритмов и программ, а также специалистов, интерпретировавших результаты вычислений. ИС, как системы, фактически не существовали, поскольку информация формировалась и накапливалась, по результатам научных или инженерных исследований, вне пространства вычислительных систем и в виде журнальных публикаций, теоретических концепций научных школ и архивов библиотечных учреждений.

Негативные последствия указанного подхода широко известны как проблематика парадигмы «*Программа-массив*», порождающая необходимость написания двух дополнительных программ на каждую пару связываемых прикладных программ (см. пункт 1.1.2).

Парадигма информационного подхода вводит в багаж технологий разработчиков ПО само понятие информации, переводя научные и инженерные знания в форматы данных, которые сохраняются во внешней (долговременной) памяти средств вычислительной техники.

Первоначально информация начала сохраняться в файловых системах ОС, подтверждая базовый тезис систем UNIX: «*Все есть файл*». Формируются первые прообразы ИС в виде комплектов документации на различные языки и системы программирования. Делаются первые попытки автоматизации доступа к документации в виде специальных утилит, например, утилиты ***man*** и ***info***.

Развитие сетевых технологий и специальных средств управления данными (СУБД) выводят информационный подход на новый уровень, ставя задачи проектирования ИС и стимулируя различные мультимедийные разработки в области представления информации. В таких условиях ИС уже рассматривается как обеспечивающая подсистема некоторого набора приложений, реализованным в виде **АРМ** (*Автоматизированного Рабочего Места*), с простейшей распределенной архитектурой показанной на рисунке 1.7.

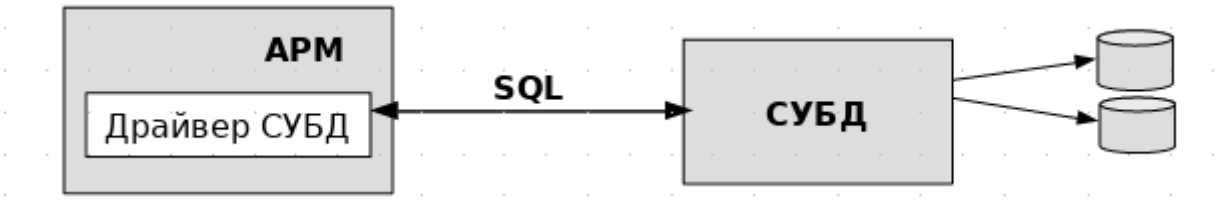


Рисунок 1.7 — ИС обслуживающая отдельное приложение

Примечание — Являясь обслуживающей подсистемой АРМ, ИС вполне удовлетворяется возможностями реляционных СУБД, предоставляя ПО АРМ информацию в виде простейших типов данных.

Дальнейшее публичное развитие парадигмы информационного подхода идёт в направлении разделения инструментальных средств ИС, согласно модели «*Клиент-сервер*», и унификации этих средств в плане отображения, обработки и хранения информации. Примеры такого развития предоставляют web-технологии, где сторона клиента реализуется в виде web-браузеров, а прикладная часть приложений переносится на сторону сервера (web-сервера). И в такой интерпретации ИС рассматриваются как web-сайты.

Парадигма АСУ ограничивает рамки рассматриваемых информационных технологий предметной областью нашей дисциплины, одновременно накладывая на целевое назначение ИС требование обеспечить приемлемый уровень управления производственной структурой уровня предприятия.

В такой проекции проектирование и создание ИС столкнулось с рядом **серьёзных препятствий**:

а) низкий уровень представления и обработки данных реляционными СУБД, ориентированными на информационное обеспечение расчётных прикладных программ;

б) большое количество разнообразных по постановкам и алгоритмам решения задач автоматизации производственно-хозяйственной деятельности предприятия;

в) непомерно большие объёмы данных, необходимые для ручного ввода в АС с целью начала ее работы и поддержки актуальности баз данных.

Результат воздействия указанных препятствий привёл к следующим **последствиям**:

а) достаточно успешной автоматизации задач уровня оперативного исполнения (АСУТП) в силу достаточно простого представления информации в устройствах контроллерного управления;

б) «*островной*» или «*кусочной*» автоматизации ряда общих задач производственно-хозяйственной деятельности: кадровое, финансовое и общее материальное и энергетическое обеспечение предприятия;

в) кризисному положению задач уровня тактического управления, которые во многом уникальны для каждого предприятия и не могут быть решены собственными силами его сотрудников.

Примечание — В рамках парадигмы АСУ, ИС формируются как обслуживающие подсистемы отдельных прикладных систем, которые проектно не связаны между собой в силу их раздельного внешнего проектирования, реализации и сопровождения.

Парадигма CALS-технологий вводит в предметную область деятельности предприятия понятие ЖЦИ — жизненного цикла изделия, которое позволяет рассматривать ИС под новым углом зрения:

а) вводится понятие **ИИС** — Интегрированной Информационной Среды;

б) вводится понятие **ИО** — Информационного Объекта;

в) вводятся и стандартизируются **Правила** информационного взаимодействия и обмена данными между субъектами **ПХД** (субъектами Производственно-Хозяйственной Деятельности).

Примечание — В таких условиях значение ИС выводится на новый уровень значимости. Теперь **ИС** не просто обеспечивает информационную потребность отдельных прикладных программ на уровне простейших типов данных, а обслуживает документооборот предприятия, манипулируя сложными информационными объектами (**ИО**).

Концепция ЖЦИ позволяет рассматривать всю деятельность предприятия как изготовление изделий — *материальных* и/или *информационных*. Изделиями могут быть отдельные ИО, ИС или вся среда ИИС, которые находятся на разных стадиях своего жизненного цикла и по разному взаимодействуют между собой. Что касается самой ИИС, то она может быть в общем случае представлена некоторым множеством *функциональных центров* (***ФЦ***), преобразующих ***входные*** материальные и информационные объекты в ***выходные*** объекты. И, в случае максимального уровня неопределённости, её можно представить рисунком 1.8.

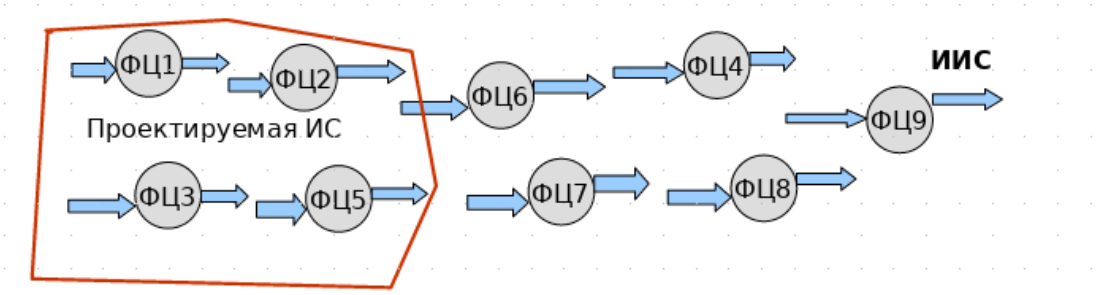


Рисунок 1.8 — Общее представление ИИС как множества функциональных центров предприятия

Таким образом, концепция ЖЦИ CALS-технологий снимает жёсткие требования к информационной архитектуре производственной деятельности предприятия, предоставляя варианты эволюционного развития этой системы в целом. Развитие ИИС может рассматриваться как выделение некоторой совокупности ***Функциональных Центров*** (***ФЦ***), описания их входов и выходов, как отдельных интерфейсов, и последующая модификация вычислительных средств и программного обеспечения для согласования этих интерфейсов.

Примечание — В условиях представленной модели ИИС, задача проектирования ИС может быть поставлена как *деятельность по дополнению выделенного набора ФС необходимым набором ПО*, обеспечивающим согласование всех их интерфейсов.

1.5 Проектирование ИС

ИС — это изделие предприятия или часть его технической производственной базы?

Рассмотрев все четыре исторически сложившихся парадигмы описания ИС, легко прийти к выводу, что наиболее современным взглядом на объект исследования является модель **ИИС** (*Интегрированной Информационной Среды*), которая согласно ГОСТ Р 50.1.031-2001 [11] обеспечивает все процессы ПХД (***Производственно-Хозяйственной Деятельности***) предприятия, сохраняя и предоставляя данные всем хозяйствующим субъектам посредством совокупности распределённых баз данных. Исходя из этой модели, ***предметная область ИС*** — выделенная, согласно целевому заданию на проектирование, совокупность **ФС** (***Функциональных Центров***) ИИС (см. рисунок 1.8), которые должны быть согласованы по входным и выходным интерфейсам.

Конкурирующей, но более адекватной, является парадигма АСУ, которая стандартами ГОСТ серий ***24.xxx***, ***34.xxx*** и ***19.xxx*** создаёт нормативную базу для проектирования ИС.

Познавательная цель данного подраздела — описать общую последовательность проектных действий и решений, необходимых для достижения студентом позитивного результата в процессе проектирования ИС.

Приступая к выполнению производственной практики или ВКР, студент сталкивается *с тремя основными вопросами*:

1) Какую ***тему работы*** выбрать и как правильно поставить задачу на проектирование?
2) Какие ***основные модели*** и технологии следует применить в процессе выполнения работы?
3) Какова ***правильная последовательность работ***, необходимая для выполнения проекта, и чем её обосновать или подтвердить?

Нужные ответы на эти вопросы зависят от специфики самой предметной области и задания полученного от руководителя практики или ВКР. На них студент должен научится отвечать самостоятельно, используя знания полученные во время учёбы в вузе. В данном подразделе изложены только общие рекомендации, которые студент должен использовать, чтобы не допускать грубых ошибок, а более подробный учебный материал по проектированию ИС изложен в последующих разделах учебного пособия.

1.5.1 Общая постановка задачи на проектирование

Студент должен отличать задачу на ***проектирование*** от ***исследовательской*** задачи.

Любая задача всегда содержит некоторую описательную часть, где присутствуют названия вещественных предметов, научных теорий или разделов знаний, а также выделяется целевая составляющая — что должно быть получено в результате решения задачи.

Целевая составляющая исследовательской задачи — изучение и описание чего-либо, что содержит большую неопределённость и заранее неизвестен результат этого изучения.

Например, нужно исследовать некоторый алгоритм или алгоритмы, найти оптимальное решение или возможность достижения чего-либо. Обычно такие задачи связаны с разработкой некоторого инструментария, прямое использование которого не является очевидным.

Целевая составляющая задачи ПИС — создание некоторой системы, включающей аппаратное и программное обеспечение при заданных ограничениях на аппаратную платформу, ОС и другие программные системы.

В общем случае постановок задачи, ПИС достаточно чётко выделяет следующие требования:

а) *автоматизировать* некоторый бизнес-процесс;

б) *обеспечить* сбор, сохранение некоторой информации (данных) и доступ к ней;

в) *осуществить* взаимодействие удалённых между собой систем.

> **Задачи ПИС** всегда имеют ограничивающие условия, которые можно рассматривать как ***внешнее управление***, а главное — они имеют одного или нескольких ***субъектов*** (*исполнителей*), несущих персональную ответственность за работу системы посредством непосредственного управления ею или посредством контроля её функционирования.

Студенту следует сразу научиться рассматривать задачу на ПИС как ***обобщённую функцию***, которую он в последующем может декомпозировать до нужного уровня подробности, при этом выделяя:

а) *входные* информационные объекты (ИО), которые ИС должна импортировать;

б) *выходные* ИО, которые ИС должна создать (экспортировать);

в) *документы*, приказы и распоряжения, ограничивающие выполнение функции ИС;

г) *список сотрудников предприятия*, которые участвуют в выполнении функции ИС или контролируют её выполнение.

> **Примечание** — Студент должен отличать задачу на проектирование ИС (***ПИС***) от задач на проектирование АС и других задач.

ИС является обеспечивающей подсистемой некоторой ***метасистемы***, поэтому задача ПИС формулируется в пределах некоторой АС (или её части) или в пределах САПР (или её части).

Классическим (эталонным) примером постановку задачи на ПИС можно считать формирование ***ТЗ*** на ***АС***, где:

1) в качестве ***базовой основы*** берётся конкретная подсистема АС и изучается её обеспечивающая подсистема ИС;

2) ***дополнительно***, выделяются и описываются расчётные подсистемы, которые не относятся к ИС, но требуются для функционирования подсистемы АС в целом;

3) относительно дополнительных подсистем принимаются ***отдельные решения*** о включении их в задание на проектирование или нет.

Если задача формулируется в пределах некоторой прикладной производственной системы, например, ***ERP*** некоторого производителя, то такие случаи выходят за рамки нашей дисциплины и здесь не рассматриваются.

> Студент должен уметь правильно оценить ***требуемый объем работ*** для поставленной задачи.

Обычно, стремясь создать полезную для производства систему и получить хорошую оценку, студент завышает свои возможности и, в результате, не успевает выполнить все необходимые работы. Нужно всегда помнить, что система всегда кажется проще, чем она есть на самом деле.

Чтобы избежать или уменьшить последствия неправильной оценки требуемого объёма работ на проектирование и реализацию ИС, следует найти и изучить один или несколько ***прототипов*** создаваемой системы.

Нужно помнить, что в проектировании полностью оригинальных систем практически не существует. Всегда существуют некоторые ***аналоги***, которые по ряду причин не могут быть использованы непосредственно. Поэтому, проводя рассуждения относительно существующего, а значит и уже описанного аналога (***прототипа***), студент имеет опорную базу, относительно которой гораздо легче описывать требования к проектируемой ИС и обосновывать доказательную составляющую проектных решений.

Знание и хорошее описание прототипов будущей ИС является показателем профессиональной подготовки студента и оценивается только с положительной стороны.

> **Примечание** — Студент должен видеть конечную цель постановки задачи: **Частное Техническое Задание** (***ЧТЗ***) на проект ИС.

Следует правильно понимать, что ***Задание*** студента на производственную практику или ВКР, написанное максимум на двух страницах текста и содержащее название темы работы, перечень графического материала и других обязательных атрибутов отчёта или пояснительной записки, частью которых оно подразумевается, является организационным документом вуза, обосновывающим легитимность учёбы студента, но не является ЧТЗ на ПИС.

Для многих направлений обучения студентов существующая практика вузов обычно не требует разработки технических заданий на учебные работы, поскольку в большинстве случаев они имеют исследовательский характер, хотя и связаны с некоторыми элементами проектирования. Обычно такое проектирование необходимо для разработки программного обеспечения, но в силу небольшой сложности создаваемых систем или по иным причинам явное создание документа ТЗ (ЧТЗ) не производится.

> **Примечание** — В рамках изучаемой дисциплины, создание документа ЧТЗ на ПИС является обязательным этапом процесса проектирования.

Поскольку ИС является обслуживающей подсистемой некоторой АС, то и написание ЧТЗ на ПИС следует создавать на основе ТЗ на АС.

Поскольку разработка ИС тесно связана с разработкой ПО, то обычно при создании ЧТЗ на ИС руководствуются требованиями ГОСТ серии 19, которые регламентируют разработку программного обеспечения.

Более подробно вопросы создания ЧТЗ на ПИС рассмотрены во втором разделе данного учебного пособия, озаглавленного как «*Концептуальное проектирование ИС*».

Студент должен правильно понимать, что написание ЧТЗ является не просто «данью традиций», требуемых формальными процедурами проектирования, а важнейшим документом, по которому всегда оценивается работа проектировщика.

> **Примечание** — Правильное составление ЧТЗ должно организовать всю последующую работу студента по созданию ИС. Подписанный и утверждённый документ ЧТЗ выступает самым авторитетным аргументом при принятии последующих проектных решений.

1.5.2 Базовые модели проектирования ИС

В общем случае при проектировании ИС могут использоваться все модели, порождённые парадигмами и технологиями, описанными в первых четырёх подразделах данного раздела. Многие из таких моделей являются достаточно специализированными и применяются только в отдельных отраслях производства.

Направления подготовки бакалавров «*Информатика и вычислительная техника*» и «*Прикладная информатика*» традиционно ориентированы на автоматизацию производственных бизнес-процессов, что связано с задачами управления и информационного обеспечения в АС, а не с общими или специфическими задачами САПР. В таком аспекте для нас наиболее близкими по предметной области являются *архитектурные модели предметной области АСУ* и *распределенная архитектурная модель ИИС*, предложенная парадигмой CALS-технологий.

Что касается *базовых моделей проектирования ИС*, то они делятся на две большие группы:

а) **Функциональные модели проектирования**, которые отражают *концептуальную часть любой системы*, выражая её будущие возможности в виде выполняемых ею функций и давая основания для сравнения её с другими аналогичными системами или прототипами;

б) **Объектные модели проектирования**, которые отражают *проект реализации конкретной системы* через архитектуру и взаимосвязь объектов абстрактных или конкретных языков программирования.

Излишне напоминать, что указанные базовые модели находятся в диалектическом противоречии, хотя предназначены для описания одной и той же системы. Они борятся за «*главенство*» в праве влияния на проектные решения, которые принимаются в процессах проектирования ИС.

> Примечание — Согласно общим принципам проектирования первенство влияния на проектные решения принадлежит *функциональным моделям*.

Главенство функциональной модели над объектной основано на принципе ***функциональной целесообразности*** создания системы, которая должна функционировать согласно поставленной цели. Сама цель является по отношению к системе внешней (*декларативной*). Её достижению подчинены все проектные решения.

Само применение функциональных моделей начинается с постановки задачи на проектирование, что должно заканчиваться формированием и последующим утверждением *частного технического задания* (ЧТЗ) на *проектирование информационных систем* (ПИС).

Более подробно эти вопросы изложены во втором разделе данного пособия, но главное, что должен помнить студент — ЧТЗ на ПИС излагается в терминах функциональных моделей, а объектные модели присутствуют в виде ограничений на задачи проектирования.

Завершается построение функциональной модели ИС на *этапе концептуального проектирования*. Этим вопросам посвящён третий раздел данного пособия, где подробно описаны синтаксис и семантика всех трёх основных функциональных моделей: IDEF0, IDEF3 и DFD. Но основной здесь является модель IDEF0, которая полностью задана рекомендациями РС Р 50.1.028-2001 [16]. Эта модель позволяет проводить декомпозицию *организационно-технической структуры организации* (предприятия) в соответствии с уровнем используемых функций, что наглядно показано на рисунке 1.9.

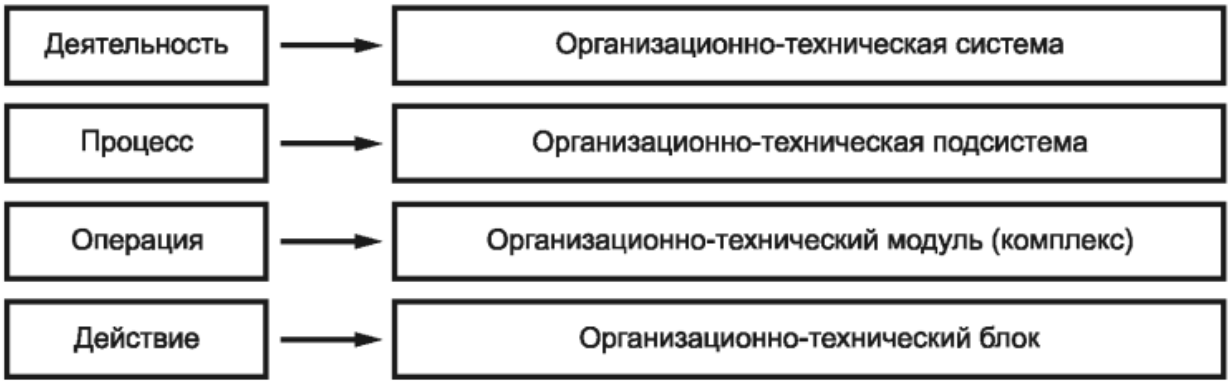


Рисунок 1.9 — Проекция функций модели IDEF0 на организационно-техническую структуру предприятия [16]

Формальная связь элементов рисунка 1.9 определяется стандартом [16], что подтверждается цитатой: «...

Организационно-техническая система — организационная структура, персонал и комплекс технических средств (оборудование), необходимые для осуществления ***деятельности***.

Организационно-техническая подсистема — часть организационно-технической системы, обеспечивающая протекание ***процесса*** (субдеятельности).

Организационно-технический комплекс (модуль) — часть организационно-технической подсистемы, предназначенная для выполнения ***операции*.**

Организационно-технический блок — часть организационно-технического комплекса, обеспечивающая выполнение ***действия***».

Необходимо заучить и уверенно использовать семантику этих функций [16]: «...

Деятельность (синонимы: *дело, бизнес*) — *совокупность процессов*, выполняемых (протекающих) последовательно или/и параллельно, преобразующих множество материальных или/и информационных потоков во множество материальных или/и информационных потоков с другими свойствами. **Деятельность** осуществляется в соответствии с заранее определённой и постоянно корректируемой **целью**, с потреблением финансовых, энергетических, трудовых и материальных **ресурсов**, при выполнении **ограничений** со стороны внешней среды. ...

Процесс (синоним: *бизнес-процесс*) — *совокупность последовательно или/и параллельно выполняемых операций*, преобразующая материальный или/и информационный потоки в соответствующие потоки с другими свойствами. **Процесс** протекает в соответствии с управляющими директивами, вырабатываемыми на основе **целей деятельности**. В ходе процесса потребляются финансовые, энергетические, трудовые и материальные **ресурсы** и выполняются **ограничения** со стороны других процессов и внешней среды. ...

Операция — *совокупность последовательно или/и параллельно выполняемых действий*, преобразующих объекты, входящие в состав материального или/и информационного потока, в соответствующие объекты с другими свойствами. Операция выполняется: а) в соответствии с **директивами**, вырабатываемыми на основе директив, определяющих протекание процесса, в состав которого входит операция; б) с потреблением всех видов необходимых **ресурсов**; в) с соблюдением ограничений со стороны других операций и внешней среды. ...

Действие — *преобразование какого-либо свойства материального или информационного объекта в другое свойство*. Действие выполняется в соответствии с **командой,** являющейся частью **директивы** на выполнение операции, с потреблением необходимых ресурсов и с соблюдением ограничений, налагаемых на осуществление операции ...».

Дополнительно, когда деятельность и процесс являются сложными функциями, допускается использование терминов [16]: «...

Субдеятельность — *совокупность нескольких процессов в составе деятельности*, объединённая некоторой частной целью (являющейся «подцелью» деятельности).

Подпроцесс — *группа операций в составе процесса*, объединённая технологически или организационно».

Теперь рассмотрим объектные модели.

Подчинённое положение объектных моделей не означает их малую значимость. Дело в том, что объектные модели связаны с реализацией ИС, а успешность (*качество*) реализации системы во многом зависит от уровня развития соответствующих технологий. В этом отношении сами объектные модели являются зависимыми от многих условий и могут существенно меняться по мере технологического развития средств вычислительной техники и инструментального обеспечения процессов создания систем. Этот факт наглядно показан в обзорах существующих парадигм (см. подразделы 1.1 — 1.4 настоящего пособия). Более подробно эта часть учебного материала изложена в четвёртом разделе, где наиболее перспективной архитектурой, в плане реализации ИС, следует считать модель распределённых сервис-ориентированных систем.

Не вдаваясь в детали реализации конкретных моделей, будем рассматривать *базовую объектную модель проектирования* в виде классической трёхзвенной архитектуры «*Клиент-сервер*», показанной на рисунке 1.10.

Рисунок 1.10 — Базовая объектная трёхзвенная архитектура «Клиент-сервер»

Бизнес-логика — *центральный метаобъект ИС*, уже реализованный как «Сервер приложений» и содержащий инструментальные средства для размещения «Сервисов» (Объектов-приложений), подразумеваемых как отдельные ***ФС*** (Функциональные центры) ***ИИС*** (Интегрированной Информационной Среды), что ранее уже показано на рисунке 1.8.

Представление — *метаобъект создания, модификации и потребления информации*, уже реализованный как «Клиент» (Набор клиентских приложений) и способный взаимодействовать с метаобъектом «Бизнес-логика» посредством сети.

Данные — *метаобъект объектной трёхзвенной архитектуры*, уже реализованный как СУБД и способный управлять по классической схеме необходимым набором баз данных.

> **Примечание** — **Главное требование** к базовой объектной модели — *обеспечить реализацию нисходящего процесса проектирования*.

Детали применения конкретной объектной трёхзвенной архитектуры «*Клиент-сервер*» рассмотрим позже в подразделе 1.6.

1.5.3 Этапы проектирования ИС

Проектирование ИС всегда связано с использованием уже готового программного обеспечения (ПО), а также с созданием, по мере необходимости, нового ПО.

Исходя из этого тезиса, студент должен иметь общее представление о ***Единой Системе Программной Документации*** (*ЕСПД*), содержащей [17]: «... комплекс государственных стандартов, устанавливающих взаимоувязанные правила разработки, оформления и обращения программ и программной документации»:

а) ГОСТ 19.001-77 — Общие положения [17];

б) ГОСТ 19.003-80 — Схемы алгоритмов и программ. Обозначение условные графические;

в) ГОСТ 19.101-77 — Виды программ и программных документов;

г) ГОСТ 19.102-77 — Стадии разработки [18];

д) ГОСТ 19.103-77 — Обозначение программ и программных документов;

е) ГОСТ 19.104-78 — Основные надписи;

ж) ГОСТ 19.105-78 — Общие требования к программным документам;

з) ГОСТ 19.106-78 — Требования к программным документам, выполненным печатным способом;

и) ГОСТ 19.201-78 — Техническое задание. Требования к содержанию и оформлению;

к) ГОСТ 19.202-78 — Спецификация. Требования к содержанию и оформлению;

л) ГОСТ 19.402-78 — Описание программы;

м) ГОСТ 19.404-79 — Пояснительная записка. Требования к содержанию и оформлению;

н) ГОСТ 19.502-78 — Описание применения. Требования к содержанию и оформлению;

о) ГОСТ 19.503-79 — Руководство системного программиста. Требования к содержанию и оформлению;

п) ГОСТ 19.504-79 — Руководство программиста. Требования к содержанию и оформлению.

Примечание — На практике студент также должен руководствоваться требованиями своего ***Задания*** и корпоративными ограничениями на проектирование.

Согласно требованиям к ТЗ на АС (см. ГОСТ 34.602-89 [9], раздел «1. ОБЩИЕ ПОЛОЖЕНИЯ»): «... Дополнительно могут быть разработаны ТЗ на части АС:

а) на подсистемы АС, комплексы задач АС и т. п. в соответствии с требованиями настоящего стандарта;

б) на комплектующие средства технического обеспечения и программно-технические комплексы в соответствии со стандартами ЕСКД и СРПП;

в) на программные средства в соответствии со стандартами ЕСПД;

г) на информационные изделия в соответствии с ГОСТ 19.201 и НТД, действующей в ведомстве заказчика АС».

Таким образом, согласно ГОСТ 19.102-77 [18], стадии и этапы работ по созданию ПО ИС представляются таблицей 1.4.

Таблица 1.4 — Стадии и этапы работ разработки ПО ИС [18]

Стадии разработки	***Этапы работ***
1. Техническое задание	Обоснование необходимости разработки программы. Научно-исследовательские работы. Разработка и утверждение технического задания.
2. Эскизный проект	Разработка эскизного проекта. Утверждение эскизного проекта.
3. Технический проект	Разработка технического проекта. Утверждение технического проекта.
4. Рабочий проект	Разработка программ. Разработка программной документации. Испытание программ.
5. Внедрение	Подготовка и передача программы.

В технически обоснованных случаях, что должно быть отражено в ЧТЗ на ПО ИС, допускается:

а) ***исключать*** вторую и третью стадии;

б) ***объединять*** и ***исключать*** этапы работ, а также вводить другие этапы работ по согласованию с заказчиком.

Конкретная необходимость использования ГОСТ серии ***19.xxx*** должна определяться дополнительно.

1.6 Учебная инфраструктура проектировщика ИС

Излишне напоминать, что любая работа, в том числе и учебная, всегда осуществляется в ограниченные сроки, по истечении которых студент должен представить результат этой работы на оценивание преподавателю или экзаменационной комиссии. В этих условиях поиск, изучение и практическая апробация необходимых инструментальных средств может занять значительное время и негативно сказаться на результатах работы.

Учебная цель данного подраздела — краткое описание конкретного состава рабочей среды и программных инструментальных средств, образующих учебную инфраструктуру проектировщика ИС и используемых для демонстрации учебных примеров в данном учебном пособии.

> Примечание — Для проведения проектных работ студенту необходима конкретная учебная программно-аппаратная инфраструктура, где такие работы могли бы быть выполнены.

Для обеспечения учебного процесса используется инфраструктурная среда ОС УПК АСУ [19], с которой студент хорошо знаком по результатам обучения дисциплине «*Операционные системы*» (*ОС*).

Непосредственно для изучаемой дисциплины подготовлена специальная рабочая область пользователя ***upk***, которая предоставляется студенту индивидуально, а также используется и сохраняется им в течение всего процесса обучения.

> Примечание — Студенту необходим учебный инструментальный прототип будущей инфраструктуры размещения проектируемой ИС, которая затем могла бы быть перемещена в производственную среду предприятия (организации).

Согласно общим современным представлениям проектируемая ИС должна представлять собой трёхзвенную распределенную сервис-ориентированную систему (см. рисунок 1.10). Учитывая, что важнейшей функцией ИС является наглядное и удобное для восприятия человеком представление информации, учебный процесс данной дисциплины ориентируется прежде всего на хорошо развитый аппарат Web-сервисов. Этот аппарат хорошо зарекомендовал себя на практике и успешно используется в учебном процессе вуза, например, портал ТУСУР.

Среди открытых и свободно доступных для скачивания, установки и последующего использования инструментальных средств является продукция организации-фонда ***Apache Software Foundation*** (ASF) [20], известного десятками своих программных изделий. Наиболее популярным из них является контейнер сервлетов ***Apache Tomcat*** [21], способный реализовывать как простейшие Web-сервисы, так и выполнять функции web-сервера.

Для целей обучения и основы прототипа проектируемой ИС будут использоваться два инструментальных средства ASF:

а) *сервер приложений **Apache TomEE*** [22], обеспечивающий распределённое размещение приложений будущей ИС;

б) *СУБД **Apache Derby*** [23], обеспечивающая приложения ИС как встроенным (*embedded*) вариантом использования, так и вариантом использования доступным через сеть.

Все указанные инструменты реализованы на основе языка Java, что обеспечивает им высокую степень переносимости между различными ОС и богатым арсеналом имеющихся приложений на языке Java.

Примечание — Студенту необходимы учебные инструментальные средства разработки ПО, с помощью которых он мог бы проводить как реализацию и тестирование ИС в целом, так и реализацию и тестирование её отдельных компонентов.

Среди бесплатных и широко доступных инструментальных средств разработки ПО, к тому же сами реализованные на языке Java, являются продукты некоммерческой организации ***Eclipse Foundation*** [24]. На момент написания данного учебного пособия эта организация предлагает тринадцать дистрибутивов различного назначения. В частности, в учебной инфраструктуре ОС УПК АСУ [19] установлен дистрибутив ***Eclipse IDE for C/C++ Developers***, используемый при изучении дисциплины «*Операционные системы*».

Непосредственно для проектных задач изучаемой дисциплины установлен и используется дистрибутив ***Eclipse IDE for Enterprise Java Developers*** [25], предназначенный для разработки на языке Java приложений уровня предприятия и хорошо интегрируемый с сервером приложений ***Apache TomEE***.

В качестве справочника или учебника по языку Java рекомендуется использовать достаточно полное руководство [26].

Примечание — Студенту необходимы учебные инструментальные средства концептуального проектирования ИС и подготовки проектной документации.

Современное проектирование ИС ориентировано на широкое использование методологии SADT, хорошо описанной в разных источниках, например, [5]. Эта методология требует построения различных диаграмм, необходимых как на этапах концептуального проектирования, так и на этапах реализации ИС.

Непосредственно в данной учебной дисциплине используются только свободно доступные продукты:

а) **Ramus Educational**, разработанный ***Ramus Soft Group*** [27] для целей построения диаграмм моделей IDEF0 и DFD;

б) **Eclipse Modeling Tools** [28], предназначенный для построения приложений на основе графических моделей.

Дополнительно для разных целей используются приложения стационарно установленного в ОС УПК АСУ ПО LibreOffice [29], продукты которого придерживаются открытого формата OpenDocument Format [30], а также широко известный браузер Mozilla Firefox [31].

Примечание — Проектировщик ИС должен хорошо знать состав, расположение и основные шаблоны применения своего инструментария.

Рассмотрим состав и варианты технологии применения инструментальных средств в отдельных пунктах данного подраздела.

1.6.1 Состав и размещение инструментальных средств

Учебная рабочая область студента имеет размер 400 МБ и находится в файле ***pis-home.ext4fs***, поэтому необходимо беречь его свободное пространство.

В целом, описание структуры и местоположение файлов ОС УПК АСУ достаточно полно описано в учебно-методическом пособии [16] и обычно рассматривается в материале первой лабораторной работы по дисциплине ОС, поэтому она здесь изучаться не будет. Далее мы считаем, что студент запустил учебную систему, подключил к ней личную рабочую область и прошёл авторизацию пользователем ***upk***. В такой ситуации, размещение всех про-

граммных средств и их адресация видны и рассматриваются в общем древовидном адресном пространстве файловой системы. В таком виде и рассматривается размещение всех используемых в данном учебном пособии объектов.

Как проектировщик, студент должен хорошо представлять себе базовые адресные пространства, обозначенные следующими каталогами общими для многих UNIX-подобных ОС:

а) **/etc** — общий каталог размещения конфигурационных файлов;

б) **/opt** — общий каталог размещения дистрибутивов дополнительных инструментальных средств;

в) **/usr/bin** — общий каталог размещения запускаемых файлов и утилит ОС;

г) **/usr/lib/jvm** — общий каталог размещения дистрибутивов языка Java;

д) **/home/upk** — домашний каталог пользователя ***upk***, к которому подключена рабочая область студента.

Примечание — **Базовой областью** подключения необходимых дистрибутивов инструментальных средств проектировщика ИС является каталог ***/opt***, содержимое которого представлено таблицей 1.5.

Таблица 1.5 — Видимость ПО дистрибутивов инструментальных средств

Каталог	***Содержимое каталога***
/opt/derby	Дистрибутив Apache Derby
/opt/eclipseEE	Дистрибутив Eclipse Enterprise Edition
/opt/eclipseMT	Дистрибутив Eclipse Modeling Tools
/opt/tomee	Дистрибутив Apache TomEE

Непосредственное подключение перечисленных дистрибутивов к каталогам таблицы 1.5 — темы отдельных лабораторных работ. Главное, чтобы студент знал о них и мог использовать в настройках своих проектов.

Примечание — **Рабочей областью** использования учебного ПО проектировщика ИС является каталог ***/home/upk***, в котором выделены специальные области показанные в таблице 1.6.

Таблица 1.6 — Рабочие области ПО инструментальных средств пользователя upk

Каталог	***Содержимое каталога***
/home/upk/derby	Рабочая область СУБД Apache Derby
/home/upk/eclipseEE	Рабочая область среды разработки Eclipse Enterprise Edition
/home/upk/eclipseMT	Рабочая область дополнительной среды разработки Eclipse Modeling Tools
/home/upk/tomee	Рабочая область сервера приложений Apache TomEE

Существенное различие между базовой и рабочей областями следующее:

а) ***базовая область*** — образует видимую область дистрибутива установленного и обычно содержит достаточно большой объем ПО и может быть использована несколькими рабочими областями;

б) ***рабочая область*** — часть установленного дистрибутива и обычно содержит файлы конфигурации и результаты применения ПО дистрибутива, например, ПО СУБД

Apache Derby содержит в рабочей области сценарии запуска и остановки СУБД, а также каталоги размещения конкретных баз данных.

Проектная инфраструктура рабочей области пользователя ***upk*** должна иметь набор каталогов, общий список и назначение которых приведён в таблице 1.7.

Таблица 1.7 — Список каталогов проектной инфраструктуры ИС

Каталог	***Содержимое каталога***
/home/upk/bin	Каталог, в который разработчик ИС помещает разработанные им запускаемые программы.
/home/upk/lib	Каталог для размещения разработанных библиотек.
/home/upk/src	Каталог с исходными текстами дополнительно разработанных программ.
/home/upk/ramus-educational-1.1.1	Установленный дистрибутив и рабочая область системы Ramus Educational
/home/upk/Документы	Каталог для размещения учебных материалов по изучаемой дисциплине и другой документации проектировщика ИС.
/home/upk/Загрузки	Каталог размещения несистемно скачиваемых файлов.
/home/upk/Рабочий стол	Каталог размещения информации на рабочем столе пользователя.

Безусловно, проектная инфраструктура включает в себя и другие элементы размещения источников информации и конфигурационных файлов, отражающих особенности построения используемой ОС.

Примечание — **Проектирование** — это всегда достижение и строгое описание конкретной упорядоченности в размещении информации и данных.

Следует заметить, что описанное размещение учебной проектной инфраструктуры разработчика ИС не является строго обязательным. Многие её элементы можно расположить по другому, но они должны быть описаны и понятны студенту, поскольку используются как в самих процессах проектирования ИС, так и в оформлении соответствующей проектной документации.

1.6.2 Технология применения инструментальных средств проектирования

Примечание — Инструментальные средства разработчика ИС должны обеспечивать создание базовой трёхзвенной архитектуры «Клиент-сервер», которая представлена рисунком 1.10.

Хотя используемые инструменты проектирования могут учитывать особенности и специальные требования предметной области ИИС предприятия, они должны обеспечивать реализацию базовой трёхзвенной архитектуры, включающей представление информации, бизнес-логику и сохранение данных.

Инструментальная среда разработки ***Eclipse IDE for Enterprise Java Developers*** [25] имеет шаблон проектирования «***Dynamic Web Project***» интегрируемый с сервером приложений ***Apache TomEE*** и ***Apache Derby***. Такая конфигурация инструментальных средств позволяет сразу реализовывать простейшую трёхзвенную архитектуру Web-сервисов, схематично показанную на рисунке 1.11.

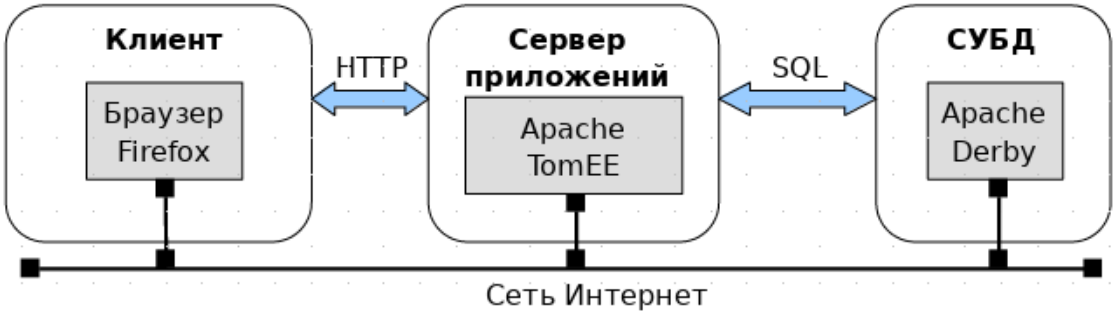


Рисунок 1.11 — Простейшая трёхзвенная архитектура Web-сервиса

Представленная архитектура Web-сервиса легко реализуется средствами учебной проектной инфраструктуры ОС УПК АСУ даже без подключения компьютера к внешней сети Интернет и рассматривается как базовая в учебном процессе данной дисциплины. Более того, используя хорошо описанные настройки, можно на одном компьютере установить несколько серверов приложений Apache TomEE и работать с ними.

Важнейшая особенность использования Apache TomEE состоит в том, что он:

а) сам имеет *большое количество реализованных технологий* создания распределенных приложений;

б) позволяет профессионально обеспечивать *мультиплексирование запросов* к нему (см. рисунок 1.12);

в) позволяет профессионально обеспечивать *демультиплексирование запросов* к различным СУБД (см. рисунок 1.13).

В среде Web-сервисов, реализация приведённых выше архитектур не вызывает затруднений, поскольку обеспечивается URL/URN-адресацией.

Примечание — Современные инструментальные средства разработчика ИС должны обеспечивать проектирование и создание распределенных систем.

Фактически архитектуры, показанные на рисунках 1.11 — 1.13, уже являются простейшими распределёнными системами.

В тех случаях, когда необходимо создать систему, распределённую на множестве функциональных центров (ФЦ) ИИС (см. рисунок 1.8), то на сервере приложений Apache TomEE следует:

а) ***реализовать*** множество независимых приложений, каждое из которых соответствует одному ФЦ;

б) ***связать*** независимые ФЦ через их интерфейсы, согласно поставленной задаче.

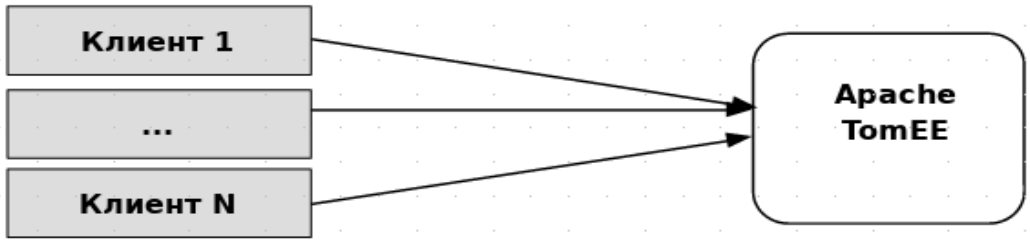


Рисунок 1.12 — Мультиплексирование запросов к серверу

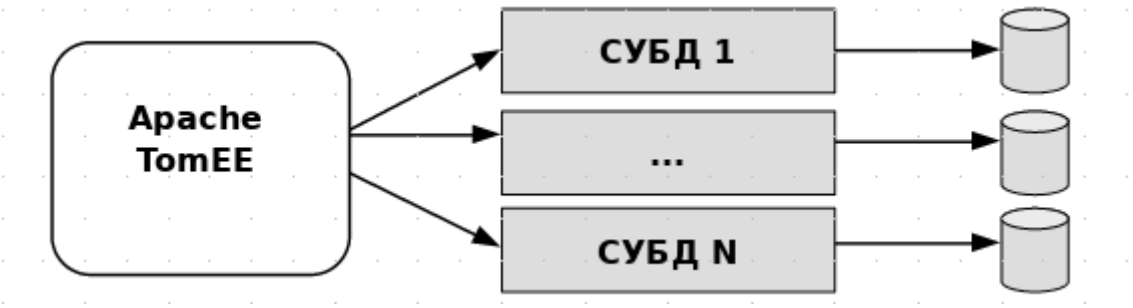


Рисунок 1.13 — Демультиплексирование запросов к СУБД

Двухзвенная часть учебной реализации такой распределенной системы показана на рисунке 1.14.

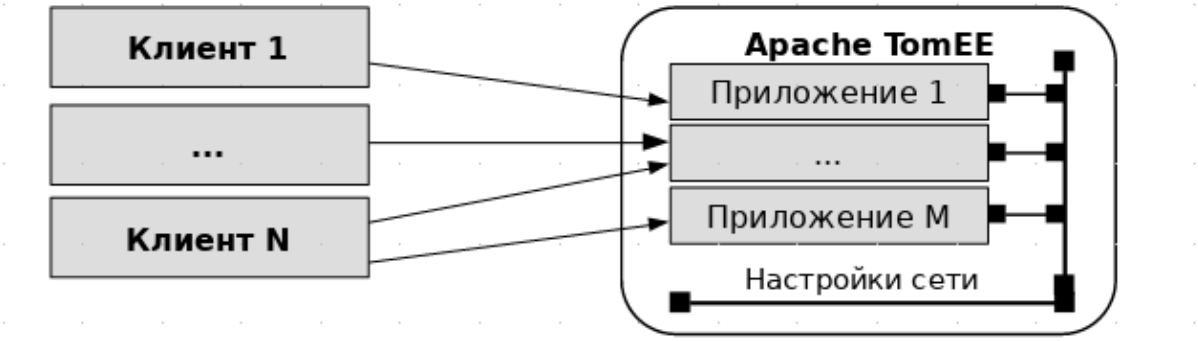


Рисунок 1.14 — Двухзвенная часть архитектуры распределённой системы

Естественно, что при необходимости такая двухзвенная система может быть дополнена требуемым доступом к СУБД.

> Примечание — Современные инструментальные средства разработчика ИС должны обеспечивать размещение компонент созданной системы в распределённой среде предметной области ИИС.

Поскольку доступ к Web-сервисам обеспечивается URL/URN-адресацией, то размещение отдельных приложений по компьютерам предметной области ИИС не вызывает затруднений. Здесь мы дополнительно учитываем возможность размещения на каждой машине сервера приложений Apache TomEE.

Что касается СУБД Apache Derby, то его размещение также не вызывает затруднений, поскольку он реализован на языке Java. Кроме того, сервер приложений способен работать практически с любыми типами СУБД.

> Примечание — **Современные распределённые системы** должны обеспечивать необходимый уровень безопасности работы с ними.

Сервер приложений Apache TomEE обеспечивает стандартный доступ к нему по протоколу HTTP. Дополнительными настройками можно обеспечить доступ по защищённому протоколу HTTPS. Имеются также инструменты для авторизации подключаемых пользователей и систем.

На этой позитивной ноте мы заканчиваем изучение теоретической части дисциплины и переходим к более практическим вопросам проектирования.

Вопросы для самопроверки

1. Какой перечень парадигм положен в основу изучаемой дисциплины? Перечислите их.
2. Какие две группы систем предлагает классификация СОД, например, в трактовке А.М. Ларионова?
3. Что такое — парадигма «Программа-массив» и в чём состоят её недостатки?
4. Как можно оценить сложность систем, придерживающихся концепции «Программа-массив»?
5. В чём состоит суть парадигмы информационного подхода в проектировании ИС?
6. Назовите основные части системного проектирования предметной области?
7. Что такое — методика SADT и какой набор стандартов был разработан специально для неё?
8. Назовите три наиболее известных стандарта, разработанных для проектирования по направлению ICAM Definitions?
9. Каким образом язык UML связан с информационным подходом проектирования ИС?
10. Каким образом технология СУБД связана с информационным подходом проектирования ИС?
11. Что такое — модель АСУ и на какие стандарты она опирается?
12. Что такое — ИС в пределах модели АС?
13. Перечислите стадии создания АС?
14. В чём состоит особенность парадигмы CALS-технологий?
15. Что такое — ИИС и где она используется?
16. На какие стандарты опирается понятие ЖЦИ?
17. Что такое — САПР и как она классифицируется?
18. Назовите базовые модели проектирования ИС.
19. Изобразите и поясните базовую трёхзвенную архитектуру распределённой системы «Клиент-сервер».
20. На основе какого сервера предлагается разрабатывать ИС?

2 ФОРМИРОВАНИЕ ТРЕБОВАНИЙ К ИС

ИС или АИС — это АС.

Начиная изучение процессов проектирования, необходимо более точно определиться с предметной областью дисциплины, а также с объектом и предметом проектирования.

Прежде всего отметим, что термин АИС был введён достаточно давно и подчёркивал использование ВМ (ЭВМ) в системах обработки информации. Такая же ситуация существует и со множеством других терминов, например, термин «*Сетевая операционная система*» подчёркивал, что дистрибутив ОС содержит сетевое программное обеспечение «*из коробки*». Сейчас подобные различия считаются неуместными и отражают только историческую хронику развития теории и практики использования ИС. С другой стороны, утверждение, что ИС — это АС, означает главное ограничение предметной области изучаемой дисциплины теоретическими представлениями парадигмы «*Автоматизированные системы*» и нормативными требованиями, основанными на стандартах ***ГОСТ серии 34.xxx***. А чтобы обосновать это утверждение, сравним ограничения на предметную область АС с аналогичными ограничениями на предметные области *концепций вычислительных систем*, *информационного подхода* и *CALS-технологий*.

Предметная область вычислительных систем включает *алгоритмическую постановку расчётных задач*, написание и тестирование программного обеспечения. В этом плане, она ограничена нормами стандартов ***ГОСТ серии 19.xxx***, поскольку исходные тексты программ являются частью такой системы как изделия. Что же касается предметной области АСУ, то наличие алгоритмов и исходных текстов программ в ней не отрицается, но рассматривается как возможная дополнительная часть соответствующего вида обеспечивающих подсистем.

Предметная область информационного подхода — *достаточно широка*, чтобы её можно было ограничить отдельной серией стандартов или характеризовать отдельной системой. В историческом плане к этой области относили системы построенные на основе первых прототипов СУБД. Их часто также называли информационными системами (ИС). Со временем этот взгляд потерял свою классификационную значимость, поскольку было разработано множество разных типов СУБД с разной степенью масштабируемости. Их стали включать в системы разной прикладной направленности, а многие реализации СУБД перешли в разряд обычного инструментального ПО.

Предметная область CALS-технологий включает в объект исследования *процессы связанные с жизненным циклом изделий* (ЖЦИ) или с *информационной поддержкой изделий* (ИПИ). В такой трактовке эта предметная область тесно связана с САПР, которые естественно сами являются разновидностью АС, и другими аналогичными системами достаточно подробно описанными в подразделе 1.4 данного учебного пособия. Сама эта предметная область достаточно полно описана и нормирована множеством международных стандартов, часть из которых переведена на русский язык, а также рядом Госстандартов РФ, имеющих статус Рекомендаций по стандартизации (РС). Имеется также множество моделей и теоретических представлений, например, ИИС (Интегрированная информационная среда), анализ и реинжиниринг бизнес-процессов, безбумажный обмен данными с использованием электронной цифровой подписи (ЭЦП) и другие, которые на прямую могут быть использованы в проектировании АС или ИС. Но полностью прямое использование стандартов CALS-технологий сталкивается со спецификой самого содержания предметной области, которое в первую очередь ориентировано на *специалистов в области конструкторского проектирования* и мало подходит для специалистов направлений «Информатика и вычислительная техника» и «Прикладная информатика».

Таким образом, учитывая направленность подготовки студентов, ИС, в пределах изучаемой дисциплины, рассматривается как разновидность АС, а её проектирование опирается на теоретическую часть и нормативные требования, изложенные в подразделе 1.3 предыдущего раздела.

Согласно стадиям и этапам создания АС, определённых ГОСТ 34.601-90 [8] (см. также пункт 1.3.3, таблица 1.2), учебная тема данного раздела соответствует первой стадии создания АС «*Формирование требований к АС*» и, применительно к тематике дисциплины, предполагает следующую последовательность этапов работ:

1. ***Обследование*** объекта и обоснование необходимости создания ИС.
2. ***Формирование*** требований пользователя к ИС.
3. ***Оформление*** отчёта о выполненной работе и заявки на разработку ИС (тактико-технического задания).

Примечание — Приступая к учебно-производственной практике, студент должен адекватно планировать объем и время выполняемых им работ.

Хотя последовательность выполнения первого этапа работ по проектированию ИС сформулирована достаточно чётко, всегда возникают вопросы:

а) С чего конкретно начинать работы?

б) Какой объем работ необходимо выполнить?

в) Что конкретно необходимо написать в отчёт?

Естественно, что качественные ответы на поставленные вопросы зависят от конкретных обстоятельств, в которых оказался студент. Тем не менее, имеются достаточно общие ориентиры, присутствующие во многих, если не во всех ситуациях.

Проектирование всегда связано с решением конкретной задачи.

В общем случае задачи на проектирование ИС формулируются как набор претензий в недостаточности информационных ресурсов некоторой деятельности предприятия или как претензии к уровню автоматизации некоторых процессов обработки информации, участвующих в такой деятельности.

В тех случаях, *когда задачи на создание ИС описаны в ТЗ на АС*, то такой источник и является основным документом, который следует изучать студенту и на основе которого можно успешно выполнить все этапы первой стадии проектирования ИС.

В тех случаях, *когда задача на проектирование ИС формулируется как учебное задание на производственную практику* и не привязана к конкретному ТЗ на АС, а отражает только некоторую перспективную идею автоматизации, необходимо начинать с подробного описания самой задачи. Такой подход принят в данном учебном пособии и изложен далее.

Проектирование — это всегда сложный целенаправленный процесс анализа предметной области объекта исследования.

Рассматривая концепцию жизненного цикла изделия , мы изучили три модельных подхода проектной деятельности: *каскадную*, *итерационную* и *спиралевидную* модели.

Обратим внимание, что классической моделью проектирования, требуемой ГОСТ 34.601-90 [8], является ***каскадная модель***, которая предполагает строгое последовательное выполнение всех стадий. Причём согласно таблице 1.3 (см. пункт 1.3.3, стр. 33), формирование требований к ИС относится к группе стадий «*До ТЗ*» и, если указанные работы будут вы-

полнены некачественно, то ТЗ на ИС скорее всего будет не согласовано и не подписано. Как следствие, последующие работы по проектированию и реализации ИС выполняться не будут.

Таким образом, ***целевое назначение*** этапа работ «*Формирование требований к ИС*» — информационное обеспечение стадий «*Разработка концепции ИС*» и «*Техническое задание*».

Изучив и описав в отчёте задачу так, чтобы она была понятна после прочтения текста и не вызывала бурных противоречивых дискуссий, студент должен:

1) *описать* ***организационную структуру предприятия***, в виде иерархии его подразделений, обязательно охватив те подразделения, с которыми ИС предположительно будет связана;
2) *выделить* ***границы объекта проектирования***, обязательно найдя и описав как минимум один прототип будущей ИС;
3) *выделить и описать* ***главный бизнес-процесс***, который будет обслуживать проектируемая ИС;
4) *сформировать и отразить в отчёте* ***общую доказательную базу***, которая будет обоснованием следующей стадии разработки концептуального проекта ИС.

Примечание — **Степень проработки** каждого из перечисленных пунктов работ должен быть минимально достаточна для позитивного вывода о необходимости продолжения проектных работ.

Следует избежать распространённых ошибок начинающего проектировщика:

а) стремление реализовать сразу какие-то части будущей ИС с целью демонстрации её будущей перспективности;

б) увлечение отдельной частью работ, например, описанием организационной структуры предприятия в ущерб другим работам, например, формированию и описанию в отчёте общей доказательной базы необходимости создания ИС.

Студент должен помнить, что в процессах проектирования всегда присутствует большая доля неопределённости. В таких условиях излишне подробное описание какой-то части выполненных работ вызывает больше вопросов, чем убеждает читателя в правильности приведённых в отчёте оценок.

Конкретные детали реализации первого этапа проектирования ИС раскроем в следующих подразделах данного пособия.

1) 2.1 — Описание учебной задачи.
2) 2.2 — Организационная структура предприятия.
3) 2.3 — Определение требований к объекту проектирования.
4) 2.4 — Требования к бизнес-моделям объекта проектирования.
5) 2.5 — Оформление отчёта по первой стадии проектирования ИС.

2.1 Описание учебной задачи

Цель данного подраздела — описание организационных аспектов прохождения студентом *начального этапа учебного процесса вуза*, называемого производственной практикой, а точнее — «Преддипломная практика».

В этом аспекте студенту необходимо:

а) *определиться* с местом (организацией) прохождения практики и конкретными сроками её прохождения (ориентировочно четыре недели);

б) *выбрать* научного руководителя и тему выполняемого проекта;

в) *сформулировать* название темы учебного проекта;

г) *зарегистрировать* название темы в управляющей структуре кафедры, с последующим выходом приказа вуза.

Большинство указанных действий выполняется под руководством специального преподавателя — «Руководителя практики» и согласно требованиям соответствующих учебно-методических пособий. Мы же, исключительно для учебного процесса данной дисциплины, конкретизируем: *общие организационные условия* прохождения производственной практики и *список действующих лиц и исполнителей* решаемой учебной задачи.

Общие организационные условия прохождения производственной практики:

а) *предприятие* прохождения производственной практики студента: г. Томск, ТУСУР;

б) *подразделение* выполнения учебных работ: кафедра АСУ;

в) *научный руководитель* студента: автор данного пособия.

Список действующих лиц и исполнителей решаемой учебной задачи:

а) **Студент** — Петров Иван Васильевич, абстрактный студент бакалавриата группы 447-1, проходящий производственную практику, выполняющий учебное задание и уполномоченный на согласование проектных документов *от исполнителя*;

б) **Научный руководитель** — Резник Виталий Григорьевич, кандидат технических наук, доцент кафедры АСУ ТУСУР, руководящий выполнением учебного задания студента и уполномоченный на утверждение проектных документов *от исполнителя*;

в) **Руководитель практики** — Григорьева Марина Викторовна, кандидат технических наук, доцент кафедры АСУ ТУСУР, руководитель производственной практики от университета и уполномоченный на согласование проектных документов *от заказчика*;

г) **Руководитель подразделения** — Романенко Владимир Васильевич, кандидат технических наук, заведующий кафедрой АСУ ТУСУР, представитель предприятия-заказчика и уполномоченный на утверждение проектных документов (*от заказчика*).

> Примечание — Проведённая конкретизация организационных условий прохождения производственной практики позволяет нам приводить конкретные примеры проектных решений, которые могут быть реально использованы студентом.

Любое описание задачи на проектирование ИС всегда проходит три состояния или *стадии конкретизации*, которые мы рассмотрим в последующих трёх пунктах:

1) *описание идеи* (учебной задачи) проектируемой ИС;

2) *согласование и регистрация задания* на обучающей кафедре;

3) *подробное описание задания*, доступное для его изложения другим лицам.

2.1.1 Описание идеи учебной задачи

Успех проектирования во многом *определяется хорошим знанием предметной области*, для которой будет выполняться проект.

Автор данного учебного пособия является преподавателем кафедры АСУ ТУСУР. Естественно, что он достаточно хорошо знает предприятие (организацию, вуз), которое занимается процессами обучения студентов. Аналогично, каждый студент уже готов для правильного восприятия объекта исследования, который включает все эти процессы.

Любой преподаватель вуза, ведущий лабораторные работы по различным дисциплинам, сталкивается со следующими **проблемами**:

а) ***размещение*** больших групп (подгрупп) студентов по рабочим местам учебных классов кафедры;

б) ***контроль*** присутствия студентов на занятиях и их надлежащее участие в учебной работе;

в) ***обеспечение*** оперативной обратной связи со студентами и оказание им своевременной помощи;

г) ***проведение*** промежуточной аттестации результатов обучения.

Перечисленные проблемы не являются новыми и каждый преподаватель справляется с ними, используя удобные для него методические приёмы и навыки. Тем не менее, плановое увеличение нагрузки, реальное увеличение численности групп (подгрупп) студентов, а также пониженная социальная ответственность студентов к процессам обучения, делают работу преподавателя все более сложной и, в результате, снижают качество обучения.

В сложившихся условиях, *актуальной задачей* является автоматизация деятельности преподавателей кафедры, в планах:

а) ***оперативного контроля*** присутствия и активности студентов на обязательных лабораторных занятиях конкретных дисциплин;

б) ***асинхронной связи*** студента и преподавателя, обеспечивающей отложенный сбор и обобщение проблемных элементов процесса обучения;

в) ***обобщающей отчётности*** результатов обучения по отдельным группам (подгруппам) и соответствующим дисциплинам, облегчающей индивидуальную оценку обучающихся.

Идейная часть заявленной задачи автоматизации состоит в создании «*Индивидуального Электронного Журнала Преподавателя*», который бы размещался у него на личном компьютере и обеспечивал указанную выше функциональность в пределах локальной сети кафедры.

Следует отметить, что предприятие (вуз) ТУСУР имеет централизованный Web-портал, обеспечивающий преподавателей и студентов вуза достаточно мощными функциональными средствами сервиса «*Электронный журнал*» и системы *Moodle*. Эти средства ориентированы на одно из перспективных направлений деятельности вуза — «*Режим удалённого обучения*». Естественно, что указанные средства могут служить в качестве прототипа идейной части предлагаемой задачи.

Студенту группы 447-1, Петрову И.В., предлагается реализация прототипа индивидуального электронного журнала преподавателя (ИЭЖП) в качестве задания на производственную практику «Преддипломная практика» на базе инфраструктуры кафедры АСУ предприятия (вуза) ТУСУР.

В качестве рекомендации по выполнению задания производственной практики, студенту Петрову И.В. предлагается:

а) в качестве *основного подхода реализации* задания использовать технологию Web-сервисов, широко используемую в современных разработках информационных систем;

б) в качестве *готовых к использованию инструментальных средств* выбирать только некоммерческие продукты, например, сервер приложений Apache TomEE, СУБД Apache Derby, среду разработки Eclipse EE и браузер Mozilla Firefox;

в) в качестве *основного языка разработки приложений* рекомендуется выбрать объектно-ориентированный язык программирования Java.

2.1.2 Задание на учебную практику

Примечание — Поскольку Петров И.В. является абстрактным студентом, то у него нет никаких возможностей отказаться от предложенного задания и будьте уверены, что он будет выполнять все рекомендации обозначенные автором данного учебного пособия.

Предварительно осмыслив и согласившись выполнять задание, студент обязан зарегистрировать его у руководителя практики от университета. Такая процедура принята во всех вузах. Более того, по тематикам выбранных студентами заданий издаётся приказ, фиксирующий тему выполняемой работы, а также оформляется документ, кратко фиксирующий ожидаемый результат практики.

Чтобы разрешить необходимые формальности, руководитель практики от университета проводит организационные собрания, где:

а) ***студент*** докладывает о выбранной теме работы, её названии и предполагаемых результатах;

б) ***руководитель практики*** рассказывает об общих требованиях к процессу обучения и отчётности результатов работы, а также обеспечивает студентов необходимой методической литературой.

Примечание — Студенту всегда первоначально кажется, что задача слишком простая и она может быть недостаточной для будущих требований, предъявляемых экзаменационной комиссией к ВКР.

Петрову И.В. кажется, что задача создания электронного журнала является слишком простой, чтобы её реализацию можно было бы в дальнейшем представить на защиту ВКР. К тому же, его смущает сильная аналогия с прототипом, достаточно профессионально и полно реализованным на портале ТУСУР. Такого же мнения придерживались и некоторые участники организационного собрания.

Подобная ситуация является типичной при выборе тематики заданий и часто приводит к предложению студента изменить название или направление будущей работы.

Не вдаваясь в детали и возможные варианты развития сюжета, научный руководитель предложил Петрову И.В. более расширенную трактовку задания в виде — «*Электронный Журнал Руководителя*» (ЭЖР), подразумевая, что частным вариантом задачи может служить первоначальная трактовка информационной системы.

В результате указанных согласовательных мероприятий, окончательное задание на производственную практику Петрова И.В. получило документальное оформление, показанное на рисунке 2.1.

Сразу отмечу, что, кроме задания показанного на рисунке 2.1, Петрову И.В. необходимо будет заполнять и другие документы обозначенные руководителем практики от университета. Учебный материал данного пособия не рассматривает подробный перечень таких документов, тем более, что они могут быть различными в зависимости от вида практики. Что ка-

сается документа «***Задание***», то этот документ приведён *по причине фиксации в нём названия темы работы*, которая должна быть отражена во всех проектных документах ИС.

Министерство науки и образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ

Кафедра автоматизированных систем управления (АСУ)

Задание

на производственную практику
«Преддипломная практика»

Студенту кафедры АСУ группы 447-1 Петрову Ивану Васильевичу

Тема работы: Электронный журнал руководителя

Индивидуальное задание: Разработать прототип информационной системы для мониторинга деятельности отдельных групп исполнителей «ЭЖР»

Время прохождения практики: с 01.09.2020 по 14.09.2020

Руководитель практики от университета,
доцент каф. АСУ, кандидат технических наук
М.В.Григорьева _____

Томск 2020

Рисунок 2.1 - Пример задания на преддипломную практику

2.1.3 Подробное описание задачи на проектирование ИС

Примечание — Перед началом процессов проектирования студент должен оформить в письменном виде подробное описание задачи.

Оформив задание на производственную практику, студент официально приступает к ***первому этапу проектирования*** — «*Формирование требования к ИС*». Перечень работ и необходимый результат этого этапа кратко описан во вводной части данной главы.

Формально этот этап предполагает:

1) описание организационной структуры предприятия;

2) выделение объекта проектирования;
3) анализ бизнес-моделей объекта проектирования;
4) оформление отчёта по выполненному этапу работ, включающему заявку на разработку тактико-технического задания ИС.

Все эти работы требуют не только достаточного времени и усилий на изучение дополнительной литературы, но и постоянного обращения к сути и ограничениям самой постановки задачи. В этих условиях, у студента «под рукой» всегда должно быть *письменное описание задачи*, в котором должны быть представлены *все важные ограничения на её постановку*.

Примечание — **Классическое проектирование** — это *анализ* (декомпозиция) предметной области в условиях достаточно большой неопределённости. Обычно эту парадигму называют стратегией проектирования «сверху-вниз».

Мысль студента мечется между элементами знаний, которые он уже освоил, и новыми идеями, которые спонтанно появляются и исчезают, оставляя в голове какие-то «мутные следы». Это приводит к тому, что он хватается за самую, на его взгляд, перспективную идею и начинает её реализовывать. Как правило, впоследствии оказывается, что он упустил что-то главное, что делает его проект необоснованным и некачественным.

Примечание — Нормальное (классическое) проектирование всегда *отталкивается от цели создания ИС* и использует стратегию «сверху-вниз».

Цель создания ИС ЭЖР — обеспечить преподавателя простыми индивидуальными средствами контроля выполнения лабораторных работ студентами, что по замыслу самого задания должно повысить качество этого процесса обучения.

В отличие от уже названного ранее прототипа (ЭЖ на центральном портале ТУСУР), цель которого официальный внешний контроль деятельности не только студентов, но и преподавателей, целевое назначение ИС ЭЖР — оказание помощи преподавателю, несущему повышенную учебную нагрузку, вызванную большим количеством студентов в группах (подгруппах).

В техническом плане, ИС ЭЖР должна представлять из себя распределённую систему, ограниченную средствами связи локальной сети кафедры АСУ, с выделенным сервером преподавателя и клиентскими станциями обучающихся студентов.

Ограничительные условия процесса функционирования ИС ЭЖР должны включать:

а) ***разделение*** участников распределенного взаимодействия по *группам* (подгруппам), *наименованию дисциплин* и *времени проведения* лабораторных работ;

б) ***архивирование*** и ***удаление*** из активного процесса уже не нужной части ИС ЭЖР для конкретной группы (подгруппы) и дисциплины;

в) ***восстановление*** в активном процессе ранее уже удалённых частей ИС ЭЖР;

г) ***авторизацию*** каждого отдельного студента в серверной части ИС ЭЖР;

д) ***регистрацию*** и ***сохранение*** в серверной части ИС ЭЖР сообщений, переданных студентом преподавателю, и его сигналов активности в пределах времени запланированного занятия.

Примечание — Отсутствуют ограничения на подробность и степень качества первичного текстового описания задачи.

Приведённое выше описание задачи не претендует на образец для подражания, тем более оно приведено в стиле рекомендаций научного руководителя для исполнения Петро-

вым И.В. Главное, чтобы студент хорошо понял и учёл наиболее важные целевые ограничения задачи и использовал их в последующей работе.

В целом при описании задачи могут быть использованы различные изображения, диаграммы или иные формы представления информации. Не требуется даже обязательного включения этой части работ в итоговый отчёт, но между научным руководителем и студентом должно быть достигнуто *единое понимание общего целевого назначения системы*.

2.2 Организационная структура управления предприятием

Примечание — Все мысли студента сосредоточены на выполнении полученного задания, поэтому он недостаточно внимания уделяет организационной структуре управления предприятием. Это — ***большая ошибка***.

Любое предприятие имеет организационную структуру управления. Не существует предприятий, которые не имеют организационной структуры управления. Необходимо понимать, что создание или функционирование любой системы в пределах предприятия подчинено иерархии управления, даже если студент этого не понимает. Также естественно, что организационная структура предприятия влияет на принятие проектных решений применительно к проектируемой системе.

Соответственно **возникают вопросы**:

а) ***где брать*** необходимую информацию?

б) ***в каком объёме*** необходимо проводить описание такой структуры?

в) ***с какой степенью подробности*** необходимо проводить описание?

В современных условиях ответ на поставленные вопросы — достаточно прост: необходимо обратиться к официальному сайту предприятия, где представлена вся публичная информация, которую студент может вполне смело использовать, не нарушая каких-либо лицензионных или иных авторских ограничений.

Применительно к нашему заданию, заходим на официальный сайт вуза по адресу ***https://tusur.ru*** и переходим к документу «*Основные сведения*», что показано на рисунке 2.2.

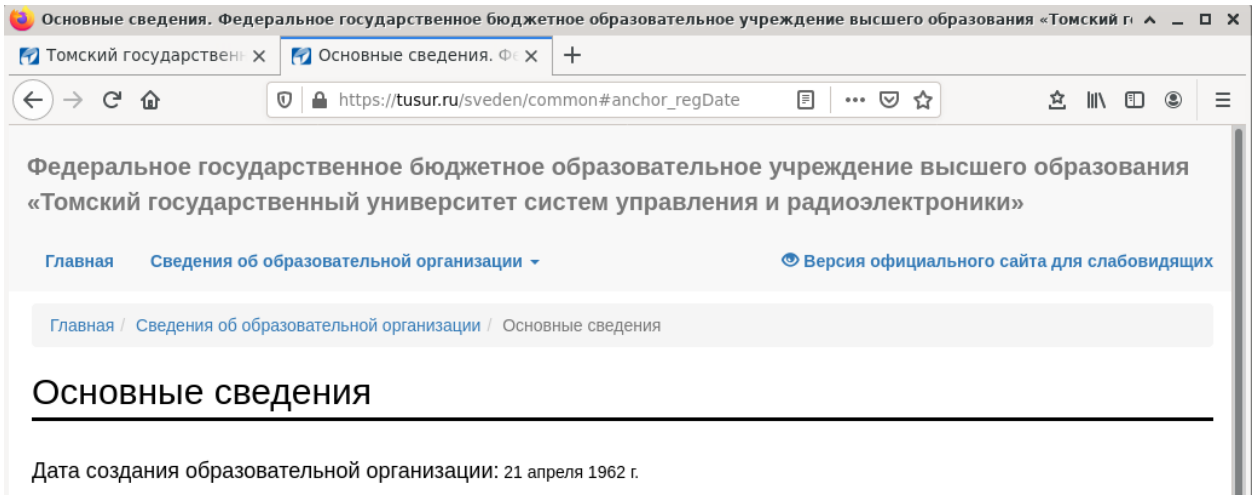


Рисунок 2.2 — Документ с основной информацией о ТУСУР

Далее по этому документу извлекаем основную представительную информацию о вузе, как это показано на рисунке 2.3.

Организационная структура любой организации имеет ярко выраженную иерархическую архитектуру.

Иерархическая структура предприятия может быть достаточно большой и сложной. Необходимо *обязательно отразить ту ее часть, которая включает* ***предметную область*** *проектируемой ИС*.

Полное наименование университета: Федеральное государственное бюджетное образовательное учреждение высшего образования «Томский государственный университет систем управления и радиоэлектроники»

Сокращённые наименования университета: ТУСУР, ФГБОУ ВО «ТУСУР», ФГБОУ ВО «Томский государственный университет систем управления и радиоэлектроники», Томский государственный университет систем управления и радиоэлектроники

Учредители образовательной организации:

Наименование учредителя	Фамилия, имя, отчество учредителя (руководителя учредителя) (ей) образовательной организации	Адрес местонахождения учредителя(ей)	Контактные телефоны	Адрес электронной почты	Адрес сайта учредителя(ей) в сети «Интернет»
Российская Федерация. Функции и полномочия учредителя Университета осуществляет Министерство науки и высшего образования Российской Федерации.	Котюков Михаил Михайлович - Министр науки и высшего образования Российской Федерации	125993, г. Москва, ул. Тверская, 11	(495) 539-55-19	info@mon.gov.ru	http://минобрнауки.рф/

Информация о месте нахождения образовательной организации:

Юридический адрес: 634050, г. Томск, пр. Ленина, 40

Дополнительный юридический адрес: 634050, г. Томск, пр. Ленина, 40

Фактический адрес: 634050, г. Томск, пр. Ленина, 40

Рисунок 2.3 — Основная представительная информация о ТУСУР

В частности, официальная организационная структура ТУСУР доступна адресу: https://storage.tusur.ru/files/133775/Strukturnaya_skhema_2020.pdf. Учитывая, что проектируемая ИС предназначена для использования в пределах подразделения «*Кафедра АСУ*», необходимая часть организационной структуры вуза может быть представлена рисунком 2.4.

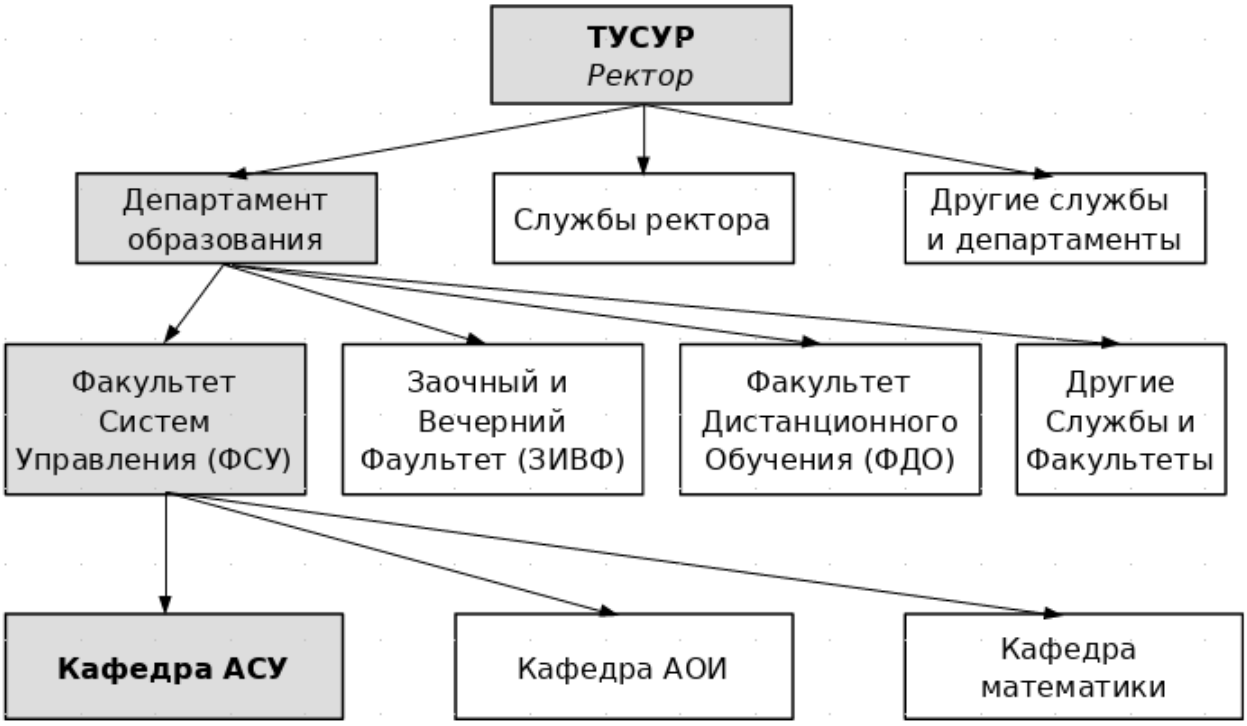


Рисунок 2.4 — Общая проектная часть организационной структуры управления вуза

Приведённый рисунок наглядно показывает основной управляющий поток, проходящий от указов и распоряжений ректора вуза, через «Департамент образования», руководя-

щую структуру факультета ФСУ и заканчивается в подразделении «*Кафедра АСУ*». Естественно, что этот рисунок должен быть включён в отчёт по первой стадии проектирования, но его интерпретацию ещё необходимо дополнить проекцией на модель АС и определением границ предметной области объекта исследования.

2.2.1 Подсистемы АС как отражение структуры управления предприятием

Типичной ситуацией для большинства предприятий (организаций, вузов) является строгое иерархическое управление «*сверху-вниз*», когда сотрудник какого-либо подразделения получает задание от своего непосредственного руководителя, выполняет работу и отчитывается перед ним за полученный результат. Распространение заданий и отчётность за них «через голову» непосредственного руководителя подразделения считаются неуместными или полностью недопустимыми событиями. Тем не менее, имеются случаи, когда такие ситуации являются не только правильными, но и входят в технологическую цепочку производственной деятельности. Такие ситуации обязательно должны быть выявлены и включены в набор требований к проектируемой системе.

Применительно к нашему учебному заданию рассмотрим организацию учебного процесса кафедры АСУ, где предполагается будущее использование проектируемой ИС.

Непосредственным начальником всех сотрудников кафедры является её заведующий, который согласовывает и получает из вышестоящих подразделений (факультетов) задания на обучение студентов, организованных в группы по списку обеспечиваемых кафедрой дисциплин. Основным поставщиком групп (подгрупп) студентов является «Факультет Систем Управления» (ФСУ), но дополнительно имеются и студенты факультетов ЗИВФ и ФДО.

Планирование процесса обучения на будущий учебный год осуществляется по окончании текущего учебного года и распределяется между сотрудниками кафедры — преподавателями.

Непосредственное обучение студентов проводят преподаватели кафедры (профессора, доценты и другой младший персонал) согласно индивидуального расписания на проведение конкретных занятий по конкретным дисциплинам. Заведующий кафедрой проводит только общие организационные мероприятия и формирует отчётность кафедры по установленному в вузе регламенту работ.

Отчётность о проведении занятий осуществляется непосредственно преподавателями, которые напрямую взаимодействуют с уполномоченными сотрудниками деканатов соответствующих факультетов, руководствуясь регламентами календарного плана обучения и требованиями рабочих программ изучаемых дисциплин.

Таким образом, рассматривая подразделения факультетов и кафедр вуза как различные уровни подсистем автоматизированной системы (АС), мы видим явное нарушение строгой иерархии управления, что для поставленной задачи проектирования графически отображается рисунком 2.5.

Неисключено, что нарушение прямой иерархии организационного управления, где сплошными стрелками показаны ***потоки групп*** студентов на обучение, а штриховыми стрелками — ***потоки отчётности*** преподавателей по результатам обучения, присутствуют и на других уровнях подразделений вуза. Но принимать решение о проведении такого дополнительного анализа иерархии управления следует только после определения границ предметной области решаемой задачи.

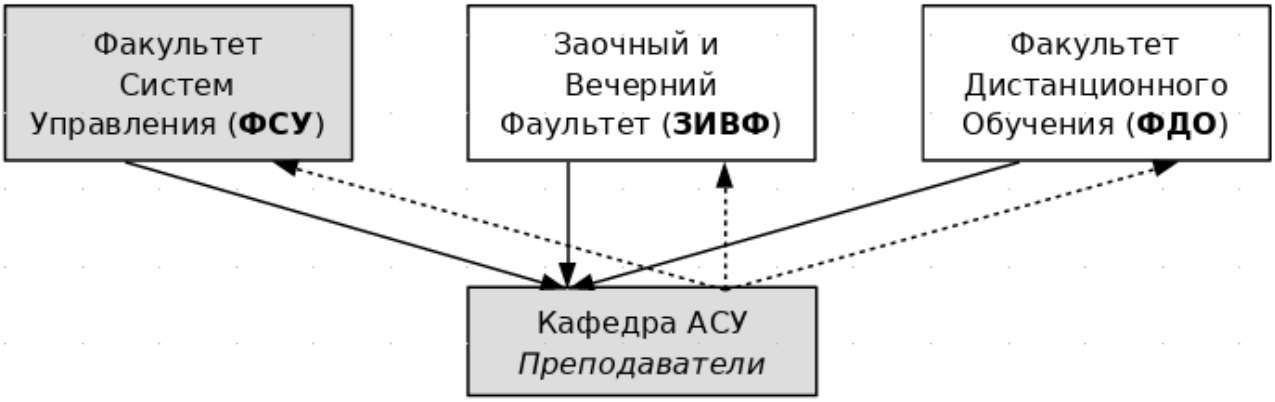


Рисунок 2.5 — Нарушение строгой иерархии управления между подразделениями факультетов и кафедрой АСУ

2.2.2 Определение границ предметной области ИС

Расширение границ предметной области любой системы всегда приводит к появлению новых связей и зависимостей между её элементами и окружающей средой. Как правило, это приводит к несоответствию между поставленными целями и итогом реализации системы.

В предыдущем пункте отмечено нарушение строгой иерархии организационного управления, которое выявлено на уровне отношений кафедры АСУ и более высокими уровнями подразделений трёх факультетов: ФСУ, ЗИВФ и ФДО. Наверно такое нарушение иерархии управления можно отметить и между другими кафедрами и факультетами вуза ТУСУР. Возможно проектировщик ИС, заметив указанную закономерность и решив воспользоваться появившейся возможностью, захочет обобщить выявленный прецедент и использовать его как аргумент в выборе каких-либо проектных решений. Наверняка он захочет обосновать перспективность будущего проекта ИС, расширив область его применения за рамки поставленной руководителем задачи.

Назначение данного пункта проекта — анализ выявленных особенностей иерархии управления процессами обучения студентов с целью задания конкретных границ предметной области проекта и устранения нежелательных последствий, вызванных незнанием студентом этих границ.

Нарушение иерархии управления, показанное на рисунке 2.5, не является характерной чертой деятельности всех предприятий и организаций. Оно отражает особенности оперативного исполнения процессов обучения в вузах.

Действительно, выявленное нарушение иерархии управления выявлено только потому, что проектируемая ИС, согласно постановке задачи, является индивидуальным инструментом отдельного конкретного преподавателя, осуществляющего учебный процесс, а объектом (продуктом) обучения является группа (подгруппа) студентов, общее руководство которыми осуществляют сотрудники деканата факультетов. В такой проекции, преподаватель периодически берёт на обучение группу студентов из общего пула групп (подгрупп), «*принадлежащих*» факультету и возвращает их обратно, после завершения занятия. Естественно, что отчётность за обучение преподаватель осуществляет через деканат факультета, а руководитель кафедры в этом процессе не участвует.

Совершенно другая картина наблюдается в масштабе годового или семестрового планирования процессов обучения. Здесь заведующие кафедрами осуществляют прямое взаимодействие с деканатами, беря на себя ответственность за обучение групп студентов в соответствии со своими долгосрочными обязательствами, а также распределяют обеспечиваемый на-

бор дисциплин между преподавателями кафедры. На таком уровне никаких нарушений в иерархии управления не наблюдается.

Завершая данный пункт проектирования, проводим фиксирование границ предметной области проектирования следующими положениями:

а) ***областью применения*** проектируемой ИС являются процессы проведения отдельных лабораторных работ по конкретным дисциплинам;

б) ***отдельное применение*** проектируемой ИС связано с процессом обучения конкретной группы (подгруппы) студентов по отдельной конкретной дисциплине, которое осуществляется в пределах отведённого для этого времени занятия.

2.2.3 Результат анализа организационной структуры вуза

Анализ организационной структуры управления предприятием всегда выполняется первым шагом проектирования, поскольку считается, что такая структура не изменяется за весь период создания и внедрения системы.

Анализ организационной структуры управления предприятием даёт хотя и грубые, но очень важные оценки границ предметной области, в пределах которой проводится проектирование ИС. Если структура управления студентом плохо изучена или есть сомнения в ее стабильности, то необходимо продолжить ее анализ, иначе следует ожидать серьёзных ошибок при принятии последующих проектных решений.

Применительно к нашему заданию, выполняемому студентом Петровым И.В., вполне обоснованно сделать вывод, что выявленные нарушения иерархии управления, представленные рисунком 2.5, не влияют на последующие проектные решения создаваемой ИС. Такой вывод сделан на том основании, что проведение промежуточной аттестации результатов обучения проводится исключительно преподавателем и не является функцией ЭЖР.

Тем не менее, научный руководитель Петрова И.В. даёт ему прямое задание на повторное проведение работ, выполненных по пункту 2.1.3, чтобы дополнить подробное описание задачи с учётом проведённого в данном подразделе исследования.

2.3 Определение требований к объекту проектирования

Завершив анализ организационной структуры предприятия и определив границы предметной области будущей системы, студент переходит ко второму этапу первой стадии проектирования, обозначенной ранее как «*Формирование требований пользователя к ИС*».

Что такое — ***формирование требований***, студенту более или менее понятно, а вот что такое — ***пользователь***, требует пояснения.

Интуитивно, пользователь — это лицо или группа лиц, которые предположительно будут работать с проектируемой системой. В таком аспекте студент обычно представляет себя, сидящего за компьютером и выполняющего те действия и функции, которые описаны в постановке задачи. Безусловно, такой подход психологически обоснован и присутствует в любой деятельности человека, но он является сильно ограниченным, поскольку опирается только на личный опыт проектировщика.

Формально (по технологии проектирования), пользователь — это будущие абстрактные исполнители, определённые заказчиком ИС в соответствии с ее целевым производственным назначением.

Между приведёнными определениями часто существует достаточно тонкое, но принципиально важное различие. Поясним это следующим высказыванием: «*Конструктор самолёта необязательно должен быть лётчиком и необязательно должен уметь пилотировать своё будущее изделие*».

Применительно к нашей задаче, студент Петров И.В. должен формулировать требования к ИС, опираясь не на свой воображаемый опыт преподавания, а на требования будущего заказчика ИС, прототипом которого может выступать автор данного пособия. Дополнительно следует помнить, что официальный заказчик системы фиксируется по результатам выполнения стадии «*Техническое задание*». До указанного времени существует только потенциальный заказчик, которым, согласно полученному заданию, является подразделение ТУСУР, именуемое как «*Кафедра АСУ*».

Подводя итог обсуждению понятия пользователь, Петров И.В. фиксирует следующие выводы:

а) ***пользователем ЭЖР*** является отдельный преподаватель кафедры;

б) ***возможные роли*** пользователя требуют дополнительного анализа задания в плане уточнения взаимодействия преподавателя и обучаемых им студентов;

в) ***дополнительным вопросом*** является возможность выделения отдельных ролей системы в качестве полноценных пользователей.

Для качественного формулирования требований к объекту проектирования, *необходимо подробное описание продукции*, выпускаемой предприятием.

Выполнение этой части проектной работы следует проводить в следующих аспектах:

а) необходимо выделить ***тот номинал продукции*** предприятия, который непосредственно взаимодействует как с пользователем, так и с ИС;

б) необходимо описать ***те свойства продукции***, которые существенно влияют на будущие проектные решения.

Учитывая большое разнообразие возможных вариантов такой деятельности, ограничимся только случаем нашего задания.

Кроме прямой учебной деятельности, ТУСУР выполняет различные виды научных работ, выпускает научный журнал, проводит конференции и выполняет другие хозяйственные

договора. Любой преподаватель как сотрудник данного вуза является участником указанных работ и несёт за их выполнение необходимую ответственность.

Прямое отношение к проектируемой системе имеет производственный процесс, связанный с обучением студентов (бакалавров и магистров) в виде отдельных циклов обработки: *лекции*, *практические занятия*, *лабораторные работы* и *различные виды исследовательских и производственных практик*.

Законченной продукцией (изделием) производства является выпускник вуза, прошедший обучение и защитивший выпускную квалификационную работу (ВКР).

Полный обучающий цикл обработки изделия составляет:

а) ***бакалавр*** — четыре года (восемь семестров);

б) ***магистр*** — два года (четыре семестра).

Проектируемые требования к системе связаны с циклом обработки изделия, именуемые как ***лабораторные работы***.

Лабораторные работы — специальный вид обучающего цикла обработки студента, выполняющийся в пределах отдельных дисциплин согласно учебному плану и тематике рабочей программы дисциплины, который проводится в составе группы (подгруппы) и обычно требует специально оборудованных классов вуза.

Технология проведения лабораторной обработки студента предполагает самостоятельное выполнение им конкретных заданий с использованием учебного материала методических пособий, консультирующих и контролирующих воздействий преподавателя.

Оценивание качества лабораторной обработки студента проводится непосредственно преподавателем и учитывается в семестре по результатам двух контрольных точек и завершающей оценки семестра: *экзамен*, *зачёт* или *зачёт с оценкой*.

Проектируемая ИС должна обеспечивать:

а) ***подготовку*** перед началом семестра исходной информации о составе группы (подгруппы) для каждой дисциплины отдельно;

б) ***регистрацию факта*** посещения и активности студента в каждом цикле выполнения лабораторных работ;

в) ***асинхронную связь*** преподавателя и студента в пределах отдельной лабораторной работы.

> На данном этапе работ может быть указан перечень инструментальных или иных средств, обеспечивающих реализацию проектируемой ИС.

В данной части работ могут приводится любые дополнительные сведения, которые по мнению проектировщика поясняют и обосновывают выдвигаемые требования к ИС. Не мешает и повторение уже ранее выдвинутых требований. Главное, чего должен избегать проектировщик, — предположений о конкретных способах реализации системы. Обычно, в дальнейшем, это приводит к ошибкам и разочарованиям.

Лучшим решением на данном этапе работ является поиск, описание и анализ возможных прототипов проектируемой системы.

2.3.1 Поиск прототипов ИС

> Грубейшей ошибкой проектировщика является утверждение, что проектируемая ИС не имеет аналогов (прототипов).

Проектировщику утверждающему, что для его системы отсутствуют аналоги или прототипы, необходимо срочно менять профессию. Мы не будем обсуждать философские и медицинские проблемы причин подобных утверждений, а сразу заявим в качестве прототипа

ЭЖР — «*Журнал успеваемости ТУСУР*», доступный каждому преподавателю и студенту ТУСУР на его центральном портале.

Безусловно, студенту следует сделать обзор различных источников и отметить ряд подобных по прикладному назначению систем, указав их в качестве возможных прототипов, и перейти к следующему пункту работ.

2.3.2 Выбор и описание прототипов ИС

Студенты и преподаватели видят «*Журнал успеваемости ТУСУР*» по разному. У преподавателя больше возможностей, но для целей проектирования они — практически одинаковы.

Чтобы не возникало разночтений, будем рассматривать систему в проекции преподавателя.

Осуществив вход на портал ТУСУР и выбрав в меню вошедшего «*Журнал успеваемости ТУСУР*», мы заходим на страницу «*Кабинет преподавателя*», где доступны две вкладки:

а) ***Ведомости*** — вкладка, перечисляющая список дисциплин закреплённых за преподавателем и групп (подгрупп) студентов изучающих указанные дисциплины;

б) ***Журналы*** — вкладка, содержащая ссылки на журналы изучаемых студентами дисциплин, которые переводят преподавателя в его личный кабинет обучающей системы ***Moodle*** (*Modular Object-Oriented Dynamic Learning Environment*).

Что касается вкладки «*Ведомости*», то преподаватель может зайти по ссылке на конкретную группу, привязанную к изучаемой дисциплине и:

а) ***выставить*** успеваемость по контрольным точкам (КТ1 и КТ2);

б) ***скачать*** ведомость для экзамена или зачёта;

в) ***перейти*** по ссылке диалога с группой, где в режиме чата может общаться со студентами посредством печати сообщений.

На вкладке «*Журналы*» преподаватель может создавать журнал для закреплённой за ним дисциплины, подключать к журналу конкретные группы студентов, а также отдельных пользователей, которые имеют регистрацию на портале ТУСУР. В дальнейшем, преподаватель может заходить в конкретный журнал дисциплины и пользоваться всеми возможностями системы Moodle по созданию электронного курса дисциплины, а также проводить дистанционное обучение и контроль успеваемости студентов. Причём, все или большинство действий преподавателя и студентов в таком журнале фиксируются в системе и доступны для контроля руководством ТУСУР.

Система Moodle обеспечивает организацию комплексного подхода для создания преподавателями учебных курсов и проведение занятий со студентами в режиме дистанционного обучения (ДО).

Система Moodle, позицируемая как система управления обучением или виртуальная обучающая среда. Она начала разрабатываться австралийским программистом Мартином Догиамасом в 1999 году. Имеет официальный web-сайт ***https://moodle.org*** [32] и реализована как свободная модульная объектно-ориентированная динамическая система, которую можно скачать и установить в следующих вариантах:

а) ***Standart Moodle*** — требуется собственный web-сервер с PHP и СУБД;

б) ***Moodle Mobile*** — доступном для Android и iOS;

в) ***Moodle Desktop*** — устанавливается на компьютеры с ОС Windows, MAC и Linux.

Система Moodle имеет достаточно большое количество документации на английском и русском языках, поэтому студент Иванов И.В. без труда найдёт множество полезных источ-

ников, а также составит хороший обзор системы и инструкций по ее применению. Мы же используем только информацию сайта https://lmslist.ru/free-sdo/obzor-moodle, на котором даётся оценка требований по установке системы и её настройкам.

На рисунке 2.6 приведены требования к аппаратной части ЭВМ, на которой предполагается установка Moodle, а также поддерживаемые системой базы данных и браузеры. Отмечается, что установка Moodle с минимальными требованиями потребует от неподготовленных пользователей порядка одного месяца работы.

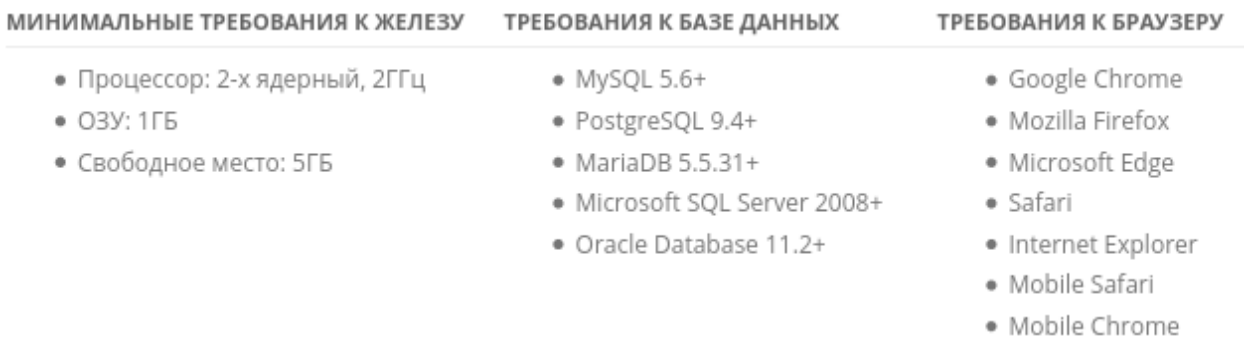


Рисунок 2.6 — Требования к системе Moodle

Другой источник — https://docs.moodle.org/archive/ru/Установка_Moodle нам сообщает, что: «*Поначалу Moodle создавался в Linux с использованием Apache, MySQL и PHP (Linux + Apache + MySQL + PHP = LAMP), но регулярно проверялся в работе в среде Windows XP/ 2000/2003 (WAMP), Solaris 10 (Sparc and x64), Mac OS X и Netware 6. Также имеется поддержка СУБД PostgreSQL, Oracle и Microsoft SQL Server*». Современные дистрибутивы Moodle требуют установки Perl и другие ограничения.

В целом, можно смело утверждать, что Moodle — достаточно зрелая инструментальная система для организации управления обучением, требующая профессиональной установки и обслуживания.

2.3.3 Результаты анализа требований

Проведя выбор и описание прототипов, необходимо обязательно сделать краткие выводы, отразив их вместе со списком использованных источников в соответствующем подразделе индивидуального отчёта студента.

Прототипов может быть много и один лучше другого. Источников — тоже много, тем более, если система популярна и долго используется. Естественно, что студенту может показаться, что проще использовать уже готовую систему, дополнив ее необходимым функционалом. В 90-е годы такая позиция приводила к ситуации, получившей название «*Островная автоматизация*», когда на предприятиях покупались и устанавливались различные системы, которые были несовместимы между собой.

Чтобы не попасть в указанную выше ловушку, студенту рекомендуется кратко описать в отчёте положительные стороны выделенных прототипов, а затем уделить особое внимание их недостаткам.

Здесь мы не будем описывать положительные стороны выбранной системы Moodle. Они и так хорошо — известны, а сразу отметим ряд её недостатков:

а) система — ***достаточно сложная*** и требует профессионального сопровождения, что является негативной характеристикой для индивидуальной системы преподавателя;

б) система имеет ***централизованную структуру*** размещения и управления, что упрощает её реализацию, но создаёт «*узкие места*» в плане доступа к ней и масштабируемости её реализации;

в) система — ***слишком специализирована***, что делает ее непригодной в качестве инструментального средства интеграции различных по назначению приложений предприятия;

г) система — ***слишком перегружена*** различными инструментальными средствами реализации обучающих технологий, но не содержит инструментальных средств автоматизации проведения лабораторных работ;

д) конфигурация системы Moodle ТУСУР ***ориентирована на общий проект*** ФДО, что не предполагает подключение к ней частных и незапланированных решений.

2.4 Требования к бизнес-моделям объекта проектирования

Содержание предыдущих двух подразделов посвящено формированию требований к предметной области и объекту проектирования ИС, где студент рассмотрел организационную структуру предприятия, выделил основные объекты, участвующие в будущем функционировании ИС, а также провёл анализ прототипов проектируемой системы.

> В данном подразделе завершается формирование требований пользователя к ИС, по которым и будет определяться полезность и перспективность самой ИС.

Заявленную часть работ по проектированию невозможно проводить без анализа содержимого предыдущих подразделов данной главы. В общем случае можно указать только на необходимость:

а) ***выделения перечня узлов*** предметной области, в которых будет создаваться и потребляться информация;

б) ***выделение связей*** между элементами бизнес-моделей, которые участвуют в функционировании бизнес-процессов.

> Главными элементами бизнес-моделей и участниками бизнес процессов являются пользователи проектируемой ИС.

Из постановки задачи, подробно описанной в подразделе 2.1, следует, что пользователями проектируемой ИС являются преподаватели и студенты, ограниченные проведением отдельных лабораторных работ по конкретным дисциплинам. Если выделить отдельный бизнес-процесс, то в нем одновременно участвуют:

а) **Преподаватель** — сотрудник кафедры АСУ, ведущий групповое обучение в пределах отдельной лабораторной работы по конкретной отдельной дисциплине;

б) **Студент** — отдельный объект обучения, которому преподаватель выдаёт задание, а затем контролирует и оценивает его выполнение;

в) **Сервер** — проектируемая ИС, реализующая функционал ЭЖР и автоматизирующая взаимодействие преподавателя и студента;

г) **Клиент** — программный компонент ЭЖР, обеспечивающий пользователям техническое взаимодействие с сервером.

Очевидно, что выделенные объекты бизнес-процессов допускают создание более специализированных объектов:

а) **Клиент-преподавателя** — ПО клиента, которое использует только преподаватель;

б) **Индивидуальный-сервер** — ЭВМ преподавателя, на котором расположено ПО клиента и сервера;

в) **Общественный-сервер** — отдельная ЭВМ, на которой расположено только ПО сервера;

г) **Клиент-студента** — отдельная ЭВМ, на которой расположено ПО клиента и которое использует только студент.

> Выделив необходимое количество главных объектов, участвующих в бизнес-процессах проектируемой ИС, перейдём к выделению мест создания, хранения и потребления информации. Не забываем, что выделенные объекты находятся также в групповых отношениях.

2.4.1 Узлы создания, потребления и хранения информации

Создателями и потребителями информации являются пользователи системы, но не только они.

В данном пункте важно выделить те источники информации, которые входят в систему из вне, а также обозначить тех внешних потребителей, которые используют созданную в системе информацию. В нашей системе таких внешних источников и потребителей информации нет, поскольку по замыслу ЭЖР является автономной (индивидуальной) системой. Вся внешняя информация поступает в ИС через пользователей: *преподавателей* и *студентов*. Следовательно, *хранилищем информации является сервер проектируемой ИС*.

Обязательно следует выделить особенности групповых отношений.

Основное групповое отношение, которое должно присутствовать в проектируемой ИС, предполагает, что один преподаватель, в пределах лабораторной работы, обучает одну группу студентов. Следовательно, в систему должен быть добавлен новый объект — «***Группа-студентов***».

Группа-студентов — именованный групповой объект пользователей, где перечисляются объекты типа **Студент**, причём отдельный студент входит только в одну группу.

В условиях распределенной архитектуры проектируемой ИС, указанное групповое отношение допускает два варианта интерпретаций, показанных на рисунках 2.7 и 2.8.

Рисунок 2.7 — Групповое отношение через «Индивидуальный-сервер»

Рисунок 2.8 — Групповое отношение через «Общественный-сервер»

Групповое отношение через «Индивидуальный-сервер», показанное на рисунке 2.7, кажется проще отношения через «Общественный-сервер», потому что во время выполнения лабораторной работы каждый студент группы обращается только к одному преподавателю.

Недостатком такой архитектуры является необходимость знания студентом *индивидуального сетевого адреса ЭВМ преподавателя*. Очевидно, что преподаватель в начале занятия должен сообщить студентам адрес и порт своего сервера.

Групповое отношение через «Общественный-сервер», показанное на рисунке 2.8, кажется достаточно сложным, поскольку требует проектирования операций мультиплексирования и демультиплексирования информационных запросов и ответов в системе. Но если учесть, что в проекте будет развернут уже готовый сервер приложений, качественно обеспечивающий распределение запросов и ответов пользователей, то вопрос о возможной сложности проектного решения снимается и не принимается как существенный аргумент.

Преимуществом такой архитектуры является большая универсальность проектного решения и *наличие заранее известного студенту сетевого адреса ЭВМ преподавателя*.

> Примечание — В дальнейшем, во избежание недоразумений, под термином ***Сервер*** будет пониматься ***Общественный-сервер***.

Следует заметить, что введение излишних ограничений, особенно на начальных этапах проектирования систем, является крайне негативным решением. Поэтому будем предполагать, что ***Общественный-сервер*** может быть размещён и на ЭВМ преподавателя.

2.4.2 Перечень связей между элементами бизнес-моделей

> Выделение и описание связей между элементами бизнес-моделей проектируемой системы является не только обязательной, но и самой сложной частью стадии формирования требований к ИС.

Заявление о сложности выделения связей между элементами бизнес-моделей является не надуманным предположением, а обобщением общественного опыта проектирования различных систем. Студентам, которые сомневаются в данном утверждении, рекомендуется перечитать подраздел 1.1 данного учебного пособия.

> Главная проблема выделения ***внешних связей*** проектируемой системы — возможное, хотя и непреднамеренное, изменение границ изучаемой предметной области.

Изменение границ изучаемой предметной области всегда приводит к появлению новых или удалению старых внешних связей системы, а также к изменению свойств внутренних связей. Это явление более подробно обсуждается в теории и практике концептуального проектирования, изложенных в следующем разделе данного пособия. Здесь же, на стадии определения требований к ИС, достаточно выделить и описать наиболее важные связи, принципиально отражающие суть решаемой задачи.

Конкретизируя ситуацию применительно к нашей учебной задаче, в явном виде можно выделить только ***управляющие внешние связи***, к которым относятся:

а) перечень и состав учебных групп, участвующих в отдельных процессах обучения (бизнес-процессах);

б) перечень преподавателей и закрепленный за ними перечень дисциплин, рассматриваемых как последовательность отдельных бизнес-процессов над закреплёнными за ними группами студентов;

в) расписание занятий, фиксирующее дату, время начала и продолжительность отдельных бизнес-процессов.

Такая ситуация значительно упрощает архитектуру проектируемой системы, позволяя выделить:

а) **Занятие** — *главный бизнес-процесс* проектируемой ИС, выполняемый по единым общим правилам и функционально объединяющий все выделенные ранее объекты бизнес-моделей;

б) **Второстепенные Бизнес-процессы** (ВБП) — набор операций с ИС ЭЖР, связанный с подготовкой списков групп студентов, привязкой к группам изучаемых дисциплин, а также других объектов и организационных мероприятий, обслуживающих главный бизнес-процесс.

> Главная проблема выделения ***внутренних связей*** проектируемой системы — адекватный учёт выделенных границ изучаемой предметной области.

Хотя вопрос адекватности кого-либо или чего-либо всегда является достаточно спорным, он всегда разрешается лишь одним способом — подробным исследованием постановки задачи. В любом случае, начинать такое исследование необходимо с анализа групповых отношений между уже выделенными объектами, являющимися участниками бизнес-процессов.

В пределах проектируемой нами ЭЖР уже выделен один групповой объект — «***Группа-студентов***». В пределах своего жизненного цикла, такой объект создаётся вне проектируемой ИС и должен быть перенесён в систему преподавателем, который будет вести занятия с этой группой хотя бы по одной дисциплине. Кроме того, таких объектов в системе может быть несколько, что требует учёта следующих их характеристик:

а) группа студентов имеет свой жизненный цикл в семестровых единицах: восемь семестров для бакалавриата и четыре семестра для магистратуры;

б) группы студентов могут иметь деления на подгруппы со своими заранее утверждёнными цифровыми обозначениями, уникальными с периодом в десять лет;

в) списочный состав группы студентов в пределах одного семестра обучения является достаточно стабильным, хотя его состав может незначительно изменяться в пределах своего жизненного цикла;

г) списочные составы различных групп студентов не пересекаются.

Таким образом, групповой объект — «***Группа-студентов***» является достаточно самостоятельной информационной единицей, которая должна существовать в проектируемой системе и иметь собственные инструменты поддержания свой целостности.

Теперь у нас имеются все основания сформировать новый групповой информационный объект, который основан на достаточно обобщённом понятии «***Учебная дисциплина***» и выделенным ранее главным бизнес-процессом проектируемой ИС, обозначенным как «***Занятие***».

> **Электронный Журнал Дисциплины** (***ЭЖД***) — групповой автономный информационный объект, статически объединяющий информационные проекции объектов «***Преподаватель***», «***Группа-студентов***» и «***Учебная дисциплина***», а также динамически формируемый информационным содержимым результатов последовательности бизнес-процессов «***Занятие***».

Приведённое определение информационного объекта ЭЖД фактически завершает исследовательскую часть данного подраздела и требует обобщения полученных результатов.

2.4.3 Результаты анализа требований

Результаты анализа требований к ИС распределены по четырём подразделам данного раздела и описывают различные аспекты проектируемой системы.

> **Цель данного пункта** — суммировать проведённые исследования, распределив выделенные объектны предметной области по категориям, удобным для завершающего этапа данной стадии — оформление результирующего отчёта.

В подразделах и пунктах данного раздела были выделены следующие категории, на основе которых может проводиться дальнейшее концептуальное проектирование системы:

а) **Пользователи** — ***Преподаватель*** и ***Студент***, являющиеся элементами множества создателей и потребителей информации, функционирующей в ИС ЭЖР;

б) **Инструментальные средства** — ***Клиент-преподавателя***, ***Клиент-студента*** и ***Сервер***, являющиеся объектами проектируемой ИС ЭЖР;

в) **Информационные объекты** — ***Группа-студентов*** и ***ЭЖД***, являющиеся главными целевыми объектами функционирования ИС ЭЖР;

г) **Бизнес-процессы** — ***Занятие*** (главный бизнес-процесс) и ***ВБП*** (Второстепенные Бизнес-процессы), являющиеся операционными элементами деятельности объектов типа ***Пользователи***;

д) **Ограничительные условия** — см. пункт 2.1.3, являющиеся целевыми требованиями к качеству выполнения ***Бизнес-процессов***.

Примечание — Обязательно должны быть выделены бизнес-процессы, относящиеся к категории ВБП, которые будут рассматриваться на стадии концептуального проектирования.

На заключительном этапе формирования отчётности, потребуется упоминание следующих бизнес процессов:

а) операции с информационными объектами типа ***Группа-студентов*** (***ОГС***), включающие создание, модификацию и удаление этих объектов;

б) операции с информационными объектами типа ***ЭЖД*** (***ОЭЖД***), включающие начальное создание объекта, архивацию и восстановление ЭЖД, а также удаление его из системы;

в) операции подключения ***Пользователей*** к главному бизнес процессу ***Занятие*** (***ОПЗ***), а также их авторизация и идентификация;

г) операции взаимодействия ***Преподавателя*** со ***Студентом*** (***ОПС***);

д) операции взаимодействия ***Студента*** с ***Преподавателем*** (***ОСП***).

Примечание — Обязательно должны быть выделены бизнес-процессы, отражающие взаимодействие проектируемой ИС с «внешним миром».

Как уже не раз было отмечено, проектируемая ИС ЭЖР не имеет прямых связей с другими системами. Такая ситуация не является типичной в практике проектирования информационных систем, а порождена условиями выбранной автором постановки задачи. Сделано это специально, чтобы не усложнять учебный материал излишними деталями, которые всегда присутствуют в реальных системах.

В качестве альтернативы, расширяющей постановку задачи ИС ЭЖР, можно рассмотреть случай формирования информационных объектов типа ***Группа-студентов***, используя информацию с портала ТУСУР.

Такое расширение постановки задачи можно предложить для последующей модификации проектируемой ИС, но только, если существующая постановка задачи будет успешно решена студентом Петровым И.В.

2.5 Оформление отчёта по первой стадии проектирования ИС

Примечание — Обычно студент недостаточно ответственно подходит к вопросам своевременного оформления результатов исследования.

Каждая стадия проектирования ИС ЭЖР, согласно ГОСТ 34.601-90 [8], должна завершаться оформлением отчёта о выполненной работе. Кроме того, первая стадия предполагает оформление заявки на продолжение работ, во время которых будет формироваться тактико-техническое задание (ТТЗ) на дальнейшее проектирование и создание системы.

Что касается заявки на ТТЗ, то стандарты ГОСТ серии 34.xxx не регламентируют структуру формата этого документа. Здесь проектировщик (***Исполнитель***) должен следовать требованиям ***Заказчика***. Соответственно, студент Петров И.В. должен руководствоваться требованиями «*Руководителя практики от университета*». Обычно, студентам ТУСУР такая заявка не требуется.

Примечание — При написании отчёта, студент должен особое внимание уделить доказательной базе о необходимости продолжения проектных работ.

2.5.1 Доказательная база на продолжение работ

Сам факт постановки задачи будущим заказчиком системы является весомым аргументом продолжения работ по проектированию ИС.

Доказательная база на продолжение работ строится на основе анализа *целевого назначения* будущей системы. Соответственно необходимо показать, что:

а) выявленные прототипы не обеспечивают целевую функциональность будущей системы;

б) имеется необходимая инфраструктура и инструментальные средства для реализации системы;

в) система может быть спроектирована в приемлемые сроки и с приемлемыми для ***Заказчика*** затратами.

Применительно к учебной задаче, доказательная база продолжения работ может быть следующей:

а) сам факт, что задача рассматривается как учебный пример и может быть элементом УМП учебной дисциплины, уже является достаточным обоснованием для продолжения работ;

б) прототип, в виде web-приложения «*Журнал успеваемости ТУСУР*» на базе системы Moodle, является сложной профессиональной системой, что не позволяет её отнести к категории индивидуальной системы преподавателя;

в) имеется необходимая инфраструктура учебных классов кафедры АСУ и инструментальные средства сервера приложений Apache TomEE, СУБД Apache Derby и системы Eclipse EE, достаточные для реализации всех функций ИС ЭЖР;

г) в проектируемой системе не требуется использование коммерческого ПО и потребность каких-либо лицензионных отчислений.

Примечание — Доказательная база на продолжение работ должна опираться только на результаты проведённых исследований и не допускать предположительных фантазий студента.

У студента могут быть какие-то особые идеи, ноу-хау или иные соображения, которые он хотел бы опубликовать. Этого не следует делать, чтобы на последующих этапах проектирования от них не пришлось отказываться.

2.5.2 Структура отчёта по первой стадии проекта

Отчёт по первой стадии проектирования студент должен рассматривать как расширенное и обоснованное описание задачи, поставленной научным руководителем и утверждённым на уровне кафедры.

Общие требования к содержанию отчётных документов проектируемых АС изложены в международном стандарте РД 50-34.698-90 [33]. Приложение 1 этого стандарта рекомендует следующую структуру основной части отчёта в виде отдельных разделов:

1) характеристика объекта и результатов его функционирования;
2) описание существующей информационной системы;
3) описание недостатков существующей информационной системы;
4) обоснование необходимости совершенствования информационной системы объекта;
5) цели, критерии и ограничения создания АС;
6) функции и задачи создаваемой АС;
7) выводы и предложения.

Примечание — Структура и правила оформления отчёта должны соответствовать требованиям ГОСТ 7.32-2017 [34].

Согласно этому стандарту структурными элементами отчёта о НИР являются:

а) титульный лист;
б) список исполнителей;
в) реферат;
г) содержание;
д) термины и определения;
е) перечень сокращений и обозначений;
ж) введение;
з) основная часть отчёта о НИР;
и) заключение;
к) список использованных источников;
л) приложения.

Полужирным шрифтом выделены обязательные структурные элементы отчёта.

Примечание — **Главное проектное требование** к содержанию отчёта по первой стадии проектных работ — наличие достаточной информации для выполнения следующей стадии: «*Разработка концепции ИС*».

Обычно результаты работ по первой стадии проектирования содержат большое количество разрозненного описательного материала. В любом случае такой материал должен входить в состав отчёта, хотя бы в качестве приложений. В отчёт не следует помещать конкретные проектные решения, оставив их на следующую стадию выполнения работ.

Вопросы для самопроверки

1. В чем состоит сходство и отличие понятий, определённых сокращениями ИС и АС?
2. Государственные стандарты какой серии являются определяющими при проектировании ИС?
3. С какой целью проводится описание организационной структуры предприятия, организации или вуза?
4. Чьи требования описываются на первой стадии проектирования ИС?
5. Какую модель проектирования определяет ГОСТ 34.601-90?
6. С какой целью выделяются границы объекта проектирования?
7. Что понимается под термином «Классическое проектирование»?
8. С какой целью проводится описание законченной продукции (изделия) предприятия?
9. Зачем необходимо описание прототипов ИС?
10. Какие бизнес-модели необходимы для описания предметной области ИС?
11. Какие узлы предметной области необходимо выделить, чтобы описать архитектуру ИС?
12. В чем состоит разница между внешними и внутренними связями ИС?
13. С какой целью формируется доказательная база проекта?
14. Зачем необходимо описание бизнес-процессов проектируемой ИС?
15. Какой ГОСТ определяет требования к отчёту по первой стадии проектирования ИС?
16. Какой ГОСТ определяет требования к отчёту по НИР?
17. С какой целью оформляется отчёт по первой стадии проекта?
18. Какой документ требуется дополнительно к отчёту по первой стадии проектирования ИС?
19. К каки последствиям приводит нарушение границ выделенной предметной области?
20. На какие цели ориентирована учебная задача, сформулированная в данной главе учебного пособия?

3 КОНЦЕПТУАЛЬНОЕ ПРОЕКТИРОВАНИЕ ИС

Концептуальное проектирование ИС — это вторая стадия проектных работ, обеспечивающая перевод требований пользователей ИС на язык её разработчиков.

Как уже было отмечено ранее, в начале 1970-х годов стала пропагандироваться методика SADT и разрабатываться технологии интегрированного автоматизированного производства *ICAM*.

SADT (***Structured Analysis and Design Technique***) — методика структурного анализа и проектирования, пропагандируемая Дугласом Т. Россом.

ICAM (***Integrated Computer-Aided Manufacturing***) — программа разработки инструментов, методов и процессов для поддержки интеграции производства, запущенная в 1976 году департаментом ВВС США.

В конечном итоге проявленные усилия привели к созданию серии стандартов IDEF (ICAM Definitions), которые и определили основы различных методологий.

Что касается *тематики концептуального проектирования*, то сам этот термин имеет достаточно широкое толкование и применяется не только в технических системах. В частности, он используется при проектировании баз данных, что соответственно предполагает инфологическое проектирование их структуры и естественным образом затрагивает технологию построения семантической модели предметной области.

Основная задача данного раздела — изучение базовых методик, обеспечивающих процесс реализации второй стадии проектирования ИС.

Для успешной реализации поставленной задачи студенту необходимо:

а) *перечитать* теоретическую часть подраздела 1.2, где описаны общепринятые стандарты, поддерживающие парадигму информационного подхода, а также раскрыты различия между функциональным и объектным аспектами описания предметной области;

б) *повторить* содержимое пункта 1.3.3, в котором перечислены этапы работ по второй стадии проектирования АС;

в) *руководствоваться* результатами проектных решений раздела 2, содержащих исходные данные для стадии концептуального проектирования.

Напомню, что согласно ГОСТ 34.601-90 [8], вторая стадия проектирования обозначается как «*Разработка концепции АС*», а в данном учебном пособии мы интерпретируем её как «*Концептуальное проектирование ИС*».

Результатом данной стадии должен быть отчёт, который в общем случае представляет собой «*Тактико-техническое Задание*» (***ТТЗ***) и на основе которого формируется, подписывается и утверждается документ «*Техническое задание*» (***ТЗ***).

Исходя из заявленной целевой установки, структура данной главы разделена на пять частей:

а) подраздел 3.1 описывает стандартные модели, предназначенные для концептуального проектирования систем;

б) подраздел 3.2 посвящён изучению функциональной модели стандарта IDEF0;

в) в подразделе 3.3 рассматривается документальное оформление результатов второй и третьей стадий проектирования;

г) в подразделе 3.4 изучается процессная модель стандарта IDEF3;

д) в подразделе 3.5 представлен учебный материал по модели DFD.

3.1 Функциональное моделирование

Ещё раз отметим, что ***в начале 1970-х годов*** Дуглас Т. Росс стал пропагандировать методику «*Структурный анализ и проектирование*» (***SADT***, *Structured Analysis and Design Technique*), а ***в конце 1970-х годов***, департамент военно-воздушных сил США реализовал идею SADT посредством стандартов ***IDEF*** (*ICAM Definitions*), где первой, если не главной, стала идея концепции функционального моделирования (***IDEF0****, Function Modeling*).

Примечание — Важность концепции функционального моделирования состоит в том, что общие характеристики всех систем всегда оцениваются по набору функций, которые проектируемая система может исполнять.

Было бы удивительным рекомендовать для использования систему, которая не содержит некоторых важных функций. Перечень функций, пусть и неявный, содержится в любой постановке задачи. Более того, перечень функций системы обязательно перечисляется в техническом задании (ТЗ) и по этому перечню определяются результаты проектирования и реализации системы.

Примечание — Процесс проектирования любой системы — это процесс анализа её свойств от общего к частному (сверху-вниз).

Инструментарий стандарта IDEF0 идеально подходит для выполнения проектных работ на стадии концептуального проектирования ИС и включён в госстандарт России, как ***ГОСТ Р 50.1.028-2001*** [16].

Действительно:

а) IDEF0 разработан на основе методики SADT и опирается на обобщённое понятие функции, преобразующей входные информационные или материальные объектные потоки в выходные потоки, учитывая при этом управляющие ограничения и «механизмы» необходимые для этого функционального преобразования;

б) IDEF0 допускает декомпозицию функций до нужного уровня детализации, что позволяет удовлетворить основные требования специалистов, которые будут заниматься реализацией системы;

в) IDEF0 позволяет проецировать иерархию функций модели на организационную структуру АС, что уже было показано ранее на рисунке 1.9 первого раздела (см. пункт 1.5.2).

Примечание — Инструментарий стандарта IDEF0 имеет и свои недостатки, которые устраняются другими моделями, такими как IDEF3 и DFD. В подразделах 3.4 и 3.5 они рассматриваются более подробно.

3.1.1 Функциональное моделирование бизнес-процессов

Функция — это не процесс. Функция обязана закончить свою работу и вернуть результат.

Понятие процесса (бизнес-процесса) является более сложным и комплексным, чем понятие функции. И прежде чем рассматривать моделирование функциями, кратко рассмотрим *методологии процессного моделирования*.

Процесс (бизнес-процесс) — это совокупность различных действий, работ или мероприятий человека или группы лиц, нацеленное на получение некоторого положительного результата для действующих акторов.

Привлекательность понятия бизнес-процесс для целей проектирования различных систем, включая ИС или АС, — вполне обоснована:

а) *интуитивная понятность* для широкого круга специалистов и различных лиц, которым кажется, что — это понятно;

б) *возможность различных уровней описательной абстракции*, позволяющей в будущем конкретизировать такие описания;

в) *широкие возможности по манипулированию различными абстракциями*, скрывая за перманентностью самого процесса конкретизацию будущего результата.

В теоретическом плане выделяются *три целевых вида* процессных моделей:

а) ***Управляющие*** — управляют функционированием систем, подобно таким как «*Корпоративное управление*» или «*Стратегическое управление*»;

б) ***Операционные*** — составляют основной бизнес компании: «*Снабжение*», «Маркетинг» и другие подобные операционные действия;

в) ***Поддерживающие*** — обслуживают основной бизнес-процесс: «*Бухгалтерский учёт*», «*Техническая поддержка*» и другие.

Примечание — **В частности**, во втором разделе учебного пособия, анализируя требования к бизнес-процессам в нашей учебной задаче (см. пункт 2.4.2), были выделены:
а) ***Занятие*** *— главный бизнес-процесс*;
б) ***ВБП*** *— второстепенные бизнес-процессы*, которые, в пункте 2.4.3, уточнены как: ***ОГС*** (операции с объектами ***Группа-студентов***), ***ОЭЖД*** (операции с электронным журналом дисциплины), ***ОПЗ*** (операции подключения преподавателей к бизнес-процессу ***Занятие***), ***ОПС*** (операции взаимодействия преподаватель-студент) и ***ОСП*** (операции взаимодействия студент-преподаватель).

Концепция процессного проектирования была подкреплена рядом модельных представлений, таких как:

а) **BPM** (*Business Process Management*) — управление бизнес-процессами, которое поддерживается набором инструментальных средств, таким как ***BPMS/BPMT*** (*Business Process Management System/Tool*);

б) **BPMN** (*Business Process Model and Notation*) — нотация и модель бизнес-процессов, разработанная группой ***Business Process Management Initiative***, поддерживается ***OMG*** (*Object Management Group*) и обеспечивает описание бизнес-моделей на языке XML;

в) **BPEL** (*Business Process Execution Language*) — язык формального описания бизнес-процессов и протоколов их взаимодействия между собой на языке XML;

г) **EPC** (*Event-driven Process Chain*) — диаграммы описания событийных цепочек бизнес-процессов, широко применяемых в системах планирования ресурсов предприятий (***ERP***);

д) **IDEF3** (*Integrated DEFinition for Process Description Capture Method*) — стандарт на методологию моделирования и документирования бизнес-процессов, происходящих в системе.

Особую популярность концепция процессного проектирования получила по результатам работ Августа-Вильгельма Шеера, разработавшего методологию моделирования бизнес-процессов предприятий и реализовавшего её в инструментальной системе ARIS (*Architecture of Integrated Information Systems, 1994 год*). Желающим познакомиться с системой ARIS и другими подходами к проектированию рекомендуется прочесть обзорную статью [35].

Примечание — **Официальной методологией** структурного анализа и проектирования (***SADT***) является функциональный подход, описываемый стандартом ***IDEF0***.

Понятие функции стандарта IDEF0 является более простым и конкретным конструктивом, чем понятие бизнес-процесса, описанного выше.

Функция — *это преобразователь* входных материальных или информационных объектов, предоставляющая на выходе достаточно конкретизированные объекты, которые могут быть использованы в дальнейшем для других функциональных преобразований. Само преобразование также должно иметь место. Если нет преобразования, то нет и функции.

Фактически, стандарт IDEF0 — это официальное описание методологии SADT, где функции являются основными элементами проекта, а бизнес-процессы описываются как различные комбинации выделенных функций.

Примечание — **Функциональный подход**, основанный на стандарте IDEF0, обеспечивает более чёткое проектное толкование бизнес-процессов, устраняя описанные выше негативные явления, присущие процессной методике проектирования.

В отличие от многих других стандартов, стандарт IDEF0 [16] имеет достаточно полное и подробное методологическое обеспечение, ориентированное на многоуровневое моделирование функций *таких систем как предприятие*. Методология IDEF0 даёт собственное более конструктивное толкование понятию «Бизнес-процесс», трактуя его как *конкретный уровень организационно-технической структуры предприятия* (см. учебный материал первого раздела: пункт 1.5.2 и рисунок 1.9):

а) ***Бизнес-процесс*** — совокупность ***Операций***;

б) ***Бизнес-процесс*** — часть более общей производственной функции — ***Деятельность***.

Рекомендации по стандартизации Р 50.1.028-2001 [16], разработанные комитетом ТК 431 «*CALS-технологии*», демонстрируют полный набор *структурных метамоделей*, которые показаны на рисунках 3.1 — 3.4.

Рисунок 3.1 — Метамодель деятельности предприятия [16]

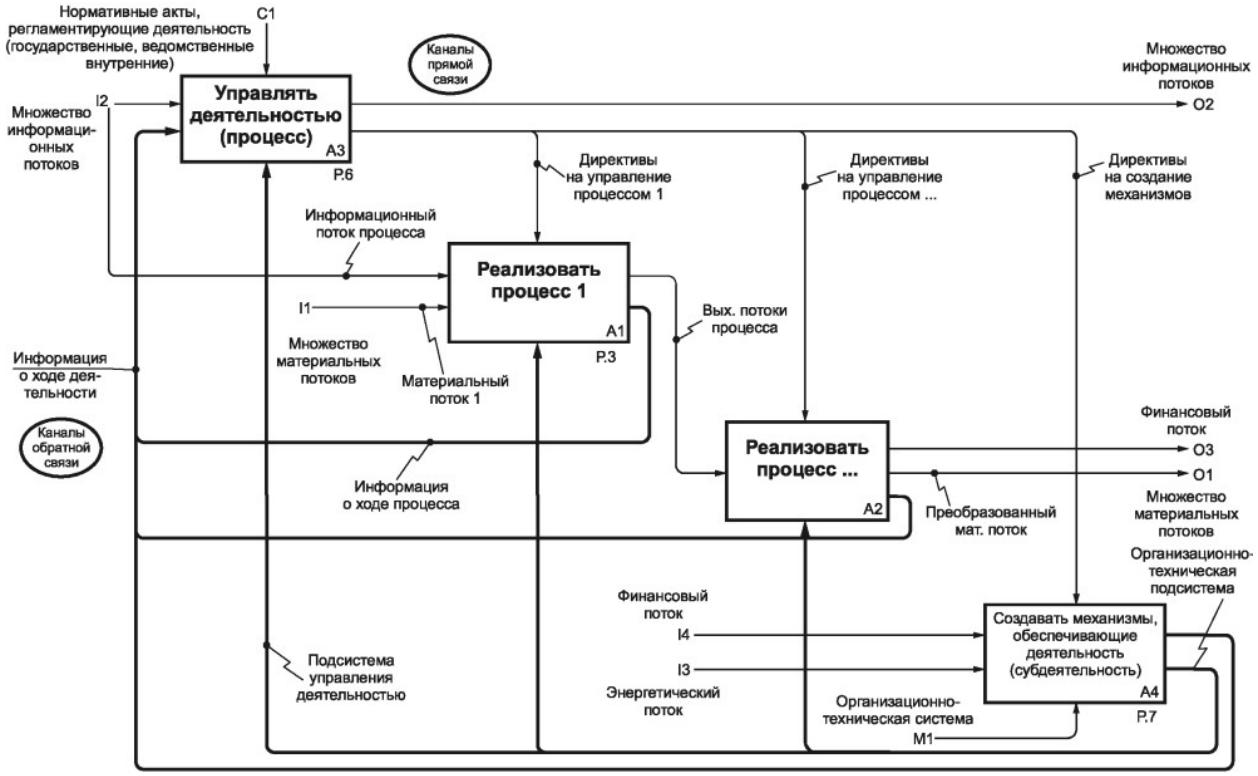


Рисунок 3.2 — Метамодель деятельности предприятия в виде процессов [16]

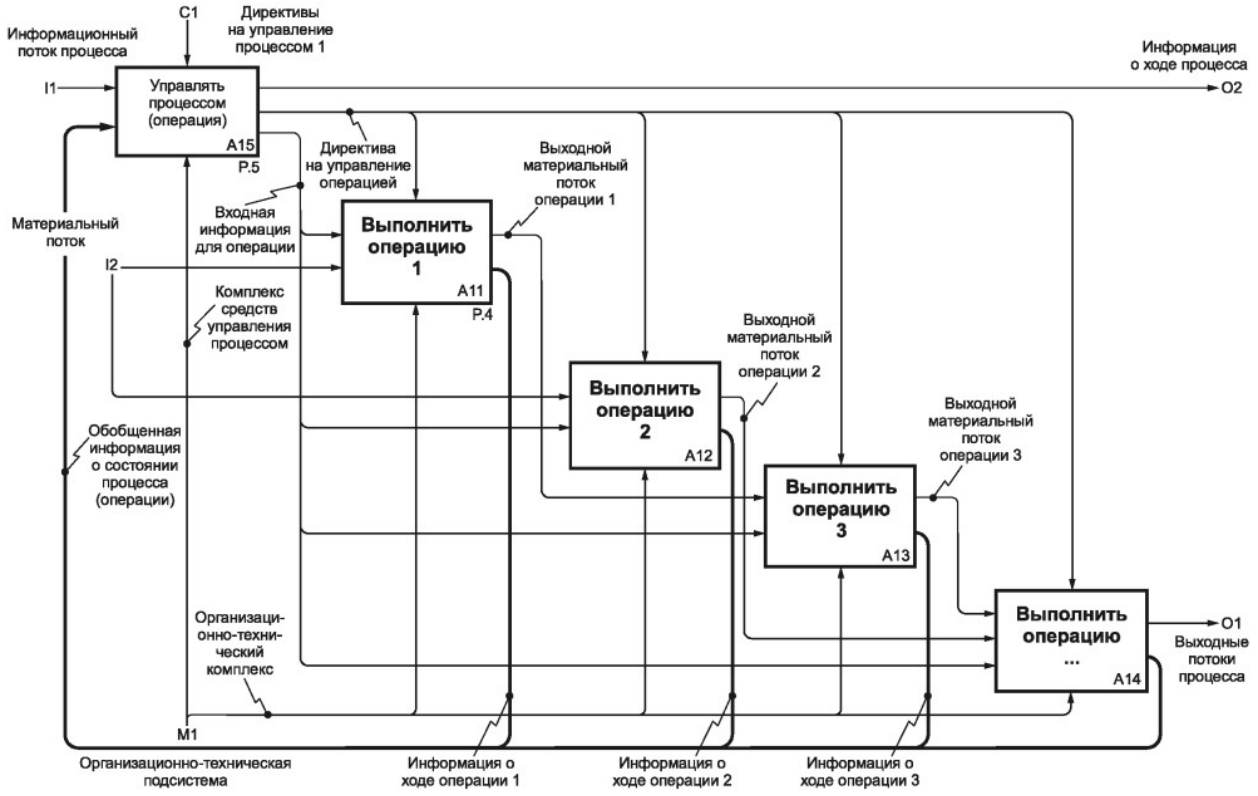


Рисунок 3.3 — Метамодель процесса предприятия в виде операций [16]

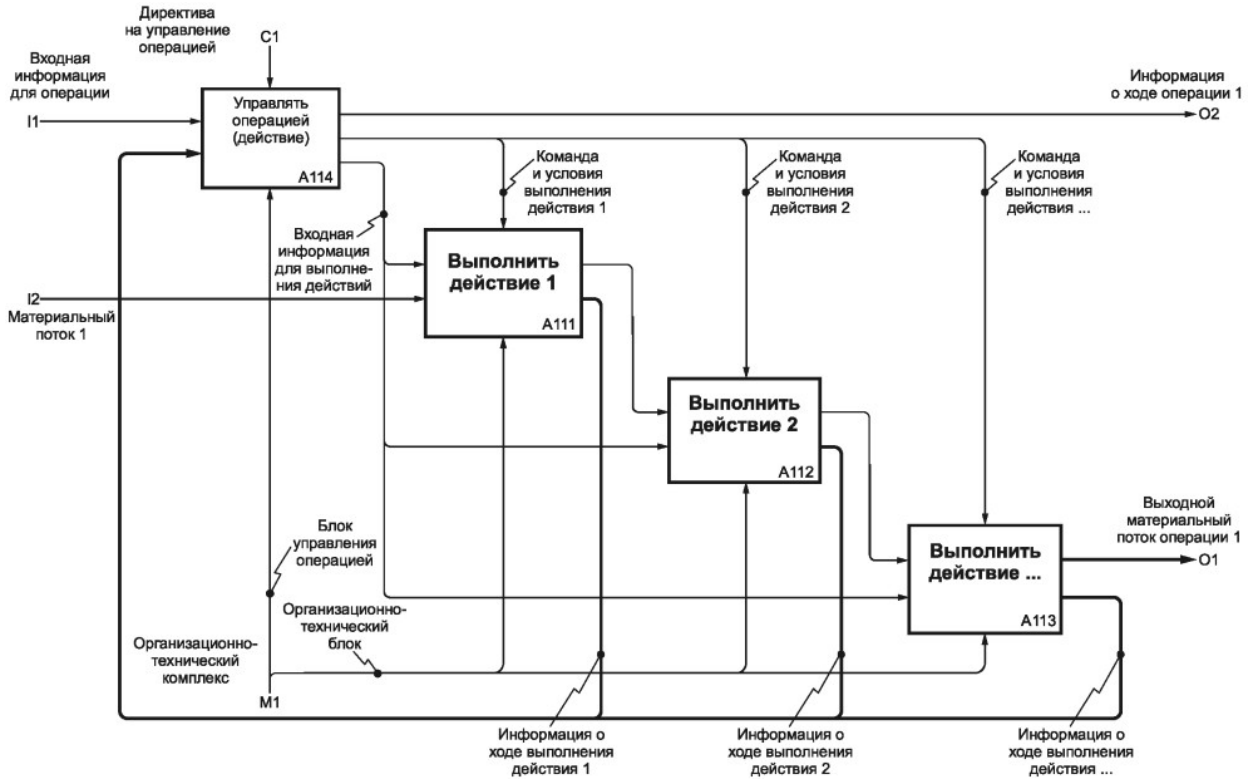


Рисунок 3.4 — Метамодель операции в виде действий [16]

Оставим подробную интерпретацию этих метамоделей до более подробного изучения синтаксиса и семантики методологии IDEF0, а сейчас обсудим место и значимость функциональных методологий на различных стадиях проектирования и создания ИС.

3.1.2 Применимость функциональных моделей к стадиям проектирования ИС

Общее понятие «*Концептуальное проектирование*» употребляется в двух семантических аспектах:

1) как *вторая стадия проектирования ИС* (АС), определённая стандартом ГОСТ 34.601-90 [8];
2) как *абстрактное, но достаточно формализованное описание системы*, достаточное для последующего успешного её проектирования и реализации.

Первая интерпретация — вполне понятна. Она требует использования известных и понятных большинству экспертов формализованных моделей для написания ТТЗ (тактико-технического задания), достаточного для написания и последующего утверждения ТЗ (технического задания).

Вторая интерпретация требует ответа на вопрос: «*Какие модели и на каких стадиях, определённых ГОСТ 34.601-90 [8], должны быть использованы, чтобы успешно завершить создание системы ИС (АС)?*».

Хотя ответ на подобный вопрос всегда субъективен и содержит большую степень неопределённости, практика проектирования даёт следующий вполне приемлемый ответ.

Первая стадия создания ИС — «Формирование требований к АС» не предполагает обязательного представления каких-либо формализованных моделей, поэтому проектировщик берёт любые из них, по своему усмотрению.

Вторая стадия создания ИС ориентирована на конкурсное создание ТТЗ и последующее написание, согласование и утверждение ТЗ, поэтому основной моделью является структурное моделирование по стандарту IDEF0.

Эскизный проект (четвёртая стадия) — использование полного набора имеющихся технологий моделирования, предполагающего анализ нескольких вариантов решений.

Технический проект (пятая стадия) предполагает возможную разработку ЧТЗ (частных технических заданий), что может потребовать использование моделей стандарта IDEF0.

Рабочая документация (шестая стадия) завершает реализацию всех программных частей ИС по уже созданным концептуальным решениям и не ориентирована на построение новых концепций.

Приведённые выше оценки применимости функциональных методологий сведены в представленную ниже таблицу 3.1.

Таблица 3.1 — Применимость функциональных моделей на разных стадиях создания ИС

Стадия создания ИС	***Применимость функциональных моделей***
1. Формирование требований к АС	Любые модели без ограничений.
2. Разработка концепции АС	IDEF0 — основная модель проектирования. IDEF3 — по мере необходимости. DFD — крайне редко.
3. Техническое задание	Тоже, что и на стадии 2
4. Эскизный проект	IDEF0 — основная модель проектирования. IDEF3 — основная модель проектирования. DFD — основная модель проектирования.
5. Технический проект	IDEF0 — по мере необходимости. IDEF3 — основная модель проектирования. DFD — основная модель проектирования.
6. Рабочая документация 7. Ввод в действие 8. Сопровождение АС	Не используются

Очевидно, что конкретная специфика проектируемых систем может изменить представленные оценки значимости тех или иных методологий. В таких случаях следует использовать более конкретные отраслевые стандарты и рекомендации.

Далее, учитывая особую роль стандарта IDEF0, изучим его применение в двух практических аспектах:

а) *как методологию функционального моделирования* (проектирования) систем, главенствующую над методологией объектного проектирования;

б) *как часть методологии концептуального проектирования* ИС, на примере учебной задачи, требования к которой уже были описаны в предыдущем разделе данного пособия.

3.2 Методология стандарта IDEF0

Базовая парадигма данного подраздела — модели стандарта IDEF0 являются основой методологии функционального подхода по проектированию ИС уровня предприятия.

В плане методологии, функциональный подход прежде всего противопоставляется объектному подходу:

а) ***функциональный подход*** — методология концептуального проектирования систем;

б) ***объектный подход*** — методология реализации систем.

Хотя функциональный подход и использует такие понятия как объекты и потоки объектов, в его основе лежит понятие функции, на что обращалось внимание как в первом разделе данного пособия, так и в учебном материале данного раздела.

> **Учебная цель данного подраздела** — детальное изучение методологии стандарта IDEF0 в объёме, достаточном для самостоятельного применения студентами полученных знаний на всех стадиях проектирования ИС.

Базовым источником изучаемого здесь учебного материала являются Рекомендации по стандартизации Р 50.1.028-2001 [16], обязательные для изучения каждым студентом.

Формальным контекстом, в рамках которого осуществляется реальное проектирование ИС, является *стандартный состав проектной группы*.

3.2.1 Стандартный состав проектной группы

Реальная проектная работа связана с подготовкой множества различных документов и выполняется группой специалистов разной квалификации, что наглядно показано на рисунке 3.5.

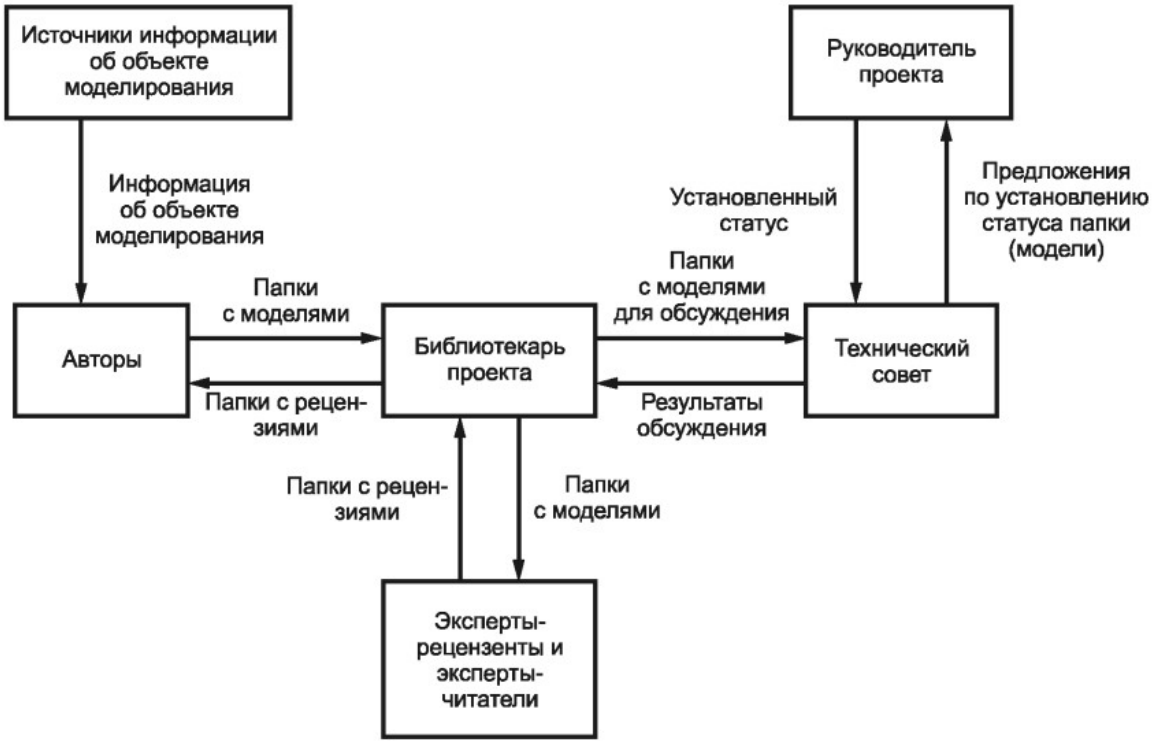


Рисунок 3.5 — Рекомендуемый стандартный состав проектной группы [16]

Руководитель проекта является *административным лицом*, которое полностью отвечает за результат проектирования, состав и взаимодействие исполнителей.

Авторы (автор) — являются *основными исполнителями*, создающими модели IDEF0 на основе источников информации об объекте моделирования.

Библиотекарь проекта — *техническое лицо*, отвечающее за сохранность результатов проектирования и взаимодействие между *авторами*, *техническим советом* и *экспертами*.

Технический совет — *коллективный орган*, назначаемый руководителем проекта и обеспечивающий коллективные согласовательные решения как по всему проекту, так и по отдельным его частям.

Эксперты — лица, обладающие специальными знаниями об объекте моделирования и высказывающие полезные критические замечания.

Примечание — **Рекомендуемый состав проектной группы** легко масштабируется от её минимального необходимого состава до уровня проектной организации и сам этот состав может рассматриваться как объект автоматизации.

Таким образом, реальная проектная деятельность по методологии IDEF0 выполняется коллективом специалистов, в котором их функции и ответственность разделены в зависимости от умения и накопленного опыта.

В частности, руководитель практики от кафедры может конкретизировать схему рисунка 3.5 и отразить её в собственных методических рекомендациях по проведению производственной практики.

Организация учебного процесса студентов хоть и ориентирована на групповое обучение, но оценивается индивидуально. Поэтому студент выполняет и оформляет свой личный проект, обусловленный *конкретной проектной задачей*.

Технология создания IDEF0-моделей предполагает построение множества различных диаграмм и другой документации, которая явно функционирует в распределённой системе взаимодействующих субъектов. В этом отношении, автоматизация работы проектной группы может рассматриваться как задача на проектирование ИС.

3.2.2 Синтаксис и семантика языка IDEF0

Примечание — **Синтаксис языка IDEF0** определяется его компонентами — *блоки*, *стрелки*, *диаграммы* и *правила*, *дополненные связями* между ними.

Синтаксис языка IDEF0 полностью описан в пятом разделе документа [16].

Блок — *прямоугольный графический компонент* (см. рисунок 3.6), описывающий функцию *активным глаголом* или *глагольным оборотом*. В правом нижнем углу блока указывается его целочисленный идентифицирующий номер.

Стрелка — *графический именованный компонент*, обозначающий информационные или материальные объекты, состоящий из одного или последовательности непрерывных прямолинейных горизонтальных и вертикальных сегментов, сопряжённых дугами в 90 градусов, имеющий наконечник и соединяющий различные блоки. Допускаются стрелки присоединённые к блоку только одним концом.

Диаграмма — *графический компонент языка IDEF0*, размещённый на отдельном листе документа и содержащий один или несколько блоков, соединённых стрелками.

Правила — *определения,* указывающие как следует применять компоненты языка IDEF0.

Рисунок 3.6 — Пример определения блока модели IDEF0 [16]

Примечание — **Семантика языка IDEF0** определяет содержание (значение) и интерпретацию указанных выше синтаксических компонент языка.

Для раскрытия семантики языка в стандарте IDEF0 широко используется технология метаопределений, когда компоненты описываются словами, которые запрещены к употреблению в самих моделях. Покажем это конкретными примерами.

Блоки именуются *активными глагольными оборотами*, в которых не употребляется слово функция. Источник [16] прямо рекомендует примеры таких имён, представленных на рисунке 3.7.

производить детали	планировать ресурсы	наблюдать
наблюдать за выполнением	проектировать систему	эксплуатировать
разработать детальные чертежи	изготовить компонент	проверять деталь

Рисунок 3.7 — Примеры глагольных оборотов для именования блоков [16]

Стрелки, называемые также *дугами* или *интерфейсными дугами*, имеют гораздо большее разнообразие, что показано на рисунке 3.8.

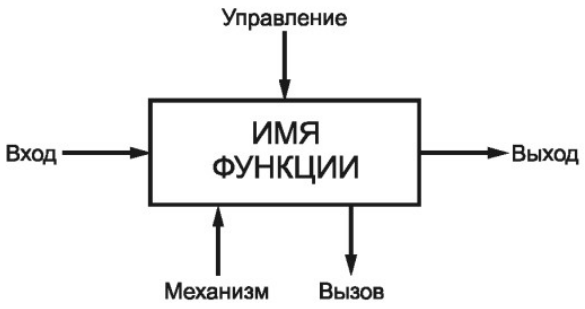


Рисунок 3.8 — Метаопределения стрелок модели IDEF0 [16]

Слова метаопределений стрелок также *не должны присутствовать в их наименованиях*. Источник [16] рекомендует использовать обороты, которые подобны примерам, показанным на рисунке 3.9.

Спецификации	Отчет об испытаниях	Бюджет
Конструкторские требования	Конструкция детали	Директива
Инженер-конструктор	Плата в сборе	Требования

Рисунок 3.9 — Примеры именования стрелок в моделях IDEF0 [16]

В целом, комбинированный пример отдельного блока показан на рисунке 3.10.

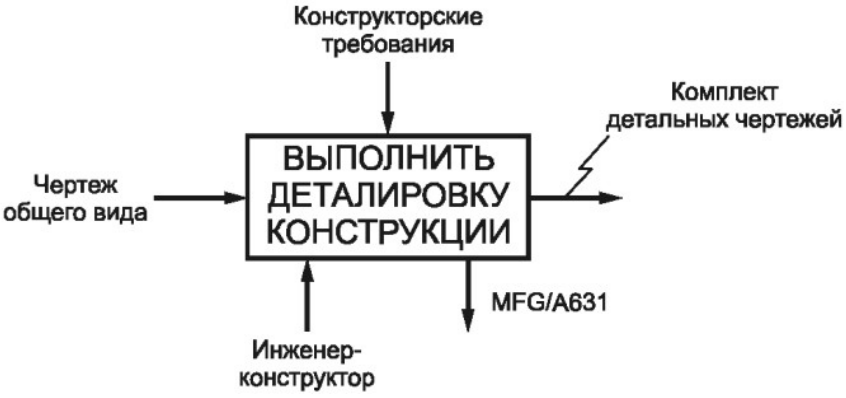


Рисунок 3.10 — Комбинированный пример блока со стрелками [16]

Правила — требования к именованию блоков и стрелок. Они дополнительно раскрывают смысловые значения *метаопределений стрелок*:

а) ***входные стрелки*** должны касаться блоков с левой стороны;

б) ***выходные стрелки*** должны выходить из блоков с правой стороны;

в) ***стрелки управления*** должны подходить и касаться блоков в верхней их части;

г) ***стрелки механизмов*** должны касаться блоков в нижней его части, причём *стрелка вызова* считается частью стрелки механизма;

д) если имеются сомнения к какой стрелке относится метка её имени, то эта неопределённость устраняется молнеобразной линией, как показано на рисунке 3.10.

Диаграммы — главные компоненты модели IDEF0. Они содержат блоки, которые в общем случае соотносятся с функциями одного уровня: *деятельность*, *процесс*, *операция* или *действие* (см. функции предоставленные на рисунке 1.9 в пункте 1.5.2, а также диаграммы с метаопределениями, представленными на рисунках 3.1 - 3.4 в пункте 3.1.1).

> Примечание — В целом, синтаксис и семантика модели IDEF0 — достаточно просты и ориентированы на однозначное понимание как разработчиками диаграмм, так и специалистами, которые читают и используют эти диаграммы.

К сожалению студенты, использующие модели IDEF0, допускают множество грубых ошибок, вызванных различными причинами. Автор данного пособия считает, что основная причина ошибок — *широкое использование учащимися объектных моделей*, которые при плохой подготовке студента не позволяют им достичь нужного понимания предмета. Поэтому дальнейшее *изложение учебного материала концентрируется на анализе ошибок*, часто допускаемых неопытным проектировщиком.

3.2.3 Контекстные диаграммы IDEF0

> Каждая модель IDEF0 должна иметь контекстную диаграмму ***A-0*** (*А минус ноль*).

Практика показывает, что большинство студентов игнорируют построение контекстной диаграммы ***A-0***, сразу представляя некоторую декомпозицию системы из нескольких блоков. *Это является грубой ошибкой*, потому что назначение контекстной диаграммы — *определить и зафиксировать границы моделируемой системы от внешней среды*, которая системой не является. В результате, если граница системы определена некачественно, то возникают различные проблемы и неопределённости с доказательной базой последующих проектных решений, а значит и дополнительные проблемы с её реализацией.

Примечание — **В общем случае**, стандартом IDEF0 допускается использование контекстных диаграмм ***A-n***, где ***n*** — количество уровней иерархии модели, в пределах которой рассматривается проектируемая система, но на практике даже не обязательно указывать уровень ***A-1***.

Обязательная контекстная диаграмма A-0 несёт следующую информационную нагрузку:

а) ***единственный блок диаграммы*** определяет *границы моделируемой системы*, концентрируя в названии блока всю функциональность её модели;

б) ***стрелки этой диаграммы*** не принадлежат моделируемой системе, а *отображают связи модели с внешним миром*, полностью ограничивая и её *внешний интерфейс*;

в) ***ЦЕЛЬ:*** — причина создания модели IDEF0, содержащая набор важных свойств, которые эта модель должна отображать;

г) ***ТОЧКА ЗРЕНИЯ:*** — должностное лицо или группа лиц, для которых создаётся или создана модель IDEF0.

Для демонстрации контекстной диаграммы ***A-0*** используем пример из источника [16], который показан на рисунке 3.11 и исполнен в инструментальной системе Ramus Educational [27].

Примечание — **Грубой ошибкой** считается *отсутствие на диаграмме A-0 «ЦЕЛИ:»* или повторение в ней названия функционального блока A0.

Следует помнить, что любая модель, включая IDEF0, только приближённо описывает реально существующую систему, поэтому «ЦЕЛЬ:» *задаёт ту нужную проекцию свойств системы*, которые будут наиболее важными на диаграммах и в пределах которых проводиться последующая декомпозиция контекстной диаграммы. Например, на рисунке 3.11 явно указывается, что последующая декомпозиция A0 будет содержать функции менеджера, возглавляющего «*Бригаду программистов*».

Также должны быть отражены все информационные потоки, присутствующие в процессах создания «Программы», по которым можно будет оценивать трудоёмкость её создания. Обычно, более подробная информация о «ЦЕЛИ:» раскрывается в текстовом документе, поясняющем назначение компонентов модели.

Что касается нашей учебной задачи, рассмотренной в следующем подразделе, то её «ЦЕЛЬ:» — *обеспечить стадию концептуального (структурного) проектирования ИС*.

Примечание — **Грубой ошибкой** считается отсутствие на диаграмме A-0 «ТОЧКИ ЗРЕНИЯ:» или указание в качестве неё позиции автора (проектировщика).

Обычно студент указывает, что это — его точка зрения. Необходимо помнить, что автор (проектировщик) является *исполнителем*, а не *заказчиком модели*.

Точку зрения формулирует заказчик модели, что требует от автора учёта всех его профессиональных особенностей и интересов в получении результата моделирования. В таких условиях необходимо обратиться к организационной структуре предприятия, исследованной на первой стадии проектирования, и указать соответствующее должностное лицо или группу лиц, которые будут оценивать проект.

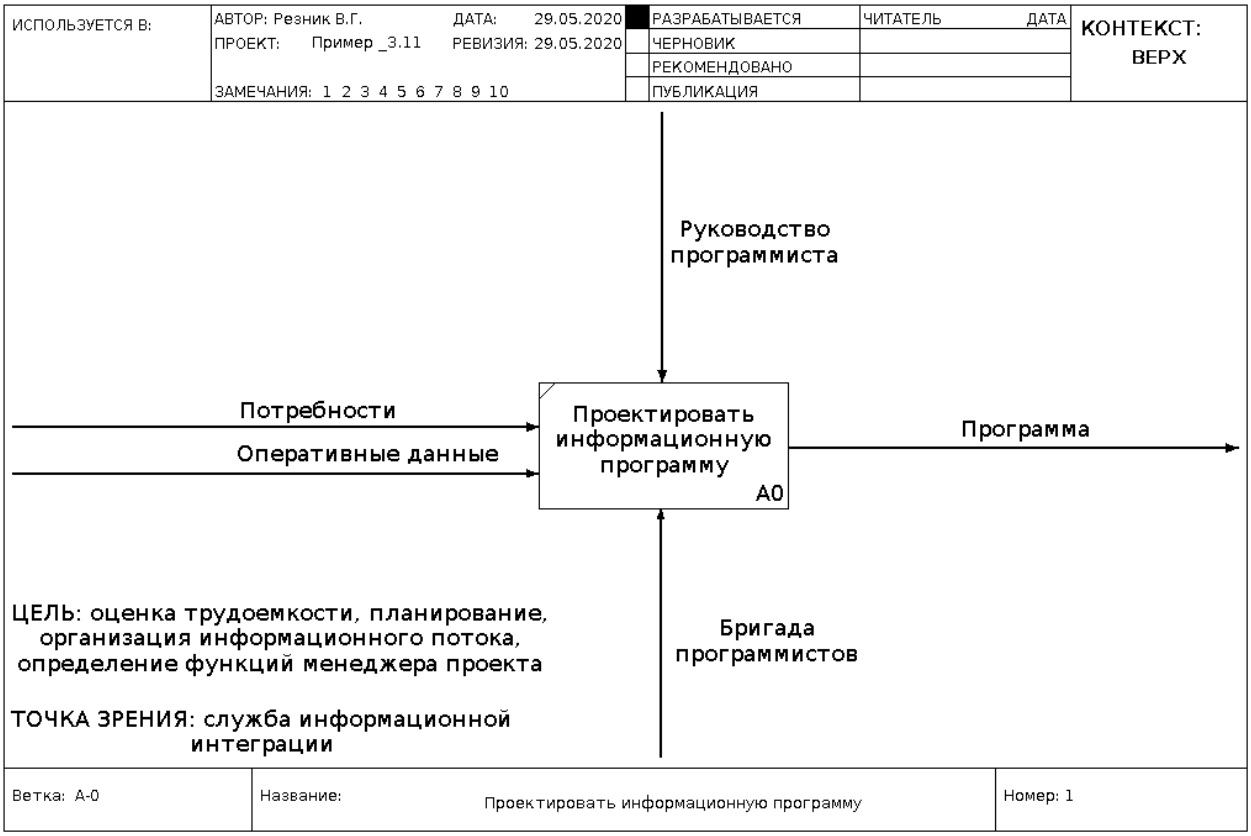


Рисунок 3.11 — Пример контекстной диаграммы [16]

3.2.4 Декомпозиция блоков диаграмм IDEF0

Обязательный состав модели IDEF0 — три типа документов: *набор графических диаграмм*, *глоссарий* и *текст*.

Набор графических диаграмм — основной результат структурного представления модели IDEF0, связанных родительскими и дочерними отношениями.

Глоссарий — документ, содержащий в алфавитном порядке список сокращений или наборов слов, использованных в именах блоков и стрелок, а также раскрывающий их точное смысловое содержание.

Текст — документ, раскрывающий и комментирующий в текстовом виде графические изображения диаграмм.

Декомпозиция блоков диаграмм тесно связана с родительскими и дочерними соотношениями функций модели IDEF0.

Родительская диаграмма — диаграмма, содержащая блок, который подвергнут декомпозиции на текущей диаграмме.

Дочерняя диаграмма — диаграмма, которая содержит декомпозицию рассматриваемого функционального блока.

Общая архитектурная декомпозиция блоков диаграмм модели IDEF0 показана на рисунке 3.12, заимствованном из источника [16].

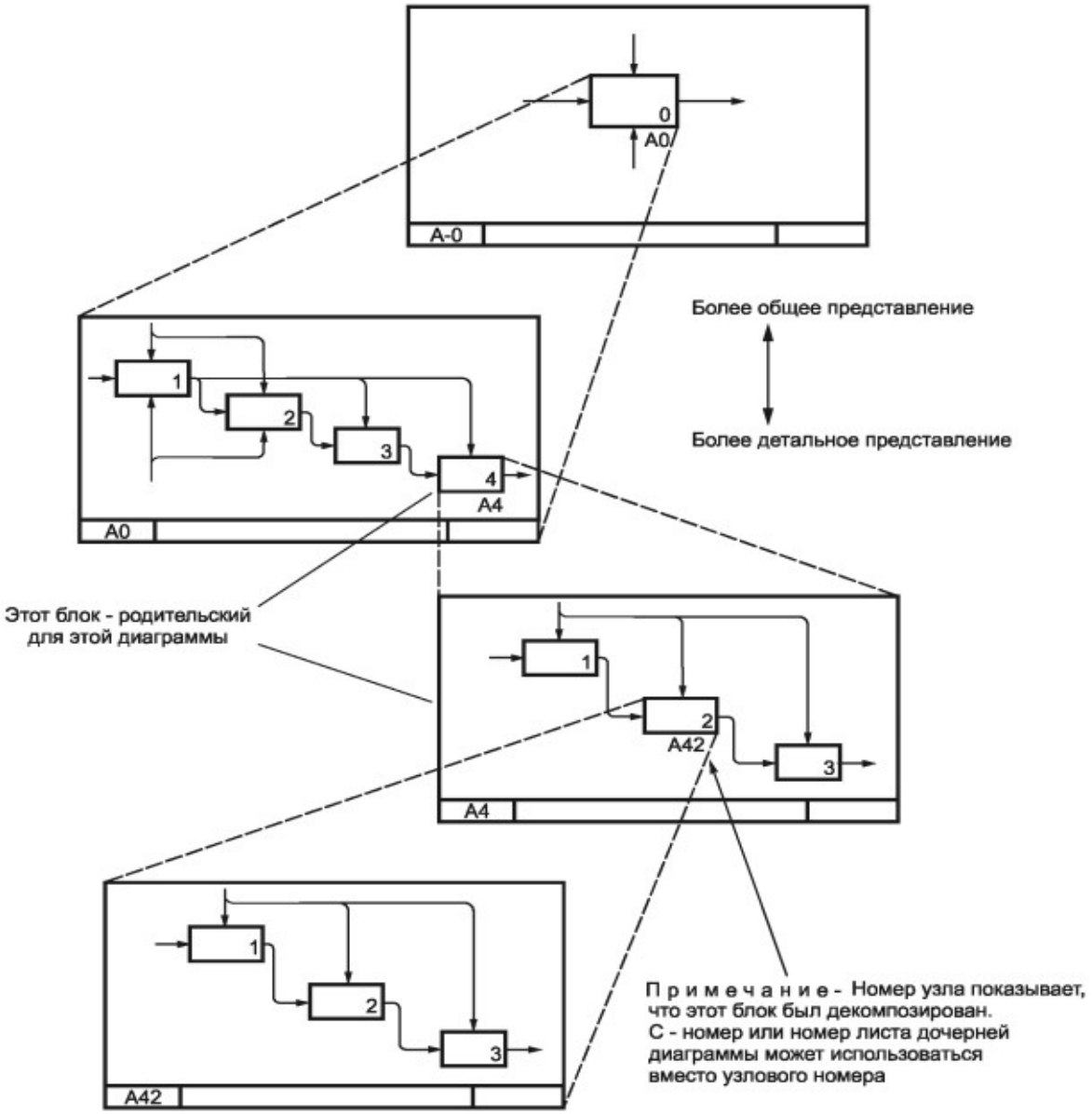


Рисунок 3.12 — Общая архитектурная декомпозиция блоков диаграмм [16]

Примечание — Все наименования блоков в наборе графических диаграмм должны быть различны, а наименования стрелок блоков родительских диаграмм передаются в дочерние диаграммы.

Первой выполняется декомпозиция единственного блока A0 контекстной диаграммы A-0. При этом, каждый отдельный блок диаграммы A0 должен иметь меньшую функциональность, чем сам блок диаграммы A-0, а в сумме они должны точно соответствовать его функциональности. Желательно, чтобы все блоки диаграммы A0 имели один уровень функциональности и имели чёткие проекции на классификацию: *деятельность*, *субдеятельность*, *процесс*, *подпроцесс*, *операция* и *действие*.

Таким образом, декомпозиция блоков обеспечивает построение иерархической модели целевой системы с нужным уровнем детализации, который определяется *не проектировщиком*, а *заказчиком моделируемой системы*.

Применительно к тематике нашей дисциплины, модель IDEF0 должна обеспечить стадию концептуального проектирования целевой ИС, включая её отчётную часть. И чтобы реа-

лизовать *целостность* и *полезную конструктивность* такой модели, используются средства точной *адресации* блоков диаграмм, *ICOM-кодирование* и *туннелирование* их дуг.

Адресация блоков диаграмм — сквозное средство идентификации компонент модели IDEF0, осуществляемое в двух аспектах:

а) ***внутреннее кодирование блоков*** определяется *в пределах самой диаграммы*, начиная с 1 и заканчивая их общим числом; *рекомендуется использовать не более шести блоков*; исключением является блок диаграммы A-0, который кодируется номером 0;

б) ***внешнее кодирование блоков*** определяется *в пределах всей модели*, как это показано на рисунке 3.13.

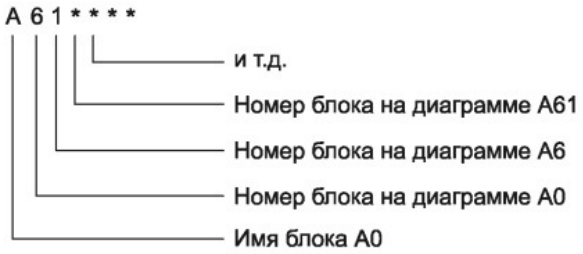


Рисунок 3.13 — Правила кодирования блоков и диаграмм модели IDEF0 [16]

ICOM-кодирование дуг — *дополнительное кодирование имён стрелок* (меток стрелок), связывающее граничные стрелки на дочерней диаграмме со стрелками родительского блока.

Синтаксис ICOM-кода представляет собой одно из буквенных обозначений I, C, O или M, после которого указывается цифра в диапазоне от 1 до максимального количества стрелок на соответствующей стороне родительского блока, увеличивающаяся по правилу: *сверху-вниз* и *слева-направо*.

Семантика используемых буквенных обозначений:

а) ***In*** — ***n***-я сверху стрелка входа (***Input***);

б) ***Cn*** — ***n***-я слева стрелка управления (***Control***);

в) ***On*** — ***n***-я сверху стрелка выхода (***Output***);

г) ***Mn*** — ***n***-я слева стрелка механизма (***Mechanism***).

Туннелирование дуг — дополнительный графический элемент для начала или конца стрелки в виде круглых скобок, имеющий собственную семантику.

Стрелка, обозначающая туннель *в месте подключения к блоку*, — представляет объекты, *не присутствующие* на следующем уровне декомпозиции (см. рисунок 3.14).

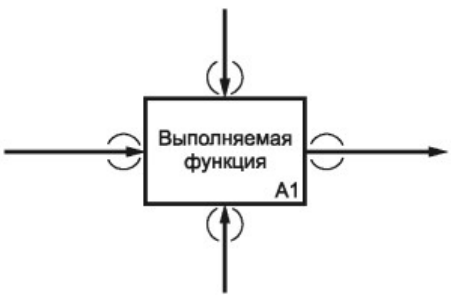


Рисунок 3.14 — Стрелки не присутствующие в декомпозиции блока [16]

Стрелка, обозначающая туннель *на своём свободном конце*, — обозначает представляемые ею объекты, *отсутствующие* на родительской диаграмме (см. рисунок 3.15).

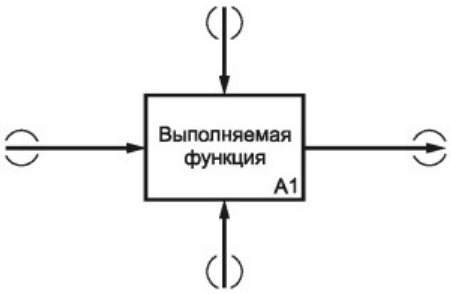


Рисунок 3.15 — Стрелки не присутствующие на родительской диаграмме [16]

Необходимость туннелирования стрелок вызвана необходимостью удаления из диаграммы «*лишних*» компонент, затрудняющих её восприятие.

3.2.5 Отношения блоков на диаграммах модели IDEF0

Семантическая выразительность графического языка IDEF0 определяется *шестью типами отношений* между блоками каждой отдельной диаграммы: *доминирование*, *управление*, *выход-вход*, *обратные связи* по входу и выходу, *выход-механизм*.

Доминирование — *отношение важности (доминирования) блоков своим расположением на диаграмме*. Более важными являются блоки, расположенные выше и левее, например, как это показано на рисунке 3.12 (см. пункт 3.2.4). Это также соответствует первоначальному расположению блоков на диаграмме: «лесенкой» сверху-вниз и слева-направо. В таком же порядке и нумеруются блоки на диаграмме.

Управление — *отношение прямого управляющего воздействия*, когда выход какого-либо блока является управляющим входом на блок с меньшим доминированием (см. рисунок 3.16).

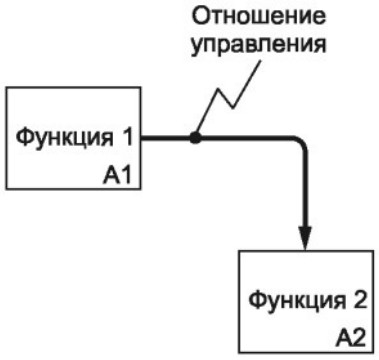


Рисунок 3.16 — Отношение управления [16]

Выход-вход — *отношение прямой передачи объектов между блоками*, когда выход какого-либо блока поступает на вход другого блока с меньшим доминированием (см. рисунок 3.17).

Обратная связь по управлению — *отношение, задающее циклическую обратную связь*, когда выход одного блока поступает на управляющий вход другого блока с большим доминированием (см. рисунок 3.18).

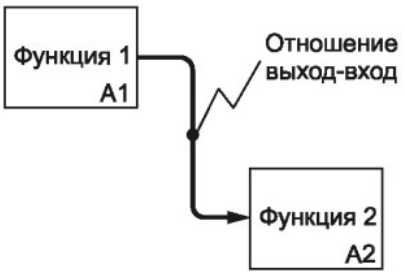


Рисунок 3.17 — Отношение выход-вход [16]

Рисунок 3.18 — Отношение обратной связи по управлению [16]

Обратная связь по входу — *отношение, задающее циклическую обратную связь*, когда выход какого-либо блока поступает на вход другого блока с большим доминированием (см. рисунок 3.19).

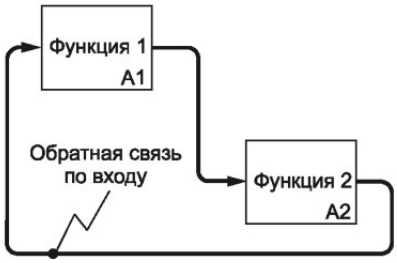


Рисунок 3.19 — Отношение обратной связи по входу [16]

Выход-механизм — наиболее сложное *отношение, задающее каким-либо блоком средства для реализации функции другого блока с меньшим доминированием* (см. рисунок 3.20).

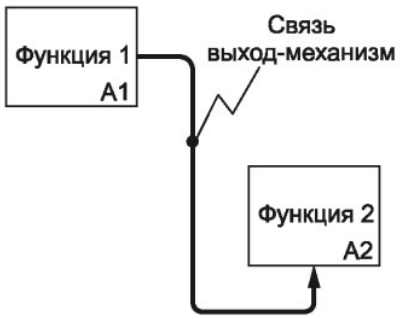


Рисунок 3.20 — Отношение выход-механизм [16]

Примечание — Допускается наличие *циклических обратных связей для одного блока*, как это показано на рисунке 3.21.

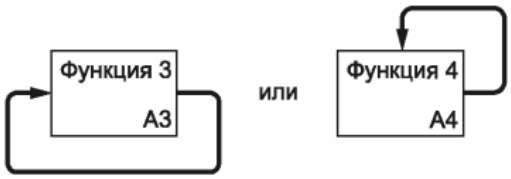


Рисунок 3.21 — Примеры двух видов циклических обратных связей для одного блока [16]

Примечание — В общем случае, между блоками допускаются комбинированные связи, включающие различные типы отношений.

Проблема наличия множества отношений между блоками является всегда актуальной, поэтому *рекомендуется на диаграмме размещать не более шести функциональных блоков*.

3.2.6 Документирование модели IDEF0

Официально, модель IDEF0 требует наличия трёх типов документов: *набора графических диаграмм*, *глоссария* и *текста*. Этот набор документов может оформляться как самостоятельно, так и в составе других документальных пакетов, например, в составе ТЗ на разработку ИС. Поскольку восприятие большого количества страниц диаграмм является достаточно затруднительным занятием, источник [16] рекомендует использовать такие дополнительные представления, как *перечень узлов* и *дерево узлов*.

Перечень узлов — *дополнительное средство документирования* части модели IDEF0, представленное в виде форматированного списка. Пример такого представления для некоторой абстрактной модели показан на рисунке 3.22.

A0 Производить продукт
A1 Планировать производство
A11 Выбрать технологию производства
A12 Оценить требуемое время и затраты на производство
A13 Разработать производственные планы
A14 Разработать план вспомогательных действий
A2 Разрабатывать и управлять графиком выпуска и ресурсами
A21 Разработать основной график
A22 Разработать график координации работ
A23 Оценивать затраты и приобретать ресурсы
A24 Следить за выполнением графика и расходом ресурсов
A3 Планировать выпуск продукции

Рисунок 3.22 — Пример перечня узлов [16]

Обратите внимание, что *перечень узлов* легко может быть представлен в электронном виде, например, в формате файла XML, в формате файла JSON или перечня пунктов меню некоторого интерфейса пользовательского приложения.

Дерево узлов — *дополнительное средство документирования* части модели IDEF0, представленное в графическом виде, например, как это показано для той же абстрактной модели на рисунке 3.23.

Соответствующее электронное представление дерева узлов также может быть использовано в интерфейсах пользовательских приложений.

Рисунок 3.23 — Пример дерева узлов [16]

К дополнительным средствам документирования следует отнести и диаграммы *FEO*.

FEO (***For Exposition Only***) — *диаграммы-иллюстрации*, которые основаны на стандартных диаграммах методологии IDEF0, но имеют различные поясняющие «украшательства», например, различный цвет блоков и стрелок, а также дополнительные обозначения и поясняющие элементы.

В целом, диаграммы FEO не являются обязательной частью документации методологии IDEF0, а используются для публичных презентаций или иных демонстрационных мероприятий.

> Примечание — **Все диаграммы стандарта методологии IDEF0** должны быть исполнены в *чёрно-белом исполнении* и не допускать посторонних форматирований и дополнительных обозначений, кроме упомянутых выше.

Завершая данный подраздел, особо отметим, что отображение стрелок на диаграммах должно строго удовлетворять следующим **общим правилам**:

а) *все отдельные сегменты стрелок* должны иметь только горизонтальное или вертикальное начертание и соединяться *с чёткими закруглениями*;
б) *пересечения различных стрелок* должны осуществляться только под прямым углом и не допускают каких-либо дополнительных начертаний;
в) *допускается ветвление стрелок*, демонстрируя параллельное распространение материальных или информационных объектов;
г) *допускается слияние стрелок*, демонстрируя объединение различных материальных и информационных объектов в один поток.

Таким образом, в данном подразделе изложены все основные синтаксические и семантические аспекты теоретической части методологии IDEF0. Все дополнительные нюансы конкретного применения этой теории, например, ICOM-кодирование граничных стрелок следует уточнять по источнику [16].

Теперь перейдём к практической демонстрации использования методологии IDEF0 для целей концептуального проектирования ИС.

3.3 Концептуальное проектирование учебной задачи

В предыдущем подразделе были кратко описаны теоретические основы построения моделей по стандарту IDEF0.

> **Цель данного подраздела** — показать как теория структурного анализа и проектирования (SADT), представленная стандартом IDEF0, может быть применена для целей концептуального проектирования ИС (вторая стадия проектирования АС).

Целевая задача на проектирование была сформирована во втором разделе данного пособия (см. подраздел 2.1), где также была выполнена *первая стадия* проектирования информационной системы «*Электронный журнал руководителя*» (сокращено — ЭЖР).

> **Цель второй стадии проектирования** — построение концептуальной модели ИС ЭЖР, используя формализацию поставленной задачи и исходных материалов собранных на первой стадии проектирования, достаточной для отчётности в виде ТТЗ и последующей реализации третьей стадии проектирования — написание и утверждение ТЗ на ИС ЭЖР.

Исполнителем второй стадии проектирования назначен абстрактный студент группы 447-1 Петров И.В., *поэтому все претензии по недостаткам проекта — к нему!*

Руководителем второй стадии проектирования является *автор данного пособия*, задача которого исправлять и комментировать ошибки, допущенные ***Исполнителем***.

> Примечание — **Методология концептуального проектирования**, выполняемая по стандарту IDEF0, требует для проектируемой ИС строгой формализации объектов и функций высшего уровня, а также — представления этой формализации в виде контекстной диаграммы A-0, с указанием «ЦЕЛИ:» и «ТОЧКИ ЗРЕНИЯ:» всего проекта.

Для лучшего разделения всей последовательности работ по проектированию, изложим учебный материал в отдельных пунктах, выделив:

а) построение контекстной диаграммы (см. пункт 3.3.1);

б) несколько этапов декомпозиции (см. пункты 3.3.2-3.3.4);

в) формирование ТТЗ учебной задачи (см. пункт 3.3.5);

г) формирование ТЗ учебной задачи (см. пункт 3.3.6).

В качестве основного инструментального средства для построения всех диаграмм проекта будет использоваться система Ramus Educational.

3.3.1 Контекстная диаграмма A-0 учебной задачи

Общие требования к контекстной диаграмме A-0 и демонстрирующий её пример подробно изложены в предыдущем подразделе (см. пункт 3.2.3).

> **Построение контекстной диаграммы A-0 для учебной задачи** требуют определения *границ* функционального блока A0, *имени* этого блока и *организационного уровня* его функции, что необходимо зафиксировать в текстовом документе, являющимся частью модели IDEF0.

Имя проекта (в инструментальной системе Ramus) зафиксируем как «***ИС ЭЖР***».

Организационный уровень блока A0 определим как ***субдеятельность***, поскольку обучение студентов является частью деятельности всего вуза.

Имя блока A0 зафиксируем как «***Выполнить семестровую учебную нагрузку преподавателя***», поскольку все занятия преподаватель выполняет в соответствии с заранее выделенным ему *планом*, включающим: *перечень дисциплин* и *учебных групп*, закреплённых за ним, а также — *расписание по всем видам занятий*, которые включены в эти дисциплины.

Сказанное выше уже определяет *внешний ресурс реализации блока A0* и *внешние управляющие ограничения* этой реализации:

а) ***стрелка механизма*** — *внешний ресурс* в виде объекта «***Преподаватель***»;

б) ***стрелки управления*** — *внешние ограничения* в виде информационных объектов: «***Учебная нагрузка***», «***Расписание занятий***» и «***УМП дисциплин***».

Внешние объекты на входе — это *внешние физические объекты* — «***Студенты на обучение***» и *внешние информационные объекты* — «***Данные о студентах***».

Внешние объекты (стрелки) на выходе — физические объекты «***Обученные студенты***» и «***Студенты без доступа***», плюс внешние информационные объекты — «***Оценки обучения студентов***».

Окончательно, дополняем диаграмму A-0 *целью* и *точкой зрения*:

а) ***ЦЕЛЬ:*** концептуальное проектирование системы ИС ЭЖР;

б) ***ТОЧКА ЗРЕНИЯ:*** заведующий кафедрой АСУ ТУСУР.

На основе проведённой формализации, Петров И.В. построил контекстную диаграмму A-0 в системе Ramus, которая показана на рисунке 3.24.

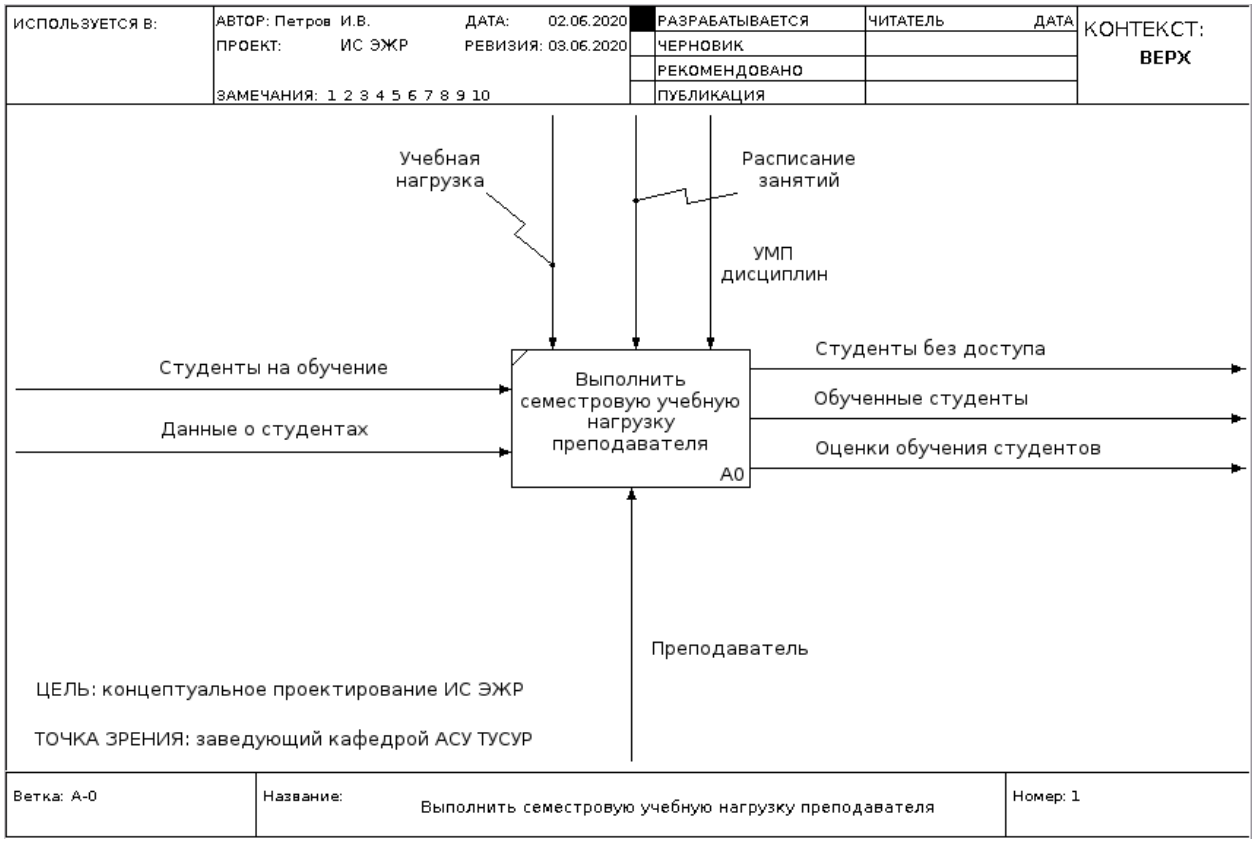


Рисунок 3.24 — Контекстная диаграмма A-0 учебной задачи

Студент Петров И.В. максимально точно и правильно построил контекстную диаграмму A-0, но на текущий момент эта диаграмма вполне понятна только самому студенту, его руководителю и возможно многим преподавателям, имеющим достаточный опыт проектирования информационных систем.

Примечание — **Высокий уровень абстракции** семантических значений компонент контекстной диаграммы А-0 порождает большую *неопределённость в толковании понятий концептуального проектирования*.

Свойства человеческого мышления — таковы, что каждый из читателей представленной диаграммы А-0 понимает её по своему, внося своё видение как в систему в целом, так и в последующую её детальную декомпозицию. В последующем, такое различное видение свойств системы и её деталей приведёт уже к явным противоречиям, которые будут активно высказываться всеми, кто читает и анализирует диаграммы проекта.

Чтобы максимально избежать последствий описанной выше общей проблемы, Петрову И.В. поручается: дополнить текстовое описание контекстной диаграммы А-0 подробным анализом того факта, что проектируемая система *является индивидуальным средством автоматизации субдеятельности преподавателя (руководителя)*, поэтому она запускается им только периодически и **для достижения следующих трёх целей**:

1) *реализации подготовительной части работ*, связанной с формированием списков групп студентов на основе входного потока (стрелки) «***Данные о студентах***» и для связи этих групп с дисциплинами, входящими во внешний ограничительный документ (стрелку) «***Учебная нагрузка***»;
2) *реализации заключительной части работ*, связанной с формированием внешнего выходного потока (стрелки) «***Оценки обучения студентов***»;
3) *реализации базовой части работ*, осуществляемой согласно заданному внешнему ограничению (стрелке) «***Расписание занятий***» для преобразования на каждом отдельном занятии внешнего входного потока «***Студенты на обучение***» во внешний выходной поток «***Обученные студенты***».

Примечание — **Основным документом**, обеспечивающим единое семантическое толкование компонент всех диаграмм проекта, является правильно составленный ***Глоссарий***.

Текстовое описание диаграммы проекта может содержать достаточное и подробное описание смыслового содержания диаграммы, но воспринимать этот текст сложно, особенно, если речь идёт о спорном толковании его отдельных компонент. Указанный недостаток семантики проекта должен устранять *глоссарий*, описывающий каждую компоненту диаграммы настолько подробно, чтобы не допустить неоднозначного толкования этого описания.

Полный глоссарий по всем диаграммам должен быть оформлен отдельным документом, содержащим список всех использованных наименований в алфавитном порядке, чтобы было удобно их находить. В данном же пункте излагаются только понятия, относящиеся к приведённой выше контекстной диаграмме А-0. ***Например***.

Выполнить семестровую учебную нагрузку преподавателя — *субдеятельность преподавателя*, именующая выполняемую функцию ИС ЭЖР.

Сама субдеятельность осуществляется посредством периодического запуска ИС:

а) как строго по расписанию занятий для выполнения базовой части работ;
б) так и в произвольное время для выполнения подготовительной и заключительной части работ.

Данные о студентах — *внешние информационные объекты*, уникально идентифицирующие каждого студента, включая принадлежность к группе обучения.

Обученные студенты — *внешние физические объекты на выходе ИС ЭЖР*, которые циклически подвергаются субдеятельности преподавателя и получают на отдельном занятии дисциплины порцию знаний, предусмотренную внешними управляющими ограничениями субдеятельности.

Оценки обучения студентов — *внешние информационные объекты на выходе ИС ЭЖР*, соответствующие результатам аттестации преподавателем знаний студентов.

Преподаватель — *внешний механизм ИС ЭЖР*, с помощью которого осуществляется функция субдеятельности проектируемой системы.

Расписание занятий — *внешнее управляющее воздействие*, определяющее дату, время и продолжительность отдельной порции субдеятельности преподавателя.

Студенты без доступа — *выходные физические объекты ИС ЭЖР*, соответствующие входным объектам «***Студенты на обучение***», которые по каким-либо причинам не получили доступ к занятию (конкретной порции субдеятельности преподавателя) и не получили (предусмотренную занятием) порцию знаний.

Студенты на обучение — *входные физические объекты ИС ЭЖР*, которые преобразуются блоком А0 диаграммы А-0 в два внешних физических потока объектов: «***Студенты без доступа***» и «***Обученные студенты***».

УМП дисциплин — *внешние управляющие объекты*, определяющие объём знаний для каждой порции субдеятельности преподавателя.

Учебная нагрузка — *внешние управляющие объекты блока А-0*, ограничивающие состав и объём субдеятельности преподавателя.

Примечание — Методика IDEF0 требует полного завершения процесса создания контекстной диаграммы А-0 перед началом работ по её декомпозиции.

3.3.2 Декомпозиция блока А0 учебной задачи

Первая задача декомпозиции любого блока модели IDEF0 — определение числа входящих в него блоков и их функциональное назначение.

Любая инструментальная система, при попытке выполнить декомпозицию выделенного блока *спросит о количестве блоков, входящих в декомпозицию*, а затем создаст новую страницу для диаграммы и разместит в ней неименованные блоки *в порядке их доминирования*. Поэтому общая практика создания диаграмм модели IDEF0 состоит в первоначальном использовании инструментов в виде *бумаги*, *карандаша*, *линейки* и *стирательной резинки*.

Что касается декомпозиции блока А0 учебной задачи, то его декомпозиция по блокам уже фактически была заявлена в предыдущем пункте как *совокупность трёх процессов*:

а) *подготовительная* часть работ;

б) *базовая* часть работ;

в) *заключительная* часть работ.

Примечание — **Функциональное название любого блока диаграммы** должно указывать, как этот блок «перерабатывает» каждую отдельную совокупность объектов входного потока в другую совокупность объектов выходного потока.

Дополним уже начатый документ глоссария тремя названиями блоков, входящих в диаграммы А0:

1) *подготовить список группы и ЭЖД* (Электронный Журнал Дисциплины);

2) *осуществить проведение занятия*;

3) *оценить работу студента*.

Подготовить список группы и ЭЖД (***A1***) — *функция* ***процесса*** *преподавателя*, входящая в его субдеятельность и соответствующая подготовительной части его работ, которая осуществляется по управляющему ограничению «***Учебная нагрузка***» и преобразует входной информационный поток объектов «***Данные о студентах***», сначала в объекты списков групп, а затем, — в объекты «***ЭЖД занятий***». ЭЖД — *Электронный журнал дисциплины*.

Осуществить проведение занятия (***A2***) — *функция* ***процесса*** *преподавателя*, входящая в его субдеятельность и соответствующая базовой части его работ, которая осуществляется под управляющими ограничениями «***Расписание занятий***» и «***УМП дисциплин***». Эта функция преобразует входные потоки «***ЭЖД занятий***» и «***Студенты на обучение***» в четыре выходных потока: «***Студенты без доступа***», «***Обученные студенты***», «***Отчёты студентов***» и «***ЭЖД после занятия***».

Оценить работу студента (***A3***) — *функция* ***процесса*** *преподавателя*, входящая в его субдеятельность и соответствующая заключительной части его работ, которая выполняется под управляющим ограничением «***УМП дисциплин***». Она преобразует входные потоки «***ЭЖД после занятия***» и «***Отчёты студентов***» в два выходных потока: «***Оценки обучения студентов***» и «***ЭЖД после оценки отчётов***».

На основании выделенных блоков, диаграмма A0 — построена и показана на рисунке 3.25, а *глоссарий дополняется определениями внутренних стрелок этой диаграммы*.

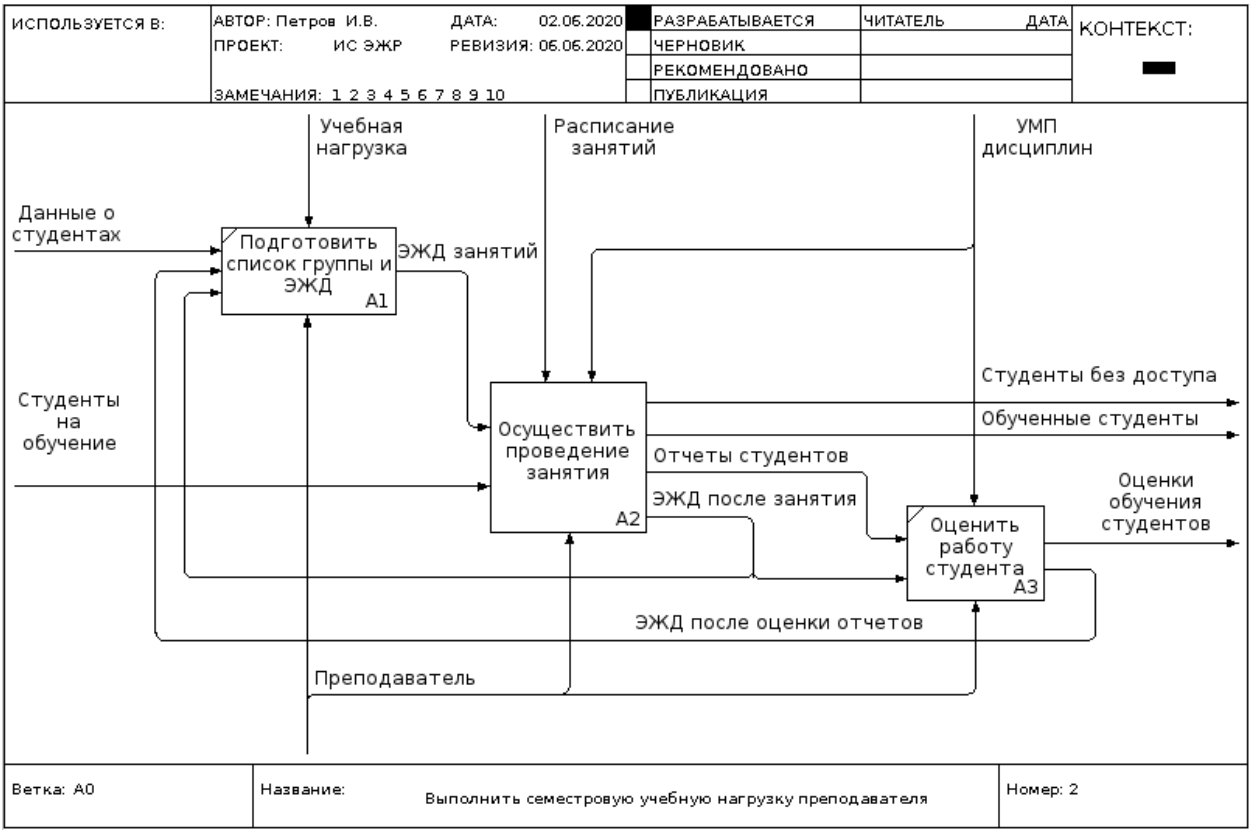


Рисунок 3.25 — Диаграмма A0, отображающая процессы ИС ЭЖР

Отчёты студентов — *индивидуальные документы студентов*, оформляемые ими в виде файлов или сообщений по результатам проведённого занятия и доступные (в каком-либо виде) преподавателю для оценивания.

ЭЖД занятий — *внутренние данные ИС ЭЖР*, формируемые для отдельных дисциплин учебной нагрузки на основе уже сформированных списков групп. Отдельный (для каждой дисциплины) ЭЖД хранится в отдельной базе данных дисциплины.

ЭЖД после занятия — *отдельный (для каждого занятия) ЭЖД*, в который добавлена информация о проведённом занятии.

ЭЖД после оценки отчётов — *отдельный (для каждого занятия) ЭЖД*, в который добавлена информация об оценках выполненных студентами работ.

Обратите внимание, что каждый ЭЖД изменяется, как после завершения занятия, так и после оценивания отчётов, циклически изменяясь от занятия к занятию.

3.3.3 Декомпозиция процессов учебной задачи

Примечание — **Декомпозиция блока А0 является обязательной**, а декомпозиция блоков последующих диаграмм выполняется по необходимости.

Представленные на рисунке 3.25 блоки диаграммы А0 соответствуют функциональному ***уровню процессов***. Поэтому далее, декомпозиция любого из этих блоков будет представлять диаграмму, блоки которой будут соответствовать, например, функциональному уровню *операций* или *действий*. Это зависит от сложности проектируемой системы.

Для учебной цели данного пункта, проведём декомпозицию только блока А2, который наиболее важен для реализации ИС ЭЖР. Этот блок автоматизирует процесс проведения занятия преподавателем и связан с достижением основной целевой функции ИС ЭЖР — повышение производительности труда преподавателя.

Что касается семантики функционирования блока А2, то здесь достаточно чётко можно выделить три аспекта работ **уровня операций**:

а) *аспект управления занятием*, осуществляемый преподавателем в виде доступа студентов к занятию, формулирования отдельных заданий занятия и предоставления студентам учебного материала для выполнения заданий;

б) *выполнение последовательности заданий студентами*, оформление ими отчётов о выполненных работах и формулирование вопросов преподавателю;

в) *аспект устранения преподавателем различных проблемных ситуаций*, связанных с выполнением студентами заданий, и ответами на все вопросы студентов.

В указанной выше проекции рассуждений, диаграмма А2 будет иметь три блока, имена которых также включаются в глоссарий проекта.

Управлять проведением занятия (***A21***) — ***операция*** *диаграммы А2*, выполняемая преподавателем в процессе проведения отдельного занятия и предполагающая следующее её содержание:

а) *подключить конкретный ЭЖД* к текущему занятию для регистрации хода его выполнения;

б) *активировать в ИС ЭЖР средства контроля доступа студентов к занятию* с целью фильтрации входного потока «***Студенты на обучение***» на выходной внешний поток «***Студенты без доступа***» и внутренний поток «***Студенты с доступом***»;

в) *определить конкретные задания* студентам для выполнения ими учебных работ;

г) *предоставить студентам учебный материал* для выполнения заданий.

Выполнить задание (***A22***) — ***операция*** *диаграммы А2*, выполняемая студентами как внутренний поток объектов — «***Студенты с доступом***». Операция осуществляется под управлением потока «***Задание***» и связана с изучением входного потока «***Учебный материал заданий***» в выходные потоки: «***Отчёты студентов***» и «***Вопросы студентов***».

Ответить на вопрос студента (***A23***) — ***операция*** *диаграммы А2*, выполняемая преподавателем под управлением «***УМП дисциплин***». Операция преобразует «***Вопросы студентов***» в «***Сообщения ответов***».

На основании выделенных блоков, диаграмма А2 имеет вид показанный на рисунке 3.26, а глоссарий проекта дополняется определениями внутренних стрелок этой диаграммы.

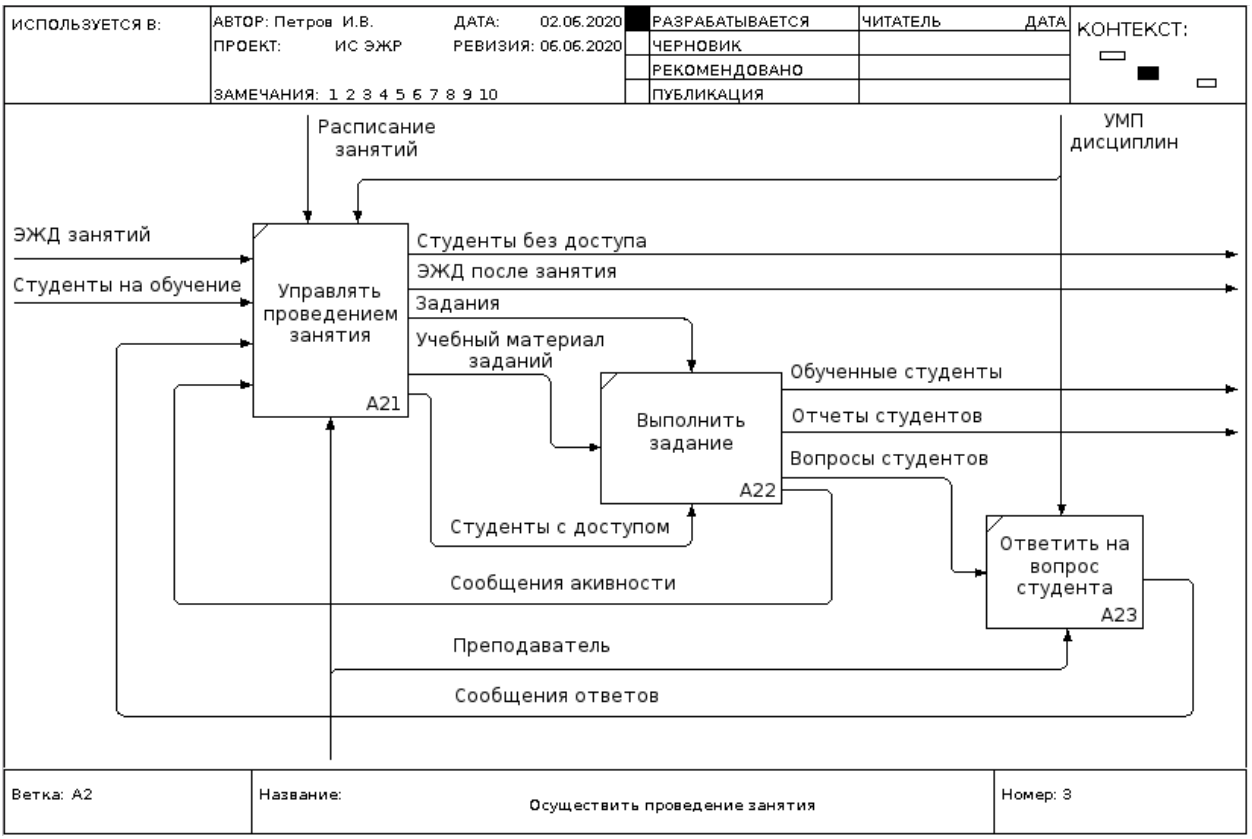


Рисунок 3.26 — Диаграмма А2, отображающая операции блока А2 ИС ЭЖР

Вопросы студентов — *выходной поток блока А22* — «***Выполнить задание***», который регистрируется в ИС ЭЖР в виде текстовых сообщений, а затем преобразуется блоком А23 в «***Сообщения ответов***».

Задания — *выходной поток блока А21*, генерируемый преподавателем как элементарные управляющие сообщения студентам. Включает отчёты студентов по предыдущим заданиям, которые преобразуются блоком А22 в объекты «***Отчёты студентов***».

Сообщения активности — *выходной поток блока А22*, генерируемый студентами, после завершения выполнения задания или после отправки вопросов преподавателю.

Сообщения ответов — *выходной поток блока А23*, генерируемый преподавателем как результат текстового ответа студентам.

Студенты с доступом — *выходной поток блока А21*, генерируемый под надзором преподавателя как результат допуска студентов к текущему занятию, который поступает на блок А22 в качестве его «механизма».

Учебный материал заданий — *выходной поток блока А21*, генерируемый преподавателем для обеспечения студентов учебным материалом, включённым в «***УМП дисциплин***». Поступает как входной поток в блок А22.

ЭЖД после занятия — *выходной поток блока А21*, генерируемый преподавателем с целью сохранения всех изменений в ЭЖД.

Следует заметить, что мы провели декомпозицию только одного блока А2 диаграммы А0. Если обозначить через $N_{ср}$ — среднее количество блоков декомпозиции на одной диаграмме и проводить декомпозицию ИС ЭЖР ***до операций***, то *среднее количество диаграмм* $D_{ср}$ можно определить формулой (3.1):

$$D_{ср} = N_{ср}^2 + 1 \quad (3.1)$$

Не трудно посчитать, что если $N_{ср} = 3$, то $D_{ср} = 10$.

Если декомпозицию проводить ***до действий***, то получаем выражение (3.2):

$$D_{ср} = N_{ср}^3 + 1 = 28 \quad (3.2)$$

3.3.4 Декомпозиция операций учебной задачи

Примечание — **Уровень декомпозиции блоков диаграмм** определяется постановкой задачи и целью, которую должна достигать проектируемая ИС ЭЖР.

Формула (3.1), приведённая в предыдущем пункте, показывает, что, даже при минимальном значении количества блоков на каждой отдельной диаграмме, общее количество необходимых диаграмм — достаточно велико. Но, если мы учтём целевое назначение стадии концептуального проектирования — написание ТТЗ и ТЗ на целевую систему, то необходимость декомпозиции некоторых блоков окажется под вопросом.

На примере диаграммы А2 (см. рисунок 3.26), отображающей *операции блока А2* (***Осуществить проведение занятия***), мы видим следующую семантику блоков:

а) *блок А21* (***Управлять проведением занятия***) — набор согласованных действий, выполняемых преподавателем во время проведения отдельного занятия (базовый вид занятия — лабораторная работа); набор и количественный объем действий этого набора порождают проблемы, описанные в постановке учебной задачи (подраздел 2.1);

б) *блок А22* (***Выполнить задание***) — набор действий, выполняемых студентами на отдельном занятии под руководством преподавателя;

в) *блок А23* (***Ответить на вопрос студента***) — набор действий преподавателя, связанный с созданием текстового сообщения студенту.

Очевидно, что для завершения стадии концептуального проектирования ИС ЭЖР достаточно будет провести декомпозицию блока А21, чем мы и займёмся в данном пункте. Также понятно, что для раскрытия набора действий блока А21 необходимо иметь достаточный опыт проведения хотя бы лабораторных работ. Поэтому далее приводится *описание перечня автоматизируемых действий преподавателя*, без обоснования необходимости их наличия.

Подключить ЭЖД к занятию (***А211***) — ***действие***, связанное с выбором необходимого для проведения занятия ЭЖД и подключения его к системе (с учётом управления «Расписание занятий»), сделав его рабочим для текущего занятия.

Активировать контроль доступа студентов (***А212***) — ***действие***, связанное с запуском некоторой серверной программы, которая будет контролировать доступ к ИС ЭЖД объектов: «Студенты на обучение», «Сообщения ответов» и «Сообщения активности».

Регистрировать в ЭЖД активность студентов (***А213***) — ***действие*** преподавателя, связанное с просмотром очереди доступных сообщений и регистрацией в ЭЖД сообщений, заслуживающих дальнейшего внимания.

Реагировать на активность студентов (***А214***) — действие преподавателя, связанное с реакцией преподавателя на важные сообщения и генерацию (под управлением «УМП дисциплин») выходных потоков объектов: «Задания», «Учебный материал заданий» и «ЭЖД после занятия».

На основании выделенных блоков, диаграмма A21 — построена и показана на рисунке 3.27, а глоссарий проекта дополняется определениями внутренних стрелок этой диаграммы.

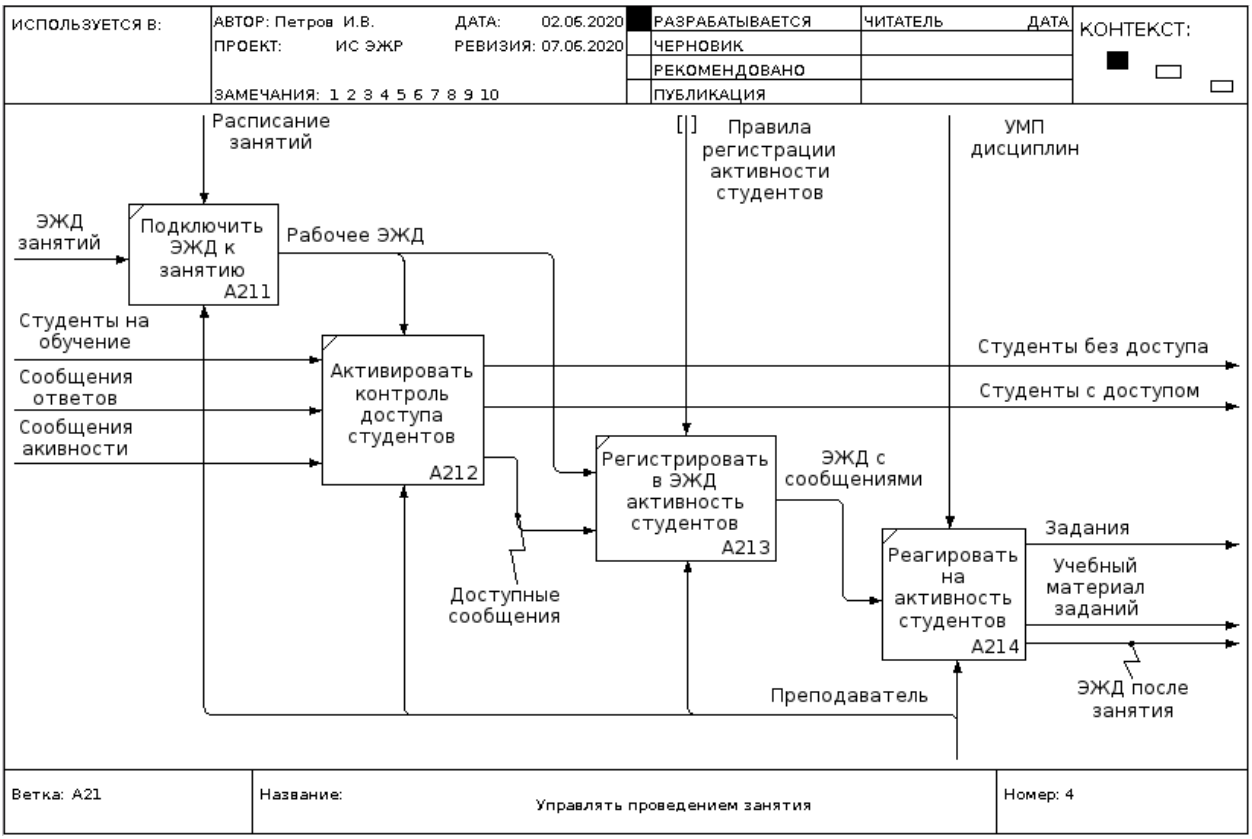


Рисунок 3.27 — Диаграмма A21, отображающая действия блока A21 ИС ЭЖР

Доступные сообщения — *выходные объекты блока A212* (Активировать контроль доступа студентов), прошедшие контроль доступа к занятию.

Рабочее ЭЖД — *выходной объект блока A211* (Подключить ЭЖД к занятию), управляющий контролем доступа всех объектов к занятию и являющийся входным объектом блока A213 (Регистрировать в ЭЖД активность студентов).

ЭЖД с сообщениями — *выходной объект блока A213* (Регистрировать в ЭЖД активность студентов), поступающий в блок A214 (Реагировать на активность студентов), где преподаватель осуществляет оперативное управление занятием.

Правила регистрации активности студентов — *входной* ***управляющий и тунелированный поток*** *объектов* (правил), для настройки действия блока A213 (Регистрировать в ЭЖД активность студентов).

> Примечание — **При описании определений глоссария** не следует экономить место и время на включении поясняющих записей о компонентах диаграммы.

Может показаться, что многие описания компонент диаграмм приведены излишне подробно, поскольку они достаточно хорошо видны и понятны на изображении самой диаграммы. Но следует помнить, что ***глоссарий*** — *это справочный документ*, в котором определения расположены в алфавитном порядке, поэтому любое пояснение читается и воспринимается в отрыве от других описаний компонент диаграмм проекта.

3.3.5 Формирование ТТЗ учебной задачи

Примечание — **ТТЗ** (тактико-техническое задание) является основным документом, на основании которого формируется отчёт по второй стадии проектирования («Разработка концепции ИС») и выполняется третья стадия проектирования ИС — «Техническое задание» (ТЗ).

Применительно к решаемой учебной задаче можно заметить, что рекомендуется дополнительно провести декомпозицию блоков А1 (***Подготовить список группы и ЭЖД***) и А3 (***Оценить работу студента***), но мы не будем этим заниматься, поскольку эти блоки реализуют вспомогательные функции и непосредственно не участвуют в главном процессе системы, который реализуется блоком А2 (***Осуществить проведение занятия***).

В целом, проведённой декомпозиции ИС ЭЖР, представленной совокупностью диаграмм А-0, А0, А2 и А21 (см. рисунки 3.24-3.27), уже достаточно, чтобы написать ТТЗ на проектируемую систему и завершить вторую стадию проектирования ИС. Структура и правила оформления отчёта по второй стадии такие же, как и при оформлении отчёта по первой стадии (см. раздел 2, пункт 2.5.2). Они должны соответствовать требованиям ГОСТ 7.32-2017 [34]. Титульный лист отчёта необходимо согласовать с руководителем практики от университета, а содержание основной части отчёта по НИР должно соответствовать международному стандарту РД 50-34.698-90 [33], где в Приложении 1 рекомендована следующая структура:

1) описание результатов изучения объекта автоматизации;
2) описание и оценка преимуществ и недостатков разработанных альтернативных вариантов концепции создания АС;
3) сопоставительный анализ требований пользователя к АС и вариантов концепции АС на предмет удовлетворения требованиям пользователя;
4) обоснование выбора оптимального варианта концепции и описание предлагаемой АС;
5) ожидаемые результаты и эффективность реализации выбранного варианта концепции АС;
6) ориентировочный план реализации выбранного варианта концепции АС;
7) необходимые затраты ресурсов на разработку, ввод в действие и обеспечение функционирования;
8) требования, гарантирующие качество АС;
9) условия приёмки системы.

Примечание — На практике, **ТТЗ** часто является основной частью более развёрнутого документа — ***ТКП*** (Технико-Коммерческого Предложения). Это — конкурсный документ, участвующий с ТКП других потенциальных исполнителей в отборе наиболее подходящих и привлекательных решений для заказчика ИС.

В целом, организация и проведение конкурсных мероприятий по выбору лучшего концептуального проекта ИС может потребовать множества дополнительных документов, определяемых соответствующей конкурсной комиссией. Возможны также дополнительные требования и к оформлению ТКП. Все эти вопросы выходят за рамки данного учебного пособия и далее не обсуждаются. Мы ограничиваемся тем, что ТТЗ (тактико-техническое задание) является завершающим документом учебной практики студентов, а отчёт по практике, основная часть которого содержит представленные выше разделы, является основным документом, полностью описывающим концепцию будущей ИС.

Что касается нашей учебной задачи, то основной исполнитель (Петров И.В.), завершает производственную практику полным концептуальным описанием ИС ЭЖР, используя материалы отчёта по первой стадии проектирования и ссылаясь на этот отчёт по мере необходи-

мости. Все вопросы по окончательному оформлению и защите учебной практики Петровым И.В. также выходят за рамки данного учебного пособия, но если он полностью придерживался указанным выше требованиям, то окончательный результат должен быть положительным.

3.3.6 Техническое задание на ИС учебной задачи

Техническое задание (ТЗ) на проектируемую ИС является основным документом, который формально (юридически) закрепляет требования к проектируемой ИС.

Согласно ГОСТ 34.601-90 [8] третья стадия проектирования АС (ИС) — «*Техническое задание*» содержит только один этап — «*Разработка и утверждение задания на создание АС*». Он выполняется на основе проектного материала первых двух стадий и в качестве результата создаёт два юридически оформленных и утверждённых (подписанных заказчиком и исполнителем) документа:

а) ***собственно документ ТЗ***, содержащий технические требования к создаваемой ИС;

б) ***договор на создание ИС***, содержащий обязательства сторон по срокам и этапам выполняемых работ, срокам и порядку оплаты выполненных работ, а также условия завершения работ и порядок устранения разногласий между заказчиком и исполнителем работ.

Примечание — Существующая практика обучения студентов уровня бакалавриата не предполагает создания полноценного ТЗ на проектируемые ИС.

В примере задания на преддипломную практику, выданного Петрову И.В. (см. раздел 2, пункт 2.1.2, рисунок 2.1), указаны только название индивидуального задания и время прохождения практики. Даже в заданиях на ВКР (выпускные квалификационные работы), выполняемых в соответствии с документом «*Образовательный стандарт вуза ОС ТУСУР 01-2013*» [36], «***Задание на бакалаврскую работу***» ограничено требованиями двух страниц текста и содержит только шесть информационных частей:

1) тема бакалаврской работы (БР);
2) срок выдачи студентом законченной БР;
3) исходные данные к работе;
4) содержание расчётно-пояснительной записки/перечень подлежащих разработке вопросов;
5) перечень графического материала (с точным указанием обязательных листов презентации);
6) дата выдачи задания.

Неудивительно, что студенты, опираясь на указанные выше исходные данные, не имеют адекватных представлений о ТЗ и его истинном назначении.

Цель дальнейшего учебного материала данного пункта — дать общее представление о документе ТЗ и указать на важные правила его оформления.

Техническое задание (ТЗ) на автоматизированную систему (АС, ИС или АИС) оформляется в соответствии с требованиями ГОСТ 34.602-89 [9].

ТЗ является достаточно сложным документом, который отражает все стадии создания системы, начиная со стадии «*Эскизный проект*». Мифы о сложности этого документа несколько преувеличены и основаны на существующей неоднозначности понимания формулировок самого стандарта.

Как целостный документ, ТЗ на АС содержит следующие шесть частей:

1) лист утверждения (ЛУ);
2) титульный лист;
3) содержание;
4) основная часть ТЗ;
5) последний лист;
6) приложения (по необходимости).

Основная часть ТЗ на АС содержит следующие девять разделов, которые допускают разделение на подразделы, пункты и подпункты:

1) общие сведения;
2) назначение и цели создания (развития) системы;
3) характеристика объектов автоматизации;
4) требования к системе;
5) состав и содержание работ по созданию системы;
6) порядок контроля и приёмки системы;
7) требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие;
8) требования к документированию;
9) источники разработки.

Примечание — **Первая проблема** формирования ТЗ на проектируемому ИС — оформление «Листа утверждения» (ЛУ).

ГОСТ 34.602-89 [9] не требует обязательного наличия листа утверждения в составе ТЗ, тем не менее, многие организации-разработчики считают обязательным его присутствие. Более того, многие документы на ИС имеют такие листы, поэтому студент должен иметь общее представление о его назначении и правилах оформления.

В целом, лист утверждения оформляется на одном или нескольких листах, содержит указание о разработчике документа, названии документа, его идентификации, а также утверждающие и согласующие подписи. Пример шаблона такого листа (ЛУ) для документа ТЗ показан на рисунке 3.28.

Прежде всего обратим внимание, что левая сторона листа утверждения отведена для утверждающих и согласующих подписей заказчика системы и других согласующих организаций, а правая сторона — для организации-разработчика (исполнителя). Сам этот лист хранится у разработчика и, в случае необходимости, его копия передаётся заказчику или другим уполномоченным сторонам.

В середине листа утверждения находятся надписи, идентифицирующие документ, а также надпись обозначения документа, которая в соответствии с требованиями ГОСТ 34.201-89 [37] имеет вид, показанный на рисунке 3.29.

Здесь «*Код документа*» — двухбуквенное обозначение, которое для технического задания равно «*ТЗ*», а «*Обозначение системы (части системы)*», согласно этому же источнику [37], показано на рисунке 3.30.

Наименование организации-разработчика

УТВЕРЖДАЮ	УТВЕРЖДАЮ
Руководитель (должность, наименование организации-заказчика АСУ)	Руководитель (должность, наименование организации-разработчика АСУ)
Личная подпись Расшифровка подписи	Личная подпись Расшифровка подписи
Дата М.П.	Дата М.П.

Наименование вида АС

Наименование объекта автоматизации

Сокращенное наименование АС

ТЕХНИЧЕСКОЕ ЗАДАНИЕ

Лист утверждения

XXXXXXXX.XXXXXXX.XXX.ТЗ-ЛУ

СОГЛАСОВАНО	СОГЛАСОВАНО
(должность)	(должность)
(ФИО)	(ФИО)
«___» _____20__ г.	«___» _____20__ г.
М.П.	М.П.
СОГЛАСОВАНО	СОГЛАСОВАНО
(должность)	(должность)
(ФИО)	(ФИО)
«___» _____20__ г.	«___» _____20__ г.
М.П.	М.П.

Рисунок 3.28 — Шаблон листа утверждения для документа ТЗ

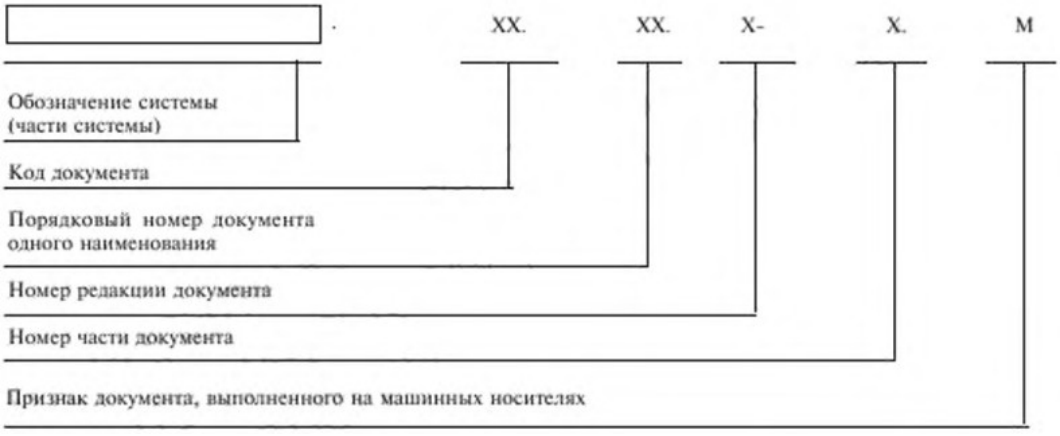


Рисунок 3.29 — Обозначение документа по ГОСТ 34.201-89 [37]

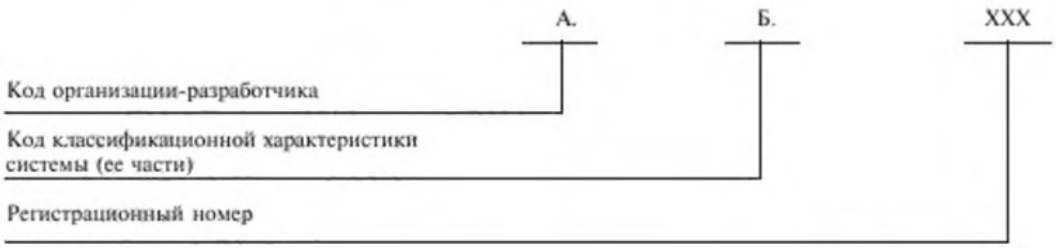


Рисунок 3.30 — Обозначение системы или её части по ГОСТ 34.201-89 [37]

Код организации-разработчика присваивается общесоюзным классификатором для предприятий, учреждений и организаций (ОКПО). Для ТУСУР это значение равно ***02069326***.

Код классификационной характеристики системы (её части) присваивают в соответствии с правилами, установленными в отрасли (например, ОКП на основе подкласса 425000 «Программно-технические комплексы для автоматизированных систем» или 507000 «Прикладные программные средства учебного назначения», см. [38]) или на основе общесоюзного классификатора подсистем и комплексов задач АСУ, который в настоящее время не поддерживается.

Регистрационный номер — номер организации-разработчика (от 001 до 999), под которым она регистрирует проектируемую систему.

Примечание — На практике, «***Обозначение системы (части системы***» — это всего лишь идентификатор проектируемой системы для хранения и извлечения документации организацией разработчиком.

Каждая организация-разработчик обычно имеет свой классификатор, которым и должен пользоваться студент. Например, образовательный стандарт ТУСУР [36] требует следующую структуру идентификатора, показанную на рисунке 3.31. При этом:

а) в качестве ***кода разработчика*** (в учебных проектах) рекомендуется использовать *аббревиатуру выпускающей кафедры*;

б) ***порядковый регистрационный номер разработки*** или номер редакции документа *назначается по указаниям кафедры*, организующей проектирование.

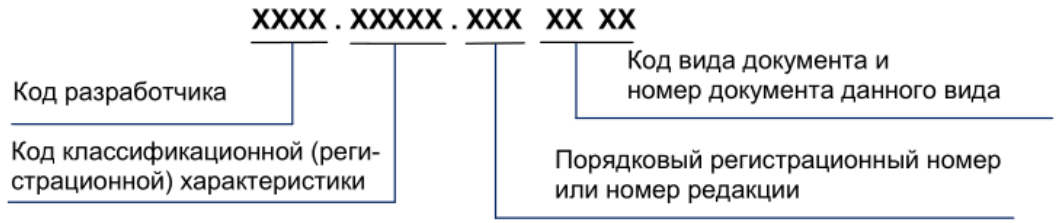


Рисунок 3.31 — Обозначение проектов по ОС ТУСУР 01-2013 [36]

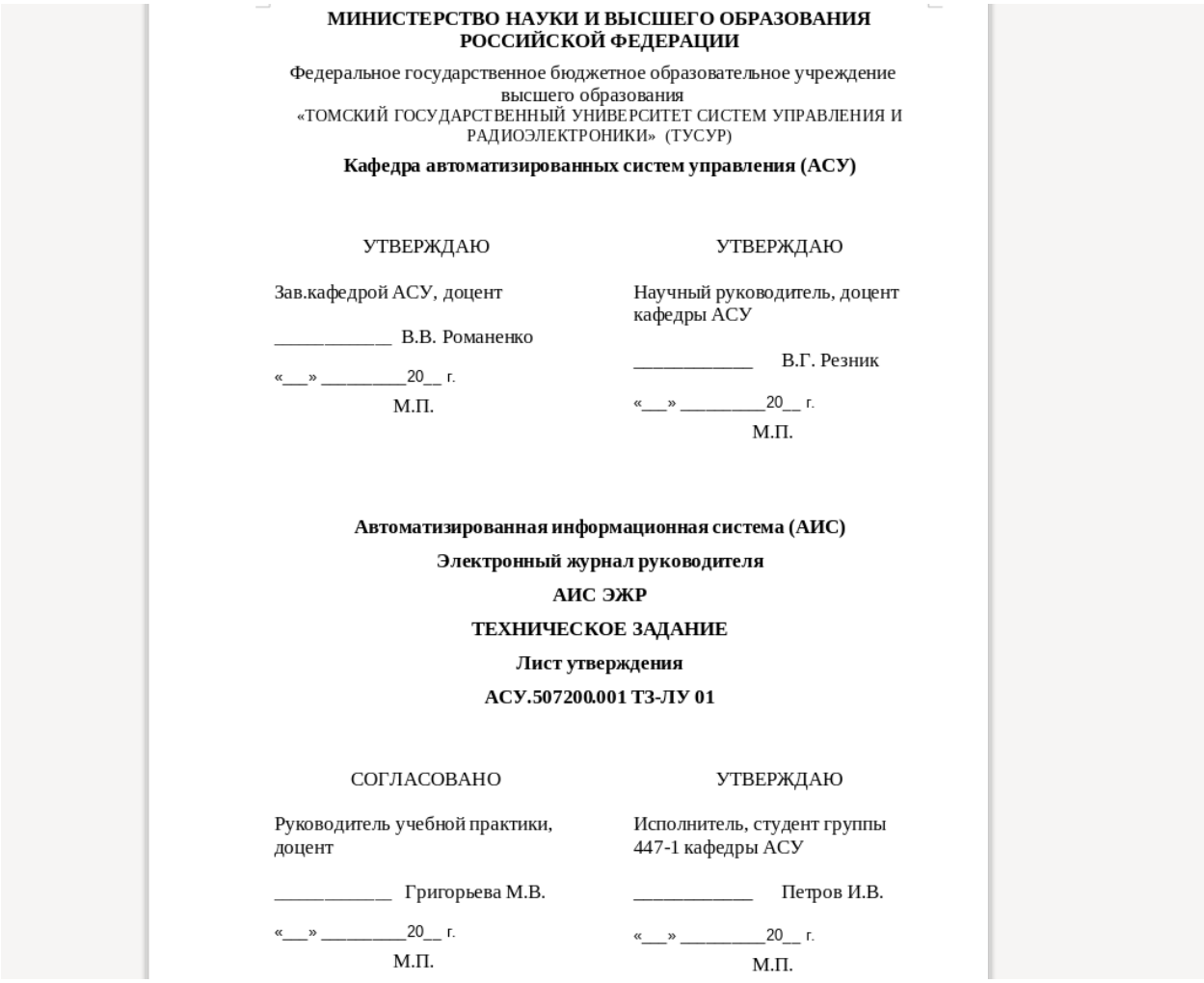
Согласно требованиям кодирования, представленным рисунком 3.31 и организационным данным учебной задачи (см. далее по тексту таблицу 3.2), лист утверждения ТЗ примет вид, показанный на рисунке 3.32.

Теперь, сам документ ТЗ, его титульный лист и завершающий лист будут иметь шифр **АСУ.507200.001 ТЗ 01** и по этому шифру заказчик может его заказывать.

Пример титульного листа ТЗ для учебной задачи показан на рисунке 3.33. Далее, на рисунке 3.34 показан завершающий лист ТЗ.

Таблица 3.2 — Организационные данные учебной задачи

Поле идентификатора	***Значение***
Код разработчика	АСУ
Код ОКП	507200 - «Программные средства для управления учебным процессом» [38]
Номер регистрации	001 (считаем, что под таким номером наша система зарегистрирована секретарём кафедры АСУ)
Код вида документа	ТЗ-ЛУ
Номер документа	01
Лица утверждающие	Зав.кафедрой — Романенко В.В. (от заказчика) Руководитель — Резник В.Г. (от разработчика)
Лица согласовывающие	Руководитель практики — Григорьева М.В. (от заказчика) Студент — Петров И.В. (от разработчика)



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)

Кафедра автоматизированных систем управления (АСУ)

УТВЕРЖДАЮ
Зав.кафедрой АСУ, доцент
_____ В.В. Романенко
«___» _____20__ г.
М.П.

УТВЕРЖДАЮ
Научный руководитель, доцент кафедры АСУ
_____ В.Г. Резник
«___» _____20__ г.
М.П.

Автоматизированная информационная система (АИС)

Электронный журнал руководителя

АИС ЭЖР

ТЕХНИЧЕСКОЕ ЗАДАНИЕ

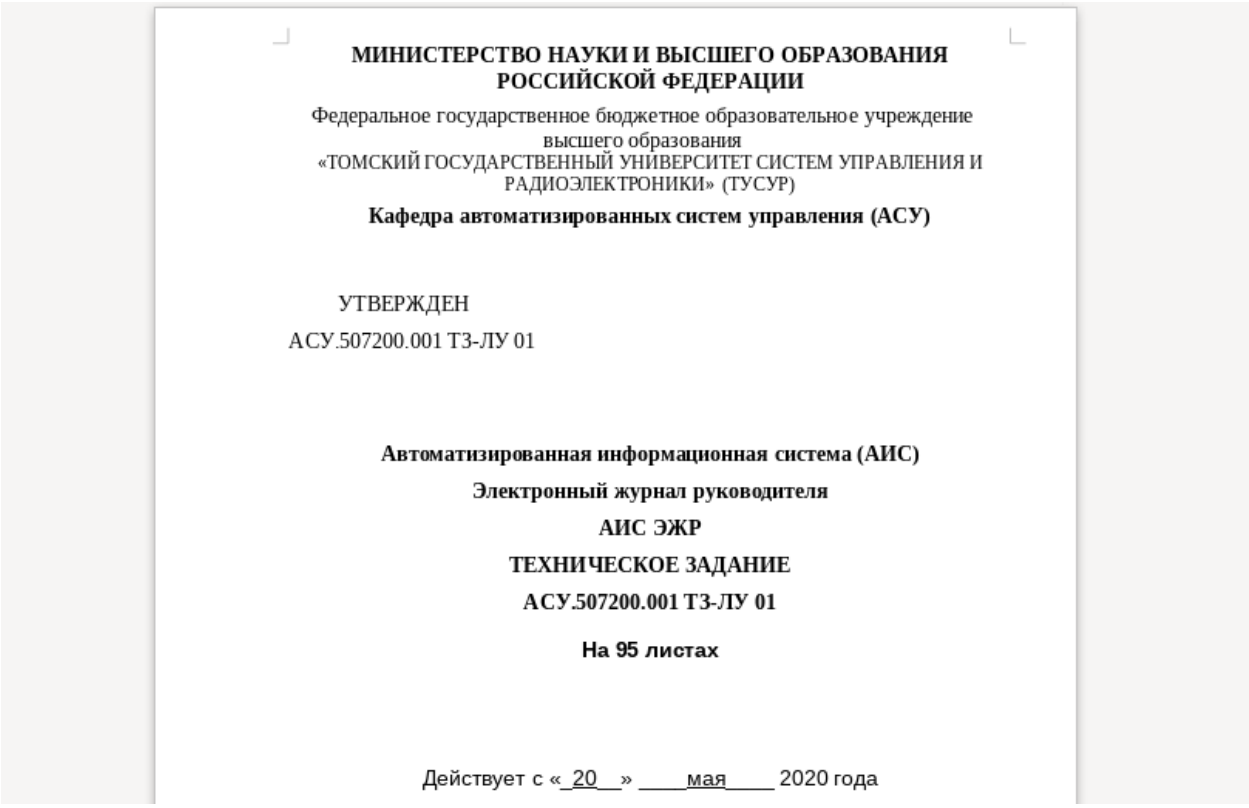
Лист утверждения

АСУ.507200.001 ТЗ-ЛУ 01

СОГЛАСОВАНО
Руководитель учебной практики, доцент
_____ Григорьева М.В.
«___» _____20__ г.
М.П.

УТВЕРЖДАЮ
Исполнитель, студент группы 447-1 кафедры АСУ
_____ Петров И.В.
«___» _____20__ г.
М.П.

Рисунок 3.32 — Лист утверждения ТЗ для учебной задачи



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)

Кафедра автоматизированных систем управления (АСУ)

УТВЕРЖДЕН
АСУ.507200.001 ТЗ-ЛУ 01

Автоматизированная информационная система (АИС)
Электронный журнал руководителя
АИС ЭЖР
ТЕХНИЧЕСКОЕ ЗАДАНИЕ
АСУ.507200.001 ТЗ-ЛУ 01

На 95 листах

Действует с «_20__» ____мая____ 2020 года

Рисунок 3.33 — Титульный лист ТЗ для учебной задачи

АСУ.507200.001 ТЗ 01
95

СОСТАВИЛИ

Наименование организации, предприятия	**Должность исполнителя**	**Фамилия, имя, отчество**	**Подпись**	**Дата**
Кафедра АСУ ТУСУР	студент	Петров И.В.		19.05.20

СОГЛАСОВАНО

Наименование организации, предприятия	**Должность исполнителя**	**Фамилия, имя, отчество**	**Подпись**	**Дата**
Кафедра АСУ ТУСУР	доцент	Григорьева М.В.		19.05.20

Рисунок 3.34 — Завершающий лист ТЗ для учебной задачи

Как отмечено выше, ГОСТ 34.602-89 [9] требует изложения содержания ТЗ в виде девяти обязательных разделов. Полный перечень всех разделов, подразделов и пунктов ТЗ приведён ниже мелким шрифтом, как справочная выдержка из источника [9]:

1.1 Полное наименование системы и ее условное обозначение.

1.2 Шифр темы или шифр (номер) договора.

1.3 Наименование предприятий (объединений) разработчика и заказчика (пользователя) системы и их реквизиты.

1.4 Перечень документов, на основании которых создаётся система, кем и когда утверждены эти документы.

1.5 Плановые сроки начала и окончания работы по созданию системы.

1.6 Сведения об источниках и порядке финансирования работ.

1.7 Порядок оформления и предъявления заказчику результатов работ по созданию системы (ее частей), по изготовлению и наладке отдельных средств (технических, программных, информационных) и программно-технических (программно-методических) комплексов системы.

2 Назначение и цели создания системы.

2.1 Назначение системы.

2.2 Цели создания системы.

3 Характеристика объектов автоматизации.

3.1 Краткие сведения об объекте автоматизации или ссылки на документы, содержащие такую информацию.

3.2 Сведения об условиях эксплуатации объекта автоматизации и характеристиках окружающей среды.

4 Требования к системе.

4.1 Требования к системе в целом.

4.1.1 Требования к структуре и функционированию системы.

4.1.1.1 Перечень подсистем, их назначение, основные характеристики, требования к числу уровней иерархии и степени централизации системы.

4.1.1.2 Требования к способам и средствам связи для информационного обмена между компонентами системы.

4.1.1.3 Требования к характеристикам взаимосвязей создаваемой системы со смежными системами, требования к ее совместимости.

4.1.1.4 Требования к режимам функционирования системы.

4.1.1.5 Требования по диагностированию системы.

4.1.1.6 Перспективы развития, модернизации системы.

4.1.2 Требования к численности и квалификации персонала системы и режиму его работы.

4.1.2.1 Требования к численности персонала (пользователей) АС.

4.1.2.2 Требования к квалификации персонала, порядку его подготовки и контроля знаний и навыков.

4.1.2.3 Требуемый режим работы персонала АС.

4.1.3 Показатели назначения.

4.1.3.1 Степень приспособляемости системы к изменению процессов и методов управления к отклонению параметров объекта управления.

4.1.3.2 Допустимые пределы модернизации и развития системы.

4.1.3.3 Вероятностно-временные характеристики, при которых сохраняется целевое назначение системы.

4.1.4 Требования к надёжности.

4.1.4.1 Состав и количественные значения показателей надёжности для системы в целом или ее подсистем.

4.1.4.2 Перечень аварийных ситуаций, по которым должно быть регламентированы требования к надёжности, и значения соответствующих показателей.

4.1.4.3 Требования к надёжности технических средств и программного обеспечения.

4.1.4.4 Требования к методам оценки и контроля показателей надёжности на разных стадиях создания системы в соответствии с действующими нормативно-техническими документами.

4.1.5 Требования к безопасности.

4.1.6 Требования к эргономике и технической эстетике.

4.1.7 Требования к транспортабельности для подвижных АС.

4.1.8 Требования к эксплуатации, техническому обслуживанию, ремонту и хранению компонентов системы.

4.1.8.1 Условия и регламент (режим) эксплуатации, которые должны обеспечивать использование технических средств (ТС) системы с заданными техническими показателями, в том числе виды и периодичности обслуживания ТС системы или допустимость работы без обслуживания.

4.1.8.2 Предварительные требования к допустимым площадям для размещения персонала и ТС системы, к параметрам сетей энергоснабжения.

4.1.8.3 Требования по количеству, квалификации обслуживающего персонала и режимам его работы.

4.1.8.4 Требования к составу, размещению и условиям хранения комплекта запасных изделий и приборов.

4.1.8.5 Требования к регламенту обслуживания.

4.1.9 Требования к защите информации от несанкционированного доступа.

4.1.10 Требования по сохранности информации при авариях.

4.1.11 Требования к средствам защиты от влияния внешних воздействий.

4.1.11.1 Требования к радиоэлектронной защите средств АС.

4.1.11.2 Требования по стойкости, устойчивости и прочности к внешним воздействия (среде применения).

4.1.12 Требования к патентной чистоте.

4.1.13 Требования по стандартизации и унификации.

4.1.14 Дополнительные требования.

4.1.14.1 Требования к оснащению системы устройствами для обучения персонала (тренажёрами, другими устройствами аналогичного назначения) и документацией на них.

4.1.14.2 Требования к сервисной аппаратуре, стендам для проверки элементов системы.

4.1.14.3 Требования к системе, связанные с особыми условиями эксплуатации.

4.1.14.4 Специальные требования по усмотрению разработчика или заказчика системы.

4.2 Требования к функциям (задачам), выполняемым системой.

4.2.1 Требования к подсистеме. Перечень функций, задач или их комплексов (в том числе обеспечивающих взаимодействие частей системы), подлежащих автоматизации.

4.2.2 Временной регламент реализации каждой функции, задачи (или комплекса задач).

4.2.3 Требования к качеству реализации каждой функции (задачи или комплекса задач), к форме представления выходной информации, характеристики необходимой точности и времени выполнения, требования одновременности выполнения группы функций, достоверности выдачи результатов.

4.2.4 Перечень и критерии отказов для каждой функции, по которой задаются требования по надёжности.

4.3 Требования к видам обеспечения.

4.3.1 Требования к математическому обеспечению.

4.3.2 Требования к информационному обеспечению.

4.3.2.1 Требования к составу, структуре и способам организации данных в системе.

4.3.2.2 Требования к информационному обмену между компонентами системы.

4.3.2.3 Требования к информационной совместимости со смежными системами.

4.3.2.4 Требования по использованию общесоюзных и зарегистрированных республиканских, отраслевых классификаторов, унифицированных документов и классификаторов, действующих на данном предприятии

4.3.2.5 Требования по применению систем управления базами данных.

4.3.2.6 Требования к структуре процесса сбора, обработки, передачи данных в системе и представлению данных.

4.3.2.7 Требования к защите данных от разрушений при авариях и сбоях в электропитании системы.

4.3.2.8 Требования к контролю, хранению, обновлению и восстановлению данных.

4.3.2.9 Требования к процедуре придания юридической силы документам, продуцируемым техническими средствами АС (в соответствии с ГОСТ 6.10.4).

4.3.3 Требования к лингвистическому обеспечению.

4.3.4 Требования к программному обеспечению.

4.3.4.1 Требования к независимости программных средств от используемых СВТ и операционной среды.

4.3.4.2 Требования к качеству программных средств, а также к способам его обеспечения и контроля.

4.3.4.3 Требования по необходимости согласования вновь разрабатываемых программных средств с фондом алгоритмов и программ.

4.3.5 Требования к техническому обеспечению.

4.3.5.1 Требования к видам технических средств, в том числе к видам комплексов технических средств, программно-технических комплексов и других комплектующих изделий, допустимых к использованию в системе.

4.3.5.2 Требования к функциональным, конструктивным и эксплуатационным характеристикам средств технического обеспечения системы.

4.3.6 Требования к метрологическому обеспечению.

4.3.6.1 Предварительный перечень измерительных каналов.

4.3.6.2 Требования к точности измерений параметров и (или) к метрологическим характеристикам измерительных каналов.

4.3.6.3 Требования к метрологической совместимости технических средств системы.

4.3.6.4 Перечень управляющих и вычислительных каналов системы, для которых необходимо оценивать точностные характеристики.

4.3.6.5 Требования к метрологическому обеспечению технических и программных средств, входящих в состав измерительных каналов системы, средств встроенного контроля, метрологической пригодности измерительных каналов и средств измерений, используемых при наладке и испытаниях системы.

4.3.6.6 Вид метрологической аттестации (государственная или ведомственная) с указанием порядка ее выполнения и организаций, проводящих аттестацию.

4.3.7 Требования к организационному обеспечению.

4.3.7.1 Требования к структуре и функциям подразделений, участвующих в функционировании системы или обеспечивающих эксплуатацию.

4.3.7.2 Требования к организации функционирования системы и порядку взаимодействия персонала АС и персонала объекта автоматизации.

4.3.7.3 Требования к защите от ошибочных действий персонала системы.

4.3.8 Требования к методическому обеспечению.

4.3.9 Требования к другим видам обеспечения системы.

5 Состав и содержание работ по созданию системы.

5.1 Перечень документов по ГОСТ 34.201, предъявляемых по окончании соответствующих стадий и этапов работ.

5.2 Вид и порядок проведения экспертизы технической документации (стадия, этап, объем проверяемой документации, организация-эксперт).

5.3 Программа работ, направленных на обеспечение требуемого уровня надёжности разрабатываемое системы.

5.4 Перечень работ по метрологическому обеспечению на всех стадиях создания системы с указанием их сроков выполнения и организации-исполнителей.

6 Порядок контроля и приёмки системы.

6.1 Виды, состав, объем и методы испытаний системы и ее составных частей.

6.2 Общие требования к приёмке работ по стадиям.

6.3 Статус приёмочной комиссии.

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

7.1 Приведение поступающей в систему информации (в соответствии с требованиями к информационному и лингвистическому обеспечению) к виду, пригодному для обработки с помощью ЭВМ.

7.2 Изменения, которые необходимо осуществить в объекте автоматизации.

7.3 Создание условий функционирования объекта автоматизации, при которых гарантируется соответствие создаваемой системы требованиям, содержащимся в ТЗ.

7.4 Создание необходимых для функционирования системы подразделений и служб.

7.5 Сроки и порядок комплектования штатов и обучения персонала.

8 Требования к документированию.

8.1 Согласованный разработчиком и заказчиком системы перечень подлежащих разработке комплектов и видов документов, соответствующих требованиям ГОСТ 34.201 и НТД отрасли заказчика; перечень документов, выпускаемых на машинных носителях; требования к микрофильмированию документации.

8.2 Требования по документированию комплектующих элементов межотраслевого применения в соответствии с требованиями ЕСКД и ЕСПД.

8.3 При отсутствии государственных стандартов, определяющих требования к документированию элементов системы, дополнительно включают требования к составу и содержанию таких документов.

9 Источники разработки.

Перечень принятых сокращений.

Примечание — Практика разработки, согласования и утверждения ТЗ на АС допускает использование только части подразделов и пунктов содержания, представленного выше.

Представленная выше структура текста ТЗ *является рекомендованной и допускает исключение подразделов и пунктов* мало относящихся к сущности проектируемой системы. Более того, допускается, по согласованию сторон, включать новые разделы подразделы, которые раскрывают или уточняют сущность решаемых системой задач.

В целом, ТЗ является сложным и достаточно объёмным документом, структура которого должна быть изучена и понятна студенту.

3.3.7 Итог применения методологии IDEF0 к проектированию ИС

Рассмотренная в данном подразделе методология IDEF0 для концептуального проектирования учебной задачи и формирования её ТЗ наглядно показывает всю сложность принятия проектных решений и оформления соответствующей документации. Более сложные задачи могут потребовать построение дополнительных поясняющих моделей, которые не реализуются средствами изученной методологии структурного проектирования.

Анализ диаграмм, представленных ранее на рисунках 3.24 — 3.27, показывает, что *не всегда понятно* в какой последовательности должны выполняться объявленные и задокументированные функции. Особенно это актуально для ***действий***, представленных и описанных на нижних уровнях декомпозиции диаграмм IDEF0.

Для построения дополнительных поясняющих моделей концептуального проектирования рекомендуется использовать методологию IDEF3.

3.4 Моделирование потоков работ IDEF3

Вторым по значимости методологией структурного моделирования бизнес-процессов SADT является стандарт IDEF3.

IDEF3 (Integrated DEFinition for Process Description Capture Method) — методология структурного моделирования и стандарт документирования процессов, происходящих в системе, показывающий причинно-следственные связи между ситуациями и событиями в понятной экспертам форме и использующий структурный метод выражения знаний о том, как функционирует система, процесс или предприятие.

Формальное описание этого стандарта доступна в англоязычной публикации [39], содержащей 236 страниц текста и иллюстраций, которая находится в свободном доступе по адресу: http://www.staratel.com/iso/IDEF/IDEF3/Idef3.pdf. Этой публикации мы и будем придерживаться, по крайней мере, в плане демонстрации рисунков. Кроме того, на просторах интернета имеются многочисленные частные публикации и трактовки этого документа, содержащие как правило методические рекомендации по его применению. В частности, краткий обзор всех стандартов IDEF содержится в учебном пособии Цукановой О.А. [40], который рекомендуется студентам для самостоятельного изучения.

> Примечание — Различные методические публикации по методологии IDEF3 делают акцент на использовании различных терминов, что затрудняет единое понимание и последующее практическое использование этой методологии.

Как и любая структурная модель, диаграммы стандарта IDEF3 содержат *узлы* и *связи* между ними, которые имеют достаточно большой набор интерпретаций, что увеличивает выразительные возможности графического языка, но затрудняет последующее понимание семантики создаваемых диаграмм моделей. Более того, IDEF3 состоит из двух методов:

а) **Process Flow Description** (PFD) — Описание технологических процессов, с указанием того, что происходит на каждом этапе технологического процесса;

б) **Object State Transition Description** (OSTD) — описание переходов состояний объектов, с указанием того, какие существуют промежуточные состояния у объектов в моделируемой системе.

Основу методологии IDEF3 составляет графический язык описания процессов. Модель в нотации IDEF3 может содержать два типа диаграмм:

а) **Описание Последовательности Этапов Процесса** — Process Flow Description Diagrams (PFDD);

б) **Сети Трансформаций Состояния Объекта** — Object State Transition Network (OSTN);

в) **Комбинированный вариант**, использующий графические элементы обоих предыдущих типов диаграмм.

Чтобы избежать различных неоднозначных толкований изучаемой методологии, ограничим изучение методологии IDEF3 только следующими четырьмя пунктами, достаточными для применения этой методологии в задачах концептуального проектирования ИС.

3.4.1 Элементы графического языка описания процессов IDEF3

Основным мотивом разработки методологии IDEF3 была необходимость предоставить концепции, синтаксис и процедуры для создания описаний системных требований к процессам. Эти требования подразумевают, что IDEF3, как метод, должен поддерживать описания следующих элементов:

1) *сценарии* организационной деятельности;
2) *роли типов пользователей* в этих организационных действиях.
3) *пользовательские сценарии* или взаимодействие пользователя с информационной системой на пользовательско-функциональном уровне;
4) *реакция системы* на пользовательские функции.
5) *пользовательские классы* и разграничение пользовательских классов.
6) *декларации сроков*, последовательности и ограничений ресурсов.
7) *объекты пользовательского интерфейса*, например, меню, ключевые слова, экраны и дисплеи.

На рисунке 3.35 показаны элементы графического языка IDEF3, которые должны обеспечить отображение заявленных описаний процессов.

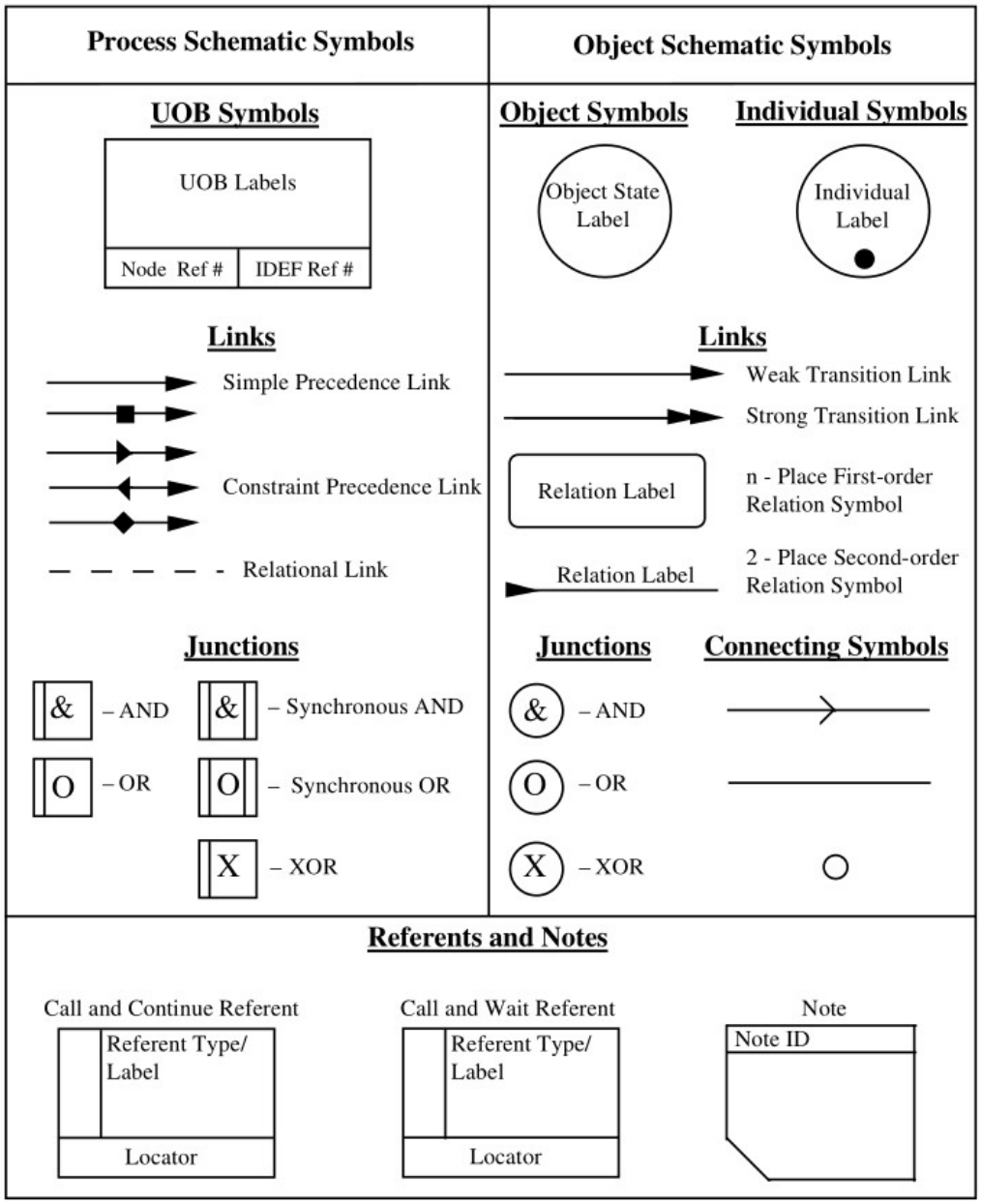


Рисунок 3.35 — Элементы языка IDEF3 [39]

Верхняя часть рисунка 3.35 разделена на две части:

а) **Process Schematics Symbols** — графические символы предназначенные для отображения схем (сценариев, диаграмм) *процессов*, таких как блоки UOB (Units of Behavior), стрелки связей блоков UOB (Links) и блоки перекрёстков (Junctions);

б) **Object Schematics Symbols** — аналогичные символы для схем состояний объектов.

В нижней части рисунка 3.35 представлены вспомогательные графические символы, названные как *Референты* (Referents) и *Замечания* (Notes).

Notes — графические элементы *Замечания* (*Примечания*) используемые для обеспечения дополнительной информации в процессе моделирования, которые могут присоединятся к диаграммам иллюстраций, текста, комментариев и другим элементам для предоставления возможности выразить идеи или концепции вместо использования относительных связей.

Referents (*Референты*) — типизированные и именованные объекты ссылок, привязанные к UOB или перекрёсткам для описания особых типов связи, которые нецелесообразно указывать ссылками *Links*.

Референты могут использоваться в схемах процессов и объектов IDEF3 для следующих целей:

а) для ссылки на ранее определённый блок UOB без дублирования его определения, чтобы указать, что другой экземпляр ранее определённого блока UOB возникает в определённой точке процесса;

б) для передачи управления или указания обратной связи в обработке других UOB.

в) для формирования ссылок или связей между схемами процессов и схемами объектов.

Референты являются активными во время активности графического элемента, к которому они подсоединены, а по цели воздействия на ссылаемый элемент диаграммы (схемы) подразделяются на два типа:

а) **Call and Continue Referent** — референт *Вызывать-и-Продолжить* ***только инициирует ссылочный элемент***, что должно произойти до завершения его собственной активности;

б) **Call and Wait Referent** — референт *Вызывать-и-Ожидать* ***инициирует ссылочный элемент и ждёт завершения его функционирования***, что должно произойти до завершения собственной активности референта.

В таблице 3.3 представлены типы референтов и обозначения их меток, используемые в диаграммах методологии IDEF3.

Таблица 3.3 — Типы и обозначения меток референтов методологии IDEF3

Типы референта	***Обозначение меток референта***
UOB	Имя функционального элемента UOB (№ UOB)
SCENARIO	Название сценария (№ Scenario)
TS (Transaction Schematic)	Название диаграммы перехода состояний (№ диаграммы перехода)
GO-TO	Имя функционального элемента UOB (№ UOB, № сценария или декомпозиции, в которой находится элемент)

Примечание — *Референты* и *Замечания* улучшают понимание диаграмм IDEF3, придают им дополнительный смысл и упрощают их построение, поскольку предназначены для «минимизации беспорядка» как в схемах (диаграммах) процессов, так и схемах объектов.

Уже перечисленные элементы графического языка IDEF3 показывают, что диаграммы, построенные по методологии IDEF3 обладают достаточно большими средствами выразительности, далеко выходящими за рамки проектирования только информационных систем (ИС).

> **Главная особенность методологии IDEF3**, отличающая её от методологии IDEF0, — возможность отображать временные последовательности выполнения процессов, работ, событий и других элементов действий, отображаемых графическими элементами *UOB*, *Object State Label* и *Junctions*.

Указанная особенность методологии IDEF3 сосредоточена и раскрывается метаопределением понятия *Сценарии*. Чтобы наиболее точно раскрыть это понятие, приведу прямой текст оригинала [39, стр. 10-12]: «Понятие сценария или истории используется в качестве базовой организационной структуры для описаний процессов IDEF3. Сценарий можно рассматривать как повторяющуюся ситуацию, набор ситуаций, описывающих типичный класс проблем, решаемых организацией или системой, или обстановку, в которой происходит процесс. Сценарии устанавливают фокус и граничные условия описания. Использование сценариев таким образом использует склонность людей описывать то, что они знают, в терминах упорядоченной последовательности действий в контексте данного сценария или ситуации. Сценарии также обеспечивают удобное средство для организации коллекций знаний, ориентированных на процесс.

Поскольку основная роль сценария заключается в связывании контекста описания процесса IDEF3, важно дать ему соответствующее имя. Названия сценариев часто принимают форму повелительного наклонения (например, глагол или глагольные фразы, такие как «Оформить заказ на покупку», «Проверить пригодность» и т. д.), а иногда могут принимать форму герундия (например, глагол, который функционирует как существительное, например, «Выполнение» или «Проверки согласованности». Хорошо подобранное название сценария гарантирует, что у пользователей описания возникнут соответствующие ассоциации с описываемыми реальными ситуациями. Правильная идентификация, характеристика и наименование сценариев является необходимым шагом к созданию ориентированных на процесс описаний процессов IDEF3. Следующие примеры являются типичными именами сценариев.

1.Разработайте дизайн штампа для боковой апертурной панели.

2. Обработка жалоб клиента.

3. Реализовать запрос на инженерное изменение.

Описание процесса IDEF3 разработано с использованием двух стратегий приобретения знаний: стратегии, ориентированной на процесс, и стратегии, ориентированной на объект. Стратегия, ориентированная на процесс, организует знания о процессах с упором на процессы и их временные, причинно-следственные и логические отношения в рамках сценария. Второе измерение организует знания о процессах, сосредоточив внимание на объектах и их поведении при изменении состояния в одном или нескольких сценариях.

Используя одну или обе эти стратегии получения знаний о процессах, пользователи IDEF3 разрабатывают описания процессов IDEF3. Обе стратегии используют базовые элементы языка IDEF3 для задания и выражения утверждений, формирующих описание.

Графические проекции информации, содержащейся в описаниях процессов, создаются с использованием графического языка IDEF3. Эти графические проекции, используемые как для непосредственной записи информации о процессе, так и в качестве механизма для отображения информации о процессе, называются схемами.

Два типа схем IDEF3 параллельны двум стратегиям получения знаний о процессах. Схема процесса IDEF3 отображает представление сценария, ориентированное на процесс. Схемы объектов поддерживают графическое отображение объектно-ориентированной информации. Схемы объектов, которые отображают объектно-центрированное представление одного сценария, называются схемами переходов. Схемы переходов, которые отображают допол-

нительные объекты и отношения между объектами для предоставления информации, устанавливающей контекст, называются расширенными схемами переходов. Схемы объектов, которые отображают объектно-центрированную информацию, охватывающую несколько сценариев, называются просто схемами объектов.

Описание процесса IDEF3 может содержать ноль или более схем процессов и ноль или более схем объектов. Например, запись о том, что конкретный объект распознаётся участниками домена, считается частью описания этого домена. Идентифицированный таким образом объект может иметь или не иметь схему объекта, связанную с ним в описании. Тем не менее, эти объекты считаются частью описания. Концепция сценария используется для организации представлений, ориентированных как на процессы, так и на объекты. Набор сценариев и информации, предназначенная для организации представлений, и является описанием процесса в стандарте IDEF3».

> Для концептуального проектирования ИС нецелесообразно использовать полную методологию стандарта IDEF3.

Можно обсуждать заявленный тезис применительно к другим стадиям проектирования ИС, но стадия концептуального проектирования требует лишь уточнения функционального описания процессов, которые полностью не раскрываются или не могут быть раскрыты средствами методологии IDEF0. Это обосновывается следующими аргументами.

Аргумент 1. Метод схем объектов стандарта IDEF3 серьёзно конкурирует с другими объектными моделями, например, стандартом UML и другими объектными технологиями, которые рассматриваются в следующем разделе данного пособия.

Аргумент 2. Находится под сомнением современная актуальность разработки инструментальных CASE-средств, ориентированных на реализацию полной методологии IDEF3. Известная инструментальная система BPWIN, поддерживающая построение диаграмм IDEF3, содержит методические описания и примеры, которые демонстрируют сценарии процессов.

Аргумент 3. Многие методические описания, демонстрирующие методику практического использования IDEF3, применяют вместо определения UOB (Unit of Behavior) определение UOW (Unit of Work), трактуя IDEF3 как методологию потоков работ, а полученные диаграммы — как WFD (Work Flow Description/Diagrams).

Следуя общей сложившейся традиции, дальнейшее изложение учебного материала данного раздела будет ориентировано на описание диаграмм потоков процессов или Process Flow Description Diagrams (PFDD).

3.4.2 Синтаксис и семантика языка IDEF3

Основным структурным функциональным элементом модели IDEF3 является блок UOB (Unit of Behavior), графический символ которого показан в левой верхней части рисунка 3.35. Этот символ обозначается *непрерывным именованным прямоугольником* и по общему семантическому назначению соответствует функциональному блоку модели IDEF0. Но поскольку диаграммы методологии IDEF3 не учитывают метапонятия связей *Управления* и *Механизма*, то отличительные особенности UOB выражаются следующими особенностями:

а) UOB содержит только стрелки (связи) входов и выходов, определяя временную очерёдность активности UOB, а также присоединение к UOB графических элементов *Референтов* и *Замечаний*;

б) именование UOB не поддерживает градацию его функционального содержания по уровням иерархии, давая UOB множественную интерпретацию: функция, процесс, действие, акт, событие, сценарий и другие подобные семантики.

Примечание — Казалось бы указанные выше особенности UOB упрощают сам процесс построения диаграмм методологии IDEF3 и наполняют UOB большим содержанием. Но, как уже было отмечено выше, многообразие толкований имён UOB приводит к неоднозначности последующей интерпретации полученных описаний потребителями уже созданных диаграмм.

Указанная проблема наглядно демонстрируется рисунком 3.36.

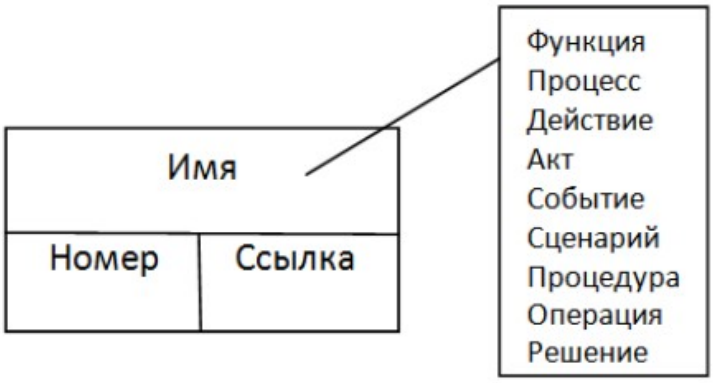


Рисунок 3.36 — Синтаксис и семантика графического элемента UOB [40]

Действительно, указание в имени UOB её функциональной интерпретации не только увеличивает длину этого имени, но и затрудняет последующее составление различных текстовых описаний получаемых диаграмм: официальной документации проектов, различных инструкций и других сопутствующих документов. Именно поэтому различные методические пособия по IDEF3 трактуют UOB как функции, действия или просто процессы. Обычно эти вариации обобщённой семантики UOB вызваны недостаточной квалификацией проектировщиков или стремлением приспособиться (угодить) ограниченным познаниям потребителей конечного продукта.

Частично указанный недостаток именования и интерпретации UOB разрешается использованием «Ссылки» на соответствующий блок диаграммы IDEF0, а также рекомендацией именовать функциональный элемент UOB активным глаголом, как это требует сама методология IDEF0.

Другой и более радикальный подход — интерпретировать функциональное содержание UOB как *Работа* и использовать это понятие при оформлении всей проектной документации. Например, авторы методического пособия [41] Уральского государственного технического университета (УГТУ), описывающего работу CASE-средств BPWIN и ERWIN, в явном виде заменяют UOB (Unit of Behavior) на UOW (Unit of Work).

Работа — синтаксическое обозначение и семантическая интерпретация функционального содержания и имени UOB, обобщающее частичные функциональные интерпретации имени UOB, показанные ранее на рисунке 3.36.

Безусловно, «механическая» подмена UOB на UOW искажает теоретическую интерпретацию стандарта IDEF3, семантически определяя её диаграммы как описание *Потока работ* (Workflow), но в плане концептуального проектирования ИС это — приемлемо, поскольку акцентирует внимание на главных свойствах проектируемых информационных систем.

В дальнейшем изложении учебного материала мы будем использовать именно термин *Работа*, учитывая его общую семантическую ограниченность, указанную выше. Именно термин *Работа* будет использоваться нами для интерпретации семантики *связей* между UOB.

Связи (Links) — важнейшие элементы любой графической диаграммы, задающие всё многообразие отношений между работами моделируемой системы.

На рисунке 3.37 показаны все виды связей стандарта IDEF3, предназначенные для процессного моделирования систем и разделённые на три группы: *связи простой очерёдности*, *связи приоритета ограничения* и *реляционные связи*.

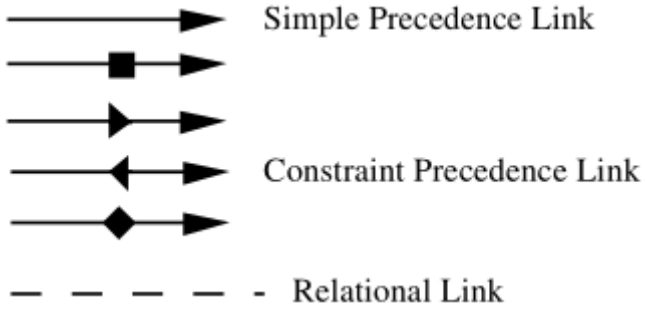


Рисунок 3.37 — Синтаксис графических элементов связей между UOB [39]

Связь простой очерёдности (Simple Precedence Link) — наиболее простой и используемый тип связи, изображаемый сплошной стрелкой с наконечником, которая показывают временной приоритет отношений между двумя функциональными блоками UOB.

На рисунке 3.38 показана связь простой очерёдности между двумя UOB с условными именами ***A*** и ***B***. Она семантически указывает, работа UOB с именем ***B*** может начаться только после завершения работы ***A*** и возможно через некоторый интервал времени.

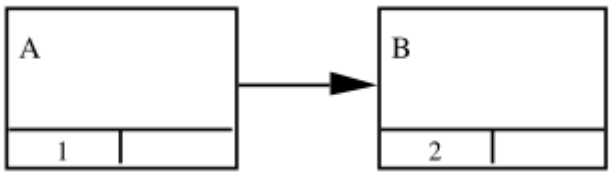


Рисунок 3.38 — Связь простой очерёдности между двумя UOB с именами A и B [39]

Реляционные связи (Relational Links) — связи обозначаемые прерывистыми линиями и не несущие никакой определённой семантики. Они могут указываться не только между различными UOB, но между сценариями и другими элементами диаграмм IDEF3. В целом они подчёркивает на существование (возможно, ограничивающее) между элементами диаграмм.

Связи приоритета ограничения (Constraint Precedence Links) — связи, включающие свойство простой очерёдности, но несущие дополнительную ограничивающую нагрузку между отношениями UOB. Такие связи имеют дополнительный графический элемент на стволе стрелки:

а) *символ треугольника* указывает своим углом направленность ограничения;

б) *символ ромба* указывает, что направленность ограничения распространяется на оба соединяемых UOB, в разных направлениях;

в) *символ прямоугольника* указывает, что свойства и направленность ограничения определены отдельным описанием или документом.

На рисунке 3.39 показаны две связи между UOB №1 с именем Sign timesheet (Подписать расписание) и UOB №2 с именем Obtain timesheet approval (Получить расписание):

а) *реляционная связь* указывает на некоторое «пользовательское» отношение;

б) *ограничивающая связь* указывает не только на временную очерёдность соединённых UOB, но и на то, что собственное расписание утверждать нельзя.

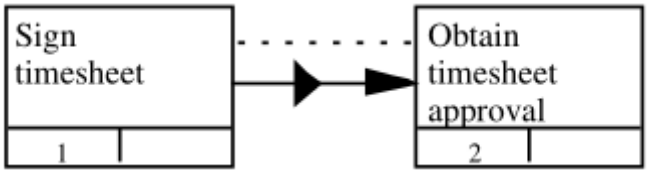


Рисунок 3.39 — Пример обозначения реляционной и ограничивающей связей между двумя UOB [39]

Методология IDEF3 поддерживает декомпозицию и собственную нумерацию блоков UOB в отображаемых диаграммах.

На рисунке 3.40 приведён абстрактный пример некоторой декомпозиции блоков UOB, одновременно поясняющий правила нумерации самих блоков UOB.

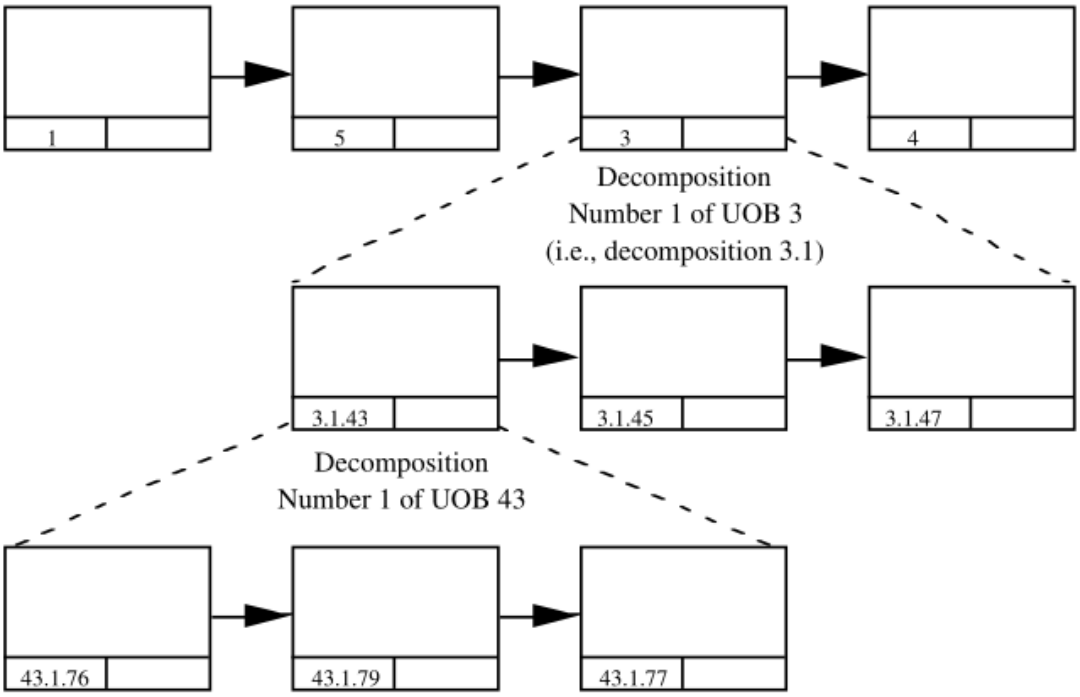


Рисунок 3.40 — Пример декомпозиции UOB в методологии IDEF3 [39]

Безусловно сам процесс декомпозиции естественным образом изменяет и уменьшает масштабность работ более нижнего уровня, но обратите внимание на нумерацию этих UOB, показанную в левой нижней их части. Хорошо видно, что блоки верхнего уровня нумеруются одним числом, а нижних уровней — тремя числами, разделёнными точками.

Правила нумерации блоков UOB:

а) каждому разработчику диаграмм проекта выдаётся личный уникальный диапазон целых чисел, которые он может использовать для нумерации UOB верхнего уровня;
б) если какой-либо UOB удаляется из диаграммы, то этот номер больше уже в проекте не используется;
в) если UOB присутствует в различных вариантах, то номер этого варианта указывается после первой цифры, отделённой точкой;
г) дочерние UOB, полученные в результате декомпозиции, нумеруются тремя целыми числами разделёнными точками: *Номер родительского UOB*; *Номер варианта родительского UOB*; *Номер UOB в пределах уровня декомпозиции*.

3.4.2 Работы и перекрёстки работ методологии IDEF3

Наиболее ценной особенностью методологии IDEF3 является возможность использования в диаграммах логических ограничений на последовательность и параллельность выполнения работ. Для реализации этой возможности используются *Перекрёстки* (Junctions).

Перекрёстки (Junctions) — геометрические элементы диаграмм методологии IDEF3, обеспечивающие отображение слияний (Fan-in) и ветвлений (Fan-out) работ с одновременным наложением на эти ситуации логических условий.

Синтаксис и семантика перекрёстков формально отображается информацией, представленной в таблице 3.4.

Таблица 3.4 — Синтаксис и семантика перекрёстков методологии IDEF3

Обозначение	***Логический тип***	***Семантика слияния***	***Семантика ветвления***
	Асинхронное AND	Все предшествующие процессы (работы) должны быть завершены	Все следующие процессы (работы) должны быть запущены
	Асинхронное OR	Один или несколько предшествующих процессов (работ) должны быть завершены	Один или несколько следующих процессов (работ) должны быть запущены
	Синхронное AND	Все предшествующие процессы (работы) завершены одновременно	Все следующие процессы (работы) запускаются одновременно
	Синхронное OR	Один или несколько предшествующих процессов (работы) завершаются одновременно	Один или несколько следующих процессов (работы) запускаются одновременно
	Эксклюзивное OR (XOR)	Только один предшествующий процесс (работа) завершён	Только один следующий процесс (работа) запускается

Все перекрёстки на диаграмме должны уникально нумероваться, причём каждый номер должен иметь префикс «***J***».

Безусловно наличие пяти логических условий создают большой потенциал для отображения большого количества вариантов построения диаграмм методологии IDEF3. Ниже, на рисунке 3.41, показан пример использования двух перекрёстков ***J1*** и ***J2*** для асинхронного согласования выполнения работ, обозначенных UOB с именами ***A***, ***B***, …, ***F*** с помощью логической операции «***И***» (AND).

Согласно правилам представленным в таблице 3.4:

а) после завершения работы ***A***, запускаются работы ***B***, ***C*** и ***D***;

б) работа A запускается только после того как завершатся работы ***C***, ***D*** и ***E***.

Естественным образом семантика указанных условий дополняется семантикой обозначений блоков UOB.

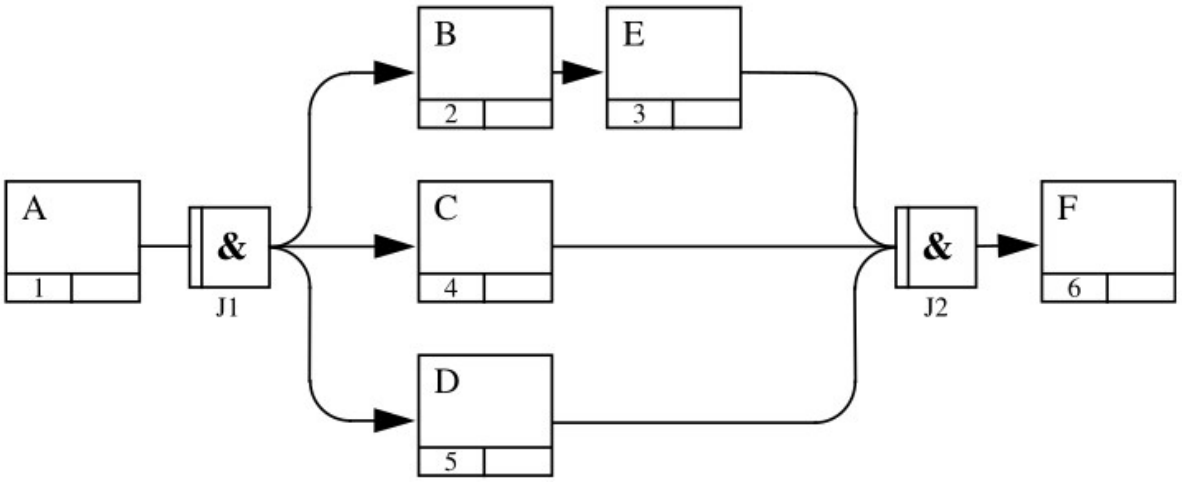


Рисунок 3.41 — Пример диаграммы IDEF3 с двумя перекрёстками [39]

3.4.3 Документирование моделей IDEF3

Кроме диаграмм, стандарт методологии IDEF3 [39] рекомендует оформление и использование множества различных текстовых документов, оформленных в виде специальных бланков. Один из примеров такого бланка показан на рисунке 3.42.

PROJECT LEADER: DATE: COMPANY: PROJECT NO.: TASK NO.:		WORKING DRAFT RECOMMENDED RELEASED	REVIEWER:	DATE:
Purpose: Context:				
List of Scenarios:	List of Objects:			
DESCRIPTION NAME:		FORM TYPE: Description Summary		

Рисунок 3.42 — Сводная форма описания IDEF3 [39]

Каждый рекомендуемый документ снабжён пояснениями и краткими инструкциями его использования. Во многом эти рекомендации соответствуют рекомендациям стандарта IDEF0.

3.4.4 Итоговая оценка применения методологии IDEF3

Методология структурного моделирования IDEF3 ориентирована научное исследование, технологическое проектирование и разработку широкого класса автоматизированных систем, ориентированных на предметные области как бизнес-процессов, так и соответствующие области технологических процессов. Внешне указанные области кажутся достаточно близкими, для студентов — возможно одинаковыми. Но между ними имеются и существенные различия.

Бизнес-процесс — это логическая последовательность действий как отдельного человека, так и группы людей, в коллективе. Сама последовательность действий может осуществляться как на высшем уровне при принятии организационных решений, так и на уровне участия людей в технологических процессах производства. Но описательная и исследовательская сущность бизнес-процессов — регламентация самих этих действий в коллективе.

В плане методологии IDEF3 здесь конкурирует с теоретическими построениями идеологии Business Process Modeling (BPM).

Технологический процесс — последовательность действий (работ), которые осуществляются непосредственно с объектами производства, хотя и при участии людей.

В плане методологии IDEF3 здесь конкурирует с теоретическими построениями идеологии объектного подхода, в частности, — с идеей создания и использования унифицированного языка моделирования UML (*Unified Modeling Language*).

Таким образом, методология IDEF3 структурно стандартизирует как процессный, так и объектный подходы, описывая одновременно и методику их применения. Указанный факт уже был отмечен в начале данного подраздела посредством выделения двух методов: PFD и OSTD. Здесь же мы акцентируем внимание на указанном факте, чтобы подчеркнуть широкий охват предметных областей изученной методологией.

С другой стороны, излишняя универсальность методологии IDEF3 не делает её полностью идеальной:

а) фактически требует построения двух видов диаграмм: процессных и объектных;

б) является во многом избыточной, например, для проектирования информационных систем (ИС);

в) слишком сложна в плане реализации инструментальных средств, полностью поддерживающих указанную методологию и автоматизирующих создание указанных инструментальных средств.

> Вывод — Применительно к практике использования методологии IDEF3 для целей проектирования ИС следует ориентироваться на необходимость отображения последовательностей работ и особенно логических условий ограничения этих последовательностей, отображаемых инструментарием перекрёстков.

Что касается стадии концептуального проектирования ИС, то применение методологии IDEF3 следует осуществлять только в крайнем случае, когда методологии IDEF0 и DFD не обеспечивают нужного результата.

3.5 Моделирование потоков данных DFD

Последней методологией структурного функционального подхода проектирования ИС, заявленного в начале данного раздела, является моделирование потоков данных или сокращённо — *DFD*.

DFD (*Data Flow Diagrams*) — диаграммы потоков данных, составляющие графическую основу *нестандартизированной методологии* структурного функционального моделирования процессов обработки данных, ориентированных на проектирование информационных систем (ИС) различного назначения.

Методологию DFD можно и удобно рассматривать как прародительницу методологии IDEF0.

Сейчас считается, что первые упоминания о DFD относятся к середине 20-го века, а популярной эта методология стала в 1974 году, с момента выхода книги Эда Йордана и Ларри Константина «Структурное проектирование». Напомним, что набор методологий, обозначаемых как IDEF (ICAM Definition) появился только в 1981 году.

Популярность методологии DFD охватила умы различных исследователей информационных систем, благодаря своей простоте, опирающейся на четыре базовых понятия: *процесс*, *поток данных*, *внешняя сущность* и *хранилище данных*. Это позволило специалистам различных предметных областей и различной квалификации быстро находить общий язык.

Процесс (Process/Activity) — функция или последовательность действий (работ) моделируемой ИС, которую нужно выполнять, чтобы обработать поток входных данных и получить результат в виде потока выходных данных. Предполагается, что процесс, с целью анализа, может подвергаться декомпозиции.

Поток данных (Data flow) — информационный объекта любого формата данных, который может быть обработан процессом, сгенерирован или потреблён внешней сущностью, а также сохранён в хранилище данных или извлечён из неё посредством процесса.

Внешняя сущность (External Entity/External Reference) — объект любой природы, находящийся за пределами моделируемой ИС и способный быть для процессов источником или потребителем потока данных.

Хранилище данных (Data store) — объект любой природы, находящийся внутри моделируемой ИС и способный сохранять или извлекать потоки данных для процессов. Хранилищами данных могут быть не только базы данных под управлением СУБД, но и файлы, бумажные или иные носители информации. Если хранилище данных выносится за пределы моделируемой ИС, то оно превращается во внешнюю сущность.

К сожалению широкая популярность DFD привели к множеству интерпретация и различным нотациям его графического языка. Тем не менее в нём сохранились черты, которые являются общими с чертами методологии IDEF0:

а) требование наличия контекстной диаграммы, предполагающей графическое отображение единственного процесса, соответствующего функции самой ИС, необходимого окружения в виде внешних сущностей, определяющих внешнюю среду ИС в виде источников и потребителей информации, а также наличия «ЦЕЛИ» и «ТОЧКИ ЗРЕНИЯ», ограничивающих последующие толкования и интерпретации самой ИС;
б) возможность декомпозиции процессов на необходимое число уровней;
в) рекомендуемая градация уровней на позиции: *концептуального* или *контекстного* уровней; *логического* уровня и *физического* уровня;
г) предполагается нумерация процессов;
д) допускаются интерпретации результатов как диаграммы «AS-IS» и «TO-BE».

Чтобы конкретизировать теоретическое и практическое использование методологии DFD, рассмотрим учебный материал в следующих трёх аспектах:

а) синтаксис и семантика языка DFD;

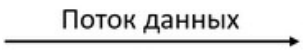
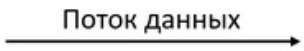
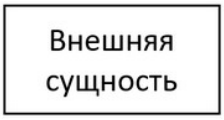
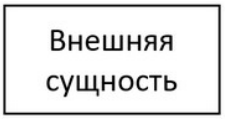
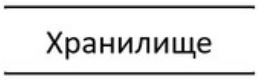
б) учебный пример построения диаграмм DFD;

в) выводы по применению методологии DFD, где подведём итоги по результатам всего раздела.

3.5.1 Синтаксис и семантика языка DFD

Все графические элементы (компоненты) нотации методологии DFD должны быть именованы, дополнительно раскрывая семантику обозначаемых компонент. Исторически создано много нотаций этой методологии, а наиболее известными являются нотации Йордана-ДеМарко (Yordon-DeMarco) и Гейна-Сарсона (Gene-Sarson), показанные в таблице 3.5.

Таблица 3.5 — Два набора синтаксиса графических элементов методологии DFD

Компонента DFD	***Нотация Йордона-ДеМарко***	***Нотация Гейна-Сарсона***
Процесс		
Поток данных		
Внешняя сущность		
Хранилище данных		

Примечание — Имена элементов нотаций, представленных в таблице 3.5, являются метаопределениями имён соответствующих элементов, поэтому их нельзя использовать в диаграммах методологии DFD.

Дополнительно следует учесть нотаций нотаций, реализуемых различными средами разработки. В частности:

а) с нотацией инструментальной среды BPWIN можно познакомиться в источнике [41];

б) с нотацией инструментальной среды Ramus Educational для методологии IDEF0 можно познакомиться в источнике [42];

в) нотация методологии DFD для среды Ramus Educational хорошо изложена в учебном пособии [43].

Учитывая, что студент уже хорошо познакомился с общими правилами построения функциональных диаграмм на примерах учебного материала технологий IDEF0 и IDEF3, дальнейшее описание применения синтаксиса и семантики методологии DFD проведём на основе конкретного учебного примера, изложенного в следующем пункте.

3.5.2 Постановка учебной задачи по методологии DFD

Ранее, для целей концептуального проектирования ИС ЭЖР, студент Петров И.В. провёл функциональное моделирование системы с помощью методологии IDEF0. Диаграммы этого моделирования представлены в разделе 3.3 данного пособия в виде рисунков 3.24 — 3.27.

Анализ диаграммы ***A2*** (рисунок 3.26), которая продублирована ниже в виде рисунка 3.43, показал, что заказчику не понятно: каким образом может быть реализован блок ***A23*** «Ответить на вопрос студента».

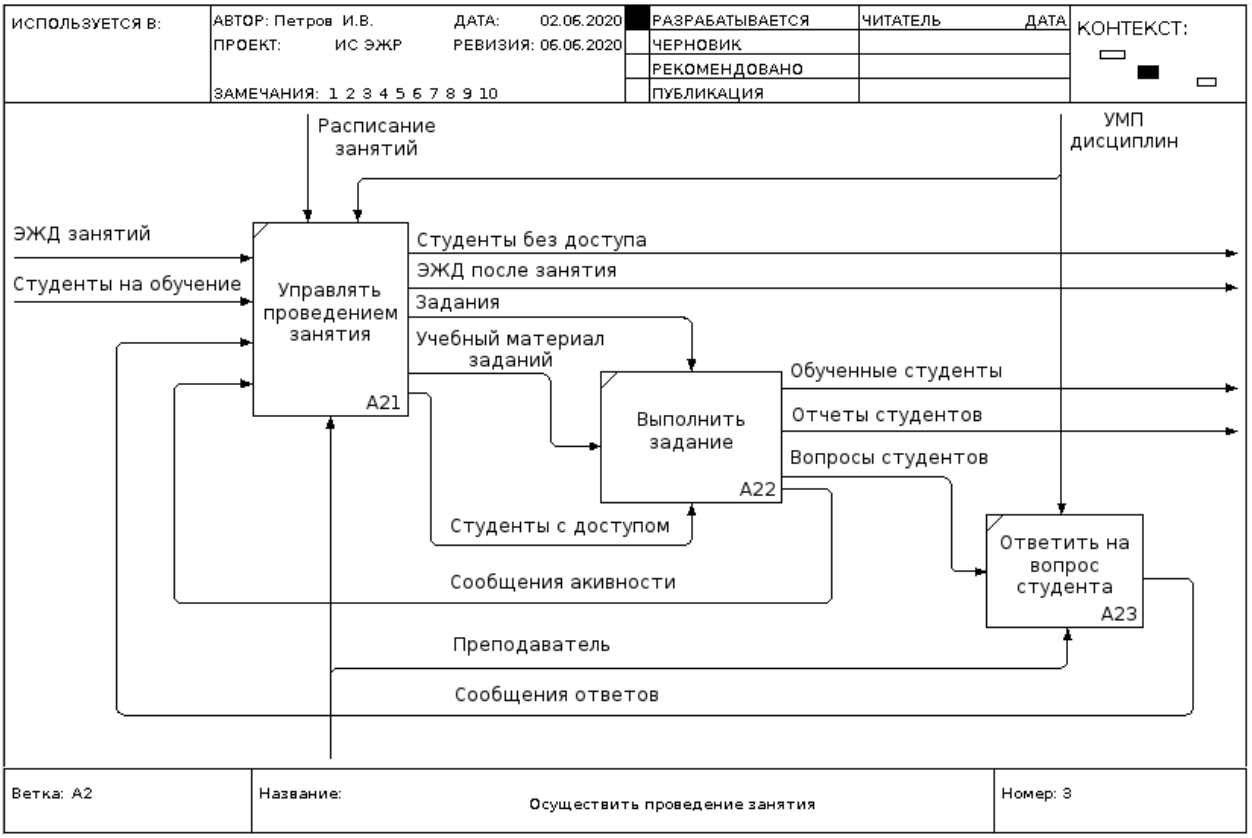


Рисунок 3.43 — Дубль рисунка 3.26, отображающего диаграмму A2 ИС ЭЖР

Объяснения Петрова И.В. о том, что блок A23 может быть реализован с использованием любой СУБД не убедил заказчика («Потенциального заказчика»), который потребовал привести дополнительные декомпозиции, более наглядно поясняющие возникший вопрос.

> Поскольку диаграммы IDEF0 не отображают хранилища данных, то было *принято решение*: пояснить структуру функционирования блока A23 посредством методологии DFD.

Рассмотрим блок A23 как отдельную маленькую информационную систему с именем «Ответить на вопрос студента».

Согласно диаграмме A2, блок A23:

а) *принимает потоки данных* «Вопросы студентов», поступающие из блока A22, и поток данных в виде текстов ответов преподавателя, которые не показаны явно на рисунке 3.43, но подразумеваются в виде связи механизма «Преподаватель»;

б) *генерирует поток данных* «Сообщения ответов», который поступает в функциональный блок A21.

Следует также обратить внимание на два дополнительных обстоятельства:

а) поток данных «Вопросы студентов» генерируется механизмом «Студенты с доступом», который в методологии DFD можно рассматривать как внешнюю сущность «Студенты с доступом»;

б) элементы потоков данных «Вопросы студентов» и «Сообщения ответов» имеют внутреннюю связь, которая преобразуется в элементы данных «Вопрос-ответ» и могут рассматриваться как элементы хранилища данных «БД вопросов и ответов».

Проведённого краткого анализа диаграммы ***A2*** уже достаточно для формулирования учебной задачи, нацеленной на реализацию принятого ранее решения.

Учебная задача — выполнить построение контекстной диаграммы и необходимую её декомпозицию для блоку ***A23*** в соответствии с правилами методологии DFD.

Решение учебной задачи начнём с формализации компонент блока A23 для уровня контекстной диаграммы и изложим необходимые операции и действия в отдельных пунктах данного подраздела.

3.5.3 Формализация компонент контекстной диаграммы блока A23

Этап формализации компонент любой методологии не только значительно облегчает последующее построение диаграмм, но и является частью проектной документации любой системы.

Поскольку методология DFD — формально не стандартизирована, то имеет смысл использовать рекомендации, например, методологии IDEF0. Мы же воспользуется результатами рассуждений предыдущего пункта, а итоговый перечень компонент представим таблицей 3.6.

Таблица 3.6 — Набора компонент методологии DFD для учебной задачи

Компонент DFD	***Семантика компонента***
Процесс «Ответить на вопрос студента»	Функциональный блок контекстной диаграммы, соответствующий функциональному блоку A23
Хранилище данных «БД вопросов и ответов»	Внутренний элемент ИС, семантически соответствующий некоторой таблице базы данных
Внешняя сущность «Студенты с доступом»	Источник генерации потока данных «Вопросы студентов»
Внешняя сущность «Преподаватель»	Источник генерации потока данных «Тексты ответов» и потребитель потока данных «Тексты вопросов»
Поток данных «Вопросы студентов»	Связь: «Студенты с доступом» - Процесс
Поток данных «Сообщения ответов»	Связь: Процесс - Хранилище данных
Поток данных «Тексты вопросов»	Связь: Процесс - «Преподаватель»
Поток данных «Тексты ответов»	Связь: «Преподаватель» - Процесс
Поток данных «Вопросы на регистрацию»	Связь: Процесс - Хранилище данных
Поток данных «Вопросы для ответа»	Связь: Хранилище данных - Процесс

Примечание — Обратите внимание, что в методологии DFD не допускаются прямые связи между внешними сущностями и базами данных. Другими словами — хранилища данных и внешние сущности имеют связи только с процессами.

Завершение процесса формализации компонент контекстной диаграммы должно завершаться формулировкой цели и точки зрения создаваемого структурного проекта.

ЦЕЛЬ: Структурное описание блока A23 в нотации DFD.

ТОЧКА ЗРЕНИЯ: Заказчик проекта ИС ЭЖР.

Используя проведённую формализацию компонентов DFD учебной задачи, перейдём к этапу реализации самой контекстной диаграммы.

3.5.4 Реализация контекстной диаграммы в системе Ramus Educational

Учитывая, что студент Петров И.В. использует инструментальное средство Ramus Educational, то у него имеются два варианта решения поставленной выше задачи:

а) *первый вариант* — взять диаграмму блока A2, показанную выше на рисунке 3.43, выделить на ней блок A23 и создать декомпозицию этого блока, указав, что используется нотация DFD;

б) *второй вариант* предполагает открытие нового проекта с созданием контекстной диаграммы в нотации DFD и проведение всех необходимых этапов её декомпозиции.

Примечание — Мы приведём пример использования второго варианта, одновременно демонстрируя все особенности реализации проекта в нотации DFD.

Запустив инструментальную среду системы Ramus, выберем режим «Создание нового проекта» и, в мастере «Свойства проекта», укажем тип нотации DFD, как это показано на рисунке 3.44.

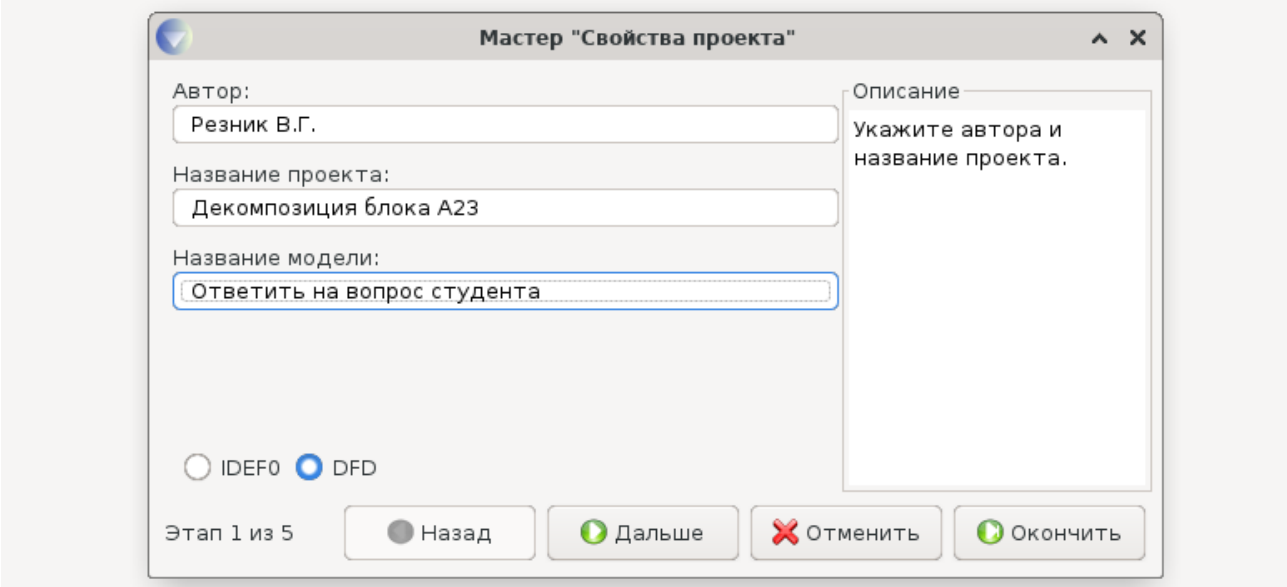


Рисунок 3.44 — Создание нового проекта в нотации DFD

Нажав кнопу «Дальше», переходим к этапу 2 и указываем, где используется проект, например, как показано на рисунке 3.45.

Нажимая кнопку «Дальше», перейдём к этапу 4, где в мастере «Свойства проекта» заполним список классификаторов, указав в них имена хранилища данных и внешних сущностей из таблицы 3.6. Результат этого этапа отображён на рисунке 3.46.

Классификатор — это набор уникальных названий объектов, являющихся особенностью реализации системы Ramus и используемых этот набор для различных целей. Имена классификаторов присваиваются *хранилищам* и *внешним сущностям*.

Для целей проектирования ИС — классификаторы можно рассматривать как *таблицы будущих баз данных*. Для этого имена классификаторов рассматриваются как названия соответствующих таблиц, а *атрибуты классификаторов* могут рассматриваться как имена и типы столбцов, указанных таблиц.

Атрибуты — единый дополнительный список объектов системы Ramus, каждый из которых создаётся как пара: «Имя атрибута» : «Тип атрибута». В дальнейшем атрибуты могут распределяться (связываться) с классификаторами.

Примечание — При создании нового проекта необходимо создать хотя бы один классификатор, иначе файл проекта может создаться ошибками.

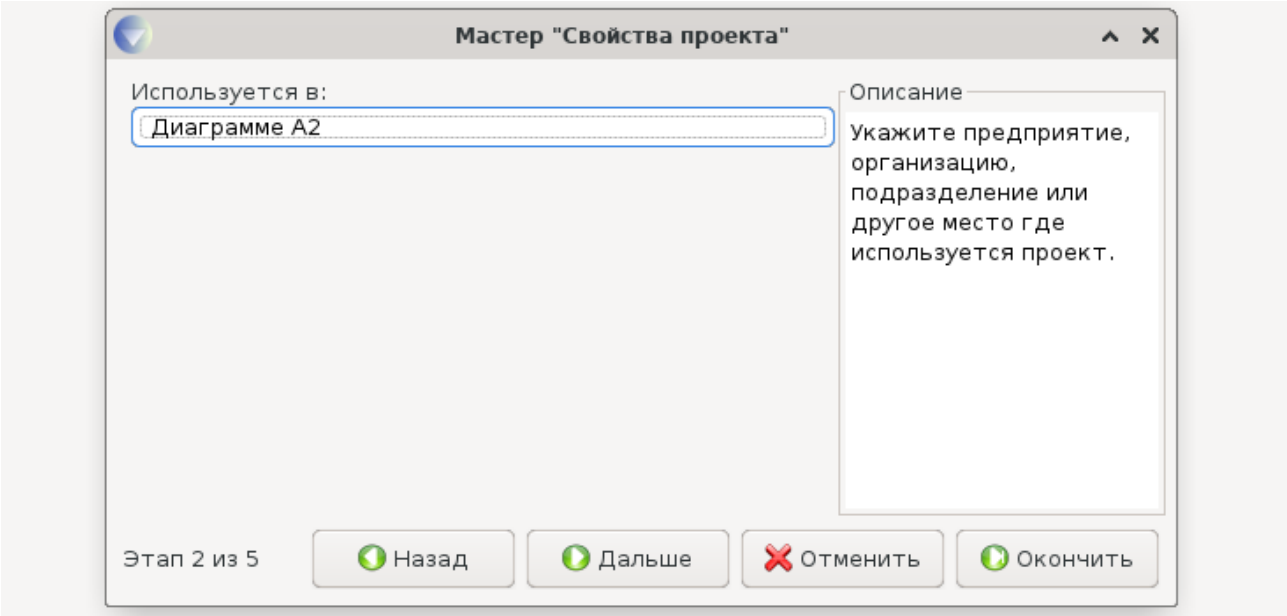


Рисунок 3.45 — Второй шаг создания проекта

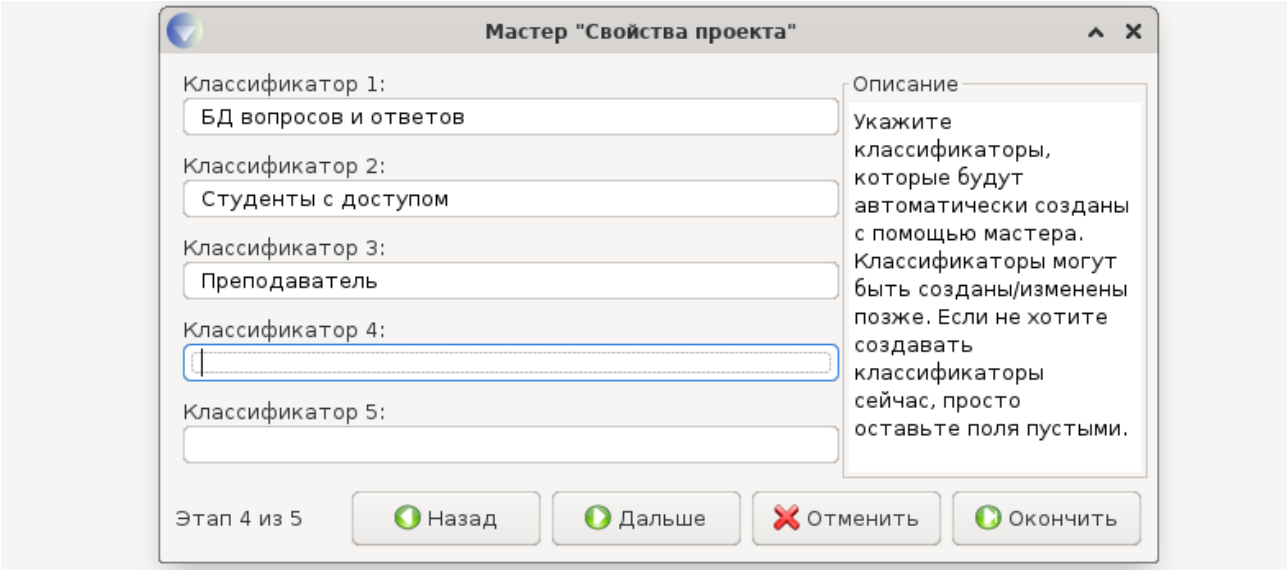


Рисунок 3.46 — Создание классификаторов для хранилища данных и внешних сущностей

После записи классификаторов можно нажать кнопку «Окончить» и система Ramus откроет пустое окно проекта.

Обязательно нужно сохранить созданный проект, нажав комбинацию клавиш ***Ctl-S***. В результате появится окно, показанное на рисунке 3.47. После нажатия клавиши «Сохранить», созданный проект будет сохранён в файле «Пример блока A23.rsf».

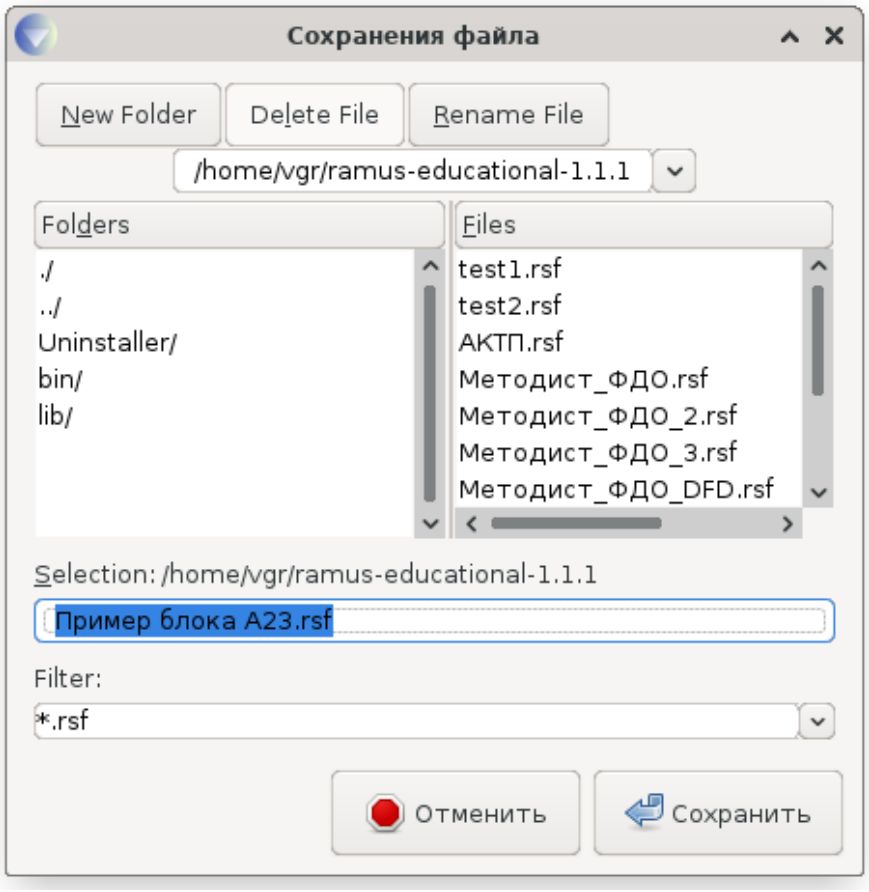


Рисунок 3.47 — Окно сохранения проекта в системе Ramus

Сохранив созданный проект, приступаем к размещению на диаграмме компонент DFD согласно информации размещённой в таблице 3.6.

Сначала размещаем компоненты процесса, хранилища данных и внешних сущностей.

Затем хранилище данных и внешние сущности привязываем к уже созданным классификаторам. Для этого:

а) выделяем курсором мыши нужный компонент и активируем контекстное меню правой кнопкой мыши;

б) в появившемся контекстном меню выбираем пункт «Редактировать активный элемент» и в появившемся окне «Свойства DFD объекта» нажимаем кнопку «Задать DFD объект»;

в) в новом окне «Выберите классификатор» выделяете мышкой нужный классификатор и закрываете окна, нажимая на клавиши «OK».

В результате указанных действий, все хранилища данных и внешние сущности будут связаны с соответствующими классификаторами и получат их целочисленные имена.

Процесс создание контекстной диаграммы Завершается добавлением компонент потоков данных, цели и точки зрения. Результат построения диаграммы показан на рисунке 3.48.

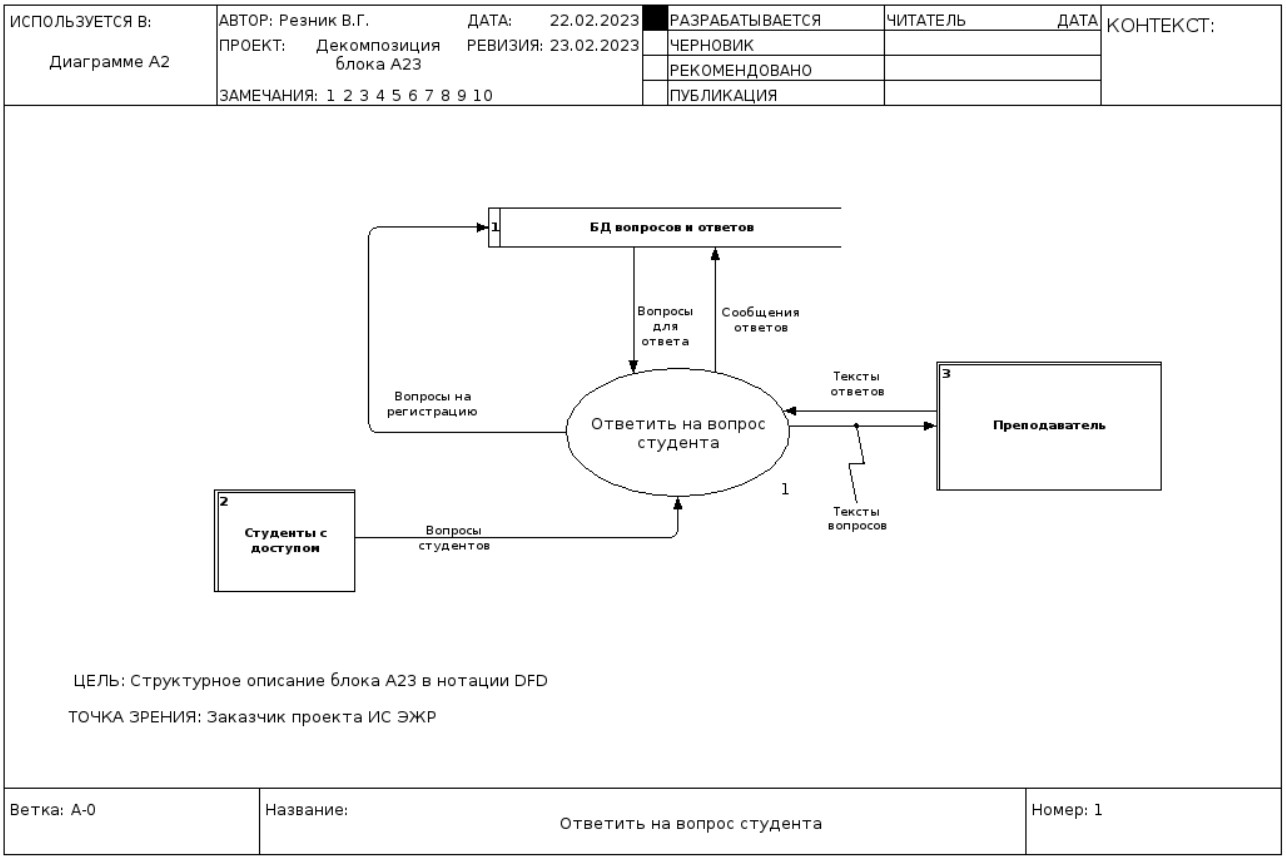


Рисунок 3.48 — Контекстная диаграмма блока A23 в нотации DFD

После согласования с заказчиком было принято решение провести декомпозицию контекстной диаграммы учебной задачи.

3.5.5 Декомпозиция контекстной диаграммы учебной задачи

В результате анализа контекстной диаграммы, студент Петров И.В. решил выделить три дочерних блока. Результат декомпозиции представлен в таблице 3.7.

Таблица 3.7 — Блоки декомпозиции процесса «Ответить на вопрос студента»

№ п/п	*Имя блока*	*Семантика блока*
1	Сохранить вопрос студента	Данный блок принимает входной поток данных «Вопросы студентов» и преобразует его в выходной поток данных «Вопросы на регистрацию», обеспечивая сохранение вопросов в хранилище данных. Такой подход обеспечивает *асинхронность функционирования внешних сущностей* «Студенты с доступом» и «Преподаватель».
2	Выбрать вопрос	Блок читает поток «Вопросы для ответа» и передаёт его во внешнюю сущность «Преподаватель».
3	Записать сообщение ответа	Блок принимает поток данных «Тексты ответов» и формирует выходной поток «Сообщения ответов», направляя его в хранилище данных «БД вопросов и ответов».

На основании информации этой таблицы, студент Петров И.В. приступил к реализации диаграммы декомпозиции учебной задачи.

Результат декомпозиции представлен на рисунке 3.49.

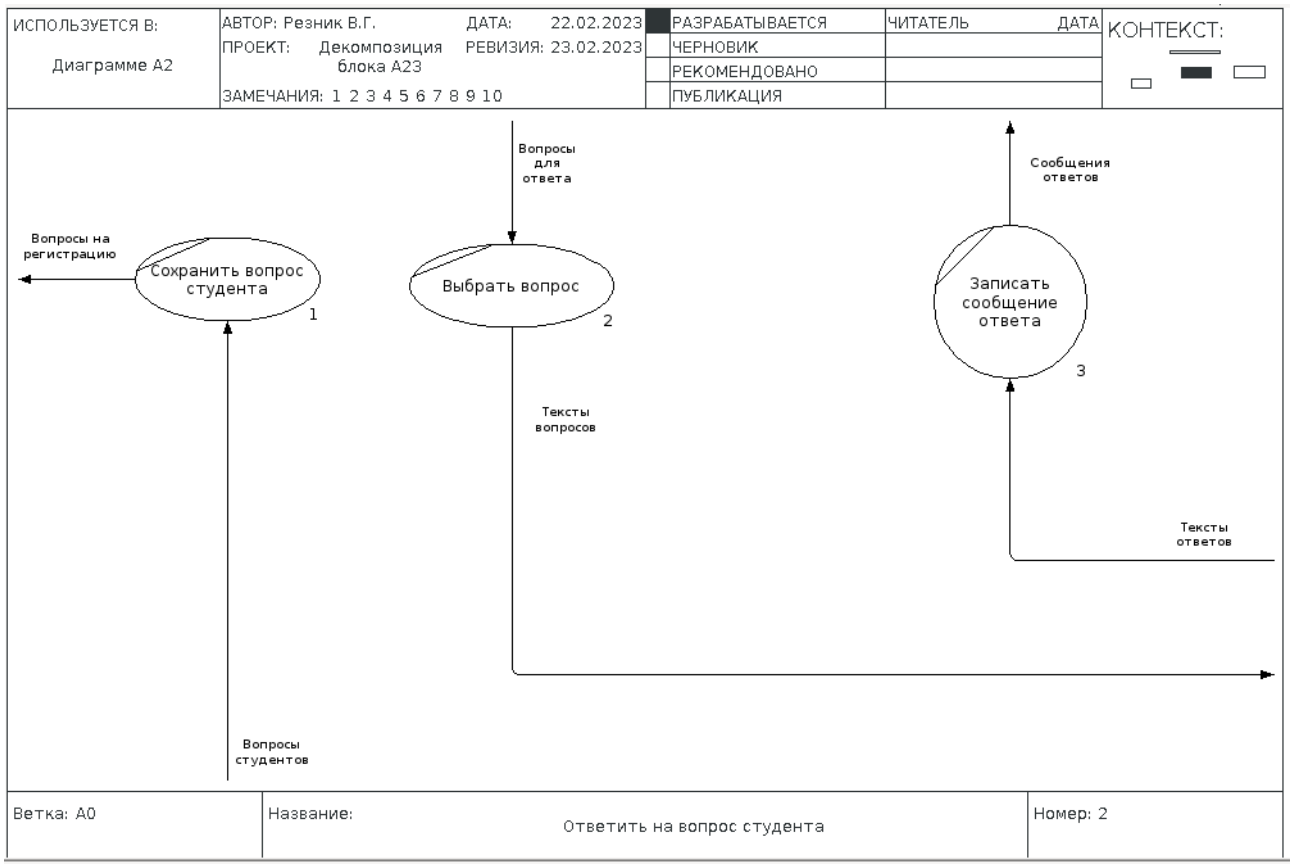


Рисунок 3.49 — Декомпозиция контекстной диаграммы с выделенными тремя процессами

Приведённая декомпозиция наглядно показывает, что:

а) процесс генерации вопросов студентами и процесс генерации ответов преподавателем осуществляются асинхронно, не задерживая и не ожидая работы друг друга; это в полной мере поясняет функционирование блока A23;

б) объекты хранилищ данных и внешних функций вынесены за поле диаграммы декомпозиции, в результате чего стрелки потоков данных направлены в разные стороны; это требует напряжение воображения читателя для восстановления всей детальной картины связей между объектами блока A23.

Примечание — В целом графическое представление рисунка 3.49 во многом даже проигрывает графическому представлению контекстной диаграммы, показанной на рисунке 3.48.

Выявленный негативный результат реализации двух диаграмм, выполненных по методологии DFD, является общим негативным свойством всех структурных диаграмм с иерархическим типом зависимостей.

Примечание — После согласования с заказчиком было принято решение о реализации одноуровневого представления диаграммы учебной задачи.

3.5.6 Итоговая одноуровневая диаграмма блока A23 в нотации DFD

При создании нового проекта система Ramus требует создание контекстной диаграммы A-0, поэтому она была реализована без внешних связей, как это показано на рисунке 3.50. Результат декомпозиции, с учётом наличия хранилища данных и внешних сущностей, представлен на рисунке 3.51. Полученный результат удовлетворил требования заказчика.

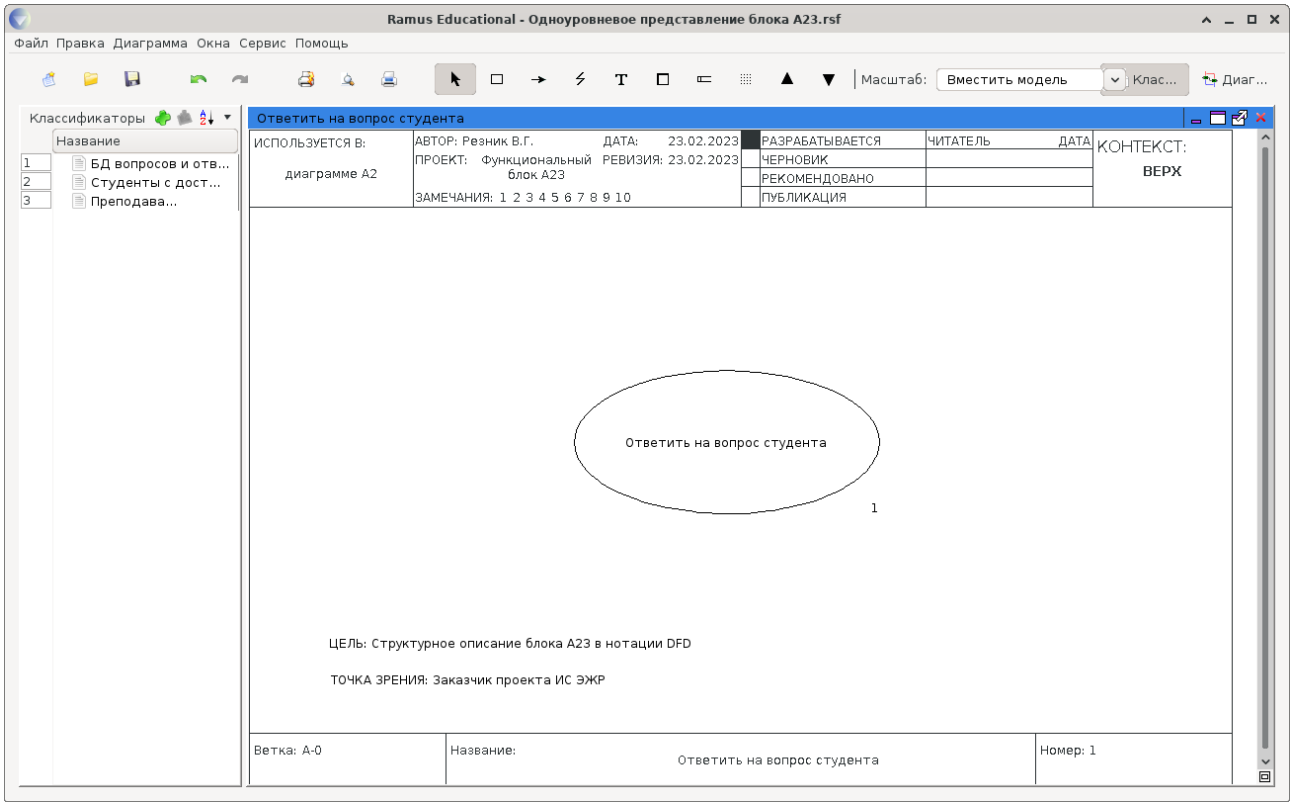


Рисунок 3.50 — Второй вариант контекстной диаграммы учебной задачи в нотации DFD

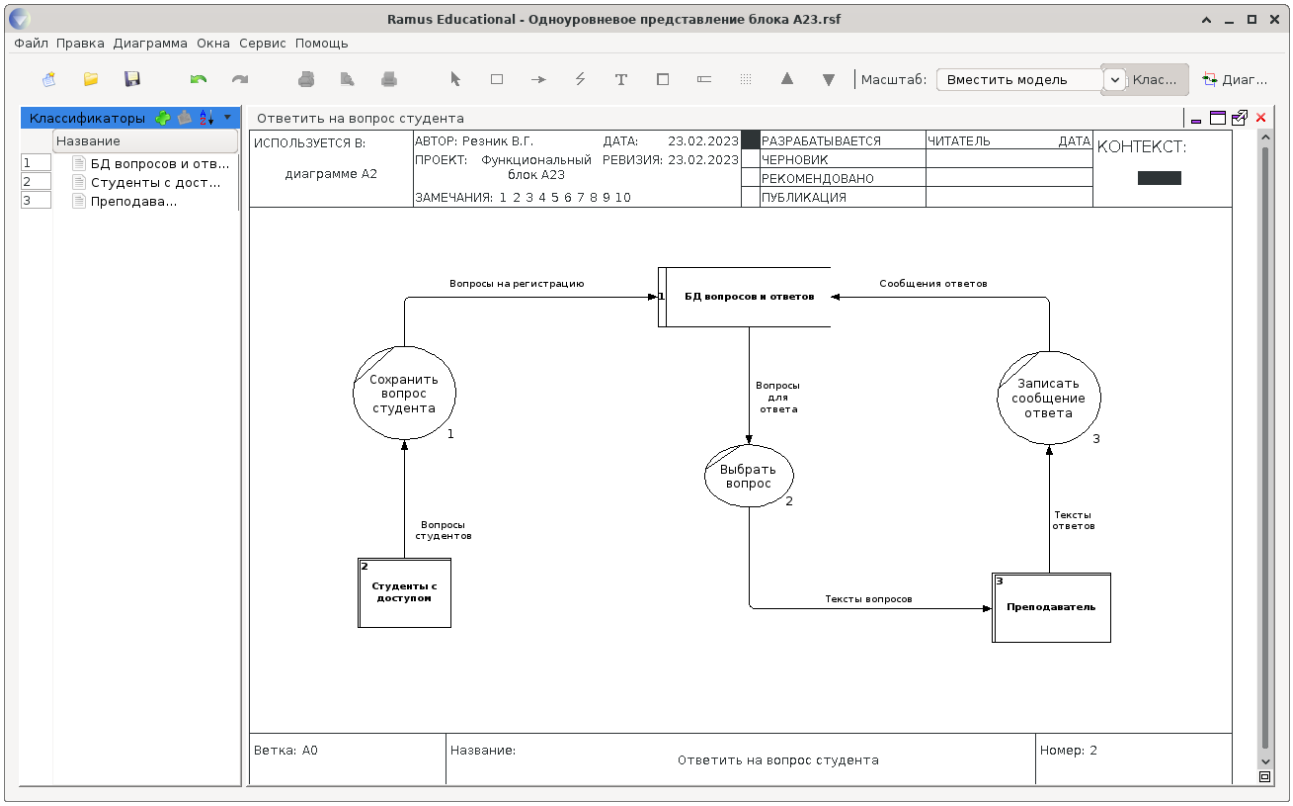


Рисунок 3.51 — Второй вариант декомпозиции учебной задачи в нотации DFD

На этом примере мы заканчиваем изучение методологии DFD и переходим к обсуждению итогов обучения по материалу всего раздела.

3.6 Выводы по применению методологий концептуального проектирования ИС

Стадия концептуального проектирования ИС предназначена для формирования ТТЗ (Тактико-Технического Задания) или ТКП (Технико-Коммерческого Предложения) с целью убедить будущего Заказчика ИС сформировать ТЗ (Техническое Задание) и заключить договор на проектирование и реализацию ИС.

Будущий (потенциальный) Заказчик воспринимает ТТЗ или ТКП как набор функций, совокупность которых должна удовлетворить его бизнес-коммерческие потребности. Именно поэтому основу стадии концептуального проектирования составляет деятельность по функциональному моделированию ИС. Более того, поскольку ИС является компонентой АС (Автоматизированной Системы), то будущий (потенциальный) Исполнитель должен использовать адекватную нотацию (формальный язык), который понятен как самому себе, так и потенциальному заказчику. Такие адекватные нотации присущи методологиям структурного моделирования: IDEF0, IDEF3 и DFD.

В общем случае, все методологии структурного моделирования связаны с описанием и декомпозицией процессов, но по разному используют свои графические нотации и интерпретацию процессов как функций.

Методология IDEF0 рассматривает функцию как *преобразователь потоков материальных и информационных объектов*, указывая в явном виде ограничения на функцию, в виде управления, и явно задавая активные субъекты (механизмы), обеспечивающие как само преобразование объектов, так и учёт ограничений. При таком потенциале IDEF0 позволяет потенциальному Исполнителю концептуально проектировать не только ИС, но и — АС. Именно поэтому методология IDEF0 является обязательной на рассмотренной здесь стадии.

Методология IDEF3 рассматривает функцию как *работу*, а процессы как *последовательности, возможно параллельные, таких работ*. На уровне концептуального проектирования как ИС, так и — АС, такая нотация может не потребоваться, поэтому данная методология рассматривается как дополнительная.

Методология DFD рассматривает функцию как *процесс, преобразующий входные потоки данных в выходные потоки данных*. В такой метасемантике эта методология казалось бы должна идеально подходить для целей концептуального проектирования ИС, но её практическое использование упирается в ряд проблем:

а) формальное отсутствие стандартизации и наличие множества графических нотаций;

б) отсутствие ограничивающих условий управления;

в) отсутствие явного выделения механизмов исполнения;

г) второстепенное значение структуры и форматов отрабатываемых потоков данных.

Все эти недостатки методологии DFD делают её второстепенной по сравнению с методологией IDEF0 и необходимой только для пояснения структурных деталей средствами, которые отсутствуют в самой методологии IDEF0.

В целом, большое многообразие проектируемых ИС требует, чтобы студент уверенно владел каждой из перечисленных методологий и максимально эффективно использовал их на стадии концептуального проектирования ИС и стадии формирования технического задания (ТЗ).

Вопросы для самопроверки

1. Что обозначают сокращения SADT и ICAM, а также в чём состоит различие обозначаемых ими понятий?
2. Каково назначение и чем заканчивается стадия концептуального проектирования?
3. Что такое — функциональное моделирование бизнес-процессов и какими средствами оно осуществляется?
4. Чем отличаются методологии IDEF0, IDEF3 и DFD и на каких стадиях проектирования ИС они применяются?
5. Перечислите стандартный состав проектной группы, рекомендуемый методологией IDEF0?
6. Что такое — контекстная диаграмма методологии IDEF0 и что на ней должно быть отображено?
7. Что такое — ICOM-кодирование и какие правила существуют для адресации блоков на диаграммах методологии IDEF0?
8. Что такое — туннелирование на диаграммах IDEF0 и как оно обозначается?
9. Перечислите обязательный набор документов для проекта IDEF0?
10. Какие типы диаграмм позволяет создавать методология IDEF3?
11. Что означает сокращение UOB применительно к методологии IDEF3?
12. Перечислите элементы графического языка описания процессов, предоставляемые методологией IDEF3?
13. Какие информационные элементы содержатся в блоке UOB?
14. Какие перекрёстки работ методологии IDEF3 вам известны?
15. Что такое — методология DFD и чем она отличается от методологий IDEF0 и IDEF3?
16. Назовите базовые элементы графической нотации методологии DFD?
17. Какие типы нотаций методологии DFD вам известны?
18. Что понимается под процессом в методологии DFD?
19. Что такое в методологии DFD — внешняя сущность и чем она отличается от потока данных?
20. Чем в методологии DFD внешняя сущность отличается от хранилища данных?

4 ОБЪЕКТНОЕ ПРОЕКТИРОВАНИЕ ИС

Напомним, что, закончив стадию концептуального проектирования, написав тактико-техническое задание (ТТЗ) на АС и проведя конкурсный отбор, заинтересованные стороны выполняют третью стадию «Техническое задание», после чего их статус официально изменяется на ***Заказчика*** и ***Исполнителя*** договора на создаваемую систему.

Фактически, после подписания «Технического задания» (ТЗ) на АС (ИС) и «Договора», ***Исполнитель*** начинает четвёртую стадию «Эскизный проект», где он, согласно уже рассмотренной ранее таблице 1.2, выполняет два этапа работ:

а) Разработка предварительных проектных решений по системе и её частям.

б) Разработка документации на АС и её части.

> Замечание — Студент, выполняя проектирование ИС по заданию ВКР, не пишет ТТЗ, ТЗ и не заключает «Договор». Тем не менее, он обязан выполнить работы, эквивалентные по содержанию, стадии «Эскизный проект», чтобы наглядно показать: базовые проектные решения, дополняющие архитектурные решения концептуального проектирования, и оценить необходимый объем работ, требующихся для создания программного обеспечения проектируемой ИС.

Безусловно на стадиях «Эскизного проекта», «Технического проекта» и «Рабочей документации» могут использоваться уточняющие методы структурного функционального моделирования, но мы считаем, что нужный уровень декомпозиций уже проведён.

> **Учебная цель** данного раздела — изучение базовых методик, обеспечивающих объектное проектирование ИС.

Основная ошибка, которую допускают большинство студентов — игнорирование результатов или вообще стадии концептуального проектирования.

Обычно, поверхностно осознав условия поставленной задачи, студент начинает объектное проектирование ИС. Для этого он обосновывает выбор языка программирования, инструментальное средство разработки программного обеспечения и СУБД. На основе выбранных инструментов начинает создаваться информационная база данных.

Как правило, выбранные инструменты и проектные действия студента осуществляются спонтанно и на основе тех знаний, которые студент уже получил в процессе обучения.

В результате, при оформлении отчёта по производственной практике или при оформлении выпускной квалификационной работы (ВКР), студент получает некоторый набор программного обеспечения и баз данных, который хоть и связан с исходной постановкой задачи, но не учитывает результаты концептуального проектирования АС.

Окончательно, стремясь удовлетворить требованиям учебного задания, студент «выдумывает» результаты концептуального проектирования и подгоняет обоснование проектных решений под уже полученные практические результаты.

Чтобы устранить описанную выше проблему, необходимо обеспечить прямую логическую связь между функциональными и объектными проектными моделями, опираясь на следующие три аспекта уже известных технологических решений:

1) *Объектные и сервисные распределенные системы* (подраздел 4.1) — как основа выбора архитектуры и инструментальных средств для реализации АС или ИС;
2) *Объектное проектирование процессов на основе языка UML* (подраздел 4.2) — как современные средства проектирования программного обеспечения;
3) *Модель «Сущность-связь»* (подраздел 4.3) — как основа для последующего проектирования баз данных.

4.1 Объектные и сервисные распределённые системы

Структурное функциональное моделирование — теоретический приём построения идеализированной целевой системы, представленной в простейших функциональных форматах, понятных как *Заказчику* так и *Исполнителю* реализуемого проекта.

Действительно, любая система интуитивно воспринимается как объект. Это восприятие не зависит от того, идёт ли речь о полноценной АС или только об отдельной её подсистеме, такой как ИС. Фактически структурное функциональное моделирование выделяет только одно (функциональное) свойство как целой системы, так и её компонент, одновременно обозначая дополнительные материальные и информационные объекты, которые «перерабатываются» обозначенными функциями.

Формально, на стадиях и этапах реализации АС или ИС необходимо «заменить» функции и подфункции целевой системы, полученные на этапе концептуального проектирования, на необходимые объекты программно-технических комплексов.

Фактически заявленный выше тезис и определяет целевую логику проектировщика АС, который на стадии «Эскизный проект» должен обоснованно определить контуры будущих объектов, функциональные свойства которых покрывают функциональные свойства элементов проекта концептуального проектирования.

Проблема реализации целевой логики проектировщика — полисемия самого понятия объект.

Поскольку семантика любой системы связана с понятием объект, то источником полисемии объекта является рекурсия декомпозиции системы подсистемы, а также технологические парадигмы использования средств вычислительной техники, кратко описанные в первом разделе данного учебного пособия:

а) *парадигма вычислительных систем* выделяет объектом программу или систему взаимодействующих программ;

б) *парадигма информационного подхода*, интерпретируемая как задача системного проектирования предметной области (см. рисунок 4.1), декларирует полное разделение семантики «Процессы» и «Объекты», оставляя за последними только две компоненты: «Модели» и «Элементная база»;

в) *парадигма АС* (АСУ), кроме трёхуровневой архитектуры, выделяет девять различных компонент обеспечения, включая информационное обеспечение (ИО АС);

г) *парадигма CALS-технологий* вообще манипулирует масштабными объектами, такими как CAE/CAD/CAM и Интегрированные информационные среды (ИИС), привязывая их к стадиям жизненного цикла изделия (ЖЦИ);

д) наконец, *парадигма распределённых объектных систем*, показанная на рисунке 4.2, предлагает свой шаблон трёхзвенной архитектуры «Клиент-сервер», включающий метапонятия «Представление», «Бизнес-логика» и «Данные», которые раскрываются как технологические метаобъекты: «Клиент», «Сервер приложений» и «СУБД».

Примечание — Очевидно, что семантика понятия «Объекты», представленная на рисунке 4.1, ориентирована только на представление и анализ метаобъекта «Данные», который представлен на рисунке 4.2, что не покрывает даже аналогичное представление объектов, принятых в современных языках ООП.

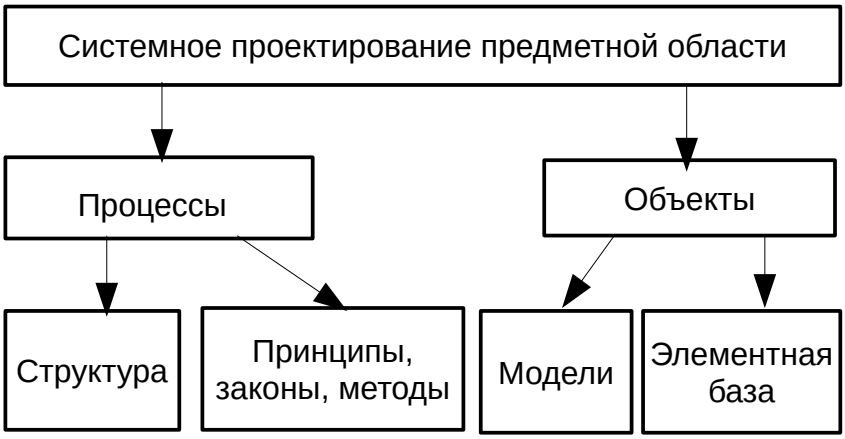


Рисунок 4.1 — Дубль рисунка 1.3, отображающего основные части системного проектирования

Рисунок 4.2 — Дубль рисунка 1.10, отображающего базовую объектную трёхзвенную архитектуру «Клиент-сервер»

Для устранения негативных последствий полисемии понятия «Объект» теория и практика анализа и реализации систем выработала два базовых подхода:

а) структурное функциональное моделирование;

б) шаблоны структурного объектного проектирования.

> **Универсальный шаблон объектного анализа и реализации систем** — «Шаблон проектирования MVC».

Шаблон проектирования MVC — специальный приём проектировщика систем, разделяющий выделенный объект анализа и реализации на три составляющие: *Model-View-Controller*, что в технологии проектирования распределённых систем представляет метамодель взаимодействия «*Проектировщика*» и «*Проектируемого объекта*» меташаблоном MVC, как это показано на рисунке 4.3, где:

а) **Проектировщик** — актор (субъект), активно воздействующий на исследуемый или проектируемый объект и способный рассматривать себя как «внешняя система» или «пользователь» объекта проектирования;

б) **Проектируемый объект** — объект декомпозиции на составляющие: *Model-View-Controller*;

в) **Model** — сущностная часть анализируемого и реализуемого объекта проектирования, доступная проектировщику только посредством взаимодействия через *Controller*;

г) **View** — преобразователь структуры и типов входной и выходной информации о компоненте *Model* в форму необходимую для проектировщика;

д) **Controller** — компонента объекта проектирования, отвечающая за взаимодействие проектировщика и составляющих *Model* и *View*.

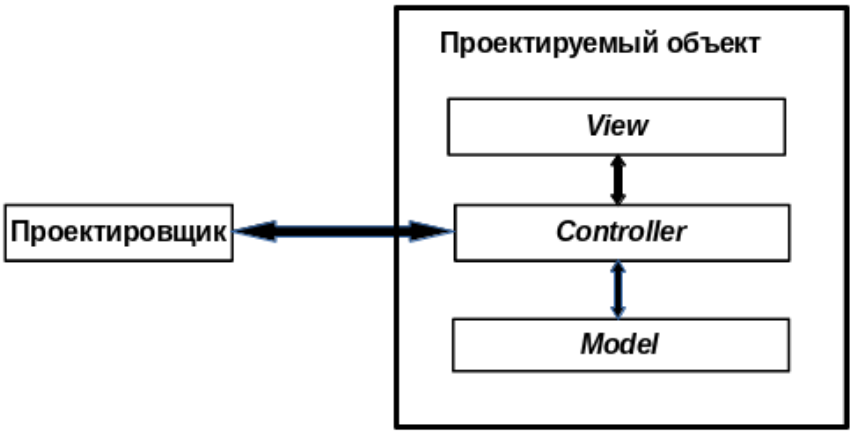


Рисунок 4.3 — Метамодель шаблона проектирования MVC

Представленный шаблон MVC задаёт нам общий теоретический конструктив, который позволяет:

а) *строить конкретизации проектируемого объекта*, уменьшая объём полисемии определения самого понятия объект;

б) *выбирать уже готовые технологические решения*, уже созданных для выделенных в проектируемом объекте компонент: Model, View и Controller.

Примечание — Развитие теории и практики распределённых систем породила множество уже готовых технологических решений, которые студент должен знать и адекватно использовать на стадиях реализации АС и ИС.

Конкретные варианты готовых технологических решений предметной области распределённых систем конкретизируем в следующих пяти пунктах, отражая историческую последовательность их появления:

а) *Технологии ООП и CORBA* (пункт 4.1.1), описывающие зарождение объектного подхода и обеспечивающие первоначальную стандартизацию технологий «Объектных распределённых систем»;

б) *Объектная технология RMI и контейнерная технология Java EE* (пункт 4.1.2), раскрывающие собственную частную реализацию «Объектных распределённых систем» средствами языка Java, а также создающие интегрированные модульные компоненты *сильно связанных* (*Strong coupling*) распределённых систем, получивших название «Контейнерные технологии Java EE»;

в) *Сервис-ориентированная архитектура слабо связанных систем* (пункт 4.1.3), поясняющая базовые принципы построения современных масштабных (*Loose coupling*) систем, ориентированных на максимальное разделение проектирования и реализации составляющих сетевой парадигмы «Клиент-сервер»;

г) *Контейнерные технологии Web-сервисов* (пункт 4.1.4), описывающие частные («облегчённые») варианты реализации компонент «Контейнерных технологий Java EE»;

д) *Проектирование ИС как Web-службы в стиле REST* (пункт 4.1.5), предлагающее вариант реализации технологии RESTfull на базе «Контейнерных технологий Java EE».

Примечание — В целом предполагается, что изложение учебного материала перечисленных выше пунктов опирается на знания, описанные в учебных пособиях [3] и [44].

4.1.1 Технологии ООП и CORBA

Формально шаблон проектирования MVC можно применить к ситуации, когда пользователь ЭВМ, включил компьютер, прошёл аутентификацию и авторизацию, после чего получил доступ к ЭВМ в виде командной строки. Тогда шаблон проектирования MVC, показанный на рисунке 4.3, можно конкретизировать рисунком 4.4.

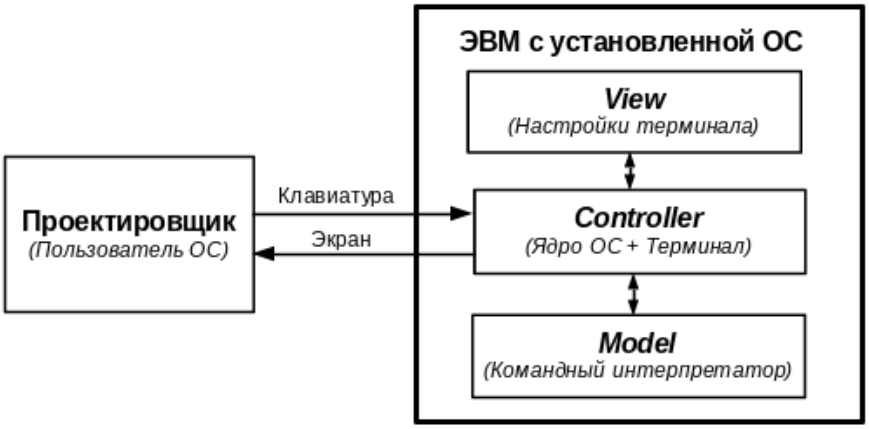


Рисунок 4.4 — Пример конкретизированной модели шаблона проектирования MVC

Представленная конкретизация шаблона MVC интересна только для учебных познавательных целей. Например, если в командной строке терминала запустить файловый менеджер *Midnight Commander*, то он заменит собой компоненту *Model* и добавит соответствующий функционал в компоненту *View*.

> **Примечание** — Технология объектно-ориентированного программирования (ООП) основана на широком использовании уже созданных программных классов, обеспечивающих реализацию нужных объектов для систем АС и ИС.

Если рассматривать целевую систему как сосредоточенную систему обработки данных (СОД), реализующую «Автоматизированное рабочее место» (АРМ) средствами технологии ООП, то простейшей конкретизацией шаблона MVC будет архитектура, показанная на рисунке 4.5.

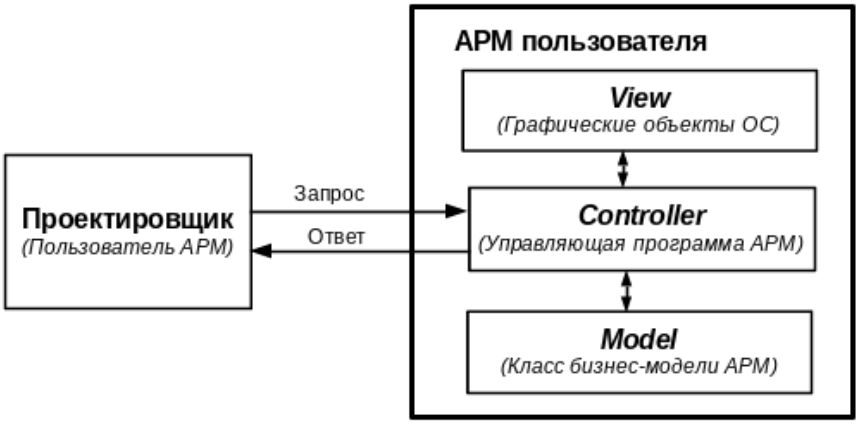


Рисунок 4.5 — Базовая конкретизация шаблона MVC для сосредоточенной объектной системы АРМ

Средства и способы реализации АРМ по шаблону рисунка 4.5 должны быть хорошо известны студенту по результатам изучения дисциплин «Объектно-ориентированное программирование» и «Основы разработки программного обеспечения». Что же касается вопроса «Является ли указанная система АС или ИС?», то ответ зависит от функционала класса *бизнес-модели АРМ*, который представлен компонентой *Model*.

> **Примечание** — В базовой конкретизации шаблона MVC, компонента *Model* может быть реализована отдельно от компоненты *Controller* и помещена в конкретную библиотеку для дальнейшего использования.

Выделение компоненты *Model* в отдельный пакет или библиотеку объектов позволяет проводить детальную декомпозицию бизнес-модели АРМ и упрощает реализацию компоненты *Controller*. Это утверждение обосновывается тем, что проектировщику, пишущему управляющую программу АРМ, достаточно знать только *интерфейсы классов*, входящих в компоненту *Model*.

> **Минимальный объём знаний**, необходимый проектировщику для практического использования объектов компоненты *Model*, — описание интерфейсов, входящих в этот компонент классов.

Интерфейс класса — специальное описание класса, в котором определены: весь состав методов, типы аргументов и возвращаемых ими значений, но отсутствует программная реализация этих методов.

> **Примечание** — Современные АС и ИС являются распределёнными системами.

В случае, когда компонента *Model* должна быть реализована на удалённой ЭВМ или требуется использовать базы данных, управляемые удалёнными СУБД, тогда *бизнес-модель* локального АРМ должна включать не только описание интерфейса вызываемой компоненты, но и — соответствующее сетевое программное обеспечение для связи и доступа к классам реальной бизнес-модели. Очевидно, что в такой ситуации разработка целевой системы значительно усложняется.

В 1989 году был создан консорциум *OMG* (*Object Management Group*), представляющий собой рабочую группу, занимающуюся разработкой и продвижением объектно-ориентированных технологий и стандартов.

Основным достижением консорциума является *проект брокерной архитектуры* взаимодействия программных объектов, получивших название *CORBA*.

CORBA (*Common Object Request Broker Architecture*) — общая архитектура брокера объектных запросов, представляющая собой технологический стандарт написания распределённых приложений. Создана для целей проектирования, разработки и развёртывания сложных объектно-ориентированных прикладных систем посредством включения в программное обеспечение «механизмов» интеграции различных изолированных систем, написанных на разных языках программирования и работающих в разных узлах сети.

> **Основная идея**, обеспечивающая интеграцию различных изолированных систем, — использование посредника (Брокера объектных запросов), позволяющего удалённым объектам посылать *заявки запросов* и получать на них *результаты ответов*, не заботясь о своём местоположении в распределённой среде и способе реализации самих удалённых объектов.

Брокер Объектных Запросов (*Object Request Broker или ORB*) — отдельная, унифицированная, сетевая программная система (сервер), поддерживающая конкретные реализации абстрактного протокола *GIOP*.

GIOP (*General Inter-ORB Protocol*) — абстрактный протокол в стандарте CORBA, обеспечивающий интероперабельность (функциональность взаимодействия) программного обеспечения брокеров и взаимодействующего с ним программного обеспечения изолированных систем.

Реализация архитектуры протокола GIOP имеет несколько базовых конкретизаций:

а) *Internet InterORB Protocol* (*IIOP или Межброкерный протокол для Интернет*) — протокол опубликованный консорциумом OMG для организации взаимодействия между различными брокерами. IIOP используется GIOP в среде сетей Интернет и обеспечивает отображение сообщений между *протоколом GIOP* и *слоем протокола TCP/IP*;

б) *SSL InterORB Protocol* (*SSLIOP*) — протокол IIOP поверх протокола SSL, поддерживающего шифрование и аутентификацию;

в) *HyperText InterORB Protocol* (*HTIOP*) — протокол IIOP поверх протокола HTTP.

Общая архитектура взаимодействия двух изолированных систем «Клиент» и «Сервер» через «Брокер CORBA» показана на рисунке 4.6.

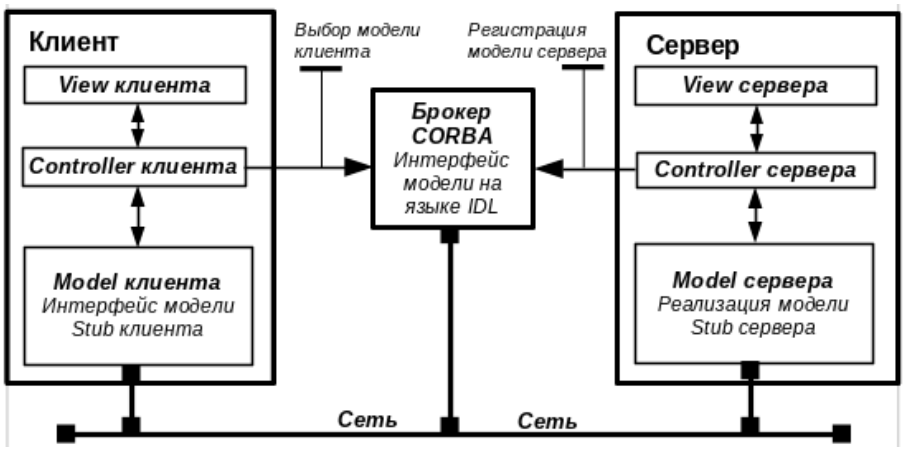


Рисунок 4.6 — Архитектура взаимодействия программ клиента и сервера через брокер CORBA

Центральное место в этой архитектуре взаимодействия программ клиента и сервера занимает сервер брокера, который позволяет программе сервера зарегистрировать на сервере брокера интерфейс реализации компоненты *Model* на языке *IDL CORBA*.

IDL CORBA (*Interface Definition Language CORBA*) — унифицированный язык описания интерфейсов распределённых объектов, разработанный рабочей группой OMG специально для обобщённой архитектуры CORBA.

> **Язык IDL CORBA** не зависит от языков реализации программ клиентов и серверов. В этом отношении он является унифицированным для всех языков программирования изолированных систем.

Применение технологии CORBA, для цели реализации программы удалённого сервера, требует выполнения следующих действий:

а) описать интерфейс компоненты *Model* на языке IDL CORBA;

б) компилировать исходный текст полученного интерфейса специальным компилятором языка IDL CORBA, который сгенерирует исходные тексты необходимого программного обеспечения на языке реализации программы сервера, включающие: описание интерфейса компоненты *Model* и тексты для «*Stub сервера*»;

в) реализовать программное обеспечение бизнес-модели сервера, согласно полученного в пункте б) исходного текста интерфейса;

г) реализовать остальные компоненты сервера и запустить его на выполнение.

В результате проведённых действий, запущенный сервер должен провести регистрацию своего интерфейса компоненты «*Model сервера*» на сервере «*Брокер CORBA*».

Обратите внимание, что интерфейс реализации «*Model сервера*» регистрируется на сервере брокера, согласно первоначального описания интерфейса на языке IDL CORBA. Это позволяет программам клиентов, реализуемых на других языках программирования, чем программа сервера, получить исходные тексты программ, включающих описание собственного интерфейса компоненты *Model* и тексты для «*Stub клиента*».

> **Примечание** — Для реализации программы клиента необходимо выполнить ту же последовательность действий, что и для программы сервера.

После запуска программы клиента, её компонента «*Controller клиента*» обеспечит связь с сервером брокера и произведёт выбор интерфейса, соответствующий интерфейсу компоненты «*Model клиента*». Если нужный интерфейс найден, то далее программа клиента работает с компонентой *Model* так, как будто реализация этой компоненты находится на компьютере клиента.

> **Технология CORBA** предназначена для реализации *сильносвязанных распределенных систем*.

Сильно связанные распределённые системы (*Strong coupling*) — распределённые системы, в которых запрашиваемые программой клиентом методы, объявленные в интерфейсе, напрямую реализуются соответствующими методами программы сервера.

Сильная связанность обеспечивает проектировщика клиентской программы иллюзией локального размещения класса на самой машине клиента. Считается, что такой подход затрудняет разработку и отладку программного обеспечения распределённых систем.

Завершая краткое описание технологии ООП и CORBA отметим, что изложенный здесь учебный материал студент изучает в дисциплине «Распределённые вычислительные системы» и имеет практические навыки применения этих технологий в проекции языка Java. Для уточнения конкретных неупомянутых здесь деталей, следует воспользоваться учебным пособием [3], в объёме второй и третьей тем (разделов).

4.1.2 Объектная технология RMI и контейнерная технология Java EE

Как было упомянуто в предыдущем пункте, объектно-ориентированный язык программирования (ООП) Java поддерживает реализацию изолированных узлов клиентов и серверов в рамках технологии CORBA. Для этого он:

а) поддерживает реализацию по крайней мере протокола IIOP, поставляемую в рамках системы исполнения (JRE), в виде пакета *org.omg.CORBA*;

б) имеет собственный компилятор *idlj*, обеспечивающий компиляцию исходного текста интерфейса на языке на языке IDL CORBA в набор исходных текстов на языке Java;

в) позволяет создавать программы клиента и сервера, обеспечивающие взаимодействие через брокер CORBA, реализованный в виде программы *orbd*.

> На языке Java реализована своя собственная технология объектных распределённых систем, получившая название — *технология RMI*.

RMI (*Remote Method Invocation*) — упрощённый (ориентированный на язык Java) вариант технологии CORBA, появившийся в 1999 году, в первой версии платформы Java EE.

Современная реализация технологии RMI использует собственный протокол *JRMP*, основанный на протоколе IIOP и имеет собственный брокер объектных запросов, реализованный в виде отдельного сервера и представленный в исполнительной среде JRE как программа *rmiregistry*. Общая схема взаимодействия программ клиента и сервера, написанных на языке Java, представлена на рисунке 4.7.

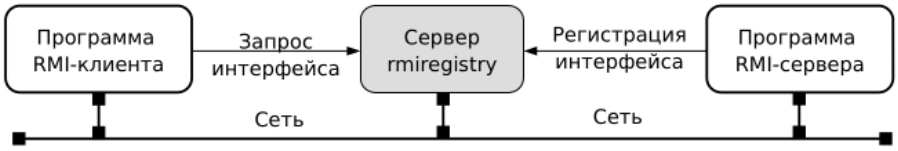


Рисунок 4.7 — Общая схема взаимодействия программ RMI-клиента и RMI-сервера

JRMP (*Java Remote Method Protocol*) — собственный протокол технологии RMI, который не требует описания интерфейсов на языке IDL CORBA, генерации «стабов клиента» (*client stub*) и «стабов сервера» (*server stub, skeleton*).

Для реализации программного обеспечения, использующего технологию RMI, платформа Java EE предоставляет инструментальный пакет *java.rmi*.

Пакет java.rmi — инструментальное средство технологии RMI, содержащее классы и интерфейсы, полностью обеспечивающие объектное проектирование *сильно связанных распределённых систем* на языке Java:

а) *описание интерфейса* компонент *Model* использует синтаксис интерфейса языка Java со следующими требованиями: целевой интерфейс должен расширять базовый интерфейс *java.rmi.Remote*, а все его методы должны быть публичными, но не должны обрабатывать уже заданное исключение *java.rmi.RemoteException*;

б) *класс компоненты Model сервера* должен включать целевой интерфейс и обязательно расширять класс *java.rmi.server.UnicastRemoteObject*;

в) *класс*, реализующий ***программу сервера***, должен использовать статические методы *bind(...)* или *rebind(...)* класса *java.rmi.Naming*, которые обеспечивают регистрацию интерфейса класса компоненты *Model сервера* на сервере брокера *rmiregistry*;

г) *класс*, реализующий ***программу клиента***, должен, при создании объектов класса компоненты Model клиента, использовать имя целевого интерфейса и статический метод *lookup(...)* класса *java.rmi.Naming* для подключения к брокеру сервера *rmiregistry*.

Таким образом, технология RMI, хотя и не является столь универсальной как технология CORBA, но значительно упрощает объём работ по созданию *сильносвязанных* распределённых систем.

> Примечание — Изложенный здесь учебный материал по технологии RMI студент подробно изучает в третьей теме дисциплины «Распределённые вычислительные системы» [3] и, кроме теоретических знаний, получает практические навыки применения этой технологии.

Настоящим технологическим прорывом в предметной области создания распределённых систем стала проектная разработка компании Sun Microsystems *контейнерных технологий Java EE*.

Контейнерные технологии — создание в среде ОС *особых инструментальных программных сред* (*контейнеров*), оснащённых специальными функциональными службами, предназначенными для упрощения разработки, отладки и эксплуатации сложных прикладных программных систем, например, АС или ИС.

Контейнерные технологии Java EE — вариант контейнерных технологий, реализующий компонентный объектно-ориентированный подход на базе виртуальной машины Java (JVM), а также функциональных платформ Java SE (Java Standart Edition) и Java EE (Java Enterprise Edition).

В целом, Java EE поддерживает четыре вида контейнеров (два «клиентских» и два «серверных»), показанных на рисунке 4.8 и предназначенных для обслуживания различных видов компонент:

а) *EJB-контейнер* — создаёт среду для серверных компонент *типа EJB*, предназначенных для реализации бизнес-логики удалённых объектов распределённых приложений; его функционирование обслуживается множеством сервисных служб, обеспечивающих доступ к нему приложений других контейнеров, а также взаимодействие с различными реализациями СУБД;

б) *Веб-контейнер* — среда, предоставляющая базовые серверные службы для управления и исполнения Веб-компонентов: EJB Light, Servlet и JSF;

в) *Контейнер клиентского приложения* — среда, включающая набор Java-классов, библиотек и других файлов, необходимых для реализации клиентских приложений и доступа к компонентам серверных контейнеров по протоколам IIOP, RMI, HTTP и HTTPS;

г) *Контейнеры апплетов* — среды, которые в настоящее время, хоть и ограничено, но реализуются большинством браузеров.

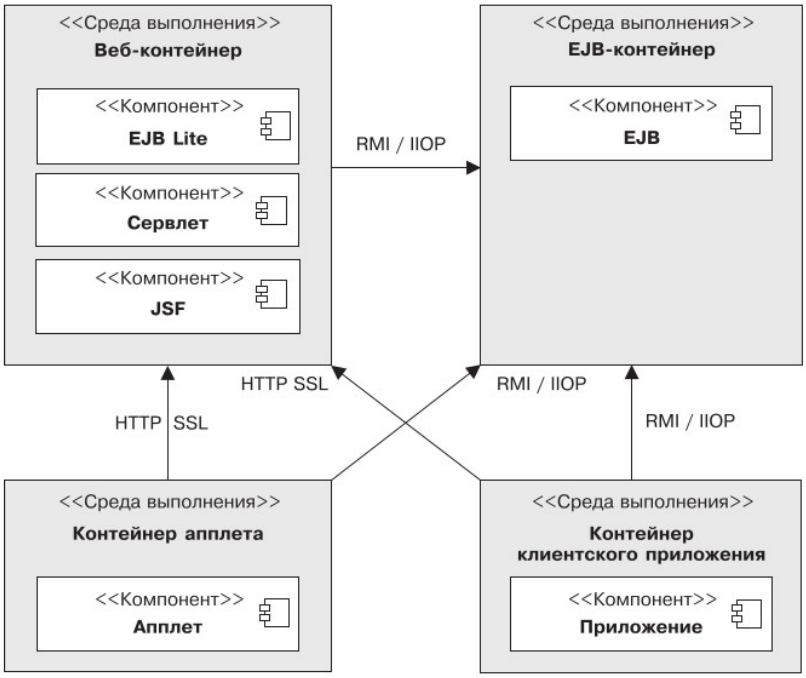


Рисунок 4.8 — Стандартные контейнеры Java EE [45]

Компоненты Java EE — специальные шаблоны классов, предназначенные для разработки распределённых приложений и прозрачно обслуживаемые различными сервисами контейнеров.

Общий набор сервисов контейнеров для Java EE версии 7 показан на рисунке 4.9 и демонстрирует всю мощь служебного программного обеспечения уже доступного проектировщикам ИС и АС для реализации прикладных систем различного назначения.

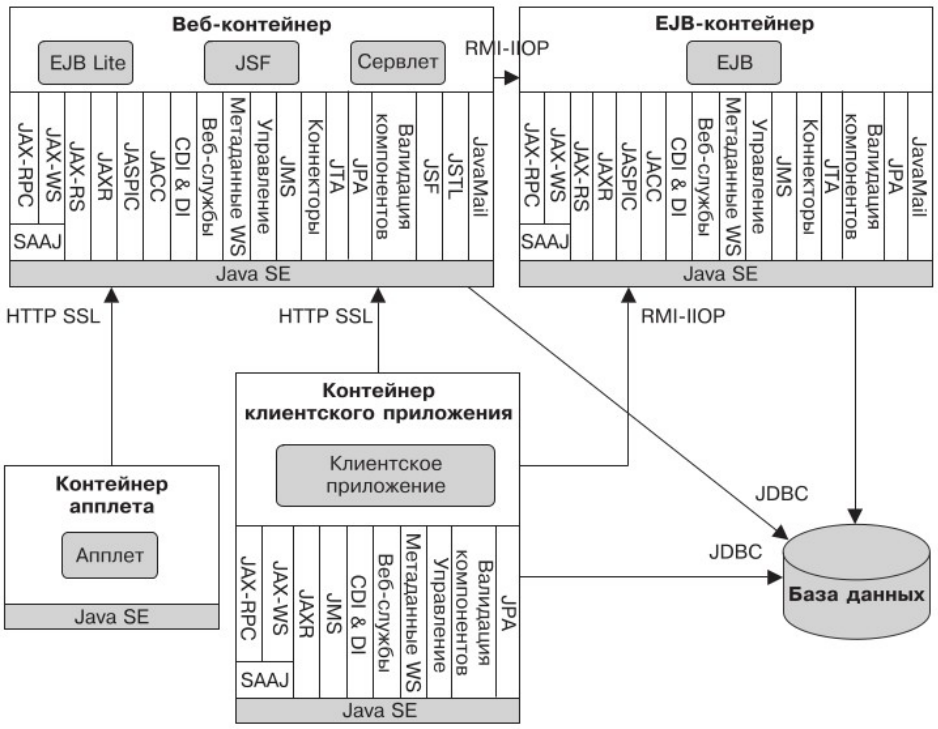


Рисунок 4.9 — Общий набор сервисов контейнеров Java EE версии 7 [45]

Базовым компонентом технологии Java EE безусловно является компонента EJB, обеспечивающая необходимый функционал для работы с базами данных.

EJB (*Enterprise Java Beans*) — корпоративная компонента предназначенная для реализации классов, непосредственно взаимодействующих с СУБД баз данных и использующая в процессе своего функционирования следующие наиболее важные сервисы:

а) **JTA** (*Java Transaction API*) — сервис управления транзакциями при взаимодействии с различными бизнес-приложениями и СУБД;

б) **JPA** (*Java Persistence API*) — сервис, обеспечивающий объектно-реляционные отображения (ORM, Object-Relational Mapping) объектов языка Java в модели реляционных баз данных;

в) **CDI & DI** (*Context and Dependency Injection & Dependency Injection*) — сервис, обеспечивающий для создаваемого компонента *контекст* и *внедрение зависимостей*.

Контейнерные технологии Java EE обеспечивают реализацию не только сильно связанных, но и *слабосвязанных распределённых систем*.

4.1.3 Сервис-ориентированная архитектура слабо связанных систем

Постепенно сильная связанность технологий CORBA и RMI стала создавать проблемы автоматизации процессов принятия решений бизнесом, поскольку сложность реализуемых этими технологиями систем не позволяла проектировать их с нужным качеством и за приемлемое для этого время.

Действительно, *бизнес-парадигма архитектуры предприятия*, показанная на рисунке 4.10 [46], требовала создания уже готовых решений, которыми могли бы воспользоваться бизнес-руководители предприятия, не дожидаясь, пока ИТ-специалисты реализуют нужные классы объектов, протестируют их и создадут приемлемое программное обеспечение.

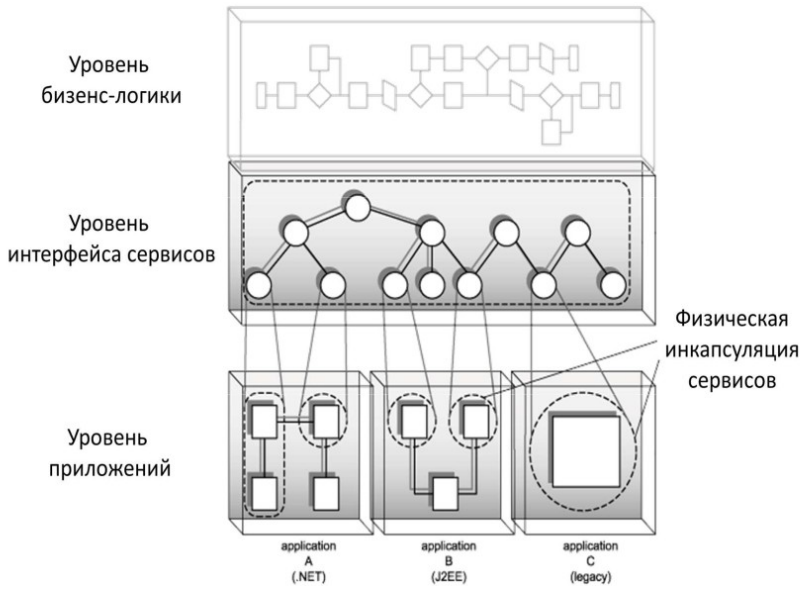


Рисунок 4.10 — Бизнес-парадигма архитектуры предприятия [46]

Бизнес-парадигма архитектуры предприятия — трёхуровневая иерархическая архитектура реализации автоматизированных систем (АС), выделяющая уровни: *бизнес-логики*, *интерфейса сервисов* и *уровень приложений*.

Уровень бизнес-логики — верхний уровень иерархии бизнес-парадигмы АС предприятия, соответствующий уровню принятия решений бизнес-руководителями предприятия, которые генерируют бизнес-процессы и становятся *потребителями сервисов* (***Service Consumer***).

Уровень интерфейса сервисов (*Service Interface*) — средний уровень иерархии бизнес-парадигмы АС предприятия, предназначенный для взаимодействия бизнес-руководителей и ИТ-специалистов предприятия.

На этом уровне обеспечиваются свойства видимости сервисов (***Visibility***), перспектива возможного использования сервисов (***Interaction***) и фактического получения результата услуги сервиса (***Real world effect***). На этом уровне ИТ-специалисты также получают статус провайдеров услуг (***Service Provider***).

Уровень приложений — нижний уровень иерархии бизнес-парадигмы АС предприятия, соответствующий уровню ИТ-специалистов, которые обеспечивают реализацию сервисов.

Этот уровень считается «невидимым» для бизнес-руководителей предприятия, поэтому он может реализовываться различными способами, например:

а) представлять слабосвязанные или сильносвязанные системы;

б) быть распределёнными или сосредоточенными системами;

в) принадлежать самому предприятию или находиться в «облаке» (*cloud computing*) на аутсорсинге (*outsourcing*).

> **Бизнес-парадигма архитектуры предприятия** реализуется теоретическими положениями и практическими решениями *сервис-ориентированных систем*.

Сервис-ориентированные системы — распределённые вычислительные системы, реализованные по теоретическим принципам сервисно-ориентированной архитектуры построения прикладных систем.

Сервис-ориентированная архитектура (*Service Oriented Architecture*, *SOA*) — это парадигма для организации и использования распределённых возможностей вычислительных систем, которые могут находиться под контролем различных доменов собственности и предоставлять единые средства для предложения, обнаружения, взаимодействия и использования возможностей для получения желаемых результатов, соответствующих измеримым предварительным условиям и ожиданиям функционирования прикладных систем.

Концептуальная идея сервис-ориентированной архитектуры, представленная автором публикации [47], показана на рисунке 4.11.

Рисунок 4.11 — Концептуальная идея сервис-ориентированной архитектуры [47]

Парадигма SOA предложила разработчикам систем новый уровень абстракции по сравнению с абстракцией классов, предлагающей только уровень объектов. Согласно рисунку 4.11, новый уровень абстракции разделяется на следующие направления конкретизации:

а) **BPM** (*Business process management*, *управление бизнес-процессами*) — концепция процессного управления организацией, рассматривающая бизнес-процессы как ресурсы;

б) **EAI** (*Enterprise Application Integration*) — общее название сервиса интеграции прикладных систем предприятия;

в) **AOP** (*Aspect-Oriented Programming*) — аспектно-ориентированное программирование; парадигма программирования, основанная на идее разделения функциональности программы на модули;

г) **Web-сервисы** (*Web-services*) — web-службы со стандартизированными интерфейсами, адресуемые уникальными URI/URL-адресами.

Дополнительно, в концептуальном плане проектирования и реализации систем, SOA выделяет следующие понятийные составляющие:

а) **Сервисный компонент** (или сервисы), которые описываются программными компонентами и обеспечивают прозрачную сетевую адресацию.

б) **Интерфейс сервиса**, который обеспечивает описание возможностей и качества предоставляемых сервисом услуг. В таком описании определяется формат сообщений для обмена информацией, а также входные и выходные параметры методов, поддерживаемых сервисным компонентом.

в) **Соединитель сервисов** — это транспорт, обеспечивающий обмен информацией между отдельными сервисными компонентами.

г) **Механизм обнаружения сервисов**, который предназначен для поиска сервисных компонентов, обеспечивающих требуемую функциональность сервиса.

> **Сервис-ориентированная архитектура систем** предлагает и реализует *слабую связанность* своих компонент.

Действительно в модели взаимодействия «Клиент-сервер», парадигма SOA предлагает клиенту «*Интерфейс сервиса*», вместо «*Интерфейса классов*», предлагаемых технологиями объектного подхода: CORBA и RMI.

Интерфейс сервиса — это ссылка на сервис, а не на удалённый объект, требуемый клиенту. Благодаря такой архитектуре собственно и реализуется идея уровня интерфейса сервисов, соответствующая бизнес-парадигме архитектуры предприятия, показанной ранее на рисунке 4.10.

> **Исторически** и, в прикладном плане, наибольшую популярность получили технологии Web-сервисов.

В 1998 году Дейв Винер, сотрудник компании UserLand Software, опубликовал протокол ***XML-RPC***, предназначенный для вызова удалённых процедур в формате XML-сообщений поверх протокола HTTP. Сам протокол разрабатывался по заказу корпорации Microsoft для обеспечения решения задач в области электронной коммерции. Последующие доработки этого протокола привели к созданию протокола ***SOAP*** (*Simple Object Access Protocol*).

Известны две спецификации этого протокола, опубликованные консорциумом W3C:

а) Simple Object Access Protocol (SOAP) 1.1;

б) SOAP Version 1.2 Part 0: Primer (Second Edition).

В марте 2001 года и в июне 2007 года, консорциум W3C публикует спецификации ***WSDL*** (*Web Service Description Language*), что обеспечивают описание сервисов в формате XML-документов и позволяет их публикацию средствами web-технологий.

Считается, что в августе 2000 года была реализована первая система ***UDDI*** (*Universal Description Discovery & Integration*) — инструмент для размещения описаний описаний WSDL. В настоящее время доступно описание версии UDDI 3.0.2, которое находится под контролем глобального консорциума ***OASIS*** (*Organization for the Advancement of Structured Information Standards*).

> **Примечание** — Более подробно многие аспекты теории и технологии парадигмы SOA изучаются в магистерской дисциплине «Распределённые сервис-ориентированные системы» [44]. Далее мы конкретизируем применение этой парадигмы для нужд реализации информационных систем (ИС).

4.1.4 Контейнерные технологии Web-сервисов

Показанные ранее рисунки 4.8 и 4.9 представляют нам стандартные контейнеры и общий набор служебных сервисов технологии Java EE. Хорошо видно, что проектировщикам и разработчикам автоматизированных и информационных систем предлагаются уже готовые:

а) **Веб-контейнер** — серверная среда для реализации компонент *EJB Lite*, *Сервлет* и *JSF*;

б) **Служебный сервис** — сервис серверной среды Веб-контейнера, предоставляющий многообразный функционал реализации прикладных сервисных компонент;

в) **Протоколы HTTP и HTTPS** — стандартизированный транспортный инструмент для доступа приложений клиентов к среде серверов приложений.

Уже наличие перечисленного инструментария позволяет с успехом выполнить этап эскизного проектирования АС или ИС.

Простейшие ИС могут быть созданы посредством реализации компонента *Сервлета*.

В 1997 году, компания Sun Microsystems предложила *технологию сервлетов*, опубликовав их спецификацию версии 1.0 и реализовав эту спецификацию на языке Java в виде пакета *javax.servlet*. В дальнейшем был разработан пакет *javax.servlet.http*, обеспечивающий интерфейсы и классы для создания высокопроизводительных Web-серверов.

В 1999 году, Apache Software Foundation реализовала спецификацию HTTP-сервлетов в широко известном Web-сервере *Apache Tomcat*.

Apache Tomcat (в старых версиях – *Catalina*) — контейнер сервлетов с открытым исходным кодом, реализующий спецификации *HTTP-сервлетов* и *JavaServer Pages* согласно общей метамодели шаблона проектирования MVC.

Конкретизация этой архитектуры, применительно к серверу Apache Tomcat, показана на рисунке 4.12.

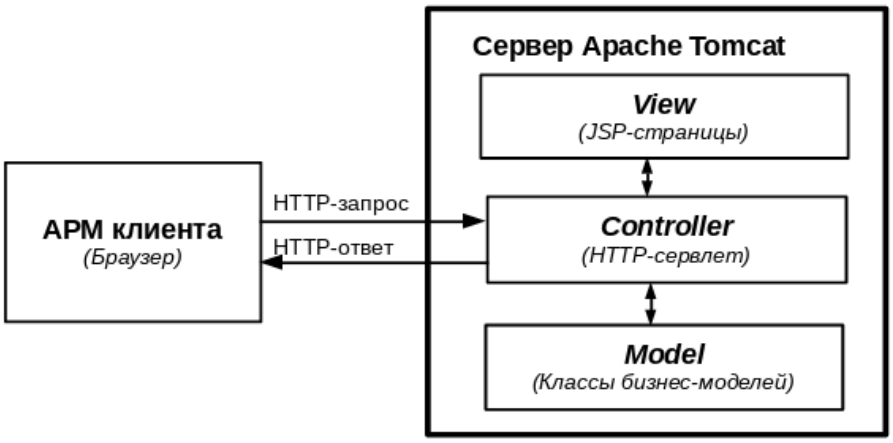


Рисунок 4.12 — MVC-архитектура взаимодействия приложения клиента и сервера Apache Tomcat

Для проектировщика информационных систем (ИС), идея шаблона MVC, реализуемая сервером Apache Tomcat, состоит в следующем:

а) компонента ***Controller*** (сервлет) реализуется публичным Java-классом, который должен расширять класс *HttpServlet*;

б) различные ***HTTP-запросы*** АРМ-клиента (*GET*, *POST*, *DELETE* и другие) могут быть обработаны соответствующими методами сервлета: *doGet(...)*, *doPost(...)*, *doDelete(...)* и другие аналогичные методы, причём в качестве аргументов им передаются объекты классов *HttpServletRequest* и *HttpServletResponse*;

в) объект типа ***HttpServletRequest*** содержит как данные от АРМ клиента, так и может дополняться данными классов компоненты ***Model***, которые проектировщик должен сам реализовать;

г) объект типа ***HttpServletResponse*** содержит результирующий ответ для АРМ клиента;

д) для обращения к компоненте ***View***, контейнер сервера предоставляет специальный класс типа *RequestDispatcher*, с помощью которого ***Controller*** может обратиться к нужной проектировщику JSP-странице и отправить результат такого обращения клиенту, в виде *HTTP-ответа*;

е) компонента ***View*** реализуется проектировщиком в виде набора *JSP-страниц*, представляющих собой текстовые файлы общего формата XHTML;

ж) ***JSP-страницы*** — специальный формат XHTML, допускающий программные вставки на языке Java и доступ к объектам запроса и ответа, формируемых сервлетом.

Технология сервлетов для каждого АРМ клиента поддерживает отдельную сессию.

Первоначально технология сервлетов предназначалась для создания высокопроизводительных Web-серверов, генерирующих и кеширующих динамические Web-страницы. Apache Tomcat является примером одного из первых таких серверов, который стал использовать контейнерные технологии предлагаемые проектом Java EE.

В частности, инструментальные средства разработки Eclipse EE имеют специальный тип проектов «Dynamic Web Project», способный инкапсулировать в проект дистрибутив сервера конкретного сервера Apache Tomcat, а затем обеспечить разработку и тестирование приложения, созданного по схеме архитектуры MVC.

Примечание — В учебной дисциплине «Распределённые вычислительные системы», студенты подробно изучают технологию сервлетов, поэтому приведённого выше описания вполне достаточно для самостоятельного для понимания и применения этой технологии в задачах проектирования информационных систем (ИС).

С 1999 года, Sun Microsystems стала формировать отдельную платформу для разработки приложений уровня предприятия — J2EE или Java EE. Новые контейнерные технологии стали добавляться к серверу Apache Tomcat, в результате чего стал формироваться дистрибутив сервера приложений, названный Apache TomEE.

В марте 2004 года, компания Sun Microsystems выпустила первую версию спецификации JSF 1.0, которая должна была усовершенствовать технологию сервлетов и перевести её на компонентную основу.

JSF (*JavaServer Faces*) — спецификация для построения компонентно-ориентированных пользовательских интерфейсов для Web-приложений, написанный на языке Java.

В последующем эта спецификация несколько раз пересматривалась и усовершенствовалась. Общая архитектура JSF 2.3, выпущенная в конце марта 2017 года, была подробно описана в монографии [45] и представлена на рисунке 4.13.

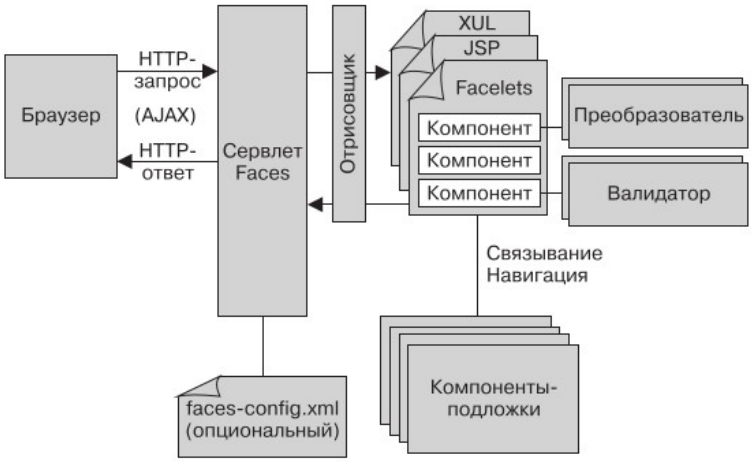


Рисунок 4.13 — Общая архитектура взаимодействия браузера и контейнерной компоненты JSF 2.3 [45]

По сути компонента JSF, показанный на рисунке 4.13 как «Сервлет Faces», представляет собой специальную реализацию сервлета, объединяющего в себе служебную функциональность всех трёх компонент шаблона MVC:

а) клиент (браузер) с помощью HTTP-запросов адресует набор XHTML-страниц, базирующихся на спецификации HTML 4.01;

б) адресуемые XHTML-страницы обрабатываются программным обеспечением, анализируя язык описания страницы (*PDL*, *Page Description Language*), обозначенный как *Facelets*;

в) на основе анализа XHTML-страниц выделяются ссылки на управляемые компоненты языка Java, обозначенные на рисунке 4.13 как «Компоненты-подложки»;

г) компоненты-подложки — разрабатываемые проектировщиком специально аннотируемые классы языка Java, которые вызываются *Сервлетом Faces*, реализуя компоненту ***Model*** (в терминологии шаблона MVC).

Преимущество технологии JSF над технологией HTTP-сервлетов и JSP состоит в возможности интеграции и отображения множества XHTML-страниц, также многоуровневая реализация компоненты *Model*.

Рекомендуемый спецификацией JSF 2.0 язык описания страниц *Facelets* позволяет объединять множество XHTML-страниц, предоставляя их браузеру как единое изображение. Это повышает повторное использование уже разработанных частей интерфейса приложений.

Что касается компонент-подложек (Java-классов), то они реализуют *Бизнес-модель* приложения, выделяя как минимум три уровня существования порождаемых ими объектов, что также наглядно показано на рисунке 4.14:

а) классы, аннотируемые как ***@RequestScoped***, порождают объекты, существующие только на время осуществления самого HTTP-запроса;

б) классы, аннотируемые как ***@SessionScoped***, порождают объекты, существующие на всё время сессии взаимодействия браузера и сервера;

в) классы, аннотируемые как ***@ApplicationScoped***, порождают объекты, существующие на время существования серверного приложения.

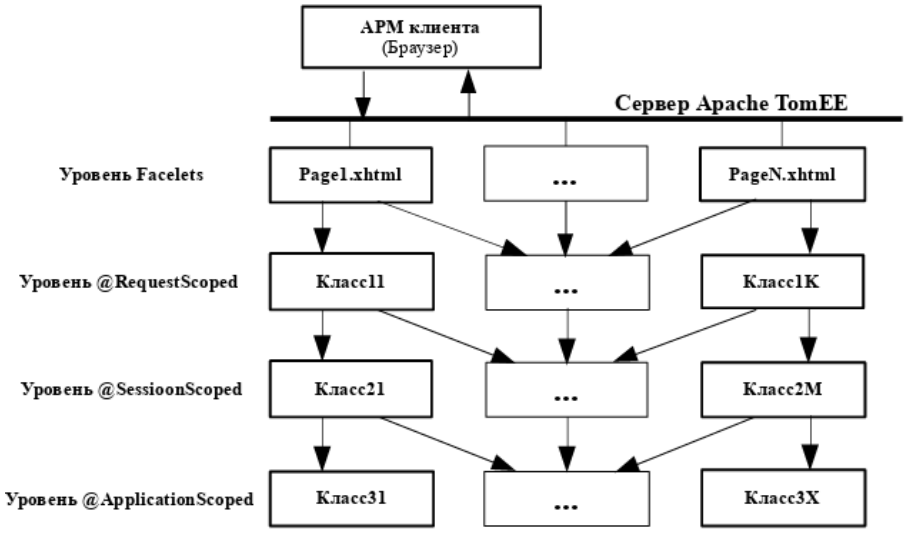


Рисунок 4.14 — MVC-архитектура взаимодействия приложения клиента и сервера согласно технологии JSF, реализуемая сервером приложений Apache TomEE

Инструментальные средства разработки Eclipse EE позволяют подключать сервер приложений Apache TomEE и, в рамках проекта «Dynamic Web Project», реализовать приложения с использованием технологии JSF.

Основные преимущества использования технологии JSF, как шаблона структурного проектирования АС и ИС:

а) возможность декомпозиции представления ***View*** отдельными компонентами *Facelets*, с последующей интеграцией этих компонент в едином представлении для *АРМ клиента* (*Браузера*);

б) возможность декомпозиции бизнес-приложения ***Model*** на матрицу различных классов, уменьшая алгоритмическую сложность реализации последовательности этапов разработки программного обеспечения;

в) возможность последовательной разработки и отладки программного обеспечения создаваемой системы.

Например, на рисунке 4.15 представлен шаблон интерфейса для уровня *Facelets*, состоящий из пяти HTML-страниц.

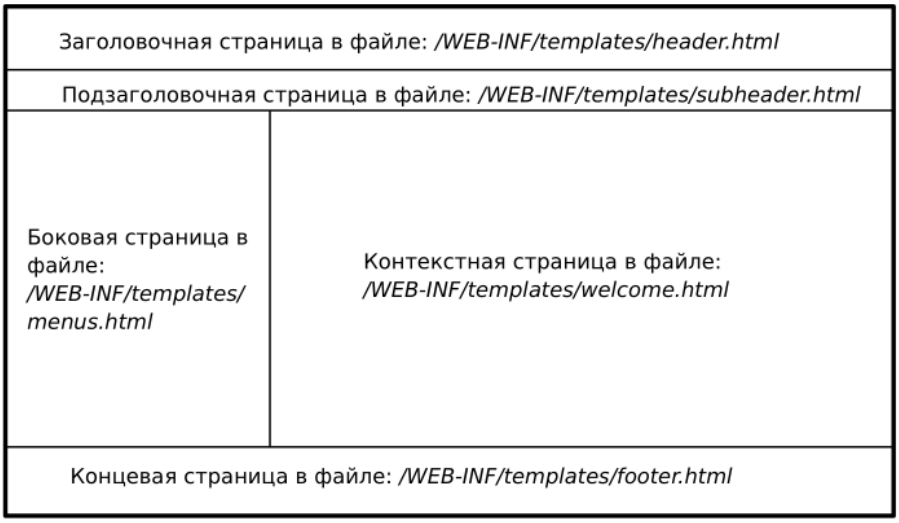


Рисунок 4.15 — Шаблон представления из пяти HTML-страниц [44]

Представленный шаблон интерфейса является *проектом интерфейса* конкретного учебного приложения, описанного источнике [44]. Первоначально он перечисляет наименования файлов, составляющих его HTML-страниц и местоположение этих страниц в пределах окна браузера, который будет обращаться к серверу Apache TomEE.

На рисунке 4.16 показана первоначальная реализация спроектированного шаблона, в пределах проекта ***labs*** инструментальной среды Eclipse EE.

Обратите внимание:

а) браузер среды Eclipse EE обращается к общедоступной, для всех браузеров, странице XHTML с именем *index.xhtml*;

б) страница ***index.xhtml***, с помощью специальных операторов языка *Facelets*, обращается, к невидимой на рисунке 4.16 и недоступной браузерам, XHTML-странице-шаблону с именем */WEB-INF/templates/lab2Templ.xhtml*, которая и организует размещение и отображение остальных пяти HTML-страниц;

в) в результате мы имеем структурную заготовку интерфейса пользователя реализуемого проекта.

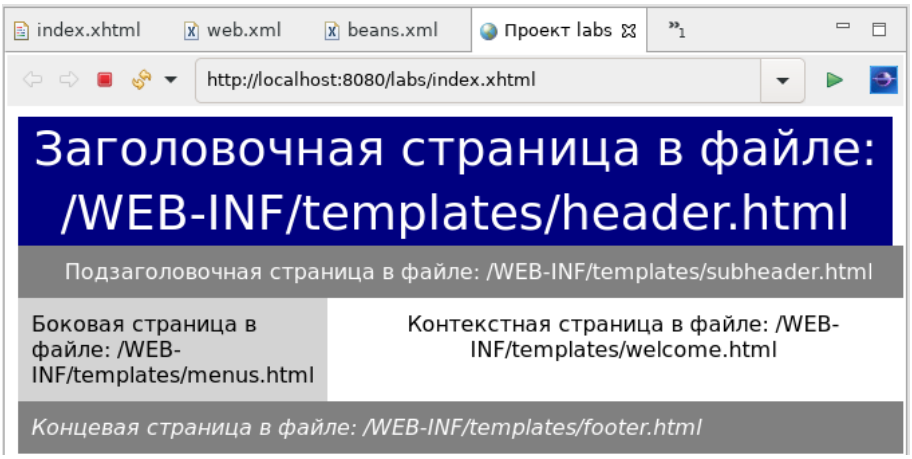


Рисунок 4.16 — Первоначальная реализация шаблона интерфейса проекта labs, состоящего из пяти проектных HTML-страниц [44]

Шаблон интерфейса на рисунке 4.16 задан набором HTML-страниц, которые имеют конкретные имена файлов и адреса размещения, в пределах реализуемого проекта. Далее:

а) для отдельных HTML-страниц создаются компоненты-подложки, реализующие функционал некоторой части бизнес-приложения ***Model*** на различных уровнях иерархии, согласно общей схеме рисунка 4.14;

б) конкретная HTML-страница заменяется XHTML-страницей, с необходимой привязкой к нужным компонентам-подложкам.

В результате последовательности указанных операций, реализуемый проект наполняется конкретным содержанием, например, как показано на рисунке 4.17.

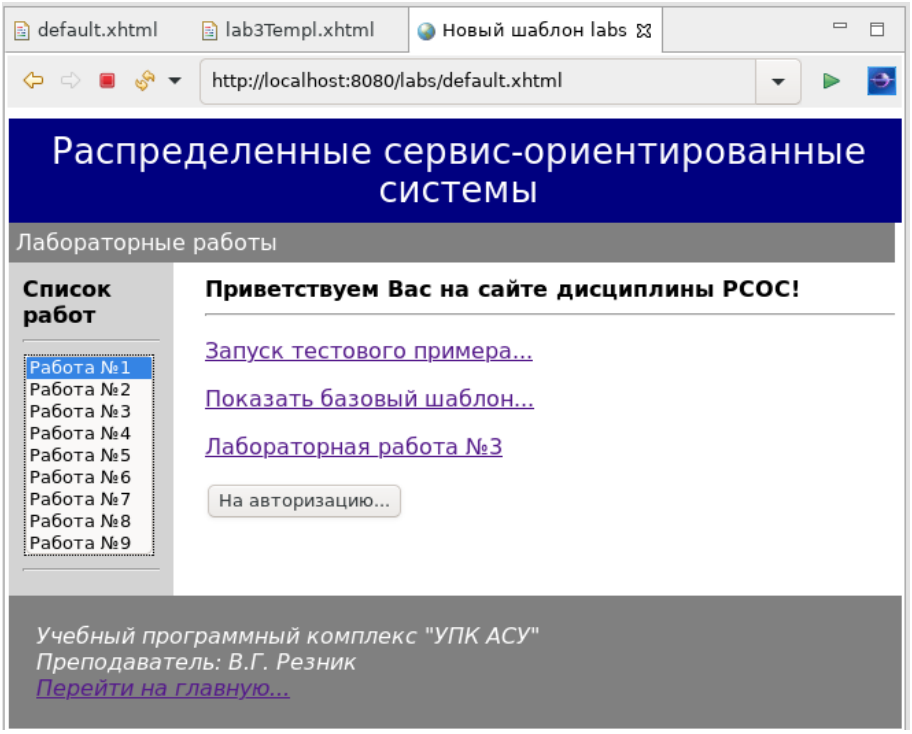


Рисунок 4.17 — Один из вариантов интерфейса проекта labs, реализуемого учебного проекта [44]

Для более подробного изучения примеров использования технологии JSF следует воспользоваться учебным пособием [44].

4.1.5 Проектирование ИС как Web-службы в стиле REST

Технология REST является современной популярной альтернативой Web-службам SOAP.

Действительно, основным недостатком модели SOA, реализуемым на основе протокола SOAP, — необходимость описания взаимодействия компонентов распределенной системы на языке WSDL, который для многих приложений является явно избыточным.

В основе языка WSDL лежит язык XML, обеспечивающий не только описание самого взаимодействия распределенных систем, но также — описание XML-схем, необходимых для синтаксического контроля таких описаний. В результате проектные решения на основе протокола SOAP становятся сложными не только для поставщиков сервисов, но главное — это является сложным для основной массы потребителей этих сервисов.

Технология REST (*Representational State Transfer*) — технология передачи состояния представления, означающая, что REST-запрос клиента к серверу содержит всю нужную информацию о желаемом ответе сервера.

> **Передача состояния представления** (*REST*) — архитектурный стиль проектирования и реализации распределенных сервис-ориентированных систем, максимально использующих возможности Web-технологий.

Основная позитивная идея сторонников REST-технологий — замена сложных описаний сетевого взаимодействия на языке WSDL на URI/URN-ресурсы, *представленные (опубликованные) в контенте уже используемых средств гипермедиа*, например, в тексте обычных страниц на языке HTML. Таким образом, потребитель сервиса, вместо чтения описаний интерфейсов и реализации программных агентов, обеспечивающих доступ к сервисам через эти интерфейсы, просто выбирает ссылку в окне браузера и получает необходимый сервис.

> Web-службы, построенные с учётом ограничений REST-технологий, называются ***RESTful-системами***.

В отличие от Web-сервисов, построенных на основе протокола SOAP, RESTful-системы не имеют официального стандарта на своё API, поэтому принято считать, что — это архитектурный стиль программирования.

RESTful — это архитектурный стиль проектирования и реализации сервис-ориентированных систем.

Считается, что идейной парадигмой технологии REST стала стандартная классификация набора действий по работе с базами данных *CRUD*, которая была введена Джеймсом Мартином в 1983 году.

CRUD — это акроним, обозначающий четыре базовых действия (функций) для работы с базами данных:

а) CREATE — создание записей (INSERT);

б) READ — чтение записей (SELECT);

в) UPDATE — модификация записей;

г) DELETE — удаление записей.

Применительно к Web-службам, термин CRUD проецируется на четыре HTTP-запроса к Web-серверам:

а) POST — запрос на создание ресурса;

б) GET — запрос на получение ресурса;

в) PUT — запрос на модификацию ресурса;

г) DELETE — запрос на удаление ресурса.

Сам термин REST был введён Роем Филдингом в 2000 году и упоминается в ключе его докторской диссертации: «Архитектурные стили и дизайн сетевых программных архитектур». Им было предложено и шесть ограничений, нарушение которых не позволяет считать сервис-приложение REST-системой:

1. **Модель «Клиент-сервер»** — предполагается приведение приложения, возможно целостного по начальной реализации, к архитектуре, где выделены поставщик сервиса и потребители сервиса. Это улучшает масштабируемость приложения и позволяет отдельным частям развиваться независимо друг от друга.
2. **Отсутствие состояния** — обеспечение на стороне сервера протокола взаимодействия без сохранения состояния. При этом, состояние сессии должно сохраняться на стороне клиента (потребителя сервиса).
3. **Кэширование ответов сервера** — способность сервера или промежуточных узлов кэшировать свои ответы. Это позволяет повысить производительность и расширяемость системы.
4. **Единообразие интерфейсов сервисов** — фундаментальное требование дизайна REST-сервисов. Унификация интерфейсов должна соответствовать четырём дополнительным условиям: идентификации ресурсов посредством URI; возможности манипуляции ресурсов на основе представлений; использованию «самозаписываемых» сообщений и применению гипермедиа как средства изменения состояния приложения.
5. **Слои взаимодействия** — использование взаимодействия клиента и сервера на основе иерархической структуры сетей.
6. **Код по требованию** (*необязательное ограничение*) — возможность расширения функциональности клиентов за счёт загрузки кода с серверов приложений в виде апплетов или сценариев.

> **Замечание** — По мнению Роя Филдинга, приложения, не соответствующие приведённым условиям, не могут называться REST-приложениями.

По мнению Филдинга, приложения, которые соответствуют указанным выше шести условиям, получают следующие преимущества:

а) *надёжность* по причине отсутствия необходимости сохранять информацию о состоянии клиента, поскольку она может быть утеряна;

б) *производительность*, за счёт использования кэша, и масштабируемость, за счёт разделения поставщиков и потребителей сервисов;

в) *простота интерфейсов и портативность компонентов*, создаваемых систем;

г) *лёгкость внесения изменений* и способность «эволюционировать» под воздействием новых требований к системам.

> **Примечание** — Программная платформа Java EE обеспечивает полную поддержку технологии RESTful с помощью инструментальных средств проекта JAX-RS.

JAX-RS (*Java API for RESTful Web Services*) — это спецификация API языка программирования Java, обеспечивающая поддержку создания Web-сервисов в соответствии с архитектурным шаблоном передачи состояния представления REST.

JAX-RS не имеет своего RFC, но содержит достаточно полное описание для версии 2.0 от 22 мая 2013 года, представленное корпорацией Oracle. Полный перечень пакетов технологи JAX-RS представлен в таблице 4.1.

Таблица 4.1 — Основные пакеты JAX-RS [45]

Пакет	***Описание пакета***
javax.ws.rs	Высокоуровневые интерфейсы и аннотации, используемые для создания Web-служб с передачей состояния представления.
javax.ws.rs.client	Классы и интерфейсы клиентского API JAX-RS.
javax.ws.rs.container	API JAX-RS контейнера.
javax.ws.rs.core	Низкоуровневые интерфейсы и аннотации, используемые для создания Web-ресурсов с передачей состояния представления.
javax.ws.rs.ext	API, предоставляющие расширения для типов, поддерживаемых в JAX-RS API.

Инструментальные средства разработки Eclipse EE позволяют подключать сервер приложений Apache TomEE и, в рамках проекта «Dynamic Web Project», реализовать серверную часть приложений ИС, как Web-службу (сервисную службу) в стиле REST.

RESTful-сервис — специальный сервлет проекта «*Dynamic Web Project*», импортирующий пакеты JAX-RS и являющийся JAVA-классом, с базовой аннотацией *@Path(...)*, и одной из аннотаций *@Stateless* или *@Singleton*.

@Stateless — аннотация определяющая, аннотируемый класс не сохраняет состояния.

@Singleton — аннотация определяющая, аннотируемый класс может продуцировать только один объект.

Наиболее важные аннотации пакета JAX-RS представлены в таблице 4.2.

Таблица 4.2 — Основные аннотации пакета JAX-RS

Аннотация	***Описание***
@Path(...)	Указывает относительный путь для класса или метода запрашиваемого ресурса.
@GET, @PUT, @POST, @DELETE и @HEAD	Указывают тип HTTP-запроса, разделяя ПО сервиса на соответствующие части.
@Produces(...)	Указывает MIME-тип ответа, генерируемый методом сервлета.
@Consumes(...)	Указывает принимаемый MIME-тип запроса, принимаемый методом сервлета от клиента.

Главной аннотацией, по которой контейнер определяет тип RESTful-сервлета, является аннотация @Path(...), имеющая общий формат:

@Path("/Путь1/Путь2/.../ПутьN") **(4.1)**

где ***ПутьK*** — слово, задающее часть пути, определённое в адресе URI-запроса.

Методы RESTful-сервлета аннотируются парами: *@Path(...)* и одной из аннотаций таблицы 4.2, указывающей тип HTTP-запроса. Причём аргументы аннотации *@Path(...)* учитываются в качестве аргументов методов RESTful-сервлета.

Технология RESTful является одним из стилей проектирования сервис-ориентированных систем, а не его стандартом.

Контейнерная технология JAX-RS предлагает общий адекватный класс ответа сервера, возвращаемый методами RESTful-сервлета и определённый выражением (4.2):

javax.ws.rs.core.Response **(4.2)**

Для реализации запросов приложений клиентов к RESTful-сервлетам, технология JAX-RS предлагает соответствующий набор классов, главным из которых является класс клиента, определённый выражением (4.3):

javax.ws.rs.client.Client **(4.3)**

Для изучения конкретных примеров вариантов использования технологии RESTful следует воспользоваться учебным пособием [44].

> **Технология RESTful** ориентирована *на максимально упрощённый удалённый доступ к хранилищам* информационных систем (ИС).

Практическое использование технологии RESTful предполагает детальное знание всех особенностей применения протокола HTTP, а также MIME-типов возвращаемых им ответов сервера. Кроме того, следует учесть, что браузеры Web-технологий обеспечивают только HTTP-запросы GET и POST.

Подводя общий итог изложенному в данном подразделе учебному материалу, проектировщику следует учесть следующие общие рекомендации:

а) *контейнерные технологии Java EE*, реализуемые серверами приложений, предлагают множество служебных сервисов, обеспечивающих реализацию распределённых приложений на достаточно высоком профессиональном уровне;

б) *наиболее адекватными для реализации ИС* являются Web-технологии серверов приложений, например Apache TomEE, предлагающие шаблоны проектирования MVC, поддерживаемые сервлетами JSP или JSF;

в) *доступ к локальным хранилищам данных ИС* следует обеспечить на сервер приложений с помощью реализации соответствующих EJB-компонент;

г) *доступ к удалённым хранилищам данных ИС* рекомендуется обеспечить разработкой классов, использующих технологию RESTful, если это конечно позволяют удалённые сервера приложений.

ЗАКЛЮЧЕНИЕ

В настоящем учебном пособии автор стремился найти наиболее доступные формы изложения достаточно сложного материала, связанного с проектированием информационных систем (ИС). Основное внимание было уделено вопросам постановки конкретных задач, сбору информации об объекте проектирования и стандартизированным методологиям концептуального проектирования ИС.

Первый раздел пособия даёт краткое описание изучаемой предметной области и базовый обзор современных ИТ-технологий, применяемых в проектировании информационных систем. Этот материал обеспечивает студента необходимым набором терминов и их определений, опирающихся на имеющиеся стандарты ИТ-технологий.

Второй раздел даёт описание начальной стадии процесса проектирования ИС. Это обеспечивает студента рядом общих методик, формально помогающих ему сформировать описание среды, в которой должна существовать проектируемая ИС.

В третьем разделе описывается методика концептуального проектирования ИС на основе функционального моделирования бизнес-процессов. Приводится достаточно подробное описание трёх графических структурных моделей IDEF0, IDEF3 (WFD) и DFD, которые достаточны для большинства стадий проектирования ИС. Даны проекции этих моделей на стадии проектирования автоматизированной системы предприятия.

Четвёртый раздел обеспечивает студента учебным материалом по технологиям объектного проектирования ИС. Учебный материал предположен в предположении, что целевая ИС может быть распределённой и требовать адекватных инструментальных средств для последующей реализации.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ 33707-2016 (ISO/IEC 2382:2015) Информационные технологии (ИТ). Словарь [Электронный ресурс] — Режим доступа: http://docs.cntd.ru/document/1200139532.
2. Орлов С. А., Цилькер Б. Я. Организация ЭВМ и систем: Учебник для вузов. 3-е изд. — СПб.: Питер, 2015. — 688 с.: ил. (Серия «Учебник для вузов»). - ISBN 978-5-496-01145-7.
3. Резник, В. Г. Распределенные вычислительные сети: Учебное пособие [Электронный ресурс] / В. Г. Резник. — Томск: ТУСУР, 2019. — 211 с. — Режим доступа: https://edu.tusur.ru/publications/9072.
4. Коцюба И.Ю., Чунаев А.В., Шиков А.Н. Основы проектирования информационных систем. Учебное пособие. – СПб: Университет ИТМО, 2015. – 206 с.
5. Норенков, И.П. Основы автоматизированного проектирования: Учеб. для вузов. 4-е изд., перераб. и доп. - М.: Изд-во МГТУ им. Н.Э. Баумана, 2009. - 430 с.: ил. - (Сер. «Информатика в техническом университете»).
6. ГОСТ 34.003-90 АВТОМАТИЗИРОВАННЫЕ СИСТЕМЫ. Термины и определения [Электронный ресурс] — Режим доступа: http://sewiki.ru/images/d/d3/GOST-34.003-90.pdf.
7. ГОСТ 34.321-96 ЭТАЛОННАЯ МОДЕЛЬ УПРАВЛЕНИЯ ДАННЫМИ [Электронный ресурс] — Режим доступа: http://docs.cntd.ru/document/gost-34-321-96.
8. ГОСТ 34.601-90 АВТОМАТИЗИРОВАННЫЕ СИСТЕМЫ. СТАДИИ СОЗДАНИЯ [Электронный ресурс] — Режим доступа: https://www.swrit.ru/doc/gost34/34.601-90.pdf.
9. ГОСТ 34.602-89 ТЕХНИЧЕСКОЕ ЗАДАНИЕ НА СОЗДАНИЕ АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ [Электронный ресурс] — Режим доступа: https://www.swrit.ru/doc/gost34/34.602-89.pdf.
10. Концепция развития CALS-технологий в промышленности России. НИЦ CALS-технологий «Прикладная логистика» [Электронный ресурс] / Е.В. Судов, А.И. Левин. – М., 2002. – 131 с. – Режим доступа: https://cals.ru/sites/default/files/downloads/mdocs/concept_ipi.pdf.
11. РС Р 50.1.031-2001 Информационные технологии поддержки жизненного цикла продукции. Терминологический словарь. Часть 1. Стадии жизненного цикла продукции [Электронный ресурс] — Режим доступа: http://docs.cntd.ru/document/1200028627.
12. ГОСТ 22487-77 ПРОЕКТИРОВАНИЕ АВТОМАТИЗИРОВАННОЕ. Термины и определения [Электронный ресурс] — Режим доступа: https://files.stroyinf.ru/Data2/1/4293772/4293772219.pdf.
13. Норенков, И.П. Автоматизированное проектирования: Учеб. для вузов. 2-е изд., перераб. и доп. - М.: 2000. - 188 с. - (Серия учебных пособий «Информатика в техниче-

ском университете») - Режим доступа: https://www.studmed.ru/view/norenkov-ip-avtomatizirovannoe-proektirovanie_dec2dce75a7.html.

14. Вендров, А. М. Проектирование программного обеспечения экономических информационных систем: Учебник. — М.: Финансы и статистика, 2002. - 352 с : ил. ISBN 5-279-02144-X.
15. Вендров, А. М. Практикум по проектированию программного обеспечения экономических информационных систем: Учеб. пособие.— 2-е изд., перераб. и доп. — М.: Финансы и статистика, 2006. - 192 с.: ил. ISBN 5-279-03106-2.
16. РС Р 50.1.028-2001 Информационные технологии поддержки жизненного цикла продукции. Методология функционального моделирования [Электронный ресурс] — Режим доступа: http://fitp.ifmo.ru/shared/files/201106/12_260.pdf.
17. ГОСТ 19.001-77 Общие положения [Электронный ресурс] — Режим доступа: https://prog-cpp.ru/wp-content/uploads/19.001-77.pdf.
18. ГОСТ 19.102-77 Стадии разработки [Электронный ресурс] — Режим доступа: https://prog-cpp.ru/wp-content/uploads/19.102-77.pdf.
19. Учебный программный комплекс кафедры АСУ на базе ОС ArchLinux [Электронный ресурс]: Учебно-методическое пособие для студентов направления 09.03.01, направление подготовки "Программное обеспечение средств вычислительной техники и автоматизированных систем" / В.Г. Резник - 2016. 33 с. — Режим доступа: https://edu.tusur.ru/publications/6238.
20. The Apache Software Foundation [Электронный ресурс] — Режим доступа: http://www.apache.org/.
21. Apache Tomcat [Электронный ресурс] — Режим доступа: https://tomcat.apache.org/.
22. Apache TomEE [Электронный ресурс] — Режим доступа: http://tomee.apache.org/.
23. Apache Derby [Электронный ресурс] — Режим доступа: http://db.apache.org/derby/.
24. Eclipse IDE 2019-12 R Packages [Электронный ресурс] — Режим доступа: https://www.eclipse.org/downloads/packages/.
25. Eclipse IDE for Enterprise Java Developers [Электронный ресурс] — Режим доступа: https://www.eclipse.org/downloads/packages/release/2020-03/r/eclipse-ide-enterprise-java-developers-includes-incubating-components.
26. Ноутон П., Шилдт Г. JAVA 2. Наиболее полное руководство в подлиннике. СПб.: БХВ-Петербург, 2008. - 1072 с. - ISBN 978-5-94157-012-6.
27. Ramus Soft Group [Электронный ресурс] — Режим доступа: https://ramus-soft-group.software.informer.com/.
28. Eclipse Modeling Tools [Электронный ресурс] — Режим доступа: https://www.eclipse.org/downloads/packages/release/2020-03/r/eclipse-modeling-tools.
29. LibreOffice International Website [Электронный ресурс] — Режим доступа: https://ru.libreoffice.org/.
30. OpenDocument Format [Электронный ресурс] — Режим доступа: http://opendocumentformat.org/.

31. Mozilla Firefox [Электронный ресурс] — Режим доступа: https://www.mozilla.org/ru/firefox/new/?redirect_source=firefox-com.
32. Moodle — Open-source learning platform [Электронный ресурс] — Режим доступа: https://moodle.org/.
33. РД 50-34.698-90 АВТОМАТИЗИРОВАННЫЕ СИСТЕМЫ. ТРЕБОВАНИЯ К СОДЕРЖАНИЮ ДОКУМЕНТОВ [Электронный ресурс] — Режим доступа: http://gostrf.com/normadata/1/4293855/4293855181.pdf.
34. ГОСТ 7.32-2017 ОТЧЁТ О НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ. Структура и правила оформления [Электронный ресурс] — Режим доступа: https://files.stroyinf.ru/Data/655/65555.pdf.
35. ARIS Toolset/BPwin: выбор за аналитиком [Электронный ресурс] — Режим доступа: http://www.interface.ru/home.asp?artId=4269.
36. Образовательный стандарт вуза ОС ТУСУР 01-2013. Работы студенческие по направлениям подготовки и специальностям технического профиля. Общие требования и правила оформления [Электронный ресурс] — Режим доступа: https://regulations.tusur.ru/documents/70.
37. ГОСТ 34.201-89 ВИДЫ, КОМПЛЕКТНОСТЬ И ОБОЗНАЧЕНИЕ ДОКУМЕНТОВ ПРИ СОЗДАНИИ АВТОМАТИЗИРОВАННЫХ СИСТЕМ [Электронный ресурс] — Режим доступа: https://www.zavodsz.ru/files/gost/0_GOST-34-201-89.pdf.
38. Классификатор ОК 005-93 [Электронный ресурс] — Режим доступа: https://classifikators.ru/okp.
39. IDEF3 Process Description Capture Method Report, 1995. — 236 (224) с. [Электронный ресурс] — Режим доступа: http://www.staratel.com/iso/IDEF/IDEF3/Idef3.pdf.
40. Цуканова О. А. Методология и инструментарий моделирования бизнес-процессов: учебное пособие – СПб.: Университет ИТМО, 2015. – 100 с.
41. Аксенов К.А., Работа с CASE-средствами BPwin, Erwin / Аксенов К.А., Клебанов Б.И. [Электронный ресурс] — Режим доступа: https://study.urfu.ru/Aid/Publication/89/1/Method_BpWin_ERwin.pdf.
42. Кара-Ушанов В.Ю., Функционально-структурное моделирование в системе Ramus Educational. - Екатеринбург, 2019. - 67 с. [Электронный ресурс] — Режим доступа: https://study.urfu.ru/Aid/Publication/13928/1/Кара-Ушанов_Ramus.pdf.
43. Грекул В.И., Проектирование информационных систем. Практикум: Учебное пособие / В.И. Грекул, Н.Л. Коровкина, Ю.В. Куприянов — М.: Национальный Открытый Университет «ИНТУИТ», 2012. — 187 с. [Электронный ресурс] — Режим доступа: https://publications.hse.ru/mirror/pubs/share/folder/1j6uku79db/direct/53441014.
44. Распределенные сервис-ориентированные системы: Учебное пособие / В. Г. Резник - 2020. 305 с. [Электронный ресурс]: — Режим доступа: https://edu.tusur.ru/publications/9404.
45. Гонсалвес Э. Изучаем Java EE 7. — СПб.: Питер, 2014. — 640 с.: ил. ISBN 978-5-496-00942-3.

46. Радченко, Г.И. Распределенные вычислительные системы / Г.И. Радченко. – Челябинск: Фотохудожник, 2012. — 184 с. — ISBN 978-5-89879-198-8.
47. SOA Архитектурные особенности и практические аспекты [Электронный ресурс] / TADVISER — 2010. — Режим доступа: http://www.tadviser.ru/index.php/татья:SOA_Архитектурные_особенности_и_практические_аспекты.
48. Арлоу, Д. UML 2 и Унифицированный процесс. Практический объектно-ориентированный анализ и проектирование / Арлоу Д., Нейштадт И. – СПб: Символ-Плюс, 2007. – 624 с. [Электронный ресурс] — Режим доступа: https://picloud.pw/media/resources/posts/2018/02/20/UML_2_и_унифицированный_процесс.pdf.

АЛФАВИТНЫЙ УКАЗАТЕЛЬ

А

Автоматизированная система....20
Автоматизированная система управления....6, 17
АС....3, 20
АСТПП....35
АСУ....6, 17

Б

Бизнес-парадигма архитектуры предприятия....157
Бизнес-процесс....133

В

ВМ....8
Внешняя сущность....134
Вычислительная сеть....8
Вычислительные комплексы....8
Вычислительные системы....8

Г

Группа-студентов....77

Д

Действие....46
Деятельность....46
Документ технический электронный....33
Документ электронный....33
Документация электронная....33
ДТЭ....33
ДЭ....33

Е

Единое Информационное Пространство....34
ЕИП....27, 31, 34

Ж

ЖЦИ....28

З

Занятие....78

И

ИАС....20
Изделие....33
ИИС....3, 27, 31
Инвариантные понятия ИПИ....27
Интегрированная автоматизированная система....20
Интегрированная информационная среда....31
Информационная система....3, 6, 21
Информационное взаимодействие....32
Информационный объект....31
ИО....31
ИПИ....27

ИС....3, 6
Итерационная модель ЖЦИ....31

К

Каскадная модель ЖЦИ....31
КИО....32
Класс информационных объектов....32
Клиент....76
Клиент-преподавателя....76
Клиент-студента....76
КО....32
Коллекция объектов....32
Контейнерные технологии....154

О

ОБДИ....32
ОБДП....32
Общая база данных о предприятии....32
Общая база данных об изделиях....32
Объектные модели проектирования....45
Операция....46
Очередь АС....23

П

Пакет java.rmi....154
Парадигма АСУ....40
Парадигма вычислительных систем....39
Парадигма информационного подхода....11, 39
Парадигма CALS-технологий....40
Перекрёстки....131
ПИС....3, 6
Поток данных....134
Правила информационного взаимодействия и обмена данными....32
Преподаватель....76
Проектирование информационных систем....3, 6
Процесс....46, 134

Р

Работа....128

С

САПР....34
Сервер....76
Сервис-ориентированная архитектура....158
Сервис-ориентированные системы....157
Сильно связанные распределённые системы....153
Система автоматизированного проектирования....34
Система телеобработки....8
Совместное использование данных....32
Спиралевидная модель ЖЦИ....31
Стадии ЖЦ изделия....28
Стадия создания АС....23
Студент....76

Сценарии....126

Т

Технологии описания предметной области....11
Технологии универсального представления данных....14
Технологический процесс....133
Типовой набор офисных приложений....15
Трёхуровневая архитектура АСУ....17

У

Уровень стратегического управления....17
Уровень тактического управления....18

Ф

Функциональные модели проектирования....45

Х

Хранилище данных....134

Ш

Шаблон проектирования MVC....148

Э

ЭВМ....8
ЭЖД....79
Экземпляр класса....32
Электронная цифровая подпись....33
Электронное хранилище....33
Электронный Журнал Дисциплины....79
Этап создания АС....23
ЭЦП....33

A

AEC CAD....35

C

CAA....35
CAAD....35
CAD....35
CAE....35
CAGD....35
CAM....35
CAPP....35
Computer-Aided Design....35
CORBA....151

D

Data Flow Diagrams....134
DFD....134

E

ECAD....35
EDA....35
EJB....156

I

ICAM....84

IDE....37
IDEF0....85
IDEF3....123
IDL CORBA....152
Information Interaction....32
Instans....32
Integrated Development Environment....37

J

Java DataBase Connectivity....38
JDBC....38
Join data using....32
JRMP....154

M

MCAD....35

R

RESTful....165

S

SADT....84

U

UML....13
Unified Modelling Language....13

V

Vault....33

Оглавление

ВВЕДЕНИЕ....3
СПИСОК УСЛОВНЫХ ОБОЗНАЧЕНИЙ И СОКРАЩЕНИЙ....5
1 ВВЕДЕНИЕ В ПРЕДМЕТНУЮ ОБЛАСТЬ ДИСЦИПЛИНЫ....6
1.1 ИС как результат развития вычислительных систем....7
1.1.1 Становление информационных технологий....8
1.1.2 Парадигма «Программа-массив»....9
1.2 ИС как парадигма информационного подхода....11
1.2.1 Технологии описания предметной области....11
1.2.2 Технологии универсального представления данных....14
1.3 ИС как подсистема АСУ....16
1.3.1 Трёхуровневая архитектура АСУ....17
1.3.2 Стандарты и виды обеспечения АС....20
1.3.3 Стадии и этапы создания АС....23
1.4 ИС как парадигма CALS-технологий....27
1.4.1 Концепция жизненного цикла изделия (ЖЦИ) применительно к ИС....30
1.4.2 Концепции ИИС и ЕИП....31
1.4.3 САПР и их классификации....34
1.4.4 CASE-средства CALS-технологий....37
1.4.5 Выводы по подразделу....39
1.5 Проектирование ИС....42
1.5.1 Общая постановка задачи на проектирование....42
1.5.2 Базовые модели проектирования ИС....45
1.5.3 Этапы проектирования ИС....48
1.6 Учебная инфраструктура проектировщика ИС....50
1.6.1 Состав и размещение инструментальных средств....51
1.6.2 Технология применения инструментальных средств проектирования....53
Вопросы для самопроверки....56
2 ФОРМИРОВАНИЕ ТРЕБОВАНИЙ К ИС....57
2.1 Описание учебной задачи....60
2.1.1 Описание идеи учебной задачи....61
2.1.2 Задание на учебную практику....62
2.1.3 Подробное описание задачи на проектирование ИС....63
2.2 Организационная структура управления предприятием....66
2.2.1 Подсистемы АС как отражение структуры управления предприятием....68
2.2.2 Определение границ предметной области ИС....69
2.2.3 Результат анализа организационной структуры вуза....70
2.3 Определение требований к объекту проектирования....71
2.3.1 Поиск прототипов ИС....72
2.3.2 Выбор и описание прототипов ИС....73
2.3.3 Результаты анализа требований....74
2.4 Требования к бизнес-моделям объекта проектирования....76
2.4.1 Узлы создания, потребления и хранения информации....77
2.4.2 Перечень связей между элементами бизнес-моделей....78
2.4.3 Результаты анализа требований....79
2.5 Оформление отчёта по первой стадии проектирования ИС....81

2.5.1 Доказательная база на продолжение работ....81
2.5.2 Структура отчёта по первой стадии проекта....82
Вопросы для самопроверки....83
3 КОНЦЕПТУАЛЬНОЕ ПРОЕКТИРОВАНИЕ ИС....84
3.1 Функциональное моделирование....85
3.1.1 Функциональное моделирование бизнес-процессов....85
3.1.2 Применимость функциональных моделей к стадиям проектирования ИС....89
3.2 Методология стандарта IDEF0....91
3.2.1 Стандартный состав проектной группы....91
3.2.2 Синтаксис и семантика языка IDEF0....92
3.2.3 Контекстные диаграммы IDEF0....94
3.2.4 Декомпозиция блоков диаграмм IDEF0....96
3.2.5 Отношения блоков на диаграммах модели IDEF0....99
3.2.6 Документирование модели IDEF0....101
3.3 Концептуальное проектирование учебной задачи....103
3.3.1 Контекстная диаграмма A-0 учебной задачи....103
3.3.2 Декомпозиция блока A0 учебной задачи....106
3.3.3 Декомпозиция процессов учебной задачи....108
3.3.4 Декомпозиция операций учебной задачи....110
3.3.5 Формирование ТТЗ учебной задачи....112
3.3.6 Техническое задание на ИС учебной задачи....113
3.3.7 Итог применения методологии IDEF0 к проектированию ИС....122
3.4 Моделирование потоков работ IDEF3....123
3.4.1 Элементы графического языка описания процессов IDEF3....123
3.4.2 Синтаксис и семантика языка IDEF3....127
3.4.2 Работы и перекрёстки работ методологии IDEF3....131
3.4.3 Документирование моделей IDEF3....132
3.4.4 Итоговая оценка применения методологии IDEF3....133
3.5 Моделирование потоков данных DFD....134
3.5.1 Синтаксис и семантика языка DFD....135
3.5.2 Постановка учебной задачи по методологии DFD....136
3.5.3 Формализация компонент контекстной диаграммы блока A23....137
3.5.4 Реализация контекстной диаграммы в системе Ramus Educational....138
3.5.5 Декомпозиция контекстной диаграммы учебной задачи....141
3.5.6 Итоговая одноуровневая диаграмма блока A23 в нотации DFD....142
3.6 Выводы по применению методологий концептуального проектирования ИС....144
Вопросы для самопроверки....145
4 ОБЪЕКТНОЕ ПРОЕКТИРОВАНИЕ ИС....146
4.1 Объектные и сервисные распределённые системы....147
4.1.1 Технологии ООП и CORBA....150
4.1.2 Объектная технология RMI и контейнерная технология Java EE....153
4.1.3 Сервис-ориентированная архитектура слабо связанных систем....156
4.1.4 Контейнерные технологии Web-сервисов....159
4.1.5 Проектирование ИС как Web-службы в стиле REST....165
ЗАКЛЮЧЕНИЕ....169
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ....170

АЛФАВИТНЫЙ УКАЗАТЕЛЬ.....174