Министерство образования и науки Российской Федерации

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

ФАКУЛЬТЕТ ДИСТАНЦИОННОГО ОБУЧЕНИЯ (ФДО)

А. И. Исакова

НАУЧНАЯ РАБОТА 1

Учебное пособие

Томск

2017

УД 001.89(075.8
ББ 72.4(2)я73
И 853

Рецензенты:

А. А. Мицель, д-р техн. наук, профессор кафедры автоматизированных систем управления ТУСУР;

А. Ю. Матросова, д-р техн. наук, профессор, заведующая кафедрой программирования Национального исследовательского Томского государственного университета

Исакова А. И.

И 853 Научная работа 1 : учебное пособие / А. И. Исакова. – Томск: ТУСУР, 2017. – 141 с.

Пособие отражает опыт организации и проведения научно-исследовательской работы преподавателями кафедры АСУ ТУСУР со студентами. Учебное методическое пособие предназначено для бакалавров направления подготовки 09.03.03 «Прикладная информатика», учебные планы которых содержат дисциплину «Научная работа», и для преподавателей, организующих научно-исследовательскую работу на кафедрах.

ОГЛАВЛЕНИЕ

Введение

Ведущую роль в повышении качества подготовки специалистов в сторону решительного поворота к развитию творческих способностей будущих специалистов призвана сыграть научная работа студентов, так как учебный процесс, сливаясь с научным трудом студентов, все более превращается в реальную профессиональную деятельность, которая в настоящее время составляет основу процесса становления будущего специалиста.

Согласно ФГОС ВПО бакалавр по направлению подготовки 09.03.03 «Прикладная информатика» [1] готовится к следующим видам профессиональной деятельности:

- проектная; производственно-технологическая;
- организационно-управленческая; аналитическая;
- научно-исследовательская.

Задача высшей школы состоит в том, чтобы сократить период адаптации студентов к учебно-исследовательской и научной работе. Решение этой задачи возможно в том случае, если с первых дней пребывания в высшей школе студент будет активно участвовать в разнообразных формах научной работы, проводимых кафедрами, факультетами.

Научно-исследовательская деятельность студентов позволяет наиболее полно проявить индивидуальность, творческие способности, готовность к самореализации личности. Несмотря на обширную нормативно-правовую базу в данной области, развитие методологии и методики исследовательской подготовки в высшей школе, на деле данному виду деятельности уделяется недостаточно внимания. Необходимо уделять внимание вопросу о готовности студентов к научно-исследовательской деятельности. Процесс исследования индивидуален и является ценностью как в образовательном, так и в личностном смысле, поэтому необходимо совершенствовать подходы к научно-исследовательской работе, для того чтобы сделать этот процесс наиболее интересным и продуктивным.

Научная работа является одним из важнейших видов деятельности профессорско-преподавательского состава, аспирантов и студентов ТУСУР. Проведение научной работы в вузе обеспечивает непрерывное совершенствование учебно-воспитательного процесса на основе фундаментальных и прикладных исследований по существующим направлениям подготовки, а также внедрение

в образовательную деятельность современных методик и педагогических технологий. Научная работа в ТУСУР организована и проводится в соответствии с действующим законодательством РФ, нормативно-правовыми актами федеральных органов управления образования, Уставом университета и Положением о системе организации НИРС ТУСУРа.

Целями данного учебного пособия являются определение главных составляющих и структуры научно-исследовательской деятельности, готовности к ней студентов; рассмотрение понятий, уровней, условий формирования, в том числе средствами проблеммного обучения, а также примера практической организации научно-исследовательской деятельности студентов.

Работа включает теоретическую часть, в которой рассматриваются понятие, структура научно-исследовательской деятельности и понятие готовности к этой деятельности, и практическую часть, в которой описывается применение системного подхода к организации научно-исследовательской работы в учебном заведении.

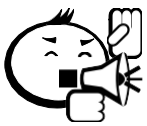
В соответствии с учебным планом направления подготовки 09.03.03 «Прикладная информатика» научная работа и подготовка выпускной квалификационной работы (ВКР) являются обязательными элементами подготовки бакалавров.

В настоящем учебном пособии изложен порядок выполнения научной работы студентами данного направления подготовки кафедры АСУ, получающими квалификацию «бакалавр» [2].

Соглашения, принятые в книге

Для улучшения восприятия материала в данной книге используются пиктограммы и специальное выделение важной информации.

Эта пиктограмма означает определение или новое понятие.

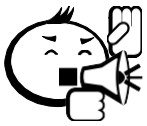


Эта пиктограмма означает «Внимание!». Здесь выделена важная информация, требующая акцента на ней. Автор может поделиться с читателем опытом, чтобы помочь избежать некоторых ошибок.

1 Цели и задачи научной работы в учебном процессе вуза

В связи с усиливающейся информатизацией и интеллектуализацией производственных технологий быстрыми темпами растет объем специальной информации – научной, технической, технологической и т. д. В этих условиях технология обучения, ориентированная на предоставление и усвоение готовых знаний, не может быть признана рациональной и перспективной. Необходимы новые технологии образования, связанные с формированием интеллектуальной культуры и повышением творческих способностей специалиста. Работа, осуществляемая в данном направлении, должна базироваться на педагогической технологии, основанной на концепции творческой деятельности. Наиболее эффективной формой ее реализации в вузе является непрерывная система работы студентов, максимальное приближение ее к учебному процессу.

Научная работа студентов является одним из важнейших средств повышения качества подготовки специалистов с высшим образованием, способных творчески применять в практической деятельности достижения научно-технического прогресса, а следовательно, быстро адаптироваться к современным условиям развития экономики.



Основные цели научной работы студента – формирование творческих способностей, развитие и совершенствование форм привлечения молодежи к научной, конструкторской, технологической и внедренческой деятельности, обеспечивающих единство учебного, научного, воспитательного процессов для повышения профессионально-технического уровня подготовки специалистов с высшим образованием, а также привитие навыков работы с научно-технической литературой, оформления отчетной документации.

Основными задачами научной работы студента являются:

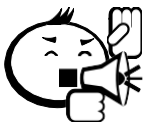
- обучение методологии рационального и эффективного добывания и использования знаний;

- совершенствование и поиск новых форм интеграции системы высшего образования с наукой и производственной деятельностью в рамках единой системы «учебно-воспитательного процесс – повышение навыков научной, творческой и исследовательской деятельности»;
- участие студентов в научных исследованиях, реальных разработках и техническом творчестве;
- освоение современных технологий в области науки, техники, производства;
- знакомство с современными научными методологиями, работа с научной литературой;
- выявление способной молодежи для дальнейшего обучения в аспирантуре, работы на кафедрах и в научных лабораториях; развитие технического творчества учащихся школ и техникумов, обеспечение отбора в вузы молодежи, проявившей склонность к технике, творчеству и науке [2].

В результате **научной работы** студенты должны **уметь**:

1) составлять литературный обзор математических методов и их программной реализации;
2) формализованно ставить задачи;
3) проводить анализ полученных результатов и давать рекомендации по их использованию;
4) уметь показать результаты в отчете;
5) защитить свою работу, отвечая на вопросы преподавателя [2].

Тематика научной работы должна быть актуальна, соответствовать современному состоянию и перспективам развития электронных информационных систем (ЭИС) на базе различных классов ЭВМ и разнообразных средств сбора, передачи и отображения информации.



Научная работа должна быть продолжением УИР и также быть связана с выпускной квалификационной работой.

При определении задач научной работы следует исходить из реальной потребности организаций, предприятий, банков, фирм в разработке и из возможности внедрения фрагментов будущей выпускной квалификационной работы на предприятии.

Сегодня основная задача вуза – переориентация дидактической системы высшей школы с обучением преимущественно информационного типа на обучение, позволяющее развивать творческие способности студентов. Развитие самостоятельности человека, формирование творческой личности – задача социальная, и решаться она должна всеми звеньями системы школа – вуз, с использованием всего многообразия методов, с учетом индивидуальных особенностей студентов. Ее решение зависит от квалификации, способностей и возможностей профессорско-преподавательского состава, а также от способностей студента.

Под самостоятельной работой в вузе понимают познавательную деятельность, выполняемую студентом самостоятельно под руководством преподавателя. Разработаны методики самостоятельной работы в виде общей и социально-предметных.

Результаты научной работы оформляются в виде отчета и презентации и защищаются в виде ответов на вопросы преподавателя. Отчет должен содержать разделы, указанные ниже, в п. 2.2, кратко изложенную теоретическую или практическую часть, полученные результаты и их обсуждение. В конце работы приводится список использованных источников и литературы.

Письменные задания для студента – первые исследовательские работы. Они имеют большое значение в формировании профессионализма. Каждая из них – это самостоятельное научно-прикладное исследование, которое является одной из форм отчетности и контроля знаний студентов, доказательством приобретения знаний по избранной проблеме, творческого осмысления соответствующей научной мысли.

Выполнение письменных заданий поможет студенту поэтапно включиться в учебно-исследовательскую, а затем в научную работу, которая способствует формированию творческих качеств и творческого отношения к своей профессии.

Научная работа – глубокое и объемное исследование избранной проблемы, это первая ступень в овладении методикой исследовательской работы. Именно эта работа поможет расширить, обобщить и систематизировать знания по изучаемой проблеме. Выполнение данного задания поможет овладеть современными методами поиска, обработки и использования информации, освоить некоторые методы исследовательской работы.

Существует достаточно большое число методических рекомендаций, предназначенных для студентов, которые имеют целью помочь освоить важнейшие этапы подготовки, написания и защиты отчета по научной работе.

Огромное влияние на изменение организационной структуры науки оказало распространение коммуникационной сети Интернет.

Разработка и совершенствование современных технологий использования информационных ресурсов во всех сферах жизни являются объективной необходимостью, обусловленной уровнем развития современного общества.

Эффективным средством решения многих вопросов самостоятельной работы студентов является использование Глобальной сети. Обычно, когда говорят о достоинствах Интернета, имеют в виду оперативность, свежесть информации, возможность ее ежедневного обновления. Но в настоящее время можно говорить и о ряде других вопросов, сопутствующих распространению Сети, например о положительном влиянии Всемирной паутины на развитие функциональных и личностных отношений творческой молодежи.

Принципиальная особенность информационной технологии состоит в том, что она не может быть непрерывной, так как соединяет работу рутинного типа и работу творческую, не поддающуюся пока формализации (принятие решений).

Из всех видов технологий информационная технология научной сферы предъявляет, пожалуй, самые высокие требования к «человеческому фактору», оказывая принципиальное влияние на квалификацию работника, содержание его труда, физическую и умственную нагрузку, профессиональные перспективы и уровень социальных отношений.

Бакалавр по направлению подготовки 230700.62 (09.03.03) «Прикладная информатика» должен решать следующие профессиональные задачи в рамках научно-исследовательской деятельности:

- применение системного подхода к автоматизации и информатизации решения прикладных задач, к построению информационных систем на основе современных информационно-коммуникационных технологий;
- подготовка обзоров, аннотаций, составление рефератов, научных докладов, публикаций и библиографии по научно-исследовательской работе в области прикладной информатики [1].

2 Организация проведения научной работы студентов

2.1 Общие положения о проведении научной работы студентов

Научная работа студентов является продолжением и углублением учебного процесса, она организуется под руководством преподавателя кафедры на конкретном предприятии.

Научно-исследовательская работа студентов, включаемая в учебный процесс, выполняется самостоятельно студентом, но под руководством преподавателя и предусматривает:

- выполнение заданий, содержащих элементы научных исследований;
- изучение теоретических основ методики, постановки, организации выполнения научных исследований по дисциплине «Научная работа».

Названная дисциплина включена в учебный план данного направления подготовки.

Научно-исследовательская работа студентов завершается обязательной подготовкой отчета, презентации, в которой представляются основные результаты работы.

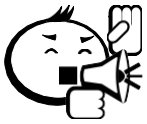
С целью активизации научной работы студентов Министерство образования Российской Федерации, отраслевые министерства и комитеты и другие заинтересованные организации проводят конкурсы на лучшую научно-исследовательскую работу студентов, выставки, конференции, олимпиады и другие мероприятия, порядок проведения которых определяется соответствующими положениями.

Научная работа является важным элементом подготовки специалистов и имеет целью развитие у студентов навыков исследовательской и проектной работы, личное получение автором работы аналитических, практических, научных результатов с использованием знаний, приобретенных в учебном процессе и самостоятельно.

Ответственный за научную работу преподаватель кафедры обязан:

- оказывать консультационную и методическую помощь по вопросам, возникающим в процессе выполнения научной работы;
- периодически (по мере необходимости) отвечать на вопросы студентов по обсуждаемой проблеме;

- принять отчет по заданным темам и презентацию, задать вопросы и по результатам ответов проставить зачет.



Следует помнить, что само по себе изучение какой-либо предметной области не может являться конечной целью научной работы – работа должна содержать элементы активного, самостоятельного исследования.

Научная работа выполняется студентом индивидуально в течение семестра. Задания на научную работу соответствуют данному направлению подготовки бакалавров и уровню учебной подготовки студентов. Работа, выполненная в течение семестра, должна обладать тематической и логической завершенностью. Работа должна быть направлена на решение теоретической либо практической задачи, результаты которой могут принести пользу для деятельности организаций, предприятий, учреждений.

2.2 Задание на научную работу

При выполнении научной работы необходимо выбрать реально существующее предприятие, изучить его информационное и программное обеспечение, выявить проблемы, которые нужно решить в плане дальнейшей автоматизации. Желательно, чтобы научная работа студента была продолжением его учебной работы и входила (как стержень и хороший задел) в выпускную квалификационную работу.

Научная работа является одним из важнейших видов деятельности любого вуза. Для бакалавров по направлению подготовки 09.03.03 «Прикладная информатика» научная работа предусматривает анализ предметной об- ласти и проектирование информационных систем на основе современных информационно-коммуникационных технологий.

При создании информационной системы в экономике проектировщику в первую очередь необходимо познакомиться с аналогами информационных систем (ИС) в рассматриваемой предметной области (например, в отделе кадров, при складском учете, электронном документообороте и т. д.), чтобы иметь представление о том, что есть на рынке программного обеспечения. Далее необходимо проанализировать все возможные среды разработки ИС и обосновать свой выбор среды разработки ИС. В последнюю очередь, приступая к про-

ектированию, надо построить информационно-логическую модель (SADT-модель) и концептуальную модель всех уровней (FR-KB-FA-модели).

Таким образом, отчет по научной работе должен включать следующие разделы:

- 1-й раздел. Обзор аналогов информационных систем.
- 2-й раздел. Обоснование среды разработки создаваемой ИС.
- 3-й раздел. Построение инфологической модели создаваемой ИС (SADT-модель).
- 4-й раздел. Построение концептуальной модели всех уровней (FR-KB-FA-модели).

Содержание всех вышеперечисленных разделов оформляется в отчет и презентацию, защитив которые студент получает «зачет» по дисциплине.

Ниже (в главах 3–5 данного учебного методического пособия) приведена справочная информация по каждому из вышеприведенных разделов.

2.3 Требования к содержанию и оформлению отчета по научной работе

При оформлении отчета по научной работе на персональном компьютере предъявляются следующие общие требования согласно образовательному стандарту вуза ОС ТУСУР 01-2013 [2]:

1. Общий **объем** машинописного текста без приложений должен составлять не менее **25** страниц, текст отчета должен быть напечатан с **интервалом 1,5** формата А4 (210×297 мм) шрифтом Times New Roman, размер шрифта 12–14 пунктов.
2. Все страницы, включая иллюстрации и приложения, нумеруются по порядку. Первой страницей считается титульный лист, на нем цифра «1» не ставится, на следующей странице проставляется цифра «2» и т. д. Порядковый номер печатается **в центре верхнего поля** страницы. Нумерация страниц должна быть сквозной от титульного листа до последнего листа текста, включая иллюстрации, таблицы, графики, диаграммы и т. д., расположенные внутри текста или после него, а также приложения.

Обязательными элементами отчета по научной работе являются: титульный лист; задание на выполнение научной работы (перечень разделов, пе-

речисленных в п. 2.2); содержание; введение; основная часть документа; заключение; список использованных источников, приложение.

Титульный лист служит обложкой документа (пример оформления представлен в приложении 1 «Пример отчета по научной работе»). На титульном листе студент должен поставить инициалы и фамилии, свои и руководителя научной работы. Вслед за титульным листом помещается содержание, в котором приводятся заголовки разделов, глав, параграфов и т. д. с указанием страниц всех частей отчета.

Содержание включает: введение, наименования всех глав, разделов, подразделов, пунктов (если они имеют наименования), заключение, список использованных источников, приложения (при наличии). Строки оглавления заканчиваются указанием ***номеров страниц***, на которых расположено ***начало*** соответствующей части документа.

Заголовок «***Содержание***» (с прописной буквы) размещают в центре строки (симметрично тексту). Наименования, включенные в содержание, записывают строчными буквами, начиная с прописной буквы. Содержание включается в общее количество страниц документа.

В разделе «***Введение***» отчета обязательно должны быть обоснованы актуальность, теоретическая и практическая значимость работы, сформулирована цель работы и перечислены задачи, решаемые для достижения поставленной цели. Объем введения, как правило, не превышает 1–2 страниц.

Основная часть, как правило, состоит из нескольких самостоятельных разделов, каждый из которых характеризуется логической завершенностью и при необходимости может делиться на подразделы и пункты (заголовок «Основная часть» в отчете не пишется!). В основной части содержится обзор рассматриваемой предметной области со ссылками на источники информации.

Рекомендуется в конце каждого раздела формулировать краткие выводы (1–2 абзаца). Разделы основной части должны быть пронумерованы, начиная с первого (*введение к отчету и заключение не нумеруются!*). Наибольший раздел не должен более чем в 2–3 раза превышать наименьший.

В разделе «***Заключение***» формулируется основной результат работы и (по пунктам) выводы по результатам выполненной работы (как правило, 3–5 выводов), а также указываются вероятные пути и перспективы продолжения работы. Объем заключения, как правило, не превышает 1–2 страниц.

Список использованных источников содержит библиографическое описание всех литературных источников, использованных в процессе выполнения

научной работы. Сведения о каждом из источников располагают в порядке их упоминания в тексте. Заголовок «Список использованных источников» записывается симметрично тексту с прописной буквы и не нумеруется.

Список должен быть пронумерован для того, чтобы можно было судить об его объеме и иметь возможность применять ссылки к тексту.

При ссылке на литературные источники в тексте, начиная с введения и далее, приводится порядковый номер источника, заключенный в квадратные скобки. При необходимости в дополнение к номеру источника указывается номер его раздела, подраздела, страницы, рисунка или таблицы. Например: [2, раздел 3], [6, приложение Б], [24, с. 66, таблица 2.4].

В библиографическом описании пунктуация используется особым образом, она выполняет две функции – обычных грамматических знаков препинаний и знаков предписанной пунктуации, т. е. знаков, имеющих опознавательный характер для областей и элементов библиографического описания. Предписанная пунктуация способствует распознаванию отдельных элементов в описаниях на разных языках.

Предписанная пунктуация предшествует элементам и областям или заключает их. Её употребление не связано с нормами языка. Каждой области описания, кроме первой, предшествует знак точка и тире. Пробелы ставятся с двух сторон предписанных знаков, кроме точки и запятой. При постановке точки и запятой пробел ставится только после знака.

В заголовке библиографического описания, содержащем имя автора, указывается фамилия с заглавной буквы, затем запятая, за которой следуют инициалы, разделенные между собой пробелом.

Оформление библиографических списков осуществляется по системе стандартов по информации, библиотечному и издательскому делу (ГОСТ 7.1–2003). Образцы библиографического описания наиболее важных типов литературных источников (с учетом требований нормативных документов) приведены в приложении 1 «Пример отчета по научной работе».

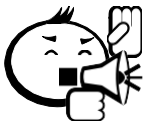
В ***приложении*** могут быть примеры первичных экономических документов (входные и выходные формы и бланки или реальные заполненные документы); информация об объекте исследования (фрагменты громоздких таблиц с описаниями сущностей и атрибутов, различные справочники предприятий и т. д.).

При оформлении текстового материала необходимо соблюдать следующие требования:

1) текст отчета должен иметь поля: левое – 30 мм, правое – 15 мм, верхнее – 20 мм, нижнее – 25 мм;
2) абзац должен начинаться на расстоянии 10 мм от левого края страницы;
3) каждая глава отчета должна начинаться с новой страницы.

Названия глав, параграфов, пунктов, подпунктов следует оформлять **по центру** и их можно писать более крупным шрифтом, чем текст. При этом цифры, указывающие их номера, не должны выступать за границу абзаца. Точка в конце названия не ставится.

Названия глав, параграфов должны соответствовать их наименованию, указанному в оглавлении. Все страницы работы должны соответствовать оглавлению.



Подчеркивания наименований глав, параграфов и фрагментов текста не допускаются!

В тексте допускаются общепринятые сокращения и такие сокращения, для которых в тексте были приведены либо полная расшифровка, либо приложенный список сокращений.

Примеры оформления отчета по научной работе приведены в приложении 1.

2.4 Порядок выполнения и защита научной работы

Для получения зачета студент должен представить иллюстративный материал (**презентация** 12–15 слайдов) и **отчет** о результатах выполнения научной работы, содержащий все разделы, перечисленные в п. 2.2 данного пособия.

Отчет и презентация высылаются для проверки преподавателю. При необходимости отчет корректируется студентом в соответствии с замечаниями, сделанными ему преподавателем.

Защита работы состоит в ответе на вопросы преподавателя по результатам проверки **иллюстративного материала** (презентации) и **отчета** по научной работе.

В презентации должны быть обязательно отражены основные итоги работы, оценка их практической значимости.

Пример оформления и содержания отчета по научной работе представлен в приложении 1. Фрагмент презентации представлен в отдельном файле.

2.5 Методологические аспекты научной работы

2.5.1 Метод проектов в научной работе

Научная работа направлена на получение аналитических результатов, относящихся к изучению выбранной предметной области, объектов, их характеристики. Особое значение имеют формы и методы научной работы. Остановимся на одном из самых продуктивных методов – методе проектов.

Метод проектов. В основу метода проектов положена идея, составляющая суть понятия «проект», его прагматическую направленность на результат, который можно получить при решении той или иной практически или теоретически значимой проблемы. Метод проектов всегда ориентирован на самостоятельную деятельность учащихся – индивидуальную, парную, групповую, которую учащиеся выполняют в течение определенного отрезка времени. Этот метод органично сочетается с индивидуальным подходом к обучению.

Метод проектов всегда предполагает решение какой-то проблемы. Решение проблемы предусматривает, с одной стороны, использование совокупности разнообразных методов, средств обучения, а с другой – необходимость интегрирования знаний, умений из различных областей науки, техники, технологии, творческих областей.

По сути дела, проект является своего рода «диссертацией студента», что позволяет проводить аналогии с настоящими диссертационными работами. Необходимым условием метода проектов является наличие значимой в исследовательском, творческом плане проблемы/задачи, требующей интегрированного знания, исследовательского поиска для ее решения (например, исследование и проведение предпроектной работы по созданию и разработке информационной системы и пр.).

Особенности метода проектов:

1. Практическая, теоретическая, познавательная значимость предполагаемых результатов может выражаться, например, в форме доклада в соответствующие службы о демографическом состоянии данного региона, о факторах, влияющих на это состояние, тенденциях, прослеживающихся в развитии данной проблемы; совместного выпуска газе-

ты, альманаха с репортажами с места событий; в охране леса в разных местностях, плане мероприятий и пр.

2. Самостоятельная (индивидуальная, парная, групповая) деятельность учащихся.
3. Структурирование содержательной части проекта (с указанием поэтапных результатов).
4. Использование исследовательских методов, предусматривающих определенную последовательность действий:
   - определение проблемы и вытекающих из нее задач исследования (использование в ходе совместного исследования метода «мозговой атаки», «круглого стола»);
   - выдвижение гипотез их решения;
   - обсуждение методов исследования (статистических методов, экспериментальных наблюдений и пр.);
   - обсуждение способов оформления конечных результатов (презентаций, защит, творческих отчетов, просмотров и пр.). Сбор, систематизация и анализ полученных данных;
   - подведение итогов, оформление результатов, их презентация;
   - выводы, выдвижение новых проблем исследования.

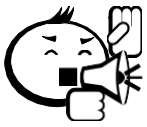
2.5.2 Рецензирование научной работы студентов

Преподаватель с кафедры на основе изучения научной работы студента представляет рецензию, в котором оцениваются:

- актуальность избранной темы, достоверность результатов, ее новизна;
- степень обоснованности научных выводов, результатов.

Объективность оценки предусматривает отражение как положительных, так и отрицательных сторон работы. Отмеченные выше критерии являются основополагающими и при оценке работы студента.

Так как рецензирование научной работы студента – это не только часть защиты работы, но и отдельный процесс, имеющий значение как процесс усовершенствования навыков анализа работ, критической оценки, оно перестает быть четко организованным процессом и обязанностью одного человека, трансформируясь в дискуссию, проводимую с участием студента, занимающегося работой над проектом.



Следует отметить один очень существенный момент: в случае использования чужого материала без ссылки на автора и источник работа студента не засчитывается.

Поэтому студент уже с первых шагов в науке должен иметь четкую уста- новку на важность библиографических ссылок и грамотное оформление списка используемой литературы.

2.5.3 Научный стиль научно-исследовательских работ

Научно-исследовательские работы (тезисы докладов, статьи, отчеты по практикам и т. д.) студентов имеют большое значение в формировании профессионализма будущего специалиста. Каждая из них – это самостоятельное научно-прикладное исследование, которое является одной из форм отчетности и контроля знаний студентов, доказательством приобретения знаний по избранной проблеме, творческим осмыслением соответствующей научной мысли.

Результаты научной работы могут быть представлены только в письменной форме и должны характеризоваться наличием научного стиля.

Основные черты научного стиля:

- точность, ясность и полнота высказывания;
- лаконичность в выражении мысли;
- широкое использование абстрактной лексики;
- употребление слов в конкретном значении;
- безличность;
- монологический характер высказывания;
- последовательность;
- тесная связь отдельных частей высказывания (с помощью развернутых синтаксических конструкций с причастными, деепричастными оборотами, перечислениями);
- использование специфической терминологии, условных знаков и обозначений.

3 Аналоги программного продукта

3.1 Характеристика программного продукта

Многие профессиональные программные продукты стоят недешево, хотя в последние годы относительная стоимость массового программного обеспечения постоянно снижается. Но на рынке для многих типов программных продуктов существуют аналоги, которые, при тех или иных условиях, могут использоватся бесплатно. Разумеется, на адаптацию и освоение таких аналогов приходится тратить время, силы и в ряде случаев финансы. Но все же следует признать, что такой путь куда честнее и безопаснее, чем использование контрафактных программ под надуманным оправданием «очень дорого» [3].

Все программы по характеру использования и категориям пользователей можно разделить на два класса (рис. 3.1) – утилитарные программы и программные продукты (изделия).

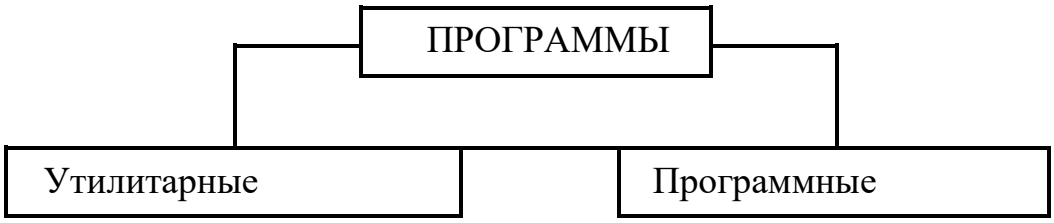


Рис. 3.1 – Классификация программ по категориям пользователей

Утилитарные программы («программы для себя») предназначены для удовлетворения нужд их разработчиков. Чаще всего утилитарные программы выполняют роль сервиса в технологии обработки данных либо являются программами решения функциональных задач, не предназначенных для широкого распространения.

Программные продукты (изделия) предназначены для удовлетворения потребностей пользователей, широкого распространения и продажи.

В настоящее время существуют и другие варианты легального распространения программных продуктов, которые появились с использованием глобальных или региональных телекоммуникаций:

- freeware – бесплатные программы, свободно распространяемые, поддерживаются самим пользователем, который правомочен вносить в них необходимые изменения;

- shareware – некоммерческие (условно-бесплатные) программы, которые могут использоваться, как правило, бесплатно. При условии регулярного использования подобных продуктов осуществляется взнос определенной суммы.

Ряд производителей используют *OEM-программы* (Original Equipment Manufacturer), т. е. встроенные программы, устанавливаемые на компьютеры или поставляемые вместе с вычислительной техникой.

Программный продукт должен быть соответствующим образом подготовлен к эксплуатации, иметь необходимую техническую документацию, предоставлять сервис и гарантию надежной работы программы, иметь товарный знак изготовителя, а также желательно наличие кода государственной регистрации. Только при таких условиях созданный программный комплекс может быть назван программным продуктом.

Программный продукт – комплекс взаимосвязанных программ для решения определенной проблемы (задачи) массового спроса, подготовленный к реализации как любой вид промышленной продукции.

Путь от «программ для себя» до программных продуктов достаточно долгий, он связан с изменениями технической и программной среды разработки и эксплуатации программ, с появлением и развитием самостоятельной отрасли – информационного бизнеса, для которой характерны разделение труда фирм-разработчиков программ, их дальнейшая специализация, формирование рынка программных средств и информационных услуг.

Программные продукты могут создаваться:

- как индивидуальная разработка под заказ;
- разработка для массового распространения среди пользователей.

При индивидуальной разработке фирма-разработчик создает оригинальный программный продукт, учитывающий специфику обработки данных для конкретного заказчика.

При разработке для массового распространения фирма-разработчик должна обеспечить, с одной стороны, универсальность выполняемых функций обработки данных, с другой стороны, гибкость и настраиваемость программного продукта на условия конкретного применения. Отличительной особенностью программных продуктов должна быть их системность – функциональная полнота и законченность реализуемых функций обработки, которые применяются в совокупности.

Программный продукт разрабатывается на основе промышленной технологии выполнения проектных работ с применением современных инструментальных средств программирования. Специфика заключается в уникальности процесса разработки алгоритмов и программ, зависящего от характера обработки информации и используемых инструментальных средств. На создание программных продуктов затрачиваются значительные ресурсы: трудовые, материальные, финансовые; требуется высокая квалификация разработчиков.

Как правило, программные продукты требуют сопровождения, которое осуществляется специализированными фирмами-распространителями программ (дистрибьюторами), реже – фирмами-разработчиками. Сопровождение программ массового применения сопряжено с большими трудозатратами – исправление обнаруженных ошибок, создание новых версий программ и т. п.

Сопровождение программного продукта – поддержка работоспособности программного продукта, переход на его новые версии, внесение изменений, исправление обнаруженных ошибок и т. п.

Программные продукты в отличие от традиционных программных изделий не имеют строго регламентированного набора качественных характеристик, задаваемых при создании программ, либо эти характеристики невозможно заранее точно указать или оценить, т. к. одни и те же функции обработки, обеспечиваемые программным средством, могут иметь различную глубину проработки. Даже время и затраты на разработку программных продуктов не могут быть определены с большой степенью точности заранее.

3.2 Основные характеристики программных продуктов

Основные характеристики программ:

- алгоритмическая сложность (логика алгоритмов обработки информации);
- состав и глубина проработки реализованных функций обработки;
- полнота и системность функций обработки;
- объем файлов программ;
- требования к операционной системе и техническим средствам обработки со стороны программного средства;
- объем дисковой памяти;
- размер оперативной памяти для запуска программ;
- тип процессора;

- версия операционной системы;
- наличие вычислительной сети и др. [4].

Программные продукты отличаются многообразием показателей качества, которые отражают следующие аспекты:

- насколько хорошо (просто, надежно, эффективно) можно использовать программный продукт;
- насколько легко эксплуатировать программный продукт;
- можно ли использовать программный продукт при изменении условия его применения и др. (рис. 3.2).

Мобильность программных продуктов означает их независимость от технического комплекса системы обработки данных, операционной среды, сетевой технологии обработки данных, специфики предметной области и т. п.

Дерево характеристик качества программных продуктов представлено на рисунке 3.2.

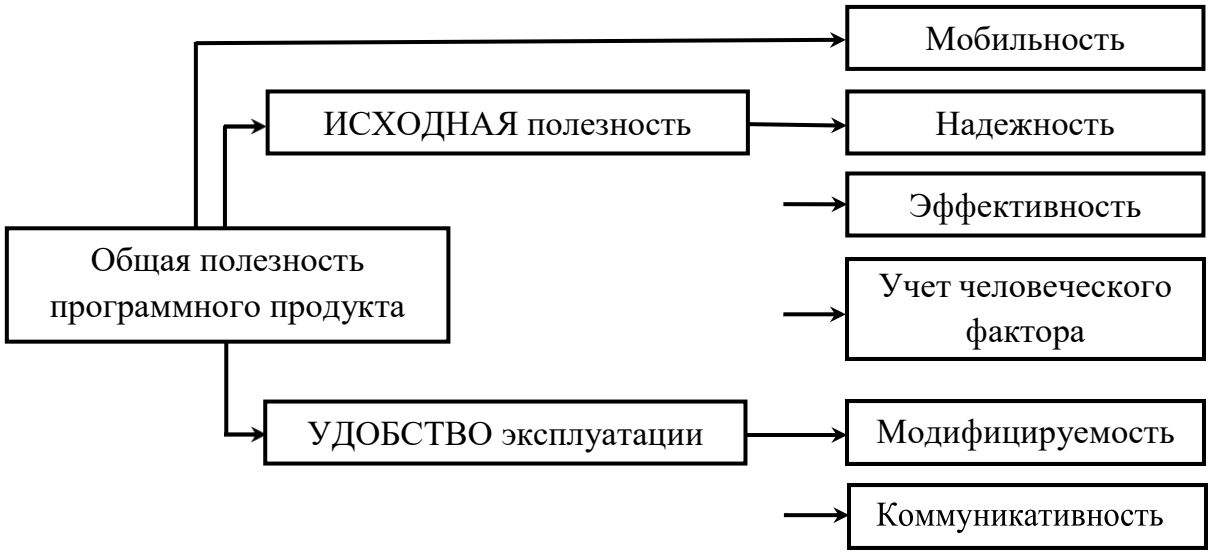


Рис. 3.2 – Дерево характеристик качества программных продуктов

Мобильный (многоплатформный) программный продукт может быть установлен на различных моделях компьютеров и операционных систем, без ограничений на его эксплуатацию в условиях вычислительной сети. Функции обработки такого программного продукта пригодны для массового использования без каких-либо изменений.

Надежность работы программного продукта определяется устойчивостью в работе программ, точностью выполнения предписанных функций обработки, возможностью диагностики возникающих в процессе работы программ ошибок.

Эффективность программного продукта оценивается как с позиций прямого его назначения – требований пользователя, так и с точки зрения расхода вычислительных ресурсов, необходимых для его эксплуатации.

Расход вычислительных ресурсов оценивается через объем внешней памяти для размещения программ и объем оперативной памяти для запуска программ.

Учет человеческого фактора означает обеспечение дружественного интерфейса для работы конечного пользователя, наличие контекстно-зависимой подсказки или обучающей системы в составе программного средства, хорошей документации для освоения и использования заложенных в программном средстве функциональных возможностей, анализ и диагностику возникших ошибок и др.

Модифицируемость программных продуктов означает способность к внесению изменений, например расширение функций обработки, переход на другую техническую базу обработки и т. п.

Коммуникативность программных продуктов основана на максимально возможной их интеграции с другими программами, обеспечении обмена данными в общих форматах представления (экспорт/импорт баз данных, внедрение или связывание объектов обработки и др.).

В условиях существования рынка программных продуктов важными характеристиками являются:

- стоимость;
- количество продаж;
- время нахождения на рынке (длительность продаж);
- известность фирмы-разработчика и программы;
- наличие программных продуктов аналогичного назначения.

Программные продукты массового распространения продаются по ценам, которые учитывают спрос и конъюнктуру рынка (наличие и цены программ-конкурентов). Большое значение имеет проводимый фирмой маркетинг, который включает:

- формирование политики цен для завоевания рынка;
- широкую рекламную кампанию программного продукта;
- создание торговой сети для реализации программного продукта (так называемые дилерские и дистрибьютерные центры);

- обеспечение сопровождения и гарантийного обслуживания пользователей программного продукта, создание горячей линии (оперативный ответ на возникающие в процессе эксплуатации программных продуктов вопросы);
- обучение пользователей программного продукта.

Спецификой программных продуктов (в отличие от большинства промышленных изделий) является также и то, что их эксплуатация должна выполняться на правовой основе – лицензионные соглашения между разработчиком и пользователями с соблюдением авторских прав разработчиков программных продуктов.

3.3 Маркетинговые исследования рынка программных средств

Если программный продукт создается под заказ и предполагается выход на рынок программных средств, маркетинг выполняется в полном объеме: изучаются программные продукты-конкуренты и аналоги, обобщаются требования пользователей к программному продукту, устанавливается потенциальная емкость рынка сбыта, дается прогноз цены и объема продаж. Кроме того, важно оценить необходимые для разработки программного продукта материальные, трудовые и финансовые ресурсы, ориентировочные длительности основных этапов жизненного цикла программного продукта [3–4].

Проектирование структуры программного продукта связано с алгоритмизацией процесса обработки данных, детализацией функций обработки, разработкой структуры программного продукта (архитектуры программных модулей) и/или структуры информационной базы (базы данных) задачи, выбором методов и средств создания программ-технологии программирования.

Программирование, тестирование и отладка программ являются технической реализацией проектных решений и выполняются с помощью выбранного инструментария разработчика (алгоритмические языки и системы программирования, инструментальные среды разработчиков и т. п.).

Для больших и сложных программных комплексов, имеющих развитую модульную структуру построения, отдельные работы данного этапа могут выполняться параллельно, обеспечивая сокращение общего времени разработки программного продукта. Важная роль принадлежит используемым при этом инструментальным средствам программирования и отладки программ, поскольку они влияют на трудоемкость выполнения работ, их стоимость, качество создаваемых программ.

Документирование программного продукта является обязательным видом работ, выполняемых, как правило, не самим разработчиком, а лицом, связанным с распространением и внедрением программного продукта. Документация должна содержать необходимые сведения по установке и обеспечению надежной работы программного продукта, поддерживать пользователей при выполнении функций обработки, определять порядок комплексирования программного продукта с другими программами. Успех распространения и эксплуатации программного продукта в значительной степени зависит от качества его документации [4].

На машинном уровне программного продукта, как правило, создаются:

- автоматизированная контекстно-зависимая помощь (HELP);
- демонстрационные версии, работающие в активном режиме по типу обучающих систем (электронный учебник) или в пассивном режиме (ролик, мультфильм) – для демонстрации функциональных возможностей программного продукта и информационной технологии его использования.

Требуется постоянная программа маркетинговых мероприятий и поддержки программных продуктов. Вначале продажа программного продукта идет вверх, затем наступает стабилизация продаж программного продукта. Фирма-разработчик стремится к максимальной длительности периода стабильных продаж на высоком уровне. Далее происходит падение объема продаж, что является сигналом к изменению маркетинговой политики фирмы в отношении данного программного продукта, требуется модификация данного продукта, изменение цены или снятие с продажи.

Эксплуатация программного продукта идет параллельно с его сопровождением, при этом эксплуатация программ может начинаться и в случае отсутствия сопровождения или продолжаться в случае завершения сопровождения еще какое-то время. После снятия программного продукта с продажи определенное время также может выполняться его сопровождение. В процессе эксплуатации программного продукта производится устранение обнаруженных ошибок.

Снятие программного продукта с продажи и отказ от сопровождения происходят, как правило, в случае изменения технической политики фирмы-разработчика, неэффективности работы программного продукта, наличия в нем неустранимых ошибок, отсутствия спроса.

Длительность жизненного цикла для различных программных продуктов неодинакова. Для большинства современных программных продуктов длительность жизненного цикла измеряется в годах (2–3 года). Хотя достаточно часто встречаются на компьютерах и давно снятые с производства программные продукты [4].

Особенность разработки программного продукта заключается в том, что на начальных этапах принимаются решения, реализуемые на последующих этапах. Допущенные ошибки, например, при спецификации требований к программному продукту, приводят к огромным потерям на последующих этапах разработки или эксплуатации программного продукта и даже к неуспеху всего проекта. Так, при необходимости внесения изменений в спецификацию программного продукта следует повторить в полном объеме все последующие этапы проектирования и создания программного продукта.

3.4 Защита программных продуктов

Программные продукты и компьютерные базы данных являются предметом интеллектуального труда специалистов высокой квалификации. Процесс проектирования и реализации программных продуктов характеризуется значительными материальными и трудовыми затратами, основан на использовании наукоемких технологий и инструментария, требует применения и соответствующего уровня дорогостоящей вычислительной техники. Это обусловливает необходимость принятия мер по защите интересов разработчика программ и создателей компьютерных баз данных от несанкционированного их использования [5–7].

Программное обеспечение является объектом защиты также и в связи со сложностью и трудоемкостью восстановления его работоспособности, значимостью программного обеспечения для работы информационной системы.

Защита программного обеспечения преследует две цели:

1) ограничение несанкционированного доступа к программам или предотвращение их преднамеренного разрушения и хищения;
2) исключение несанкционированного копирования (тиражирования) программ [5].

Программный продукт и базы данных должны быть защищены по нескольким направлениям от воздействия:

1) человека – хищение машинных носителей и документации программного обеспечения; нарушение работоспособности программного продукта и др.;
2) аппаратуры – подключение к компьютеру аппаратных средств для считывания программ и данных или их физического разрушения;
3) специализированных программ – приведение программного продукта или базы данных в неработоспособное состояние (например, вирусное заражение), несанкционированное копирование программ и базы данных и т. д.

Самый простой и доступный способ защиты программных продуктов и базы данных – ограничение доступа.

Контроль доступа к программному продукту и базе данных строится путем:

- парольной зашиты программ при их запуске;
- использования ключевого внешнего носителя для запуска программ;
- ограничения программ или данных, функций обработки, доступных пользователям, и др. [6].

Могут также использоваться и криптографические методы защиты информации базы данных или головных программных модулей.

Программные системы защиты от несанкционированного копирования предотвращают нелицензионное использование программных продуктов и баз данных. Программа выполняется только при опознании некоторого уникального некопируемого ключевого элемента.

Таким ключевым элементом могут быть:

- внешний носитель, на котором записан не подлежащий копированию ключ;
- определенные характеристики аппаратуры компьютера;
- специальное устройство (электронный ключ), подключаемое к компьютеру и предназначенное для выдачи опознавательного кода.

Программные системы защиты от копирования программных продуктов:

- идентифицируют среду, из которой будет запускаться программа;
- устанавливают соответствие среды, из которой запущена программа, той, для которой разрешен санкционированный запуск;
- вырабатывают реакцию на запуск из несанкционированной среды;
- регистрируют санкционированное копирование;

- противодействуют изучению алгоритмов и программ работы системы [5].

Идентификация среды компьютера обеспечивается за счет:

1) закрепления месторасположения программ на жестком магнитном диске (так называемые неперемещаемые программы);
2) привязки к номеру BIOS (расчет и запоминание с последующей проверкой при запуске контрольной суммы системы);
3) привязки к аппаратному (электронному) ключу, вставляемому в порт ввода-вывода, и др. [7].

На Западе наиболее популярны методы правовой защиты программных продуктов и баз данных, которые включают:

- патентную защиту;
- закон о производственных секретах;
- лицензионные соглашения и контракты;
- закон об авторском праве [5].

Различают две категории прав:

- экономические права, дающие их обладателям право на получение экономических выгод от продажи или использования программных продуктов и баз данных;
- моральные права, обеспечивающие защиту личности автора в его произведении.

Во многих странах несанкционированное копирование программ в целях продажи или бесплатного распространения рассматривается как государственное преступление, карается штрафом или тюремным заключением. Но, к сожалению, само авторское право не обеспечивает защиту новой идеи, концепции, методологии и технологии разработки программ, поэтому требуются дополнительные меры их защиты.

Патентная защита устанавливает приоритет в разработке и использовании нового подхода или метода, примененного при разработке программ, удостоверяет их оригинальность.

Статус ***производственного секрета*** для программы ограничивает круг лиц, знакомых или допущенных к ее эксплуатации, а также определяет меру их ответственности за разглашение секретов. Например, используется парольный доступ к программному продукту или базе данных, вплоть до паролей на отдельные режимы (чтение, запись, корректировку и т. п.). Программы, как лю-

бой материальный объект большой стоимости, необходимо охранять от кражи и преднамеренных разрушений.

Лицензионные соглашения распространяются на все аспекты правовой охраны программных продуктов, включая авторское право, патентную защиту, производственные секреты. Наиболее часто используются лицензионные соглашения на передачу авторских прав.

Лицензия – договор на передачу одним лицом (лицензиаром) другому лицу (лицензиату) права на использование имени, продукции, технологии или услуги. Лицензиар увеличивает свои доходы сбором лицензионных платежей, расширяет область распространения программного продута или базы данных; лицензиат извлекает доходы за счет их применения.

В лицензионном соглашении оговариваются все условия эксплуатации программ, в том числе создание копий. На каждой копии программы должны быть те же отметки, что и на оригинале:

- знак авторского права (обычно ©) и название разработчика, года выпуска программы, прочих ее атрибутов;
- знак патентной защиты или производственного секрета;
- торговые марки, соответствующие использованным в программе другим программным изделиям (обычно – ™ и название фирмы-разработчика программного продукта);
- символ зарегистрированного права на распространение программного продукта (обычно ®).

Существует несколько типов лицензий на программные продукты.

Исключительная лицензия – продажа всех имущественных прав на программный продукт или базу данных, покупателю лицензии предоставляется исключительное право на их использование, а автор или владелец патента отказывается от самостоятельного их применения или предоставления другим лицам.

Это самый дорогой вид лицензии, к нему прибегают для монопольного владения с целью извлечения дополнительной прибыли либо с целью прекращения существования на рынке программных средств программного продукта.

Простая лицензия – лицензиар предоставляет право лицензиату использовать программный продукт или базу данных, оставляя за собой право применять их и предоставлять на аналогичных условиях неограниченному числу лиц (лицензиат при этом не может сам выдавать сублицензии, может лишь продать копии приобретенного программного продукта или базы данных).

Такой вид лицензии приобретают дилер (торговец) либо фирмы-производители, использующие купленные лицензии как сопутствующий товар к основному виду деятельности. Например, многие производители и фирмы, торгующие компьютерной техникой, осуществляют продажу вычислительной техники с установленным лицензионным программным обеспечением (операционная система, текстовый редактор, электронная таблица, графические пакеты и т. д.).

Этикеточная лицензия – лицензия на одну копию программного продукта или базы данных. Данный тип лицензии применяется при розничной продаже. Каждый официальный покупатель заключает лицензионное соглашение с продавцом на их использование, но при этом сохраняется авторское право разработчика.

Экономические отношения между лицензиаром и лицензиатом могут строиться различным образом. За право пользования программным продуктом или базой данных выплачивается единовременное вознаграждение (паушальный платеж), которое и является фактической ценой лицензии. Возможны и периодические отчисления лицензиару за право пользования в виде ***роялти*** – фиксированная ставка в определенные интервалы времени в течение действия лицензионного соглашения, как правило, процент от стоимости программных продуктов или баз данных.

Закон об охране программных продуктов и компьютерных баз данных автором признает физическое лицо, в результате творческой деятельности которого они созданы. Автору независимо от его имущественных прав принадлежат личные авторские права: авторство, имя, неприкосновенность (целостность) программ или баз данных.

Авторское право действует с момента создания программного продукта или базы данных в течение всей жизни автора и 50 лет после его смерти. Автор может:

- выпускать в свет; распространять и модифицировать;
- воспроизводить в любой форме, любыми способами;
- осуществлять любое иное использование программного продукта или базы данных [4].

Авторское право не связано с правом собственности на материальный носитель. Имущественные права на программный продукт или базу данных могут быть переданы частично или полностью другим физическим или юридическим лицам по договору. Имущественные права относятся к категории наследуемых.

Если программный продукт или база данных созданы в порядке выполнения служебных обязанностей, имущественные права принадлежат работодателю.

Программные продукты и базы данных могут использоваться третьими лицами – пользователями на основании договора с правообладателем.

Лицо, правомерно владеющее экземпляром программы или базы данных, вправе, без получения дополнительного разрешения правообладателя, осуществлять любые действия, связанные с функционированием программного продукта или базы данных в соответствии с ее назначением, в том числе:

- исправлять явные ошибки;
- адаптировать программный продукт или базу данных;
- изготавливать страховые копии и т. д.

4 Среды разработки информационной системы

4.1 Состав информационных систем

Информационная система *– это взаимосвязанная совокуп- ность средств, методов и персонала, используемых для хранения, обработки и выдачи информации в интересах достижения поставленной цели.*

Современное понимание информационной системы предполагает использование в качестве основного технического средства переработки информации компьютера. Кроме того, техническое воплощение информационной системы само по себе ничего не будет значить, если не учтена роль человека, для которого предназначена производимая информация и без которого невозможно ее получение и представление [8].

Необходимо понимать разницу между компьютерами и информационными системами. Компьютеры, оснащенные специализированными программными средствами, являются технической базой и инструментом для информационных систем. Информационная система немыслима без персонала, взаимодействующего с компьютерами и телекоммуникациями.

В нормативно-правовом смысле информационная система определяется как «организационно упорядоченная совокупность документов (массив документов) и информационных технологий, в том числе и с использованием средств вычислительной техники и связи, реализующих информационные процессы»[1].

В целом информационные системы определяются следующими свойствами:

- любая информационная система может быть подвергнута анализу, построена и управляема на основе общих принципов построения систем;
- информационная система является динамичной и развивающейся;
- при построении информационной системы необходимо использовать системный подход;

[1]Закон РФ «Об информации, информатизации и защите информации» от 20.02.1995 г., № 24-ФЗ.

- выходной продукцией информационной системы является информация, на основе которой принимаются решения;
- информационную систему следует воспринимать как человеко-машинную систему обработки информации [9].

Внедрение информационных систем может способствовать:

- получению более рациональных вариантов решения управленческих задач за счет внедрения математических методов;
- освобождению работников от рутинной работы за счет ее автоматизации;
- обеспечению достоверности информации;
- совершенствованию структуры информационных потоков (включая систему документооборота);
- предоставлению потребителям уникальных услуг;
- уменьшению затрат на производство продуктов и услуг (включая информационные) [10].

Общую структуру информационной системы можно рассматривать как совокупность подсистем независимо от сферы применения. В этом случае говорят о структурном признаке классификации, а подсистемы называют обеспечивающими. Таким образом, структура любой информационной системы может быть представлена совокупностью обеспечивающих подсистем, среди которых обычно выделяют информационное, техническое, математическое, программное, организационное и правовое обеспечение.

Информационное обеспечение *– совокупность единой си- стемы классификации и кодирования информации, унифицирован- ных систем документации, схем информационных потоков, цирку- лирующих в организации, а также методология построения баз данных.*

Назначение подсистемы *информационного обеспечения* состоит в своевременном формировании и выдаче достоверной информации для принятия управленческих решений.

Для создания информационного обеспечения необходимо:

- ясное понимание целей, задач, функций всей системы управления организацией;

- выявление движения информации от момента возникновения и до ее использования на различных уровнях управления, представленной для анализа в виде схем информационных потоков;
- совершенствование системы документооборота;
- наличие и использование системы классификации и кодирования;
- владение методологией создания концептуальных информационно-логических моделей, отражающих взаимосвязь информации;
- создание массивов информации на машинных носителях, что требует наличия современного технического обеспечения.

4.2 Системы управления базами данных (СУБД)

4.2.1 Классификация СУБД

По **степени универсальности** различают два класса СУБД:

1) системы общего назначения;
2) специализированные системы [11].

СУБД общего назначения не ориентированы на какую-либо предметную область или на информационные потребности какой-либо группы пользователей. Каждая система такого рода реализуется как программный продукт, способный функционировать на некоторой модели ЭВМ в определенной операционной системе, и поставляется многим пользователям как коммерческое изделие. Такие СУБД обладают средствами настройки на работу с конкретной базой данных. Использование СУБД общего назначения в качестве инструментального средства для создания автоматизированных информационных систем, основанных на технологии баз данных, позволяет существенно сокращать сроки разработки, экономить трудовые ресурсы. Этим СУБД присущи развитые функциональные возможности и даже определенная функциональная избыточность.

Специализированные СУБД создаются в редких случаях при невозможности или нецелесообразности использования СУБД общего назначения [12].

СУБД общего назначения – это сложные программные комплексы, предназначенные для выполнения всей совокупности функций, связанных с созданием и эксплуатацией базы данных информационной системы.

Рынок программного обеспечения ПК располагает большим числом разнообразных по своим функциональным возможностям коммерческих систем

управления базами данных общего назначения, а также средствами их окружения практически для всех массовых моделей машин и для различных операционных систем.

Используемые в настоящее время СУБД обладают средствами обеспечения целостности данных и надежной безопасности, что дает возможность разработчикам гарантировать большую безопасность данных при меньших затратах сил на низкоуровневое программирование. Продукты, функционирующие в среде Windows, выгодно отличаются удобством пользовательского интерфейса и встроенными средствами повышения производительности.

Рассмотрим основные характеристики некоторых СУБД – лидеров на рынке программ, предназначенных как для разработчиков информационных систем, так и для конечных пользователей.

По **технологии обработки** данных базы данных подразделяются на централизованные и распределенные [11–12].

Централизованная база данных хранится в памяти одной вычислительной системы. Если эта вычислительная система является компонентом сети ЭВМ, возможен распределенный доступ к такой базе. Этот способ использования баз данных часто применяют в локальных сетях ПК.

Распределенная база данных состоит из нескольких, возможно пересекающихся или даже дублирующих друг друга частей, хранимых в различных ЭВМ вычислительной сети. Работа с такой базой осуществляется с помощью системы управления распределенной базой данных (СУРБД).

По **способу доступа** к данным базы данных разделяются на базы данных с ***локальным доступом*** и базы данных с ***удаленным (сетевым) доступом***. Системы централизованных баз данных с сетевым доступом предполагают различные *архитектуры* подобных систем:

- файл-сервер;
- клиент-сервер [12].

Файл-сервер. Архитектура систем БД с сетевым доступом предполагает выделение одной из машин сети в качестве центральной (сервер файлов). На такой машине хранится совместно используемая централизованная БД. Все другие машины сети выполняют функции рабочих станций, с помощью которых поддерживается доступ пользовательской системы к централизованной базе данных. Файлы базы данных в соответствии с пользовательскими запросами передаются на рабочие станции, где в основном и производится обработка. При большой интенсивности доступа к одним и тем же данным производительность

информационной системы падает. Пользователи могут создавать на рабочих станциях локальные БД, которые используются ими монопольно. Концепция файл-сервер условно отображена на рисунке 4.1.

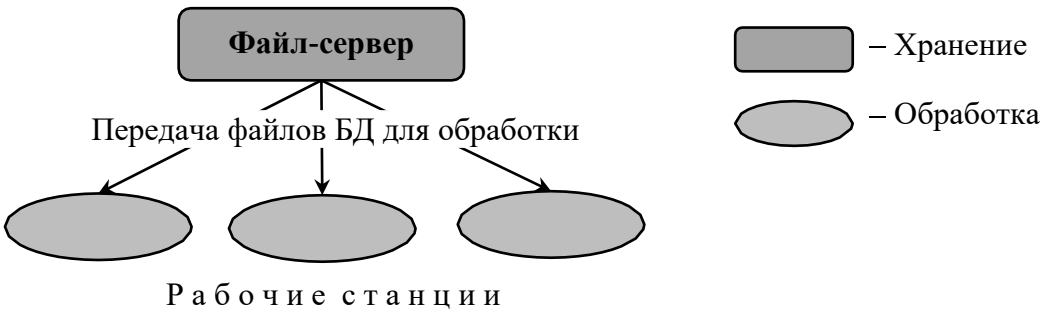


Рис. 4.1 – Схема обработки информации в БД по принципу файл-сервер

Клиент-сервер. В этой концепции подразумевается, что помимо хранения централизованной базы данных центральная машина (сервер базы данных) должна обеспечивать выполнение основного объема обработки данных. Запрос на данные, выдаваемый клиентом (рабочей станцией), порождает поиск и извлечение данных на сервере. Извлеченные данные (но не файлы) транспортируются по сети от сервера к клиенту. Спецификой архитектуры клиент-сервер является использование языка запросов SQL. Концепция клиент-сервер условно изображена на рисунке 4.2.

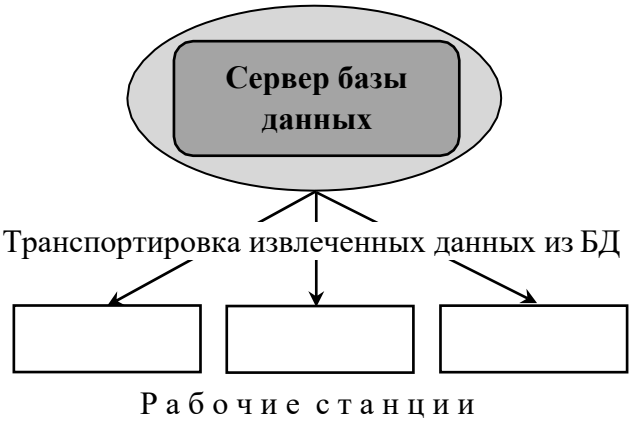


Рис. 4.2 – Схема обработки информации в БД по принципу клиент-сервер

Системы управления базами данных разделяются на серверные и клиентские [13].

Примеры серверных СУБД: Caché, DB2, Firebird, Informix, Ingres, InterBase, MSDE, MS SQL Server, MySQ, mSQL, Oracle, Pervasive SQL, PostgreSQL, Sybase ASE, Sybase ASA, Sybase IQ, Teradata, ЛИНТЕР, Mnesia, H2.

Примеры клиентских СУБД: DataFlex, dBase, MS Access, Paradox, OpenOffice.org Base, Sav Zigzag.

4.2.2 Корпоративные СУБД

Функционирование практически любого современного предприятия немыслимо без манипуляции данными, связанными с его производственной деятельностью. Нередко эффективность его деятельности и конкурентоспособность на рынке товаров или услуг непосредственно связаны с тем, актуальны ли эти данные и доступны ли они обращающимся к ним пользователям (причем нередко не только пользователям локальной сети, но и посетителям корпоративного Web-сервера и сотрудникам, обращающимся к ним с помощью мобильных устройств). С этой целью применяются различные архитектуры физического хранения данных, такие как Storage Area Network (SAN) или Network Attached storage (NAS), а также системы управления базами данных, предназначенные для логической организации данных и осуществления доступа к ним.

Корпоративные данные большинства компаний сейчас хранятся в реляционных СУБД.

В простейшем случае корпоративная информационная система, использующая СУБД, состоит из двух основных компонентов: сервера баз данных, управляющего данными и выполняющего поступающие от клиентских приложений запросы, и самих клиентских приложений, обеспечивающих интерфейс пользователя и посылающих запросы к серверу. Именно сервер баз данных может манипулировать файлами, в которых хранятся данные, выполнять пользовательские запросы, поддерживать ссылочную целостность данных, обеспечивать доступ к ним, осуществлять резервное копирование данных и протоколировать операции, связанные с их изменением.

К современным реляционным СУБД предъявляются следующие требования:

- ***масштабируемость***, то есть способность одновременно обслуживать большее количество пользовательских запросов с той же скоростью при пропорциональном этому количеству увеличении объема предоставляемых ресурсов (процессоров, оперативной памяти и т. д.);
- ***доступность***, то есть постоянная возможность получения ответа на запрос;

- ***надежность***, то есть минимальная вероятность сбоев, а также наличие средств восстановления данных после сбоев, резервирования и дублирования;
- ***управляемость***, то есть простота администрирования и конфигурирования, а нередко и наличие средств автоматического конфигурирования (обычно набор средств администрирования включает средства создания баз данных и их объектов, инструменты репликации данных между различными серверами, утилиты управления пользователями и группами, средства мониторинга событий, средства просмотра планов выполнения запросов, утилиты миграции из других СУБД);
- ***наличие средств защиты*** данных от потери и несанкционированного доступа;
- ***поддержка стандартных механизмов доступа к данным*** (ODBC, JDBC, OLE DB, ADO) [14, 15].

Как правило, отсутствие какого-либо из этих признаков приводит к тому, что даже у неплохой по другим потребительским свойствам СУБД область применения оказывается весьма ограниченной.

Так, СУБД с плохой масштабируемостью, успешно применявшаяся при небольшом обрабатываемом объеме данных, оказывается непригодной при его росте, и нередко ее приходится заменять на другую; при этом неизбежны определенные затраты на переписывание серверного кода. Лишние затраты на администрирование обычно тоже никому не нужны. Плохие масштабируемость и доступность влекут за собой дополнительные затраты рабочего времени сотрудников, простои, а также потерю компанией клиентов, отчаявшихся дождаться нужных данных на корпоративном сайте и вынужденных обратиться на сайт конкурента.

Именно поэтому лидеры рынка корпоративных СУБД стремятся к производству продуктов, удовлетворяющих всем вышеуказанным требованиям. Кроме того, как правило, подобные продукты существуют для нескольких платформ, а нередко и в разных редакциях, предназначенных для решения разнообразных задач или обслуживания различного количества данных и пользователей. Из последних тенденций развития корпоративных СУБД следует отметить поддержку XML и Web-сервисов XML.

Существует два типа реляционных баз данных:

1) оперативные, или OLTP (OLTP – On-Line Transaction Processing), базы данных. Обычно в эти базы данных осуществляется интенсивный ввод данных, а вот число адресованных к ним запросов невелико;
2) хранилища данных, применяемые, как правило, в аналитических приложениях и системах поддержки принятия решений. К ним обычно адресуется большое число запросов, но ввод данных в них не столь интенсивен [13].

Отметим, что многие современные СУБД с успехом поддерживают создание баз данных обоих типов – все определяется тем, как будет спроектирована структура данных. Однако нередко для создания хранилищ данных применяются специальные СУБД, способ хранения данных в которых особым образом оптимизирован для ускорения выполнения запросов. И, как правило, создание OLAP-хранилищ, основанных на нереляционных многомерных базах данных, требует наличия отдельных серверных продуктов.

В заключение отметим, что существующие на сегодняшний день возможности СУБД ведущих производителей отражают современные тенденции развития информационных систем, такие как использование многопроцессорных систем и распределенной обработки данных, создание распределенных систем, применение средств быстрой разработки приложений, создание систем поддержки принятия решений с использованием аналитической обработки данных, а также все возрастающие требования к надежности и отказоустойчивости.

4.2.3 Обоснование выбора СУБД при проектировании информационной системы

Ниже приведены основные требования к СУБД, которые надо учитывать при проектировании информационной системы.

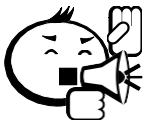
1. **Основные показатели СУБД**. Система управления базами данных (СУБД) отвечает за агрегирование данных и их последующее хранение и обработку.

СУБД управляется на языках работы с базами данных (БД), например SQL (Structured Query Language). СУБД основаны на реляционной модели данных. Реляционная модель – представление БД в виде таблиц для действий над записями на языке SQL. Реляционные системы – это системы «автоматической навигации». SQL – более абстрактный язык, чем C++, т. к. способ запроса остается на выбор оптимизатора СУБД. «Постреляционная СУБД» – наличие в ре-

ляционной СУБД файлов управления данными, не вписывающихся в реляционную модель, т. е. объектов.

Ранее данные хранились только в алфавитно-цифровой форме, классифицировались по стандартным типам (строки, целые числа и т. д.). Теперь сюда включаются и бинарные объекты: изображения, видео и большие фрагменты текста, по которым может происходить поиск.

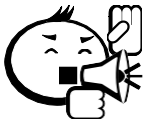
Другим необходимым элементом СУБД является встроенный язык программирования для автоматизации процедур обслуживания системы и обработки данных внутри СУБД ее собственными средствами. Пользовательские приложения взаимодействуют в СУБД в рамках двух- или трехуровневой клиент-серверной архитектуры. Следовательно, физический сервер, на который установлена СУБД, называется сервером БД. Администрирование СУБД включает в себя создание БД, управление и обслуживание инфраструктуры сервера [16].



Выбор СУБД зависит от тех приложений, которыми она будет управляться.

Выбор СУБД – прерогатива разработчика, а не пользователя.

Ведущие поставщики СУБД: IBM, Oracle и Microsoft.



При выборе СУБД необходимо руководствоваться такими показателями, как масштабируемость, быстродействие (как в выборе транзакций, так и в построении сложных аналитических выборок), работа с XML и кластерные решения.

В среднем скорости работы IBM, DB/2, MS SQL и Oracle примерно одинаковы. На общем фоне выделяются только Cache из-за новизны подхода и особой идеологии архитектуры [17].

Масштабируемость. Чем больше данных, тем сложнее ими управлять. Например, СУБД Oracle10g существует в нескольких вариантах, с разными схемами лицензирования. Для всех версий существует одно ядро, все версии совместимы.

Мультиплатформенность. Oracle и IBM DB/2 также расширяют возможности масштабирования: можно менять аппаратную платформу и ОС на более соответствующую растущим потребностям бизнеса без потерь данных, смены прикладного ПО и переподготовки администратора БД.

Кластерные технологии в приложении к СУБД, например по технологии Oracle RAC, повышают надежность системы, упрощают масштабируемость и снимают расходы на развитие инфраструктуры.

Различные СУБД отличаются характерными чертами. Например, IBM DB/2 имеет собственную высокопроизводительную кластерную структуру, которая позволяет переходить от больших RISC-серверов в качестве серверов БД к мейнфреймам. Oracle поддерживает XML DB. Oracle и IBM DB/2 поддерживают SQLJ, что особенно важно в телекоммуникации.

2. **Требования к «рабочему месту» веб-разработчика**. Анализ «рабочего места» разработчика подразумевает разговор о веб-серверах, СУБД и языках написания сценариев, т. е. «жизненно необходимого» пакета веб-программиста.

Веб-сервер – это сервер, принимающий HTTP-запросы от клиентов и вызывающий HTTP-ответы, как правило, вместе с HTML-страницей, изображением или мультимедийным объектом. Говоря о веб-сервере, имеют в виду как программное обеспечение (ПО), так и отдельный компьютер, на котором это ПО установлено.

Клиент (веб-браузер) подает веб-серверу запросы на получение ресурсов, обозначаемых URL-адресами.

Дополнительные функции веб-сервера:

- ведение журнала обращений пользователя к ресурсам;
- аутентификация и авторизация пользователей;
- поддержка динамически генерируемых страниц;
- поддержка HTTPS защищенных соединений с клиентами [18].

Существуют также легкие веб-серверы, например lighttpd, litespeed, mongrel.

Они имеют следующие параметры:

- эффективность (быстрота реакции);
- масштабируемость (для многих пользователей);
- безопасность (не выходит ли за дозволенные рамки операций; шифрование; делает ли более уязвимыми соседей);
- работоспособность: проверяется в режимах отказа и аварийности;
- соответствие стандартам (протокол RFC в разновидностях);
- гибкость (можно ли настроить сервер для принятия большого числа запросов или динамических страниц, требующих значительных вычислений);

- требования к платформе: на каких платформах установлен;
- управляемость: легко ли установить и обслуживать сервер [18].

Легкие веб-серверы могут конкурировать с Apache и IIS, а также соединяться с ними для ускорения работы.

В чем «легкость» веб-серверов?

- Веб-сервер является простым, удобно устанавливаемым, нетребовательным и устойчивым (некоторые стали громоздкими, например Java Web Server, AOLServer и Zeus).
- Весь веб-сервер способен поместиться в одном файле, машины могут иметь встроенные, доступные через веб управляющие консоли без сложной разработки и накладных расходов.
- Веб-серверы имеют открытый исходный код.
- Сервис-ориентированные архитектуры SOA довольно капризны, легкие серверы быстро устанавливают SOA для демонстраций.
- Некоторые легкие веб-серверы ускоряют передачу видео и изображений Apache веб-сервером.

Веб-серверы обычно пишутся на C++, Erlang, Java, Lisp, Perl, Pythonu Tcl. Зачем использовать редкий язык?

- В целях образования: программист получает опыт работы с языком.
- На C++ некоторые файлы довольно громоздкие, в языках высокого уровня удается существенно уменьшить размер файлов.
- Языки высокого уровня облегчают экспериментальные возможности программиста: веб-серверы являются удобным экспериментальным материалом.
- Легкая модификация: добавление HTTP-сервера к существующему приложению может потребовать увеличение исходного кода только на несколько строк.

Все «легкие» веб-серверы являются узконаправленными объектами. В некоторых приложениях упор идет на быстродействие, в некоторых – на безопасность или экономию памяти. Ниже приведены «легкие» веб-серверы с примерами:

- Ультралегкие (Cheetah Server, DustMote, fnord, ihttpd, mattous, Scrinchy, ZWS).
- Высокопроизводительные серверы (cghttpd, darkhttpd, Gatling, Kernux, lighttpd, Light Speed Web Server, Miniature JWS, Yaws).

- Как классы используются библиотеки (EHC, Embedded TCL Web Server).
- Веб-серверы, написанные на Python (cdServer, edna).
- Веб-серверы, написанные на Perl и других, менее известных языках (Camlserver, dhttpd, DNHTTPD, Jellybean, lns.http, Mongrel, Nanoweb, Naridesh, OpenAngel, Xavante, XSP).
- Веб-серверы, написанные на C++ (ABYSS, Anti_Web HTTPD, MHTTPD, mini_httpd, Nullhttpd, Seminode, thttpd) [18].

3. **Требования к аппаратным и программным ресурсам**. Рассмотрим минимальные требования к аппаратным и программным ресурсам для работы веб-сервера Apache, а значит, и всей работы веб-программиста [19]. Как известно, Apache изначально был разработан для работы на платформах Unix, теперь же существуют и конфигурации, работающие под Windows XP или Vista. Поэтому было бы логично раздельно рассмотреть системные требования для разных типов платформ.

Согласно исследованию, проведенному netcraft.com, большинство акций разработчиков принадлежали Apache (47,17%) и Microsoft (23,34%). Из всех активных сайтов 51,12% работали на Apache, 23,99% – на Microsoft. Касательно распределения рынка акций топ-серверов, Apache принадлежит 66,82% всех акций, Microsoft принадлежит 18,25%. Вышеуказанные цифры довольно ясно демонстрируют, что Apache захватил более 50% рынка [19].

4. **Требования к обеспечению информационной безопасности**. Хакерским атакам подвержены все серверы. Это может быть как крупная корпорация, так и малая сеть, в которых хакерские программы сканируют весь IP-диапазон и ищут уязвимые места в сети. Даже если работать в локальной сети, а не в Интернете, существует вероятность заражения троянами через флэш-носители или ноутбуки.

Самые распространенные угрозы и проблемы, возникающие вследствие хакерских атак, а также меры, при помощи которых можно ликвидировать эти угрозы, приведены ниже.

- Угроза доступа к файлам .httppasswd и .httpaccess. Данные файлы отвечают за авторизацию пользователей. Бороться с этой угрозой можно регулярно проходя авторизацию.
- Угроза доступа через уязвимости приложений. Функциональный уровень приложений может вызывать проблемы из-за ошибок кода и утечек (например, если в программе существуют скрытые «лазейки»

быстрого доступа к данным, которые изначально программист писал «под себя» для ускорения процессов отладки приложения и просто забыл убрать). Бороться с данной угрозой можно посредством недопуска к коду программы. Кроме того, можно дописывать ядро, если работа идет под Unix.

- Угрозы межскриптовой уязвимости (например, посредством тегов HTML). С этой угрозой можно бороться посредством фильтрации определенных символов прокси веб-сервером.
- Проблемы переполнения буфера памяти в Apache. Неконтролируемые потоки данных способны привести к отказу сервера авторизовать реальных пользователей. Бороться с такой проблемой можно при помощи регулярного обновления сервера Apache.
- Сигналы к различным процессам. Хакер может посылать сигналы различным процессам и тем самым «убивать» или «порождать» лишние процессы. В нескольких версиях Apache с этой угрозой можно бороться посредством ограничения прав доступа пользователей к файловой системе и процессам.

Ко всем возможным угрозам применяются следующие контрмеры:

- постоянное обновление Apache; мониторинг активности;
- определение вторжения; понимание конфигурации;
- регулярная проверка посредством антивирусных программ;
- отключение ненужных модулей и ненужных файлов Apache;
- механизмы аутентификации; безопасный доступ администратора [19].

5. **Критерии выбора платформы для разработки веб-приложения**.

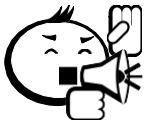
При написании веб-приложения необходимо учитывать особенности самого веб-приложения и те программные пакеты, которыми оно будет дополнено [18].

Необходимо исходить из того, что пишется веб-приложение в основном на HTML, оно имеет PHP-сценарии, обращающиеся к базам данных MySQL, использует JavaScript почтовый сервер, обращается к XML-файлам, ссылается на CSS-файлы. Кроме того, зачастую требуется участие Flash-сценариев, сценариев Java, CGI-сценариев; существует возможность регистрировать и авторизовать пользователей.

В настоящее время существуют версии большинства приложений, поддерживаемые на той или иной платформе (например, Flash был изначально разработан под Windows, но в настоящее время существуют версии, работающие

под разновидностями Unix). То есть обычно программисты руководствуются набором приложений, поддерживаемым веб-сервером, предоставляющим хостинг. Если веб-приложение имеет небольшое количество Unix-ориентированных скриптов (например, на Perl или на Python), насыщено графикой, видео, Flash-скриптами и т. д., то есть смысл выбрать именно Windows-платформу, а не Unix. Кроме того, если приложение написано именно под Windows и является частью веб-приложения, то есть смысл либо полного, либо частичного перехода на платформу Windows (здесь есть смысл экономической оценки: стоит ли покупать или переписывать приложение под Unix или дешевле сменить платформу).

Если в приложении присутствует большое число скриптов, написанных на Perl, Python, C++, кроме того, приложение выполняет параллельные процессы с другими приложениями, то есть смысл выбрать платформу Unix.



Другим важным моментом в вопросе выбора платформы является ориентация приложения на браузеры.

Например, если идет ориентация на IE, то он не будет работать на плат- форме Unix. Если разрабатывается Java-апплет, к примеру J-Ads2, то при закач- ке на сервер он должен работать. Для этого сервер должен поддерживать один из соответствующих браузеров (Netscape 3.x, Netscape 4.0x, 4.x, IE4.x, 5.x или выше, Opera с плагином Java, NeoPlanet...), поддерживающих AWT. Здесь необходимо отметить, что Netscape 4.61 работает только под Mac OS, который не поддерживает AWT. При работе с Java-машиной предпочтительнее платформа Windows.

Необходимо учесть, что веб-приложение и сама система становятся более уязвимыми для вирусов, поскольку многие вирусы, например «черви», пишутся как апплеты или сервлеты, т. е. как Java-приложения.

4.3 Клиентские приложения, обеспечивающие интерфейс пользователя

В настоящее время существует большое количество инструментальных средств для разработки интерфейса, поддерживающих различные методы его реализации [20].

Основное назначение тех средств разработки пользовательских графических интерфейсов, которые разрабатываются и поставляются отдельно от окон-

ной системы, – облегчение создания нового графического интерфейса за счет использования существующих параметризованных заготовок [20]. Как видно, в принципе это те же самые идеи, на которых основана объектно-ориентированная библиотека оконной системы X Xt Intrinsics.

И действительно, наиболее распространенный пакет, предназначенный для быстрой и качественной разработки графических пользовательских интерфейсов, Motif, который был спроектирован и разработан в североамериканском консорциуме OSF, в основном является развитием идей Xt Intrinsics. Motif – это сугубо коммерческий продукт. Дело дошло до того, что компания OSF запатентовала внешний интерфейс продуктов, входящих в состав Motif, чтобы не дать кому-нибудь возможность воспроизвести этот интерфейс.

В Калифорнийском университете г. Беркли был создан альтернативный механизм под названием Tcl/Tk. Этот механизм основан на наличии специализированного командного языка, предназначенного для описания графических пользовательских интерфейсов, соответствующего интерпретатора и библиотеки ранее разработанных заготовок интерфейсов. Пакет Tcl/Tk распространяется (вместе с полной документацией) свободно, и многие профессиональные программисты находят его более удобным, чем Motif.

1. ***Пакет Motif***.

Motif (официальное название этого продукта – OSF/Motif) представляет собой программный пакет, включающий оконный менеджер, набор вспомогательных утилит, а также библиотеку классов, построенных на основе Xt Intrinsics. Для конечных пользователей оконных систем, опирающихся на Motif, основной интерес представляет менеджер окон.

Для разработчиков же графических интерфейсов важны все три компонента Motif. Новый интерфейс разрабатывается в графическом же режиме с использованием оконного менеджера. При этом полезно использование утилит Motif и необходимо использование библиотеки классов Motif.

Библиотека классов Motif является расширением библиотеки Xt Intrinsics с целью придания этой библиотеке практического смысла (по-другому можно сказать, что Motif – это то, чем должен был бы быть Xt, если бы при его создании ставились коммерческие цели). Все графические объекты (правильнее сказать, классы) Xt Intrinsics включаются в библиотеку классов Motif, хотя в ней используются другие имена.

Но Motif существенно расширяет возможности Xt Intrinsics. В его библиотеке поддерживается большое число классов, позволяющих создавать меню,

«нажимаемые» кнопки и т. д. Основное назначение этих классов – определение новых виджетов, связанных с окнами.

Однако в Motif поддерживается и новый вид графических объектов (их классов) – так называемые гаджеты (gadgets). Гаджет отличается от виджета тем, что соответствующий класс также может использоваться для создания элементов интерфейса, но графический объект не привязывается к определенному окну. При отображении на экран гаджета используется окно объекта, относящегося к суперклассу класса гаджета.

Основная идея Motif: развитая библиотека классов языка Си++, возможности применения этих классов при использовании обычного стиля программирования и поддержка визуального программирования с немедленным отображением получающихся графических объектов.

2. ***Язык и интерпретатор Tcl/Tk***.

Продукт Tcl/Tk в действительности представляет собой два связанных программных пакета, которые совместно обеспечивают возможность разработки и использования приложений с развитым графическим пользовательским интерфейсом [21]. Название Tcl относится к командному языку инструментальных средств – tool command language. Это простой командный язык для управления приложениями и расширения их возможностей. Язык Tcl является «встраиваемым»: его интерпретатор реализован в виде библиотеки функций языка Си++, так что интерпретатор может быть легко пристыкован к любой прикладной программе, написанной на языке Си++.

Tk (рекомендуемое произношение – «ти-кей») является библиотекой Си-функций, ориентированной на облегчение создания пользовательских графических интерфейсов в среде оконной системы X (т. е., по сути дела, некоторый аналог Xt Intrinsics). С другой стороны, аналогично тому, как это делается в командных языках семейства shell, функции библиотеки Tk являются командами языка Tcl, так что любой программист может расширить командный репертуар языка Tcl путем написания новой функции на языке Си++.

Совместно Tcl и Tk обеспечивают четыре преимущества для разработчиков приложений и пользователей (мы используем здесь авторские тексты разработчиков). Во-первых, наличие командного языка Tcl дает возможность в каждом приложении использовать мощный командный язык. Все, что требуется от разработчика приложения, чтобы удовлетворить его специфические потребности, – это создать несколько новых команд Tcl, требующихся приложению (и, возможно, другим приложениям – явно традиционный стиль командного

программирования в ОС UNIX). После этого нужно связать прикладную программу с интерпретатором Tcl и пользоваться полными возможностями командного языка [21].

Вторым преимуществом использования Tcl/Tk является возможность быстрой разработки графических интерфейсов. Многие интересные оконные приложения могут быть написаны в виде скриптов языка Tcl без привлечения языков Си или Си++ (а Tcl позволяет скрыть многие несущественные детали). Как утверждают разработчики Tcl/Tk, пользователи оказываются способными к созданию новых графических интерфейсов уже после нескольких часов знакомства с продуктом. Другой особенностью языка Tcl, способствующей быстрой разработке оконных приложений, является то, что язык является интерпретируемым. Можно опробовать новую идею интерфейса, выражающуюся в сотнях или тысячах строк кода на языке Tcl, без потребности вызова новых программных средств, путем простого нажатия на клавишу мыши (не наблюдая существенных задержек при использовании современных рабочих станций).

Третьим преимуществом языка Tcl является то, что его можно применять в качестве языка «склейки» приложений. Например, любое основанное на Tcl и использующее Tk оконное приложение может направить свой скрипт любому другому аналогично ориентированному приложению. С использованием Tcl/Tk можно создавать приложения, работающие в стиле мультимедиа, и опять же они смогут обмениваться скриптами, поскольку пользуются общим интерпретатором командного языка Tcl и общей внешней библиотекой Tk [21].

Наконец, четвертым удобством интегрированного пакета Tcl/Tk является удобство пользователей. Для написания нового приложения в среде Tcl/Tk достаточно выучить несколько совершенно необходимых команд и этого окажется достаточно.

3. ***Microsoft Expression Blend – инструмент создания интерфейсов***.

Появление языка описания пользовательских интерфейсов XAML (произносится – «за́ммель») и новой среды разработки Expression Blend позволяет заметно ускорить и облегчить проектирование и построение пользовательских интерфейсов как для веб-, так и для настольных приложений [20].

Данный язык позволяет описывать внешний вид и поведение интерфейсных элементов, устанавливать взаимодействие этих элементов с различными данными и событиями. Допускает прямое подключение к Common Language Runtime (CLR), что обеспечивает большую гибкость при проектировании ПО.

Blend обладает разветвленными возможностями для построения качественных интерфейсов и его главной возможностью является создание пользовательских библиотек-стилей, содержащих интерфейсные элементы с заранее заданным внешним видом и поведением. Blend является мощным приложением для создания пользовательских интерфейсов.

Blend – программа для создания уже готовых интерфейсов, т. е. дизайнер выдает программистам готовый интерфейс, не требующий их вмешательства в графическое решение. Программист только подключает интерфейс к процедурному коду. Данная концепция является достаточно новаторской как для дизайнеров интерфейсов, так и для программистов. С точки зрения дизайнера интерфейсов, Blend является дополнительным инструментом при проектировании интерфейсов, так как проектирование интерфейсов – это не только и даже не совсем внешний вид, а в первую очередь взаимодействие программы и человека, а инструменты, позволяющие это делать в Blend, малоразвиты или отсутствуют совсем.

Преимущества новой технологии.

Прежде всего, гибкость при создании приложений, которая обеспечивается наличием современных средств визуализации и новых технологий:

- Векторная графика: теперь интерфейс состоит полностью из векторных объектов (интерфейсные элементы, графика, пиктограммы).
- Новые экранные шрифты и новая технология попиксельного позиционирования изображения на экране.
- Одна программа может содержать несколько интерфейсов (разные разрешения, веб- и настольные приложения и т. д.) [20].

Дизайнер и программист могут одновременно работать над одним проектом, каждый выполняя свою функцию, что обеспечивает гибкость при создании приложений и увеличивает скорость работы.

5 Проектирование информационной системы

5.1 Этапы создания информационных систем

Создание электронных информационных систем (ЭИС), осуществляется на основе требований со стороны предполагаемых пользователей, которые, как правило, изменяются в процессе разработки. С точки зрения теории принятия решений процесс проектирования ЭИС – это процесс принятия проектно-конструкторских решений, направленных на получение описания системы (проекта ЭИС), удовлетворяющего требованиям заказчика [22–25].

*Под **проектом ЭИС** понимают проектно-конструкторскую и технологическую документацию, в которой представлено описание проектных решений по созданию и эксплуатации ЭИС в конкретной программно-технической среде.*

*Под **проектированием ЭИС** понимается процесс преобразования входной информации об объекте проектирования, о методах проектирования и об опыте проектирования объектов аналогичного назначения в соответствии с ГОСТом в проект ЭИС.*

С этой точки зрения проектирование ЭИС сводится к последовательной формализации проектных решений на различных стадиях жизненного цикла ЭИС: планирования и анализа требований, технического и рабочего проектирования, внедрения и эксплуатации ЭИС.

Объектами проектирования ЭИС являются отдельные элементы или их комплексы функциональных и обеспечивающих частей. Так, функциональными элементами в соответствии с традиционной декомпозицией выступают задачи, комплексы задач и функции управления. В составе обеспечивающей части ЭИС объектами проектирования служат элементы и их комплексы информационного, программного и технического обеспечения системы.

Осуществление проектирования ЭИС предполагает использование проектировщиками определенной технологии проектирования, соответствующей масштабу и особенностям разрабатываемого проекта [23].

***Технология проектирования ЭИС** – это совокупность ме- тодов и средств проектирования ЭИС, а также методов и*

средств организации проектирования (управления процессом создания и модернизации проекта ЭИС) (рис. 5.1).

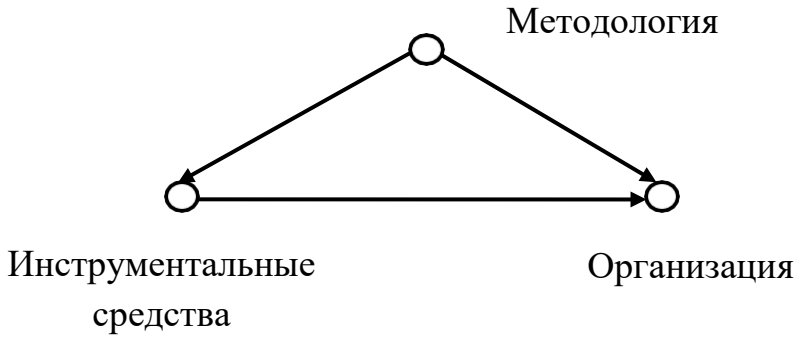


Рис. 5.1 – Состав компонентов технологии проектирования

В основе технологии проектирования лежит технологический процесс, который определяет действия, их последовательность, состав исполнителей, средства и ресурсы, требуемые для выполнения этих действий.

Так, технологический процесс проектирование ЭИС в целом делится на совокупность последовательно-параллельных, связанных и соподчиненных цепочек действий, каждое из которых может иметь свой предмет. Действия, которые выполняются при проектировании ЭИС, могут быть определены как неделимые технологические операции или как подпроцессы технологических операций [24]. Все действия могут быть собственно проектировочными, которые формируют или модифицируют результаты проектирования, и оценочными, которые вырабатывают по установленным критериям оценки результатов проектирования.

Таким образом, технология проектирования задается регламентированной последовательностью технологических операций, выполняемых в процессе создания проекта на основе того или иного метода, в результате чего стало бы ясно, не только ***что*** должно быть сделано для создания проекта, но и ***как, кому*** и в ***какой последовательности*** это должно быть сделано.

Принято выделять два уровня представления модели данных – логический и физический [25].

Цель моделирования данных на **логическом уровне** состоит в обеспечении разработчика ИС концептуальной схемой базы данных в форме одной модели или нескольких локальных моделей, которые относительно легко могут быть отображены в любую систему баз данных.

Логический уровень – *это абстрактный взгляд на данные, на нем данные представляются так, как выглядят в реальном мире, и могут называться так, как они называются в реальном мире, например «Отдел», «Фамилия сотрудника».*

Объекты, модели, представляемые на логическом уровне, называются сущностями и атрибутами. Логическая модель данных может быть построена на основе другой логической модели, например модели процессов. Такая модель данных является универсальной и никак не связана с конкретной реализацией СУБД (системы управления базой данных). Построение логической модели ИС до ее программной разработки или до начала проведения архитектурной реконструкции столь же необходимо, как наличие проектных чертежей перед строительством большого здания. Хорошие модели ИС позволяют наладить плодотворное взаимодействие между заказчиками, пользователями и командой разработчиков.

*На **физическом уровне** данные, напротив, зависят от конкретной СУБД, фактически являясь отображением системного каталога. В физической модели содержится информация обо всех объектах БД.*

Поскольку стандартов на объекты БД не существует, физическая модель зависит от конкретной реализации СУБД. Следовательно, одной и той же логической модели могут соответствовать несколько разных физических моделей.

Логические модели. На логическом уровне проектирования строится так называемая визуальная модель объекта [24].

Визуальные модели обеспечивают ясность представления выбранных архитектурных решений и позволяют понять разрабатываемую систему во всей ее полноте.

Визуальное моделирование не способно раз и навсегда решить все проблемы, однако его использование существенно облегчает достижения таких целей, как:

- повышение качества программного продукта,
- сокращение стоимости проекта,
- поставка системы в запланированные сроки [23].

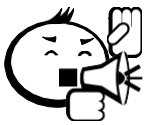
Существует множество подходов к построению таких моделей: продукционные, фреймы, графовые модели, семантические сети, модель «сущность – связь» (ERD), UML и т. д.

Физические модели. Логическая модель данных должна быть отображена в компьютеро-ориентированную даталогическую модель, «понятную» СУБД [24]. В процессе развития теории и практического использования баз данных, а также средств вычислительной техники создавались СУБД, поддерживающие различные даталогические модели.

5.2 Последовательность создания информационной модели данных

Процесс создания информационной модели данных начинается с определения концептуальных требований ряда пользователей. Концептуальные требования могут определяться и для некоторых задач (приложений), которые в ближайшее время реализовывать не планируется. Это может несколько повысить трудоемкость работы, однако поможет наиболее полно учесть все нюансы функциональности, требуемой для разрабатываемой системы, и снизит вероятность ее переделки в дальнейшем. Требования отдельных пользователей интегрируются в едином «обобщенном представлении». Последнее называют концептуальной моделью.

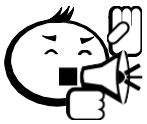
Концептуальная модель *представляет объекты и их взаимосвязи без указания способов их физического хранения.*



Концептуальная модель является, по существу, моделью предметной области.

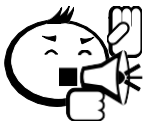
При проектировании концептуальной модели все усилия разработчика должны быть направлены в основном на структуризацию данных и выявление взаимосвязей между ними без рассмотрения особенностей реализации и вопросов эффективности обработки. Проектирование концептуальной модели основано на анализе решаемых на этом предприятии задач по обработке данных. Концептуальная модель включает описания объектов и их взаимосвязей, представляющих интерес в рассматриваемой предметной области и выявляемых в результате анализа данных. Здесь имеются в виду данные, используемые как в

уже разработанных прикладных программах, так и в тех, которые только будут реализованы [26].



Концептуальная модель транслируется затем в модель данных, совместимую с выбранной СУБД.

Возможно, что отраженные в концептуальной модели взаимосвязи между объектами окажутся впоследствии не реализуемыми средствами выбранной СУБД. Это потребует изменения концептуальной модели. Версия концептуальной модели, которая может быть обеспечена конкретной СУБД, называется логической моделью.

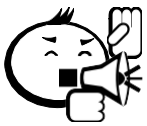


Логическая модель отражает логические связи между элементами данных вне зависимости от их содержания и среды хранения.

Логическая модель данных может быть реляционной, иерархической или сетевой.

Пользователям выделяются подмножества этой логической модели, называемые внешними моделями, отражающие их представления о предметной области.

***Внешняя модель** соответствует представлениям, которые пользователи получают на основе логической модели, в то время как концептуальные требования отражают представления, которые пользователи первоначально желали иметь и которые легли в основу разработки концептуальной модели.*



Логическая модель отображается в физическую память.

***Физическая модель**, определяющая размещение данных, ме- тоды доступа и технику индексирования, называется **внутренней моделью системы**.*

Внешние модели никак не связаны с типом физической памяти, в которой будут храниться данные, и с методами доступа к этим данным. С другой стороны, если концептуальная модель способна учитывать расширение требований к системе в будущем, то вносимые в нее изменения не должны оказывать влияния на существующие внешние модели. Основное различие между указанными выше тремя типами моделей данных (концептуальной, логической и физической) состоит в способах представления взаимосвязей между объектами. При проектировании БД нам потребуется различать взаимосвязи между объектами, между атрибутами одного объекта и между атрибутами различных объектов.

Моделирование данных проводится как поуровневый спуск от концептуальной модели к логической, а затем к физической модели.

5.3 Методология IDEF0

Создание современных информационных систем представляет собой сложнейшую задачу, решение которой требует применения специальных методик и инструментов. В последнее время среди системных аналитиков и разработчиков значительно вырос интерес к CASE (Computer-Aided Software/System Engineering) – технологиям и инструментальным CASE-средствам, позволяющим максимально систематизировать и автоматизировать все этапы разработки программного обеспечения [25].

Технология создания информационных систем предъявляет особые требования к методикам реализации и программным инструментальным средствам.

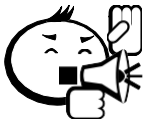
Реализацию проектов по созданию ИС принято разбивать на стадии анализа (прежде чем создавать ИС, необходимо понять и описать бизнес-логику предметной области), проектирования (необходимо определить модули и архитектуру будущей системы), непосредственного кодирования, тестирования и сопровождения.

Для создания ИС жизненно необходим инструмент, значительно (в несколько раз) уменьшающий время разработки ИС.

На современном рынке средств разработки ИС достаточно много CASE-средств, например ERwin и BPwin, которые были разработаны фирмой Logic Works. После слияния в 1998 г. Logic Works с PLATINUM technology они выпускаются под логотипом PLATINUM technology.

На начальных этапах создания ИС необходимо понять, как работает организация, которую собираются автоматизировать. Никто в организации не знает,

как она работает в той мере подробности, которая необходима для создания ИС. Руководитель хорошо знает работу в целом, но не в состоянии вникнуть в детали работы каждого рядового сотрудника. Рядовой сотрудник хорошо знает, что творится на его рабочем месте, но плохо знает, как работают коллеги. Поэтому для описания работы предприятия необходимо построить модель [25].

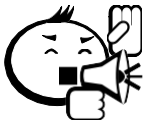


Такая модель должна быть адекватна предметной области, следовательно, она должна содержать в себе знания всех участников бизнес-процессов организации.

. . . . . .

Наиболее удобным языком моделирования бизнес-процессов является IDEF0, предложенный более 20 лет назад Дугласом Россом (SoftTech, Inc.) и называвшийся первоначально **SADT** – Structured Analysis and Design Technique. В дальнейшем это подмножество SADT было принято в качестве федерального стандарта США под наименованием IDEF0. Подробные спецификации на стандарты IDEF можно найти на сайте http://www.idef.com.

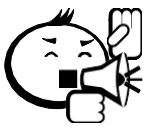
. . . . . .



В IDEF0 система представляется как совокупность взаимодействующих работ или функций. Такая чисто функциональная ориентация является принципиальной – функции системы анализируются независимо от объектов, которыми они оперируют. Это позволяет более четко смоделировать логику и взаимодействие процессов организации.

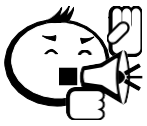
Под моделью в IDEF0 понимают описание системы (текстовое и графическое), которое должно дать ответ на некоторые заранее определенные вопросы.

Моделируемая система имеет границу. Взаимодействие системы с окружающим миром описывается как вход (нечто, что перерабатывается системой), выход (результат деятельности системы), управление (стратегии и процедуры, под управлением которых производится работа) и механизм (ресурсы, необходимые для проведения работы). Находясь под управлением, система преобразует входы в выходы, используя механизмы.



Процесс моделирования какой-либо системы в IDEF0 начина- ется с определения контекста, т. е. наиболее абстрактного уровня описания системы в целом. В контекст входят определение субъекта моделирования, цели и точки зрения на модель.

Под субъектом понимается сама система, при этом необходимо точно установить, что входит в систему, а что лежит за ее пределами, другими словами, мы должны определить, что мы будем в дальнейшем рассматривать как компоненты системы, а что как внешнее воздействие. На определение субъекта системы будет существенно влиять позиция, с которой рассматривается система, и цель моделирования – вопросы, на которые построенная модель должна дать ответ.



Первоначально необходимо определить область (Scope) моделирования.

Описание области – как системы в целом, так и ее компонентов – является основой построения модели. Хотя предполагается, что в течение моделирования область может корректироваться, она должна быть в основном сформулирована изначально, поскольку именно область определяет направление моделирования и когда должна быть закончена модель. При формулировании области необходимо учитывать два компонента – широту и глубину [23].

Широта подразумевает определение границ модели, что будет рассматриваться внутри системы, а что снаружи.

Глубина определяет, на каком уровне детализации модель является завершенной. При определении глубины системы необходимо не забывать об ограничениях времени – трудоемкость построения модели растет в геометрической прогрессии от глубины декомпозиции. После определения границ модели предполагается, что новые объекты не должны вноситься в моделируемую систему; поскольку все объекты модели взаимосвязаны, внесение нового объекта может быть не просто арифметической добавкой, но в состоянии изменить существующие взаимосвязи.

IDEF0-модель предполагает наличие четко сформулированной цели, единственного субъекта моделирования и одной точки зрения.

Диаграммы IDEF0. Основу методологии IDEF0 составляет графический язык описания бизнес-процессов. Модель в нотации IDEF0 представляет собой совокупность иерархически упорядоченных и взаимосвязанных диаграмм. Каждая

диаграмма является единицей описания системы и располагается на отдельном листе.

Модель может содержать четыре типа диаграмм:

1) контекстную диаграмму (в каждой модели может быть только одна контекстная диаграмма);
2) диаграммы декомпозиции;
3) диаграммы дерева узлов;
4) диаграммы только для экспозиции (FEO) [25].

1. **Контекстная диаграмма** является вершиной древовидной структуры диаграмм и представляет собой самое общее описание системы и ее взаимодействия с внешней средой. После описания системы в целом проводится разбиение ее на крупные фрагменты.

2. Этот процесс называется функциональной декомпозицией, а диаграммы, которые описывают каждый фрагмент и взаимодействие фрагментов, называются **диаграммами декомпозиции**. После декомпозиции контекстной диаграммы проводится декомпозиция каждого большого фрагмента системы на более мелкие и так далее, до достижения нужного уровня подробности описания. После каждого сеанса декомпозиции проводятся сеансы экспертизы – эксперты предметной области указывают на соответствие реальных бизнес-процессов созданным диаграммам. Найденные несоответствия исправляются, и только после прохождения экспертизы без замечаний можно приступать к следующему сеансу декомпозиции. Так достигается соответствие модели реальным бизнес-процессам на любом и каждом уровне модели. Синтаксис описания системы в целом и каждого ее фрагмента одинаков во всей модели.

3. **Диаграмма дерева узлов** показывает иерархическую зависимость работ, но не взаимосвязи между работами. Диаграмм деревьев узлов может быть в модели сколь угодно много, поскольку дерево может быть построено на произвольную глубину и необязательно с корня.

4. **Диаграммы для экспозиции (FEO)** строятся для иллюстрации отдельных фрагментов модели, для иллюстрации альтернативной точки зрения либо для специальных целей.

Работы обозначают поименованные процессы, функции или задачи, которые происходят в течение определенного времени и имеют распознаваемые результаты. Работы изображаются в виде прямоугольников. Все работы должны быть названы и определены. Имя работы должно быть выражено отглагольным существительным, обозначающим действие (например, «Изготовление детали», «Прием заказа» и т. д.). Работа «Изготовление детали» может иметь, например, следу-

ющее определение: «Работа относится к полному циклу изготовления изделия от контроля качества сырья до отгрузки готового упакованного изделия». При создании новой модели (меню File/New) автоматически создается контекстная диаграмма с единственной работой, изображающей систему в целом (рис. 5.2).

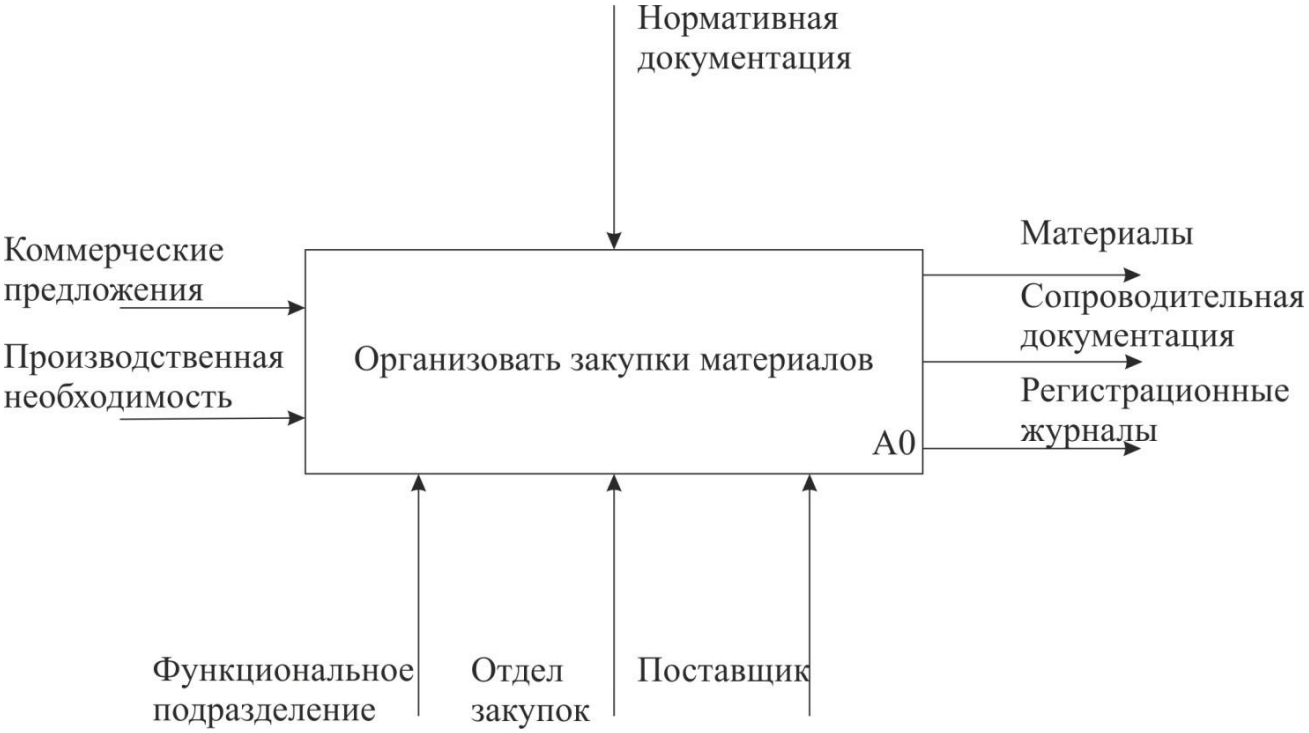


Рис. 5.2 – Пример контекстной диаграммы

Диаграммы декомпозиции содержат родственные работы, т. е. дочерние работы, имеющие общую родительскую работу. Декомпозировать работу на одну работу не имеет смысла: диаграммы с количеством работ более восьми получаются перенасыщенными и плохо читаются. Для обеспечения наглядности и лучшего понимания моделируемых процессов рекомендуется использовать от трех до шести блоков на одной диаграмме.

Если оказывается, что количество работ недостаточно, то работу можно добавить в диаграмму, щелкнув сначала по кнопке на палитре инструментов, а затем по свободному месту на диаграмме.

Работы на диаграммах декомпозиции обычно располагаются по диагонали от левого верхнего угла к правому нижнему.

Такой порядок называется порядком доминирования. Согласно этому принципу расположения в левом верхнем углу располагается самая важная работа или работа, выполняемая по времени первой. Далее вправо вниз располагаются менее важные или выполняемые позже работы. Такое расположение облегчает чтение диаграмм, кроме того, на нем основывается понятие взаимосвязей работ (рис. 5.3).

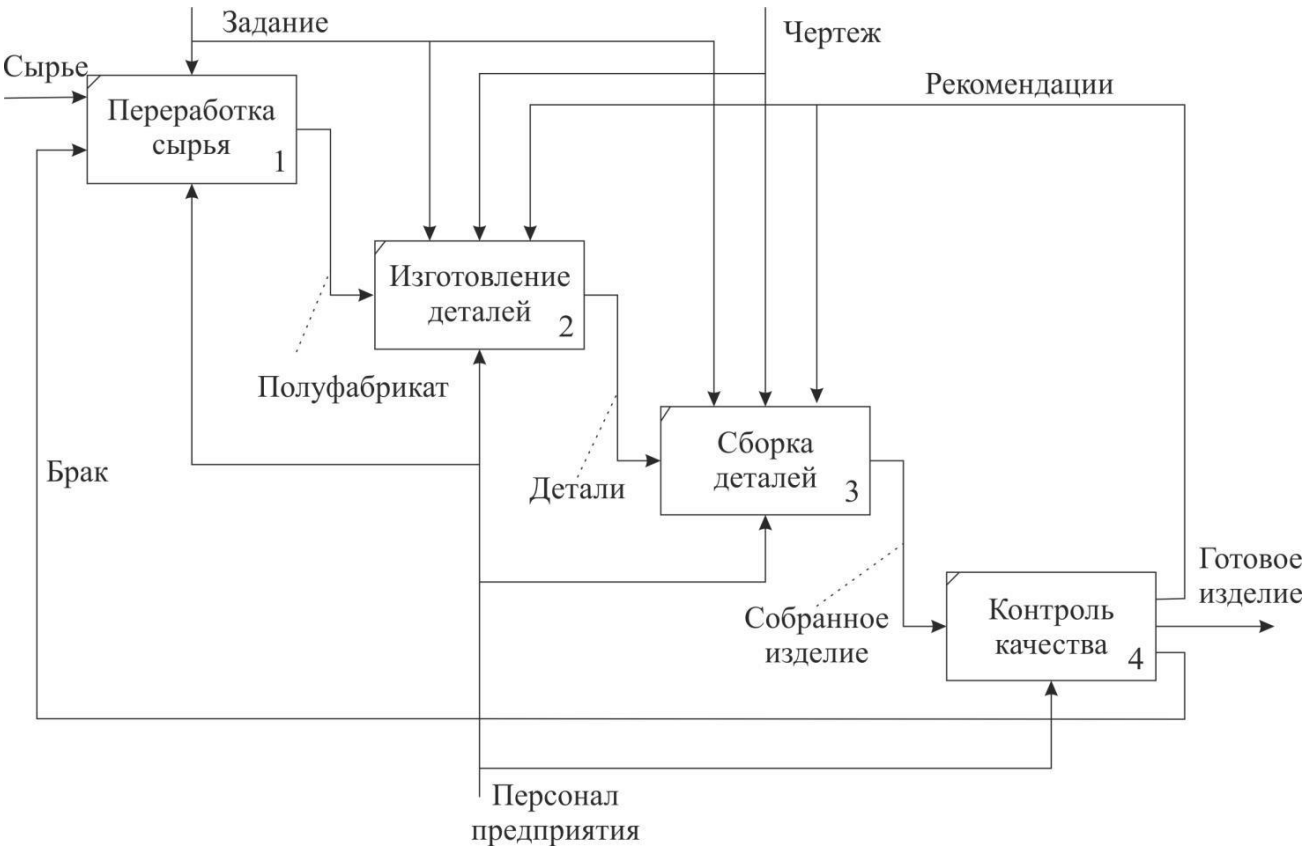


Рис. 5.3 – Пример диаграммы декомпозиции «Изготовление изделия»

Каждая из работ на диаграмме декомпозиции может быть в свою очередь декомпозирована. На диаграмме декомпозиции работы нумеруются автоматически слева направо. Номер работы показывается в правом нижнем углу. В левом верхнем углу изображается небольшая диагональная черта, которая показывает, что данная работа не была декомпозирована.

Взаимодействие работ с внешним миром и между собой описывается в виде стрелок [23, 25].

В IDEF0 различают пять типов стрелок (рис. 5.4).

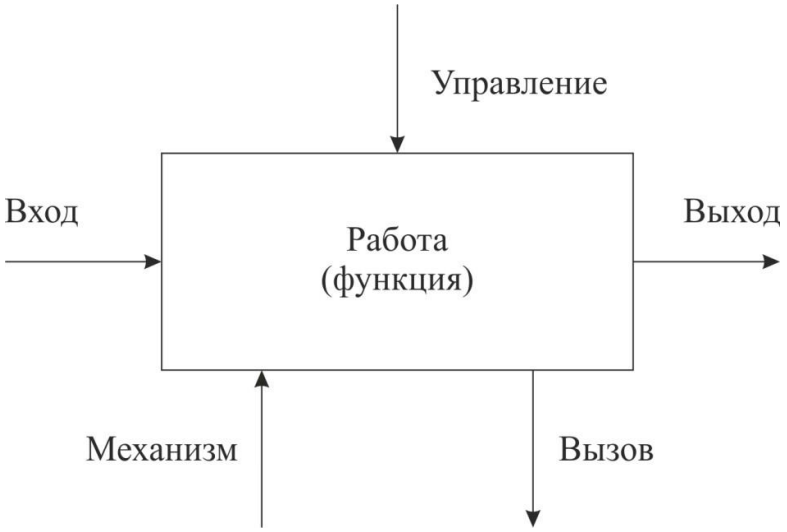


Рис. 5.4 – Основные элементы графической нотации IDEF0

Вход (Input) – материал или информация, которые используются или преобразуются работой для получения результата (выхода). Допускается, что работа может не иметь ни одной стрелки входа. Каждый тип стрелок подходит к определенной стороне прямоугольника, изображающего работу, или выходит из нее. Стрелка входа рисуется как входящая в левую грань работы. При описании технологических процессов (для этого и был придуман IDEF0) не возникает проблем определения входов. Действительно, «Сырье» на рисунке 5.3 – это нечто, что перерабатывается в процессе «Переработка сырья» для получения результата. При моделировании ИС, когда стрелками являются не физические объекты, а данные, не все так очевидно. Например, при «Приеме пациента» карта пациента может быть и на входе и на выходе, между тем качество этих данных меняется. Другими словами, в нашем примере для того, чтобы оправдать свое назначение, стрелки входа и выхода должны быть точно определены с тем, чтобы указать на то, что данные действительно были переработаны (например, на выходе – «Готовое изделие»). Очень часто сложно определить, являются ли данные входом или управлением. В этом случае подсказкой может служить то, перерабатываются/изменяются ли данные в работе или нет. Если изменяются, то скорее всего это вход, если нет – управление.

Управление (Control) – правила, стратегии, процедуры или стандарты, которыми руководствуется работа. Каждая работа должна иметь хотя бы одну стрелку управления. Стрелка управления рисуется как входящая в верхнюю грань работы. На рисунке 5.3 стрелки «Задание» и «Чертеж» – управление для работы «Изготовление деталей». Управление влияет на работу, но не преобразуется работой. Если цель работы – изменить процедуру или стратегию, то такая процедура или стратегия будет для работы входом. В случае возникновения неопределенности в статусе стрелки (управление или вход) рекомендуется рисовать стрелку управления.

Выход (Output) – материал или информация, которые производятся работой. Каждая работа должна иметь хотя бы одну стрелку выхода. Работа без результата не имеет смысла и не должна моделироваться. Стрелка выхода рисуется как исходящая из правой грани работы. На рисунке 5.3 стрелка «Готовое изделие» является выходом для работы «Изготовление изделия».

Механизм (Mechanism) – ресурсы, которые выполняют работу, например персонал предприятия, станки, устройства и т. д. Стрелка механизма рисуется как входящая в нижнюю грань работы. На рисунке 5.3 стрелка «Персонал предприятия» является механизмом для работы «Изготовление изделия». По усмотрению аналитика стрелки механизма могут не изображаться в модели.

Вызов (Call) – специальная стрелка, указывающая на другую модель работы. Стрелка вызова рисуется как исходящая из нижней грани работы. Стрелка вызова используется для указания того, что некоторая работа выполняется за пределами моделируемой системы. В BPwin стрелки вызова используются в механизме слияния и разделения моделей.

Граничные стрелки. Стрелки на контекстной диаграмме служат для описания взаимодействия системы с окружающим миром. Они могут начинаться у границы диаграммы и заканчиваться у работы, или наоборот. Такие стрелки называются граничными. Имена вновь внесенных стрелок автоматически заносятся в словарь (Arrow Dictionary).

Диаграммы IDEF0 имеют двойную нумерацию. Во-первых, диаграммы имеют номера по узлу. Контекстная диаграмма всегда имеет номер A-0, декомпозиция контекстной диаграммы – номер A0, остальные диаграммы декомпозиции – номера по соответствующему узлу (например, A1, A2, A3 и т. д., см. рис. 5.5).

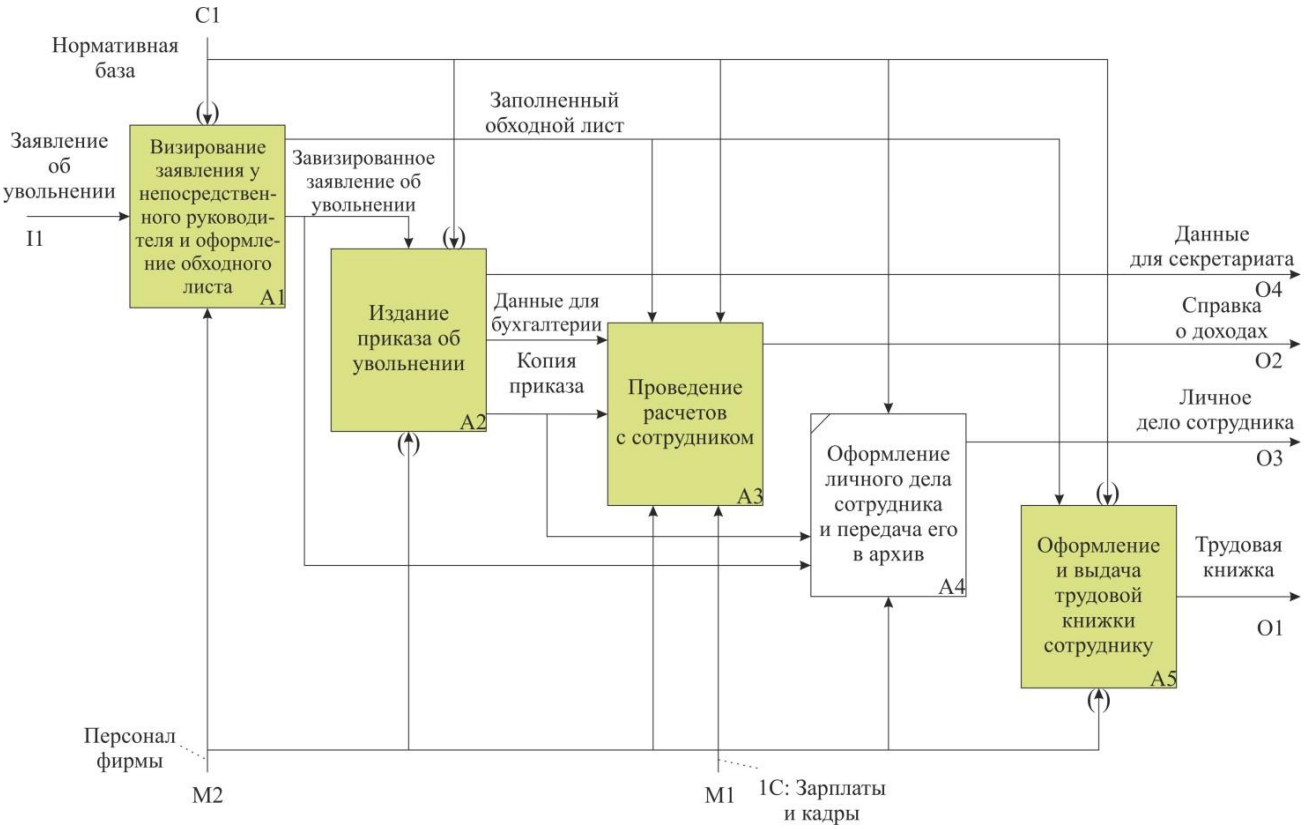


Рис. 5.5 – Пример диаграммы, разработанной с помощью стандарта IDEF0

BPwin автоматически поддерживает нумерацию по узлам, т. е. при проведении декомпозиции создается новая диаграмма и ей автоматически присваивается соответствующий номер. В результате проведения экспертизы диаграммы могут уточняться и изменяться, следовательно, могут быть созданы различные версии

одной и той же (с точки зрения ее расположения в дереве узлов) диаграммы декомпозиции.

5.4 Физическая и логическая модели данных

ERwin имеет два уровня представления модели – логический и физический [26].

Логический уровень – это абстрактный взгляд на данные, на нем данные представляются так, как выглядят в реальном мире, и могут называться так, как они называются в реальном мире, например «Постоянный клиент», «Отдел» или «Фамилия сотрудника». Объекты модели, представляемые на логическом уровне, называются сущностями и атрибутами (подробнее о сущностях и атрибутах будет рассказано ниже). Логическая модель данных может быть построена на основе другой логической модели, например на основе модели процессов. Логическая модель данных является универсальной и никак не связана с конкретной реализацией СУБД.

Физическая модель данных, напротив, зависит от конкретной СУБД, фактически являясь отображением системного каталога. В физической модели содержится информация о всех объектах БД. Поскольку стандартов на объекты БД не существует (например, нет стандарта на типы данных), физическая модель зависит от конкретной реализации СУБД. Следовательно, одной и той же логической модели могут соответствовать несколько разных физических моделей. Если в логической модели не имеет значения, какой конкретно тип данных имеет атрибут, то в физической модели важно описать всю информацию о конкретных физических объектах – таблицах, колонках, индексах, процедурах и т. д. Разделение модели данных на логические и физические позволяет решить несколько важных задач.

Документирование модели. Многие СУБД имеют ограничение на именование объектов (например, ограничение на длину имени таблицы или запрет использования специальных символов – пробела и т. п.). Зачастую разработчики ИС имеют дело с нелокализованными версиями СУБД. Это означает, что объекты БД могут называться короткими словами, только латинскими символами и без использования специальных символов (т. е. нельзя назвать таблицу предложением, только одним словом). Кроме того, проектировщики БД нередко злоупотребляют «техническими» наименованиями, в результате таблица и колонки получают наименования типа *RTD_324* или *CUST_A12* и т. д. Полученную в результате структуру могут понять только специалисты (а чаще всего только авторы модели), ее невозможно обсуждать с экспертами предметной области. Разделение модели на логическую и физическую позволяет решить эту проблему. На физическом

уровне объекты БД могут называться так, как того требуют ограничения СУБД. На логическом уровне можно этим объектам подобрать синонимы – имена более понятные неспециалистам, в том числе на кириллице и с использованием специальных символов. Например, таблице *CUST_A12* может соответствовать сущность *Постоянный клиент*. Такое соответствие позволяет лучше задокументировать модель и дает возможность обсуждать структуру данных с экспертами предметной области.

Масштабирование. Создание модели данных, как правило, начинается с создания логической модели. После описания логической модели проектировщик может выбрать необходимую СУБД и ERwin автоматически создаст соответствующую физическую модель. На основе физической модели ERwin может сгенерировать системный каталог СУБД или соответствующий SQL-скрипт. Этот процесс называется прямым проектированием (Forward Engineering). Тем самым достигается масштабируемость – создав одну логическую модель данных, можно сгенерировать физические модели под любую поддерживаемую ERwin СУБД. С другой стороны, ERwin способен по содержимому системного каталога или SQL-скрипту воссоздать физическую и логическую модель данных (Reverse Engineering). На основе полученной логической модели данных можно сгенерировать физическую модель для другой СУБД и затем сгенерировать ее системный каталог. Следовательно, ERwin позволяет решить задачу по переносу структуры данных с одного сервера на другой. Например, можно перенести структуру данных с Oracle на Informix (или наоборот) или перенести структуру dbf-файлов в реляционную СУБД, тем самым облегчив решение по переходу от файл-серверной к клиент-серверной ИС. Заметим, однако, что формальный перенос структуры «плоских» таблиц на реляционную СУБД обычно неэффективен. Для того чтобы извлечь выгоды от перехода на клиент-серверную технологию, структуру данных следует модифицировать. Процессы прямого и обратного проектирования будут рассмотрены ниже.

Для создания моделей данных в ERwin можно использовать две нотации: IDEF1X и IE (Information Engineering). Методология IDEF1X была разработана для армии США и широко используется в государственных учреждениях США, финансовых и промышленных корпорациях. Методология IE, разработанная Мартином (Martin), Финкельштейном (Finkelstein) и другими авторами, используется преимущественно в промышленности.

ERwin имеет несколько уровней отображения диаграммы: уровень сущностей, уровень атрибутов, уровень определений, уровень первичных ключей и уро-

вень иконок. Переключиться между первыми тремя уровнями можно с использованием кнопок панели инструментов.

5.5 Подходы к концептуальному моделированию

Задача проектировщика БД на этапе концептуального моделирования состоит в том, чтобы на основании локальных представлений пользователей найти обобщенное представление информации, свойственное природе предметной области (ПО) как целого [27].

Для этого необходимо выполнить анализ ПО, то есть:

- уяснить цели предприятия, для которого создается БД;
- выявить функции или действия, направленные на достижение целей, и правила бизнеса, принятые на предприятии;
- выявить данные, необходимые для выполнения функций;
- выделить объекты, вовлеченные в деятельность, и выявить связи между ними [28].

Результатом анализа является концептуальная (информационная) модель предметной области (ПО). С точки зрения проектировщика она содержит спецификации логического макета БД – определения базовых отношений и правил целостности данных. Существуют два основных подхода к проектированию концептуальной модели – формальный и семантический.

Формальный подход включает два этапа моделирования:

1) анализ ПО с целью выявления полного перечня хранимых атрибутов и правил бизнеса;
2) нормализация универсального отношения до требуемого уровня [28].

Результатами анализа являются определения схемы универсального отношения и множества межатрибутных связей, существующих в нем. Результат нормализации – система отношений, связанных по типу «родитель – потомок» или «супертип – категория». Выполняя нормализацию, обычно стремятся строить такие проекции универсального отношения, которые можно интерпретировать как объекты ПО или факты связи объектов.

Основной недостаток формального подхода состоит в том, что он требует проведения детального анализа ПО ***до*** начала проектирования логического макета БД. Поэтому методики, основанные на формальном подходе, мало пригодны для решения сложных задач.

Семантический подход, в отличие от формального, предполагает *параллельное* выполнение анализа ПО и проектирование логического макета БД.

В основе подхода лежат понятия ER-модели данных. Процесс проектирования включает три этапа [28].

На *первом этапе* проектировщик формирует общие представления о компонентах бизнеса и их отношениях и идентифицирует основные сущности и связи. При этом он не стремится получить детальную информацию о свойствах объектов ПО и их взаимосвязях.

На *втором этапе* представления проектировщика детализируются до уровня идентификаторов экземпляров сущностей и типов связей. Здесь окончательно определяется состав сущностей модели и специфицируются ограничения целостности сущности и ссылочной целостности – первичные и альтернативные ключи, внешние ключи, типы сущностей и связей. Результат этапа – логический макет БД с точностью до ключей.

На *третьем этапе* формируются окончательные представления о составе атрибутов сущностей и полностью определяются схемы всех отношений, соответствующих сущностям. Как правило, все отношения схемы находятся в 3 нормальной форме (ЗНФ).

Выделяя сущности и определяя связи между ними, проектировщик опирается на свои текущие представления о ПО и здравый смысл. На каждом этапе он может согласовать свои представления с представлениями конечных пользователей. Поэтому грубые ошибки моделирования при разумном использовании семантического подхода – редкость. Этот подход к проектированию представляется вполне естественным и понятным. Кроме того, он очень эффективен, если проектировщик придерживается определенной дисциплины проектирования и использует подходящие средства для документирования работ.

Методологии семантического подхода. В настоящее время существуют несколько методологий анализа и проектирования логического макета БД, реализующих семантический подход [27].

Основным компонентом любой из них является графический язык определения данных – система графических нотаций для основных понятий *ER-модели*. Этими средствами проектировщик может точно, ясно и наглядно сформулировать свои текущие представления о ПО в виде диаграмм.

Второй компонент – *глоссарий, содержащий однозначные определения имен сущностей и атрибутов*. Он раскрывает те аспекты модели, которые невозможно отразить графическими средствами. Методологии различаются, в основном, нотациями графических языков определения данных и обладают рядом общих достоинств.

Во-первых, все они поддерживают *параллельное развитие анализа ПО и проектирования структуры БД и правил целостности данных*. Нет никакой необходимости выявлять весь перечень хранимых данных и правил бизнеса еще до начала проектирования. Для выделения базовых отношений не нужно использовать громоздкие и сложные процедуры нормализации универсального отношения.

Во-вторых, простота нотаций графических языков, строго определенные правила именования сущностей, атрибутов и связей, а также содержащиеся в глоссарии определения имен обеспечивают точную передачу представлений автора модели и однозначную (и одинаковую!) понимаемость модели всеми участниками разработки – аналитиками, экспертами и проектировщиками структур хранения данных.

В-третьих, строго определенная иерархия уровней модели открывает возможность согласования представлений аналитика с представлениями конечных пользователей системы на всех этапах разработки.

В-четвертых, в настоящее время существует несколько десятков товарных CASE-систем (*Computer Aided System Engineering*), поддерживающих семантические методологии на всех этапах жизненного цикла системы от формулировки замысла до выпуска проектной документации. Многие из них обеспечивают отображение окончательных диаграмм модели на физические структуры данных распространенных СУБД, а также создание триггеров ссылочной целостности, как стандартных, так и оригинальных.

В силу указанных причин использование семантического подхода значительно снижает трудозатраты на ранних этапах проектирования системы, упрощает и облегчает верификацию моделей и обеспечивает создание высококачественных спецификаций систем баз данных.

5.6 Уровни представления диаграмм

Методология информационного моделирования IDEF1X (Integrated DEFinitions 1 eXpanded) различает три уровня графического представления информации, три уровня диаграмм или три уровня логической модели, отличающихся по глубине представления информации о данных:

- уровень «сущность – связь» (Entity Relationship Level, ER);
- уровень ключей (Key-Based Level, KB);
- уровень атрибутов (Fully Attributed Level, FA) [28–29].

1. **Уровень «сущность – связь» (ER level)**. Это уровень наименее детального представления информации. Он используется на начальной стадии моделирования, когда еще не выяснены или не поняты до конца свойства сущностей и связей. На диаграммах ER-уровня сущности и связи представлены только их именами.

Диаграмма «сущность – связь» представляет собой модель данных верхнего уровня. Она включает сущности и взаимосвязи, отражающие основные бизнес-правила предметной области. Такая диаграмма не слишком детализирована, в нее включаются основные сущности и связи между ними, которые удовлетворяют основным требованиям, предъявляемым к ИС. Диаграмма «сущность – связь» может включать связи многие-ко-многим и не включать описание ключей. Как правило, ERD используется для презентаций и обсуждения структуры данных с экспертами предметной области.

На ER-уровне сущности не различаются как зависимые или независимые, а соединения – как идентифицирующие и неидентифицирующие. Сущности не содержат горизонтальных линий, отделяющих область ключей от области данных. Имена сущностей вписываются в обозначающие их прямоугольники.

Диаграмма ER-уровня может показывать категории, но указывать дискриминаторы кластеров необязательно.

На ER-уровне допустимы неспецифические соединения. Для изображения соединений можно использовать как сплошные, так и штриховые линии. Это не специфицирует соединения.

Связь является логическим соотношением между сущностями. Каждая связь должна именоваться глаголом или глагольной фразой (Relationship Verb Phrases).

…… ……

*Под **сущностью в IDEF1X** понимается **отношение**. Сущности изображаются на диаграммах именованными прямоугольниками, в которые вписываются имена атрибутов.*

***Сущность** – это объект, событие или концепция, информация о которых должна сохраняться.*

…… ……

На ER-уровне независимые и зависимые сущности не различаются, атрибуты не указываются, а имена сущностей вписываются в обозначающие их прямоугольники.

Имя связи выражает некоторое ограничение или бизнес-правило и облегчает чтение диаграммы, например:

- каждый ТОВАР <*хранится на*> СКЛАДе;
- каждая НАКЛАДНАЯ <*содержит*> ТОВАР;
- каждый ПОКУПАТЕЛЬ < *получает*> НАКЛАДНую (рис. 5.6).

Рис. 5.6 – ER-диаграмма со связями между сущностями

Связь показывает, какие именно товары содержатся в накладной и какой именно покупатель получает конкретную накладную с товаром, который хранится на складе. По умолчанию имя связи на диаграмме не показывается. Для отображения имени следует в контекстном меню, которое появляется, если щелкнуть левой кнопкой мыши по любому месту диаграммы, не занятому объектами модели, выбрать пункт Display Options/Relationship и затем включить опцию Verb Phrase.

На логическом уровне можно установить идентифицирующую связь один-ко-многим, связь многие-ко-многим и неидентифицирующую связь один-ко-многим (соответственно это кнопки слева направо в палитре инструментов).

2. **Уровень ключей (Key-Based Level, KB)**. На этом уровне в диаграммах отражаются имена первичных и внешних ключей сущностей и спецификации связей. Диаграмма KB-уровня объявляет уникальные идентификаторы экземпляров сущностей и ограничения ссылочной целостности.

Модель данных (KB), основанная на ключах, – более подробное представление данных. Она включает описание всех сущностей и первичных ключей и предназначена для представления структуры данных и ключей, которые соответствуют предметной области.

Ключи. Каждый экземпляр сущности должен быть уникален и отличаться от других атрибутов.

Первичный ключ (primary key) *– это атрибут или группа атрибутов, однозначно идентифицирующая экземпляр сущности.*

Атрибуты первичного ключа на диаграмме не требуют специального обозначения – это те атрибуты, которые находятся в списке атрибутов выше горизонтальной линии. При внесении нового атрибута в диалоге Attribute Editor, для того чтобы сделать его атрибутом первичного ключа, нужно включить флажок Primary Key в нижней части закладки General. На диаграмме неключевой атрибут можно внести в состав первичного ключа, воспользовавшись режимом переноса атрибутов (кнопка в палитре инструментов).

ERwin автоматически производит миграцию ключевых атрибутов родительской сущности в дочернюю сущность, где они становятся внешними ключами.

Атрибуты ключа не должны содержать нулевых значений.

Каждая сущность должна иметь по крайней мере один потенциальный ключ. Многие сущности имеют только один потенциальный ключ. Такой ключ становится первичным. Некоторые сущности могут иметь более одного возможного ключа. Тогда один из них становится первичным, а остальные – альтернативными ключами.

Альтернативный ключ (Alternate Key) *– это потенциальный ключ, не ставший первичным. ERwin позволяет выделить атрибуты альтернативных ключей, и по умолчанию в дальнейшем при генерации схемы БД по этим атрибутам будет генерироваться уникальный индекс.*

Ключи могут быть **сложными**, т. е. содержащими несколько атрибутов. Сложные первичные ключи не требуют специального обозначения – это список атрибутов выше горизонтальной линии.

На диаграмме атрибуты альтернативных ключей обозначаются как (AK*n.m*), где *n* – порядковый номер ключа, *m* – порядковый номер атрибута в ключе. Когда альтернативный ключ содержит несколько атрибутов, (AK*n.m*) ставится после каждого. Например, атрибуты ***Фамилия, Имя, Отчество*** и

Дата рождения входят в альтернативный ключ № 1 (AK1), ***Номер паспорта*** составляет альтернативный.

Для того чтобы стать первичным, потенциальный ключ должен удовлетворять следующим требованиям.

Уникальность. Два экземпляра не должны иметь одинаковых значений возможного ключа.

Компактность. Сложный возможный ключ не должен содержать ни одного атрибута, удаление которого не приводило бы к утрате уникальности. Тогда второй ключ, претендующий на роль первичного, не является компактным, так как удаление энергичности не приведет к утрате уникальности.

Диаграммы KB-уровня должны удовлетворять следующим правилам для ключей.

1. На диаграммах KB- и FA-уровней каждая сущность должна иметь первичный ключ.
2. Сущность может иметь несколько альтернативных ключей.
3. Как первичный, так и альтернативный ключ может быть либо одиночным атрибутом, либо группой атрибутов.
4. Отдельный атрибут может быть частью более чем одного ключа, первичного или альтернативного.
5. Атрибуты, входящие в первичный и альтернативный ключи, могут быть как собственными атрибутами сущности, так и присоединенными через связь с другой сущностью.
6. Первичный и альтернативные ключи должны содержать только те атрибуты, которые необходимы для уникальной идентификации экземпляров сущности.
7. Каждый неключевой атрибут должен неприводимо зависеть от первичного ключа, если он составной.
8. Каждый атрибут, не являющийся частью первичного ключа или какого-либо из альтернативных, должен функционально зависеть только от первичного ключа и каждого из альтернативных [28, 30].

Правила для внешних ключей приведены ниже.

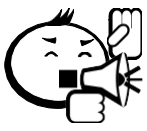
- Каждая сущность, являющаяся потомком в специфической связи или категорией в категоризационной связи, должна содержать множество атрибутов – внешних ключей, переданных связью. Конкретный атрибут может быть элементом нескольких таких множеств. Число атрибутов в каждом множестве внешних ключей должно совпадать с чис-

лом атрибутов первичного ключа родительской или родовой сущности.

- Первичный ключ родовой сущности должен передаваться как первичный ключ каждой категории.
- Потомок не может содержать двух полных внешних ключей, которые соотносят с каждым его экземпляром один и тот же экземпляр одного и того же предка, если эти внешние ключи не переданы через различные пути связей, включающие по крайней мере одну промежуточную сущность между этим предком и потомком.
- Каждый присоединенный атрибут потомка или категории должен быть атрибутом первичного ключа связанной с ним родительской или родовой сущности. Обратно, каждый атрибут первичного ключа родительской или родовой сущности должен быть присоединенным атрибутом связанного с нею потомка или категории.
- Каждое имя роли, назначенное присоединенному атрибуту, должно быть уникальным, и в одно и то же имя всегда должен вкладываться один и тот же смысл. Один и тот же смысл не может вкладываться в разные имена, если они не являются псевдонимами.
- Присоединенный атрибут может быть частью более чем одного множества внешних ключей при условии, что он имеет одно и то же значение в этих множествах в некотором фиксированном экземпляре сущности. Такому присоединенному атрибуту может быть назначено имя роли.
- Каждый внешний ключ должен ссылаться на один и только один атрибут первичного ключа родителя [28, 30].

3. **Уровень атрибутов (Fully Attributed Level, FA)**. Диаграмма FA-уровня показывает имена всех атрибутов сущностей и связей и полностью определяет структуру и взаимосвязи данных.

Полная атрибутивная модель – наиболее детальное представление структуры данных: представляет данные в третьей нормальной форме и включает все сущности, атрибуты и связи [28, 31].



Цель моделирования – создание FA-диаграммы. Она является графическим представлением структуры реляционной базы данных с полностью определенными схемами отношений.

Диаграмма FA-уровня должна содержать все, что содержит диаграмма KB-уровня и, кроме того, все неключевые атрибуты. На KB- и FA-уровнях в полной мере действуют все правила синтаксиса, изложенные выше.

Основные компоненты диаграммы ERwin – это сущности, атрибуты и связи. Каждая сущность является множеством подобных индивидуальных объектов, называемых экземплярами. Каждый экземпляр индивидуален и должен отличаться от всех остальных экземпляров. Атрибут выражает определенное свойство объекта. С точки зрения БД (физическая модель) сущности соответствует таблица, экземпляру сущности – строка в таблице, а атрибуту – колонка таблицы.

Построение модели данных предполагает определение сущностей и атрибутов, т. е. необходимо определить, какая информация будет храниться в конкретной сущности или атрибуте.

Сущности должны иметь наименование с четким смысловым значением, именоваться существительным в единственном числе, не носить «технических» наименований и быть достаточно важными для того, чтобы их моделировать. Именование сущности в единственном числе облегчает в дальнейшем чтение модели. Фактически имя сущности дается по имени ее экземпляра. Примером может быть сущность ***Заказчик*** (но не *Заказчики*!) с атрибутами ***Номер заказчика, Фамилия заказчика*** и ***Адрес заказчика***. На уровне физической модели ей может соответствовать таблица ***Customer*** с колонками ***Customer_number***, ***Customer_name*** и ***Customer_address***.

Для внесения сущности в модель необходимо (убедившись предварительно, что вы находитесь на уровне логической модели – переключателем между логической и физической моделью служит раскрывающийся список в правой части панели инструментов) «кликнуть» по кнопке сущности на панели инструментов (ERwin Toolbox), затем «кликнуть» по тому месту на диаграмме, где необходимо расположить новую сущность. Щелкнув правой кнопкой мыши по сущности и выбрав из всплывающего меню пункт Entity Editor, можно вызвать диалог Entity Editor, в котором определяются имя, описание и комментарии сущности.

Каждая сущность должна быть полностью определена с помощью текстового описания в закладке Definition. Закладки Note, Note 2, Note 3, UDP (User Defined Properties – Свойства, определенные пользователем) служат для внесения дополнительных комментариев и определений к сущности. В прежних версиях ERwin закладкам Note2 и Note3 соответствовали окна Query и Sample.

Закладка Definition используется для ввода определения сущности. Эти определения полезны как на логическом уровне, поскольку позволяют понять, что это за объект, так и на физическом уровне, поскольку их можно экспортировать как часть схемы и использовать в реальной БД (CREATE COMMENT on entity_name).

Закладка Note позволяет добавлять дополнительные замечания о сущности, которые не были отражены в определении, введенном в закладке Definition. Здесь можно ввести полезное замечание, описывающее какое-либо бизнес-правило или соглашение по организации диаграммы.

В закладке Note 2 можно задокументировать некоторые возможные запросы, которые, как ожидается, будут использоваться по отношению к сущности в БД. При переходе к физическому проектированию записанные запросы помогут принимать такие решения в отношении проектирования, которые сделают БД более эффективной.

Закладка Note 3 позволяет вводить примеры данных для сущности (в произвольной форме).

В закладке Icon каждой сущности можно поставить в соответствие изображение, которое будет отображаться в режиме просмотра модели на уровне иконок.

Часто приходится создавать производные атрибуты, т. е. атрибуты, значение которых можно вычислить из других атрибутов. Примером производного атрибута может служить ***Возраст сотрудника***, который может быть вычислен из атрибута ***Дата рождения сотрудника***. Такой атрибут может привести к конфликтам; действительно, если вовремя не обновить значение атрибута ***Возраст сотрудника***, он может противоречить значению атрибута ***Дата рождения сотрудника***. Производные атрибуты – ошибка нормализации, однако их вводят для повышения производительности системы – если необходимо узнать возраст сотрудника, можно обратиться к соответствующему атрибуту, а не проводить вычисления (которые на практике могут быть значительно более сложными, чем в приведенном примере) по дате рождения.

В IDEF1X различают зависимые и независимые сущности. Тип сущности определяется ее связью с другими сущностями. Идентифицирующая связь устанавливается между независимой (родительский конец связи) и зависимой (дочерний конец связи) сущностями. Когда рисуется идентифицирующая связь, ERwin автоматически преобразует дочернюю сущность в зависимую. Зависимая сущность изображается прямоугольником со скругленными углами (сущ-

ность ПОЗИЦИЯ ЗАКАЗА на рисунке 5.7). Экземпляр зависимой сущности определяется только через отношение к родительской сущности, т. е. в структуре на рисунке 5.7 информация о позиции заказа не может быть внесена и не имеет смысла без информации о заказе, который его размещает. При установлении идентифицирующей связи атрибуты первичного ключа родительской сущности автоматически переносятся в состав первичного ключа дочерней сущности. Эта операция дополнения атрибутов дочерней сущности при создании связи называется миграцией атрибутов. В дочерней сущности новые атрибуты помечаются как внешний ключ – (FK).

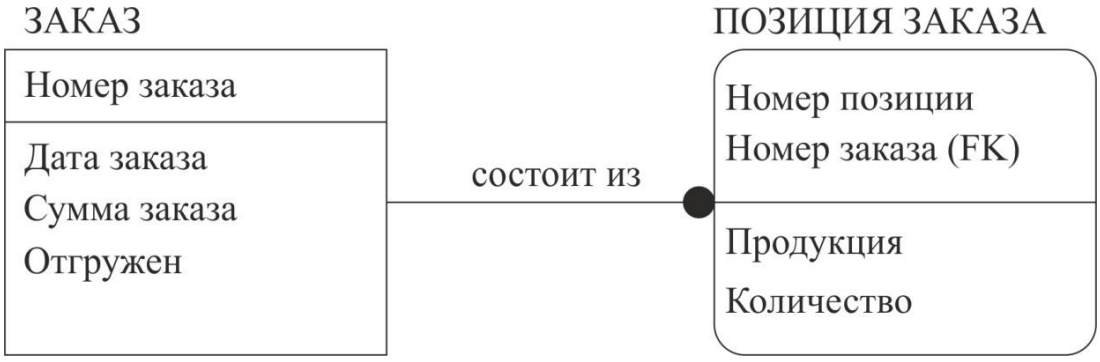


Рис. 5.7 – Идентифицирующая связь между независимой и зависимой таблицей

Синтаксис графического языка IDEF1X обеспечивает однозначное представление ограничений ссылочной целостности в диаграммах KB- и FA-уровня. Правила именования и определения сущностей, доменов и атрибутов дают возможность задать ограничения на значения в текстовых документах, сопровождающих диаграмму. В силу этого трансляция FA-диаграммы в тексты описания таблиц БД на языке конкретной СУБД оказывается чисто формальной процедурой и может выполняться автоматически.

5.7 Основные правила стандарта IDEF1X

Основные правила стандарта IDEF1X приведены ниже.

1. Сущности, атрибуты и домены обязательно именуются. Именем может быть только имя существительное, возможно с определениями. В качестве имен допускаются аббревиатуры и акронимы.

2. Имя должно быть уникальным и осмысленным. Формальное определение имени обязательно включается в глоссарий модели.

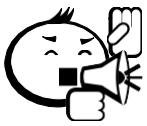
Создавая модель, проектировщик стремится сформировать ясное представление о ПО, в частности о том, какие сведения будут храниться в БД. Этого

можно добиться, только сформулировав точные и однозначные определения смысла каждого имени, введенного в модель [28].

Например, что такое ДЕТАЛЬ? Это неделимая часть изделия, или она сама может собираться из других деталей? Может ли ИЗДЕЛИЕ быть ДЕТАЛЬю? Наша фирма только закупает ДЕТАЛи или может их производить?

Или что такое ФИЛЬМ? Это лента, лежащая в нашей фильмотеке, или это произведение киноискусства? Нас интересуют любые фильмы или только фильмы определенного жанра?

От ответов на эти вопросы зависит организация данных и их взаимосвязи. Поэтому прежде чем строить диаграммы, необходимо получить точные ответы на все подобные вопросы и зафиксировать документально определения смысла имен. Без этого модель будет неоднозначной и противоречивой.



Имя сущности, атрибута или домена должно иметь единственный смысл, и этот смысл всегда должен выражаться этим именем. Тот же смысл не может вкладываться в другое имя, если оно не является псевдонимом или синонимом основного.

Например, атрибут *дата* не может иметь смысл даты *начала ИЛИ окон чания* отчетного периода. Совершенно непонятно, как интерпретировать значения этого атрибута в различных кортежах отношения.

Сущности и атрибуты всегда именуются в единственном числе.

Имя должно относиться к одному экземпляру сущности или значению атрибута.

Соблюдение этих правил обеспечивает интерпретацию диаграмм фразами естественного языка и точную передачу смысла, вложенного в имена автором модели.

3. Атрибут есть свойство или характеристика, общая для некоторых или всех экземпляров сущности. Атрибут является конкретизацией домена в контексте сущности.

4. На диаграммах KB- и FA-уровней каждая сущность имеет не менее одного атрибута. Каждый атрибут может быть собственным атрибутом сущности или присоединенным (мигрировавшим), полученным от другой сущности через связь.

5. Каждый атрибут является собственным атрибутом точно одной сущности.

6. Присоединенный атрибут должен быть частью первичного ключа передавшей его сущности (родительской или родовой). Присоединенный атрибут помечается символом «(FK)», следующим за именем атрибута.

В совокупности эти требования означают, что никакие две сущности не могут иметь одноименных атрибутов, если они не связаны каким-либо отношением. В последнем случае одноименными могут быть атрибуты, которые являются частью первичного ключа родителя и частью внешнего ключа потомка.

7. Каждый экземпляр сущности должен иметь определенное значение каждого атрибута, являющегося частью первичного ключа.

8. Не может быть экземпляра сущности, имеющего более чем одно значение какого-либо из атрибутов.

9. Атрибуты, не являющиеся частью первичного ключа, могут иметь неопределенные значения. Такие атрибуты помечаются символом «(O)», следующим за именем атрибута (Optional – необязательный).

10. Если атрибут является собственным атрибутом одной сущности и присоединенным атрибутом другой, то либо он имеет одинаковые имена в обеих сущностях, либо помечается именем роли как присоединенный.

11. Определение атрибута должно содержать ссылку на имя домена.

12. На KB-диаграммах показываются только атрибуты, входящие в состав первичных, альтернативных и внешних ключей. Атрибуты альтернативных ключей обязательно помечаются символом «(AK*n*)», где *n* – номер альтернативного ключа. Все атрибуты, входящие в состав одного и того же альтернативного ключа, помечаются одним и тем же значением *n*. На FA-диаграмме показываются все атрибуты каждой сущности.

13. Соединение – это один из двух видов связей, используемых в языке IDEF1X. Стандарт определяет соединение как ассоциацию между двумя сущностями или между экземплярами одной и той же сущности.

14. В диаграммах IDEF1X представляются только бинарные и унарные связи (соединения), а также иерархии (связи категоризации). Отсутствие обозначений для представления связей высшей арности не является ограничением модели. Неспецифические соединения могут быть показаны только на диаграммах ER-уровня.

15. Соединению присваивается имя, выражаемое глагольным оборотом. Имя зрительно привязывается к дуге, изображающей соединение. Имена соединений могут быть неуникальными в пределах диаграммы.

16. Имя каждого соединения одной и той же пары сущностей должно быть уникальным во множестве имен связей этих сущностей.

17. Имена специфических соединений выбираются так, чтобы можно было построить осмысленную фразу, составленную из имени родительской сущности, имени связи, выражения кардинальности и имени потомка.

18. Связь может быть поименована «от родителя» и «от потомка». Имя «от родителя» обязательно.

19. Если связь не именуется со стороны потомка, то имя «от родителя» должно выбираться так, чтобы связь легко читалась и со стороны потомка.

20. Неспецифические соединения обязательно именуются с обеих сторон.

21. Сущность может иметь несколько альтернативных ключей.

22. Как первичный, так и альтернативный ключ может быть либо одиночным атрибутом, либо группой атрибутов.

23. Отдельный атрибут может быть частью более чем одного ключа, первичного или альтернативного.

24. Атрибуты, входящие в первичный и альтернативный ключи, могут быть как собственными атрибутами сущности, так и присоединенными через связь с другой сущностью.

25. Первичный и альтернативные ключи должны содержать только те атрибуты, которые необходимы для уникальной идентификации экземпляров сущности.

26. Каждый неключевой атрибут должен неприводимо зависеть от первичного ключа, если он составной.

27. Каждый атрибут, не являющийся частью первичного ключа или какого-либо из альтернативных, должен функционально зависеть только от первичного ключа и каждого из альтернативных [29].

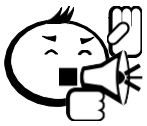
5.8 Дополнения к модели и лексические соглашения стандарта IDEF1X

Диаграммы модели сопровождаются дополнительными материалами, уточняющими и поясняющими ее смысл. Имеются два вида дополнений: ***глоссарий*** и ***примечания*** [17, 26–30].

Глоссарий является обязательным дополнением. Он содержит описания отдельных диаграмм модели и определения сущностей, атрибутов и доменов.

Обязательные компоненты глоссария:

- **имена** – уникальные, осмысленные и соответствующие природе ПО наименования сущностей, атрибутов и доменов;
- **определения** – краткие, точные, однозначные тексты, обеспечивающие правильное понимание смысла имен, одинаковое для любого читателя.



Определение должно быть одинаково применимо в любом контексте, в котором встречается имя.

Кроме того, глоссарий может содержать **список псевдонимов**. Каждому имени могут соответствовать в различных контекстах псевдонимы. Необходи- мость их использования обусловлена тем, что в различных частях моделируе- мой ПО может использоваться разная терминология. Поэтому одни и те же ве- щи могут называться разными именами. Полный список псевдонимов должен сопровождать каждое имя.

Примечания – это необязательный вид дополнений. Они могут пояснять смысл диаграмм и их элементов, содержать описания путей связей, сообщать о функциях, связанных с тем или иным объектом и т. п.

Примечания нумеруются числами натурального ряда. На диаграмме ссылка на примечание помещается около соответствующего объекта в круглых скобках.

Для того чтобы обеспечить наглядность и читаемость диаграмм, стандарт 1DEF1X рекомендует придерживаться ряда ***соглашений*** относительно построения имен и фраз на диаграммах.

Имена. В именах сущностей и атрибутов используются только буквы, цифры, а также знаки-разделители: «дефис», «подчерк» и «пробел». Имя должно начинаться с **буквы**. Части составного имени отделяются дефисом, подчерком или пробелом. Эти разделители не различаются.

Рекомендуется имена сущностей на диаграммах и в текстах глоссариев и замечаний всегда писать **ПРОПИСНЫМИ** буквами.

Например, глоссарий может содержать такое определение имени ПОСТАВЩИК: «Юридическое лицо, заключившее с фирмой договор на поставку

одного или нескольких видов ДЕТАЛей для одного или нескольких видов ИЗДЕЛИЙ».

Это определение содержит сведения не только о смысле, но и о связях определяемой сущности с другими сущностями ПО. Выделение имен этих сущностей регистром облегчает зрительное восприятие фактов связи.

Помимо вышеперечисленных разделителей в именах используются еще «•» и «/». Точка отделяет имя роли от основного имени атрибута. Имя роли предшествует точке, основное имя следует за точкой. Ни перед, ни после точки не допускаются другие разделители. Слэш разделяет разнонаправленные имена связи. Кроме того, он разделяет имя сущности или связи и его ***идентификатор***.

Литература

1. Федеральный государственный образовательный стандарт высшего профессионального образования (ФГОС ВПО) третьего поколения по направлению подготовки 230700 «Прикладная информатика» (квалификация (степень) «бакалавр»), утвержден Приказом Министерства образования и науки Российской Федерации от 22 декабря 2009 г. № 783.
2. Исакова, А. И. Научная работа : методические указания по самостоятельной и индивидуальной работе студентов всех форм обучения для направления бакалавриата 230700 – «Прикладная информатика» / А. И. Исакова ; Министерство образования и науки Российской Федерации, Томский государственный университет систем управления и радиоэлектроники, кафедра автоматизированных систем управления. – Томск : ТУСУР, 2013. – 16 с. [Электронный ресурс]. – Режим доступа: http://asu.tusur.ru/learning/bak230700/d53/b230700_d53_work.doc (дата обращения: 03.11.2017).
3. Грибова, В. В. Методы и средства разработки пользовательского интерфейса: современное состояние / В. В. Грибова, А. С. Клещев // Программные продукты и системы. – 2001. – № 1. – С. 2–6. [Электронный ресурс]. – Режим доступа: http://www.swsys.ru/index.php?page=article&id=765 (дата обращения: 03.12.2015).
4. Волченков, Е. Программная инженерия. Стандартизация пользовательского интерфейса / Е. Волченков. – М., 2002 [Электронный ресурс]. – Режим доступа: http://tizer.adv.vz.ru (дата общаения: 03.11.2017).
5. Основы информационной безопасности : учеб. пособие для вузов / Е. Б. Белов [и др.]. – М. : Горячая линия-Телеком, 2006. – 544 с.
6. Нечаев, Д. В. Методика защиты персональных данных : доклад / Д. В. Нечаев // Научная сессия ТУСУР-2010. – Томск : В-Спектр. – Ч. 3. – С. 176–178.
7. Требования о защите информации, не составляющей государственную тайну, содержащейся в государственных информационных системах :

методический материал / Федеральная служба по техническому и экспортному контролю. – М. : ГНИИИ ПТЗИ ФСТЭК России, 2013. – 38 с.

8. Избачков, Ю. С. Информационные системы : учеб. пособие для вузов / Ю. С. Избачков, В. Н. Петров. – 2-е изд. – СПб. : Питер, 2005. – 655[1] с.
9. Информационные технологии в экономике и управлении : учебник для бакалавров / Санкт-Петербургский государственный университет экономики и финансов ; ред. В. В. Трофимов. – М. : Юрайт, 2013. – 479 с.
10. Исакова, А. И. Информационные системы : учеб. пособие для студентов специальности 080801 / А. И. Исакова ; Министерство образования и науки Российской Федерации, Томский государственный университет систем управления и радиоэлектроники, кафедра автоматизированных систем управления. – Томск : Факультет дистанционного обучения, ТУСУР, 2010. – 202 с.
11. Кузин, А. В. Базы данных : учеб. пособие для вузов / А. В. Кузин, С. В. Левонисова. – 4-е изд., стереотип. – М. : Академия, 2010. – 320 с.
12. Марков, А. С. Базы данных. Введение в теорию и методологию : учебник для вузов / А. С. Марков, К. Ю. Лисовский. – М. : Финансы и статистика, 2006. – 510 с.
13. Полонская, Е. В. Сравнительный анализ возможностей СУБД POSTGRESQL. MYSQL и MS ACCESS : материалы доклада / Е. В. Полонская // Научная сессия ТУСУР-2009. – Томск : В-Спектр, 2009. – Ч. 1. – С. 237–240.
14. Литвин, Пол. ACCESS 2002: Разработка корпоративных приложений : пер. с англ. / Пол Литвин, Кен Гетц, Майк Гунделой. – СПб. : Питер, 2003. – 848 с.
15. Петрова, Г. В. Краткий обзор СУБД (ORACLE, INTERBASE, ACCESS) в области обеспечения безопасности : доклад, тезисы доклада / Г. В. Петрова, Е. А. Мошников // Научная сессия ТУСУР-2008. – Томск : В-Спектр, 2008. – Ч. 3. – С. 171–173.
16. Сенченко, П. В. Организация баз данных : учеб. пособие / П. В. Сенченко ; Томский государственный университет систем управления и радиоэлектроники. – Томск : 2004. – 171 с. [Электронный ресурс]. – Режим доступа:

http://edu.tusur.ru/training/publications/2881 (дата обращения: 03.12.2015).

17. Кузин, А. В. Базы данных : учеб. пособие для вузов / А. В. Кузин, С. В. Левонисова. – 5-е изд., испр. – М. : Академия, 2012. – 320 с.
18. Маккоу, Алекс. Веб-приложения на JavaScript: практическое руководство / А. Маккоу ; пер. Н. Вильчинский. – СПб. : ПИТЕР, 2012. – 288 с.
19. Ли, Джеймс. Использование Linux, Apache, MySQL и PHP для разработки WEB-приложений : пер. с англ. / Джеймс Ли, Брент Уэр ; пер. А. Н. Узниченко. – М. : Вильямс, 2004. – 429[3] с.
20. Тидвелл, Дженифер. Разработка пользовательских интерфейсов : пер. с англ. / Дж. Тидвелл ; пер. Е. Шикарева. – СПб. : Питер, 2008. – 416 с.
21. Колисниченко, Д. Н. Разработка Linux-приложений : научно-популярное издание / Д. Н. Колисниченко. – СПб. : БХВ-Петербург, 2012. – 431 с.
22. Гвоздева, Т. В. Проектирование информационных систем : учеб. пособие для вузов / Т. В. Гвоздева, Б. А. Баллод. – Ростов н/Д : Феникс, 2009. – 512 с.
23. Туманов, В. Е. Проектирование реляционных хранилищ данных : справочное издание / В. Е. Туманов, С. В. Маклаков. – М. : ДИАЛОГ-МИФИ, 2007. – 336 с.
24. Мытник, С. А. Проектирование информационных систем : учеб. пособие / С. А. Мытник ; Федеральное агентство по образованию, Томский государственный университет систем управления и радиоэлектроники, кафедра автоматизации обработки информации. – Томск : ТМЦДО, 2008. – 163 с.
25. Золотов, С. Ю. Проектирование информационных систем : учеб. пособие / С. Ю. Золотов ; Министерство образования и науки Российской Федерации, Томский государственный университет систем управления и радиоэлектроники, факультет дистанционного обучения. – Томск : Эль Контент, 2013. – 87 с.
26. Коннолли, Томас. Базы данных. Проектирование, реализация и сопровождение. Теория и практика / Томас Коннолли, Каролин Бегг. – 3-е изд. : пер. с англ. – М. : Издательский дом «Вильямс», – 2003. – 1440 с.

27. Диго, С. М. Базы данных : Проектирование и использование : учебник для вузов / С. М. Диго. – М. : Финансы и статистика, 2005. – 590 с.
28. Сибилёв, В. Д. Базы данных : учеб. пособие / В. Д. Сибилёв ; Федеральное агентство по образованию, Томский государственный университет систем управления и радиоэлектроники, кафедра автоматизированных систем управления. – Томск : ТУСУР, 2007. – 279 с.
29. Советов, Б. Я. Базы данных: теория и практика : учебник для бакалавров / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовской. – 2-е изд. – М. : Юрайт, 2012. – 464 с.
30. Давыдова, Е. М. Базы данных : учеб. пособие / Е. М. Давыдова, Н. А. Новгородова ; Министерство образования и науки Российской Федерации, Томский государственный университет систем управления и радиоэлектроники, кафедра комплексной информационной безопасности электронно-вычислительных систем. – 3-е изд., перераб. и доп. – Томск : В-Спектр, 2012. – 128 с.

Приложение 1
Пример 1 оформления отчета по научной работе

Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра автоматизированных систем управления (АСУ)

АВТОМАТИЗАЦИЯ ПРОЦЕССОВ СТРАХОВАНИЯ

Отчёт по дисциплине «Научная работа 1»

Студент гр. ____
_____И. О. Фамилия
«___»_____201_ г.

Руководитель
доцент каф. АСУ,
канд. техн. наук
_____А. И. Исакова
«___»_____201_ г.

Томск 201_

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель дисциплины «Научная работа 1» - глубокое закрепление теоретических знаний, получаемых студентами при изучении дисциплин учебного плата, повышение требовательности к себе, аккуратности, точности в выполнении заданий и научной активности. Приобщение к будущей профессии, таким образом, приобретает творческий характер и стимулирует креативную деятельность студентов.

Цель научной работы:

- рассмотреть аналоги автоматизации документооборота предприятия;

- выбрать среду разработки ИС;

-проектирование информационно-логической SADT-модели информационной системы «ЗАО «Макс»;

-спроектировать концептуальную модель базы данных.

1 ЦЕЛИ, ЗАДАЧИ И ФУНКЦИИ ИС

В условиях перехода к рыночной экономике страхование принадлежит к числу наиболее быстро развивающихся отраслей хозяйственной деятельности. Рыночная экономика, и, прежде всего негосударственный сектор народного хозяйства, предъявляет спрос на различные виды страхования, так как частная собственность, в отличие от государственной, нуждается во всеобъемлющей страховой защите. Она не имеет за своей спиной финансовых гарантий со стороны государства и хочет обезопасить себя от последствий возможных рисков.

По своей сути страхование представляет собой создание целевых фондов денежных средств, предназначенных для защиты имущественных интересов населения в частной и хозяйственной жизни от стихийных бедствий и других непредвиденных, случайных по своей природе чрезвычайных событий, сопровождающихся ущербами.

Страховой фонд как экономическая категория представляет собой резерв материальных или денежных средств, предназначенный для возмещения ущербов.

На макроэкономическом уровне страхование выполняет следующие функции:

Обеспечения непрерывности общественного воспроизводства;

Освобождения государства от дополнительных расходов;

Стимулирования научно-технического прогресса;

Защиты интересов пострадавших лиц в системе отношений гражданской ответственности;

Концентрации инвестиционных ресурсов и стимулирования экономического роста.

В России, несмотря на все сложности становления рыночных отношений, формируется круг крупных страховых компаний. До сих пор продолжается тенденция по увеличению их удельного веса на российском страховом рынке. Одно из ведущих мест среди первых ста по величине компаний занимает Московская Акционерная Страховая Компания ЗАО «Макс». При осуществлении страховой деятельности ЗАО «Макс» основное внимание

уделяет индивидуальному подходу к каждому клиенту и неукоснительному выполнению всех взятых на себя обязательств. При разработке и внедрении новых страховых продуктов специалисты компании руководствуются принципом их соответствия потребностям клиента по удобству условий, доступности тарифов и качеству сервиса.

В процессе сотрудничества с компанией клиент может быть уверен, что в случае ДТП все финансовые вопросы возьмет на себя страховая компания. Это определяет комплекс задач, которые должна выполнять разрабатываемая информационная система:

– создание клиентской базы данных для осуществления быстрого поиска нужного клиента, для просмотра необходимой информации о нем, в случае его повторного обращения в страховую компанию, а также для составления отчетной документации;

– оформление страхового договора и расчет страхового взноса, чтобы уменьшить количество ошибок при расчете и оформлении документа;

– хранение информации о страхователе для повторного использования его данных, если он решит продлить договор страхования или заключить новый договор;

– хранение информации о транспортном средстве для переоформления договора, в случае продажи транспортного средства другому лицу, тоже застрахованного в страховой компании зао «макс»;

– хранение информации о страховых агентах для составления отчета по агентам. из общей суммы всех заключенных договоров одним агентом берется процент и зачисляется в его заработную плату;

– контроль системы оплаты страхового договора для составления расчетно-платежной ведомости и передачи ее в бухгалтерию;

– изменение статуса страхового договора (расторгнутый, переоформленный, пролонгированный) для внесения изменений по желанию клиента;

– печать и выдача страхового документа страхователю, для подтверждения его права на страховую защиту;

– формирование статистической отчетности для бухгалтерии.

Выбранные задачи имеют общую информационную базу и нормативно-справочную информацию. Для решения данного комплекса задач необходимо внедрение информационных технологий, автоматизированных информационных систем. При разработке информационной системы будет использована СУБД. Страховые документы будут храниться в единой базе, перемещаться, и обрабатываться с помощью компьютерной сети организации. Ведь компьютерная сеть, связывает компьютеры на рабочих местах организации. При этом любые действия с документами, осуществляемые на конкретном компьютере, могут автоматически собираться и накапливаться на сервере (центральном компьютере) сети.

Автоматизированная система представляет возможность производить оперативный и эффективный обмен информацией между всеми участками производственного процесса, позволяет сократить время, требуемое на подготовку конкретных задач, исключить возможных появлений ошибок подготовки отчётной документации.

Разработка АИС страховой компании по движению договоров ОСАГО, направлена на улучшение качества работы с клиентами за счет автоматизации продаж страховых полисов, контроля за качеством обслуживания и устранения ошибок, связанных с оформлением страхового договора. Сотрудникам компании программное обеспечение позволит получить сведения о досрочно прекращенных и заключенных ранее договорах по ОСАГО, произведенных выплатах, выданных бланках строгой отчетности, моментально проверять данные, вести и формировать реестры для РСА.

За счет внедрения ИС повысится производительность работы пользовательских приложений, которые позволят оперативно взаимодействовать с клиентами и выйти на качественно новый уровень их обслуживания.

Таким образом, внедрение системы автоматизации обработки страховых документов предусматривает комплекс организационно-технических мероприятий, решающих большинство перечисленных выше проблем. Главной составной частью информационной системы является создание единой информационной клиентской базы, сведения о каждом клиенте и заключенном

с ним договоре будут надежно храниться в центральной базе данных компании.

Следовательно, автоматизация позволит:

1) Контролировать процесс страхования;
2) Осуществлять быстрый поиск страховых документов в базе данных;
3) Значительно сократит время на оформление страховых документов;
4) Расширенные возможности по получению статистических и аналитических данных, позволят планировать стратегию страхования, снизить затраты и увеличить объем страхования;
5) Повысит рост производительности труда, за счет высокого быстродействия системы;
6) Уменьшит ошибки, связанные с оформлением полиса и подготовкой отчетной документации;
7) Обеспечит удобство в работе по заполнению страхового полиса;
8) Вводить, хранить и управлять данными о клиентах и объектах страхования;
9) Предоставлять статистические отчеты для анализа страхования;
10) Составлять отчеты о количестве застрахованных клиентах.

В конечном счете, при внедрении информационной системы повысится оперативность управления ресурсами организации, выявятся ее резервы и перспективы роста для укрепления позиций на рынке страхования.

Разрабатываемая информационная система страховой компании должна устранить все недостатки в работе путем снижения затрат и увеличения объема страхования.

2 АНАЛОГИ ИНФОРМАЦИОННЫХ СИСТЕМ ДЛЯ АВТОМАТИЗАЦИИ ПРОЦЕССОВ СТРАХОВАНИЯ

Ниже приведены имеющиеся на рынке решениями, с аналогичными объектами и целями автоматизации:

1) 1С:Мобильный Страховой Офис [1].

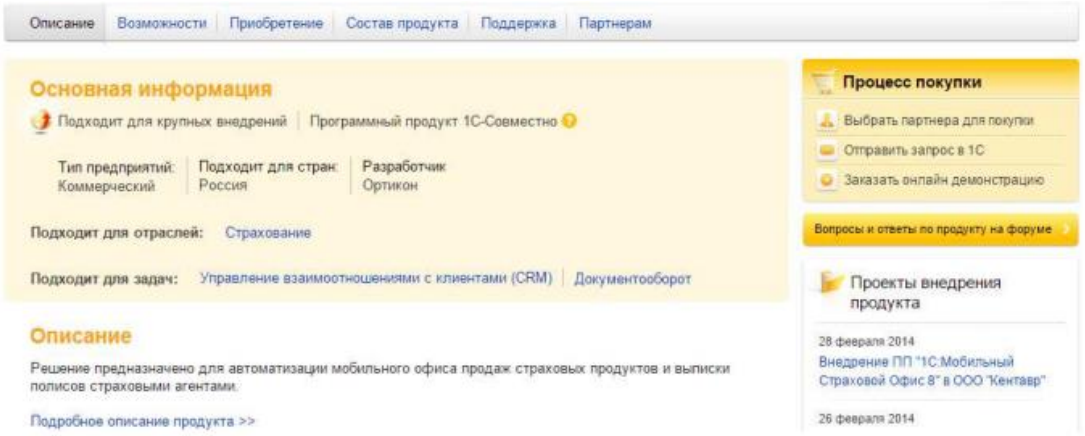


Рисунок 1 - Home Page «1С:Мобильный Страховой Офис» на официальном сайте производителя

Решение 1С:Мобильный Страховой Офис (рис. 1) предназначено для автоматизации мобильного офиса продаж страховых продуктов и выписки полисов страховыми агентами. Программный продукт "1С:Мобильный страховой офис" позволяет автоматизировать процесс выписки полисов страховыми агентами, а также автоматизировать точки продаж страховых организаций и страховых брокеров. Программа содержит удобный конструктор страховых тарифов, на основании которого система автоматически строит калькулятор, обеспечивающий быструю предпродажную подготовку и расчет оформляемых страховых полисов. Страховой агент или брокер могут настроить систему для расчета продуктов разных страховых организаций, тем самым предоставить страхователю выбор оптимального для него продукта. Программный продукт "1С:Мобильный страховой офис" позволяет пользователю сохранять и быстро находить информацию по клиенту, событиям, связанных с их обслуживанием, вести и хранить электронную переписку, не забыть про сроки окончания договоров и другие важные события.

Функциональные возможности конфигурации "1С:Мобильный страховой

офис":

– конструирование страховых продуктов, включая настройку тарифной сетки;

– учет объектов страхования по типам, с настройкой произвольного количества свойств;

– создание и хранение договоров и дополнительных соглашений по страхованию;

– ввод оплат по договорам страхования;

– учет бланков строгой отчетности;

– механизм управления контактами для точки продаж;

– обмен данными.

2) Система ПАРУС-Страхование 6. X [2].

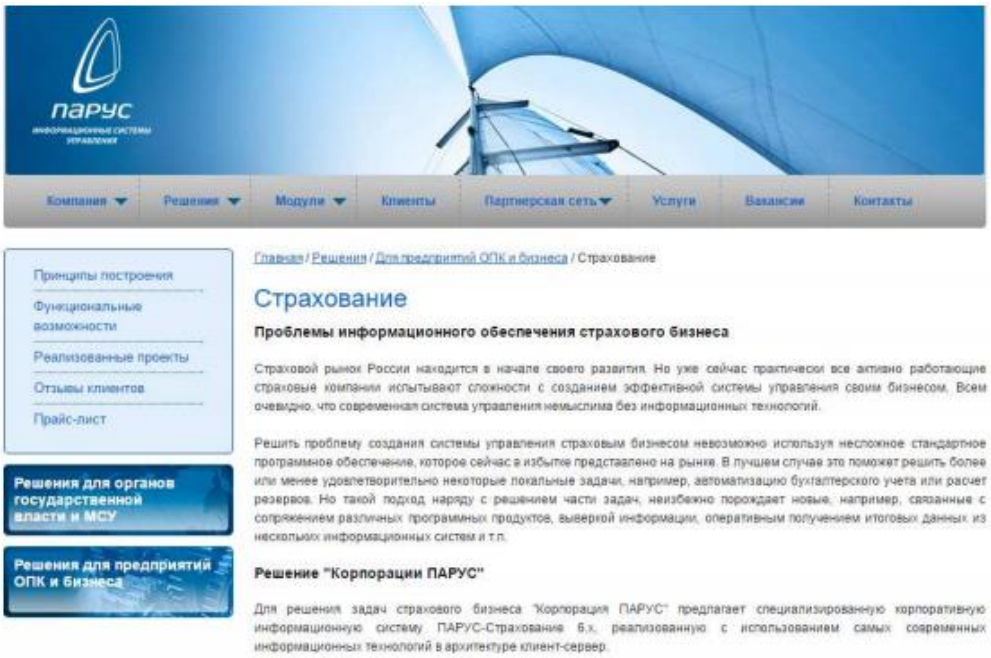


Рисунок 2 - Домашняя страница системы «ПАРУС-Страхование 6.x»

Для решения задач страхового бизнеса "Корпорация ПАРУС" предлагает специализированную корпоративную информационную систему ПАРУС-Страхование 6.x (рис. 2), реализованную с использованием самых современных информационных технологий в архитектуре клиент-сервер. Данная система позволяет автоматизировать все основные бизнес-процессы страховой деятельности. В ее состав входит множество различных модулей. Рассмотрим функциональность модуля "Страхование имущества и ответственности",

который предназначен для сопровождения договоров классического страхования:

- регистрация брокеров и агентов;
- регистрация и сопровождение договоров страхования (полисов);
- регистрация и учет использования любых бланков строгой отчетности;
- настройка системы на конкретные правила страхования;
- учет страховых событий и убытков;
- учет комиссионного вознаграждения;
- учет графика уплаты страховой премии и реального движения денежных средств по договору страхования (полису);
- формирование отчетности по видам страхования, относящимся к страхованию имущества и ответственности.

3) КОРУС Консалтинг «CRM для страховых брокеров» [3].

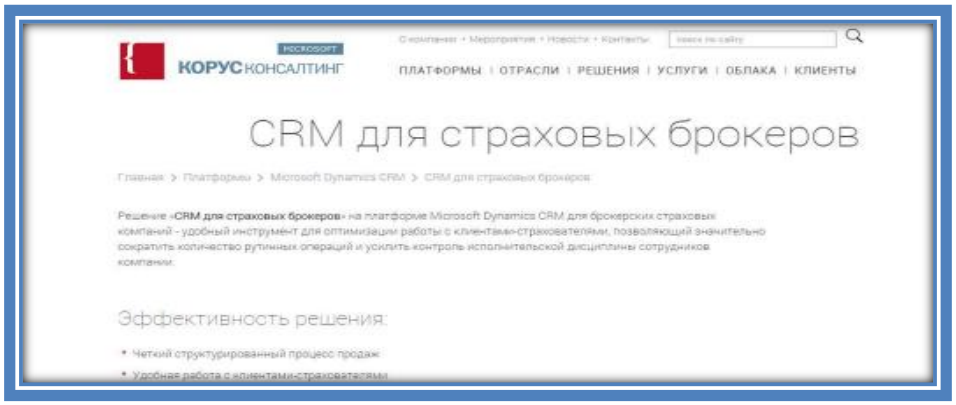


Рисунок 3 - Описание решения КОРУС Консалтинг «CRM для страховых брокеров» на официальном сайте производителя

Решение «CRM для страховых брокеров» (рис.3) на платформе Microsoft Dynamics CRM для брокерских страховых компаний - удобный инструмент для оптимизации работы с клиентами-страхователями, позволяющий значительно сократить количество рутинных операций и усилить контроль исполнительской дисциплины сотрудников компании.

Функциональность решения:

- заключение и обслуживание договоров страхования;
- возобновление договоров;
- управление бланками строгой отчетности;

- хранение документов;
- формирование отчетов.

3. ОБОСНОВАНИЕ ВЫБОРА СРЕДЫ РАЗРАБОТКИ

На сегодняшний день известно более двух десятков форматов данных настольных СУБД, однако наиболее популярными следует признать Access, Paradox, FoxPro и dBase.

Рассмотрим каждую из этих СУБД в отдельности.

dBase и Visual dBase

Первая промышленная версия СУБД dBase – dBase II (принадлежащая тогда компании Ashton-Tate, приобретенной позже компанией Borland) появилась в начале 80-х годов. Благодаря простоте в использовании, нетребовательности к ресурсам компьютера и, что не менее важно, грамотной маркетинговой политике компании-производителя этот продукт приобрел немалую популярность [7].

Хранение данных в dBase основано на принципе «одна таблица — один файл» (эти файлы обычно имеют расширение *.dbf). MEMO-поля и BLOB-поля (доступные в поздних версиях dBase) хранятся в отдельных файлах (обычно с расширением *.dbt). Индексы для таблиц также хранятся в отдельных файлах. При этом в ранних версиях этой СУБД требовалась специальная операция реиндексирования для приведения индексов в соответствие с текущим состоянием таблицы [7].

Формат данных dBase является открытым, что позволило ряду других производителей заимствовать его для создания dBase-подобных СУБД, частично совместимых с dBase по форматам данных. Например, весьма популярная некогда СУБД FoxBase (разработанная Fox Software, Inc. и ныне принадлежащая Microsoft) использовала формат данных dBase для таблиц, однако форматы для хранения MEMO-полей и индексов были своими собственными, несовместимыми с dBase.

После покупки dBase компанией Borland этот продукт, получивший впоследствии название Visual dBase, приобрел набор дополнительных возможностей, характерных для средств разработки этой компании и для

имевшейся у нее другой настольной СУБД – Paradox. Среди этих возможностей были специальные типы полей для графических данных, поддерживаемые индексы, хранение правил ссылочной целостности внутри самой базы данных, а также возможность манипулировать данными других форматов, в частности серверных СУБД, за счет использования BDE API и SQL Links [7].

В настоящее время Visual dBase принадлежит компании dBase, Inc. Его последняя версия — Visual dBase 7.5 имеет следующие возможности:

– средства манипуляции данными dBase и FoxPro всех версий;

– ядро доступа к данным Advantage Database Server фирмы Extended Systems и ODBC-драйвер для доступа к данным этой СУБД;

– средства визуального построения запросов.

Paradox

В конце 80-х — начале 90-х годов Paradox, принадлежавший тогда компании Borland International, был весьма популярной СУБД, в том числе и в нашей стране, где он одно время занимал устойчивые позиции на рынке средств разработки настольных приложений с базами данных.

Принцип хранения данных в Paradox сходен с принципами хранения данных в dBase — каждая таблица хранится в своем файле (расширение *.db), MEMO- и BLOB-поля хранятся в отдельном файле (расширение *.md), как и индексы (расширение *.px).

Однако, в отличие от dBase, формат данных Paradox не является открытым, поэтому для доступа к данным этого формата требуются специальные библиотеки. Например, в приложениях, написанных на C или Pascal, использовалась некогда популярная библиотека Paradox Engine, ставшая основой Borland Database Engine (BDE). Эта библиотека используется ныне в приложениях, созданных с помощью средств разработки Borland (Delphi, C++Builder), в некоторых генераторах отчетов (например, Crystal Reports) и в самом Paradox. Существуют и ODBC-драйверы к базам данных, созданным различными версиями этой СУБД [7].

Microsoft FoxPro и Visual FoxPro

FoxPro ведет свое происхождение от настольной СУБД FoxBase фирмы Fox Software. Разрабатывая FoxBase в конце 80-х годов, эта компания преследовала цель создать СУБД, функционально совместимую с dBase с точки зрения организации файлов и языка программирования, но существенно превышающую ее по производительности. Одним из способов повышения производительности являлась более эффективная организация индексных файлов, нежели в dBase, – по формату индексных файлов эти две СУБД несовместимы между собой [8].

По сравнению с аналогичными версиями dBase, FoxBase и более поздняя версия этого продукта, получившая название FoxPro, предоставляли своим пользователям несколько более широкие возможности, такие как использование деловой графики, генерация кода приложений, автоматическая генерация документации к приложениям и т.д.

Впоследствии этот продукт был приобретен компанией Microsoft. Его последние версии (начиная с версии 3.0, выпущенной в 1995 году) получили название Visual FoxPro. С каждой новой версией этот продукт оказывался все более и более интегрирован с другими продуктами Microsoft, в частности с Microsoft SQL Server, – в состав Visual FoxPro в течение нескольких последних лет входят средства переноса данных FoxPro в SQL Server и средства доступа к данным этого сервера из Visual FoxPro и созданных с его помощью приложений [8].

Последняя версия этого продукта — Visual FoxPro 6.0, доступна и отдельно, и как составная часть Microsoft Visual Studio 6.0. Отличительной особенностью этой настольной СУБД от двух рассмотренных выше является интеграция этого продукта с технологиями Microsoft, в частности поддержка COM (Component Object Model — компонентная объектная модель, являющаяся основой функционирования 32-разрядных версий Windows и организации распределенных вычислений в этой операционной системе), интеграция с Microsoft SQL Server, возможности создания распределенных приложений, основанных на концепции Windows DNA (Distributed interNet Applications) [8].

Visual Fox Pro 6.0 предоставляет следующие возможности [8]:

– средства создания COM-объектов и объектов для Microsoft Transaction Server, позволяющих создавать масштабируемые многозвенные приложения для обработки данных;

– средства доступа к данным серверных СУБД, базирующиеся на использовании OLE DB (набор COM-интерфейсов, позволяющий осуществить унифицированный доступ к данным из разнообразных источников, в том числе из нереляционных баз данных и иных источников, например Microsoft Exchange);

– средства доступа к данным Microsoft SQL Server и Oracle, включая возможность создания и редактирования таблиц, триггеров, хранимых процедур;

– средства отладки хранимых процедур Microsoft SQL Server;

– средство визуального моделирования компонентов и объектов, являющиеся составными частями приложения – Visual Modeller;

– средство для управления компонентами приложений, позволяющее осуществлять их повторное использование.

Итак, тенденции развития этого продукта очевидны: из настольной СУБД Visual FoxPro постепенно превращается в средство разработки приложений в архитектуре «клиент/сервер» и распределенных приложений в архитектуре Windows DNA. Впрочем, эти тенденции в определенной степени характерны для всех наиболее популярных настольных СУБД: и dBase, и Paradox также позволяют осуществлять доступ к наиболее популярным серверным СУБД.

Microsoft Access

В отличие от Visual FoxPro, фактически превратившегося в средство разработки приложений, Access ориентирован в первую очередь на пользователей Microsoft Office, в том числе и не знакомых с программированием. Это, в частности, проявилось в том, что вся информация, относящаяся к конкретной базе данных, а именно таблицы, индексы (естественно, поддерживаемые), правила ссылочной целостности, бизнес-правила, список пользователей, а также формы и отчеты хранятся в одном

файле, что в целом удобно для начинающих пользователей.

Microsoft Access – это функционально полная реляционная СУБД. В ней предусмотрены все необходимые вам средства для определения и обработки данных, а также для управления ими при работе с большими объемами информации [9].

В состав Access входят [9]:

– средства манипуляции данными Access и данными, доступными через ODBC (последние могут быть «присоединены» к базе данных Access);

– средства создания форм, отчетов и приложений; при этом отчеты могут быть экспортированы в формат Microsoft Word или Microsoft Excel, а для создания приложений используется Visual Basic for Applications, общий для всех составных частей Microsoft Office;

– средства доступа к данным серверных СУБД через OLE DB;

– средства создания клиентских приложений для Microsoft SQL Server;

– средства администрирования Microsoft SQL Server.

Система Access – это набор инструментов конечного пользователя для управления базами данных. В ее состав входят конструкторы таблиц, форм, запросов и отчетов. Эту систему можно рассматривать и как среду разработки приложений. Используя макросы или модули для автоматизации решения задач, можно создавать ориентированные на пользователя приложения такими же мощными, как и приложения, написанные непосредственно на языках программирования. При этом они будут включать кнопки, меню и диалоговые окна [9].

Access специально спроектирован для создания многопользовательских приложений, где файлы базы данных являются разделяемыми ресурсами в сети. Система Access поддерживает обработку транзакций с гарантией их целостности. Кроме того, предусмотрена защита на уровне пользователя, что позволяет контролировать доступ к данным отдельных пользователей и целых групп.

Основываясь на изложенных выше данных в качестве СУБД разрабатываемой системы был выбран Microsoft Access.

4 ФУНКЦИОНАЛЬНЫЙ СОСТАВ ИС ПО СИНТАКСИСУ МЕТОДОЛОГИИ SADT

Моделирование бизнес-процессов страховой компании ЗАО «Макс» осуществлено с помощью программы ALLFUSION PROCESS MODELER V 7.1.

В результате комплексного анализа информационной структуры и бизнес-процессов страховой компании ЗАО «Макс» построена функциональная модель, изображенная на рисунке 4. Контекстная диаграмма представляет собой общую деятельность страховой компании.

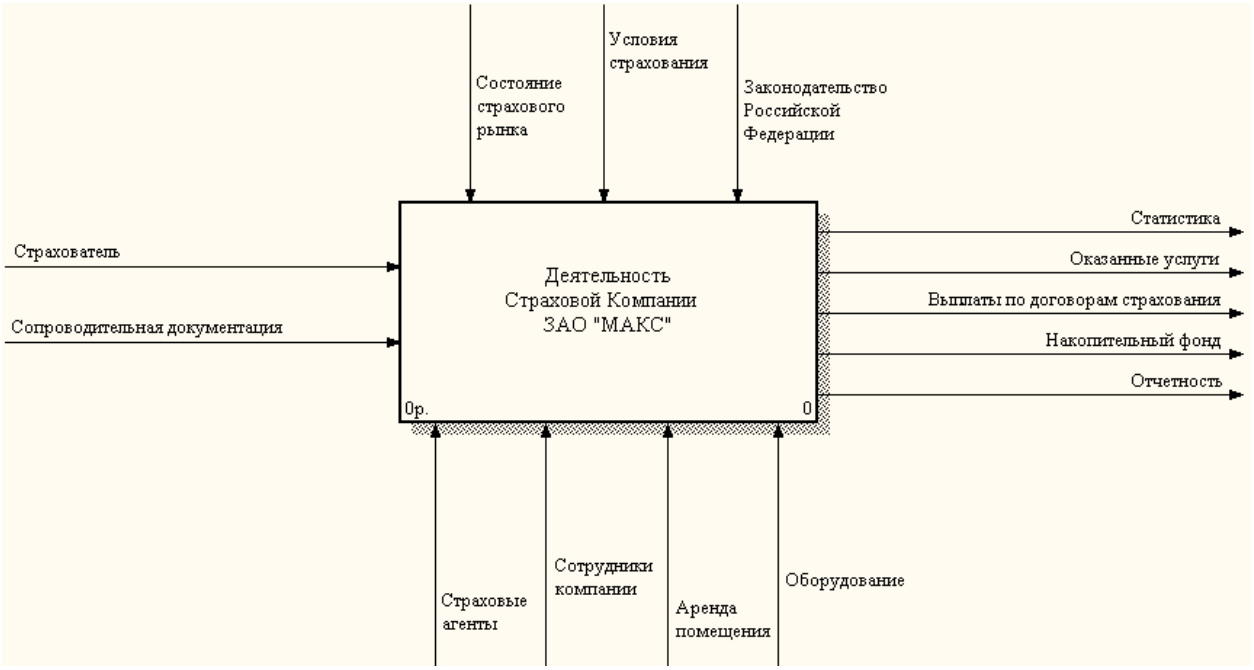


Рисунок 4 - Контекстная диаграмма страховой компании ЗАО «Макс»

На данной диаграмме бизнес-процессов страховой деятельности входной информацией является: «Страхователь» и «Сопроводительная документация».

Механизмами (ресурсами) являются: «Страховые агенты, Сотрудники компании, Аренда помещения и Оборудование».

Управлением являются: «Состояние страхового рынка, Условия страхования и Законодательство РФ».

Выходами являются: «Статистические данные, Оказанные услуги, Выплаты по договорам страхования при страховых случаях, сформированный Накопительный фонд за счет сбора страховых премий и Отчетность».

В научной работе будет рассматриваться оформление, расчет и движение страхового полиса по обязательному страхованию автогражданской ответственности (ОСАГО).

При декомпозиции контекстной диаграммы (рис. 5) видна деятельность компании, которая состоит из: «Заключения страхового договора, Формирования страхового фонда, финансово-экономического анализа страховой деятельности и Урегулировании убытка - возмещение ущерба при страховом случае».

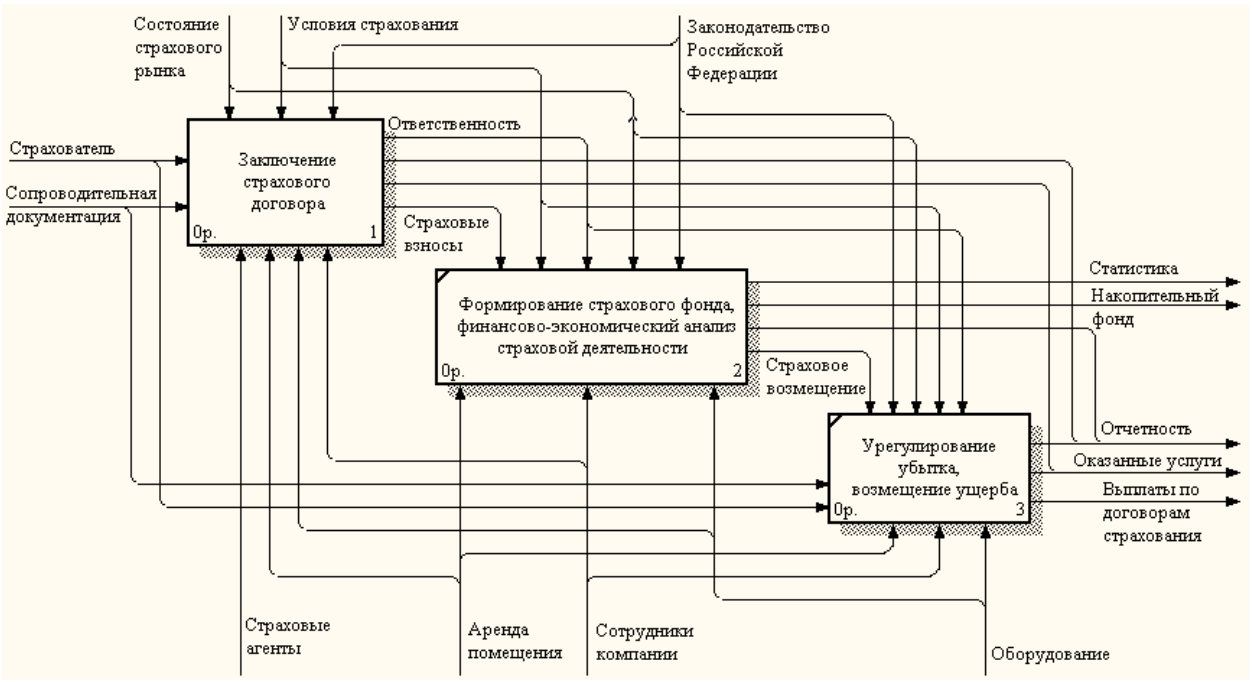


Рисунок 5 - Декомпозиция бизнес процессов страховой деятельности ЗАО «Макс»

В страховой компании основной функцией является заключение страхового договора, за счет которого формируется страховой фонд из полученных страховых премий.

Бизнес процесс заключения страхового договора состоит из: «Поступления заказа от клиента, Оформлении страхового полиса и Выдачи полиса страхователю» (рис. 6).

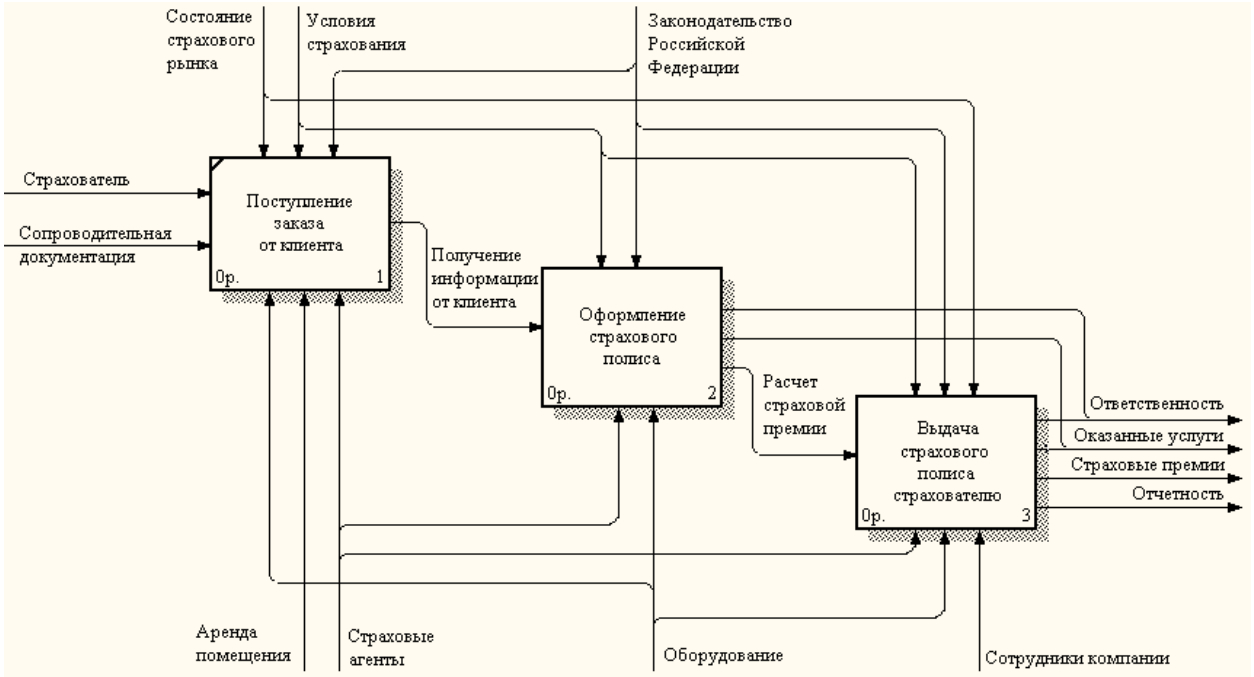


Рисунок 6 - Диаграмма бизнес процесса заключения страхового договора

Основными составляющими страховой компании ЗАО «Макс» по договорам ОСАГО являются: оформление страхового полиса и расчет страховой премии (рис. 7).

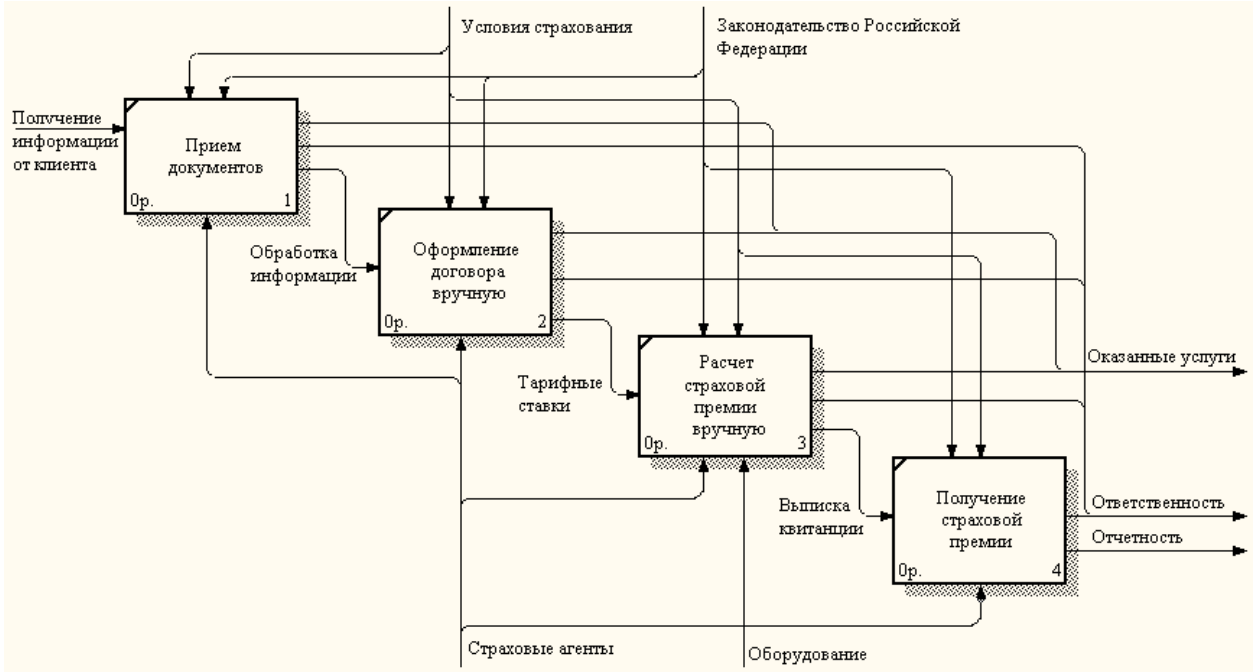


Рисунок 7 - Декомпозиция бизнес процесса заключения договора ОСАГО «Как-Есть»

На данной диаграмме бизнес-процессов входной информацией является: «Получение информации от клиента: данные об автомобиле, собственнике, лицах допущенных к управлению, сроке страхования, периоде использования транспортного средства».

При выдаче страхового полиса страхователю делается его копия, которая движется внутри страховой компании (рис. 8).

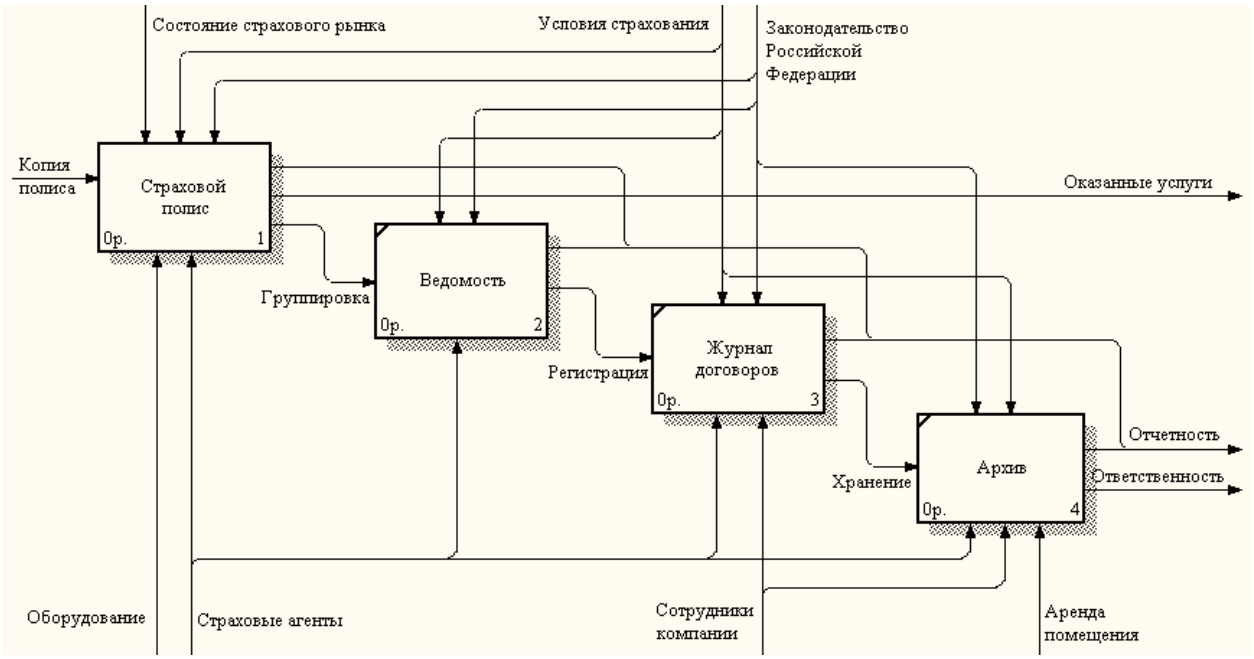


Рисунок 8 - Декомпозиция бизнес процесса движения страхового полиса внутри страховой компании «Как-Есть»

Заключенные страховые договора группируются в ведомости и регистрируются в журнале договоров, а затем хранятся в архиве компании.

Страхователь может внести изменения в заключенный ранее страховой договор, например, переоформить его, добавив еще одного водителя в список лиц, допущенных к управлению транспортным средством или наоборот сделать без ограничения лиц, допущенных к управлению. А также полис можно восстановить (в случае утери), переоформить, расторгнуть (если например транспортное средство собираются продать) или пролонгировать, то есть продлить период использования транспортного средства. Диаграмма бизнес процесса возможных действий со страховым полисом изображена на рисунке 9.

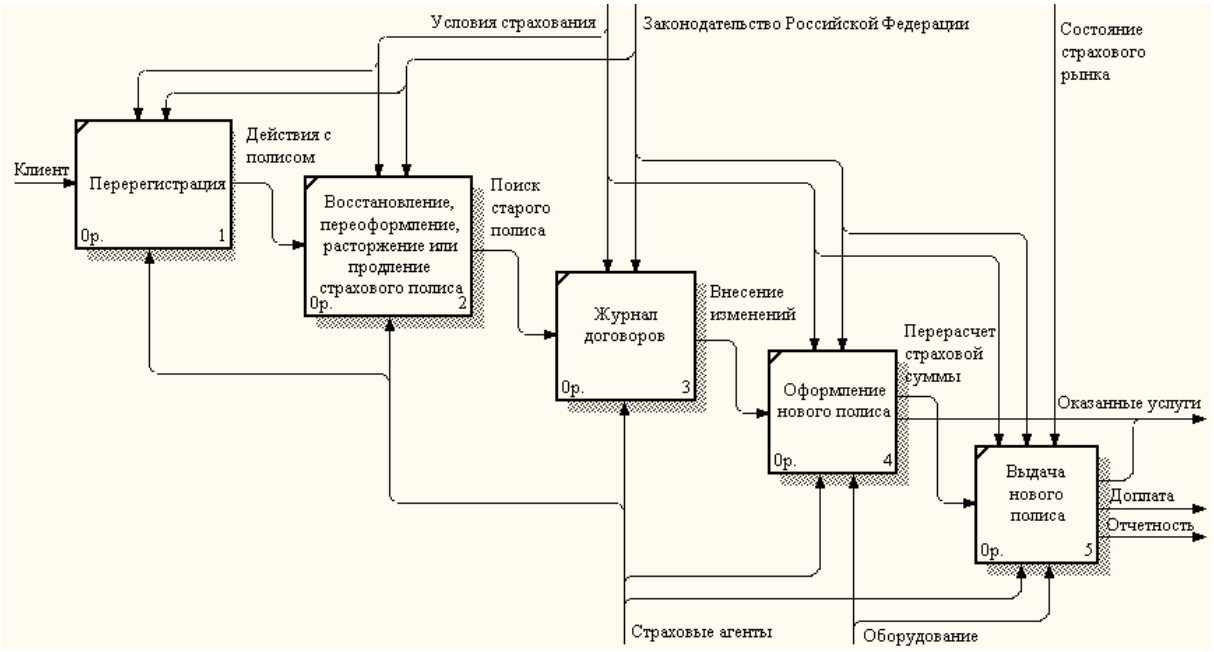


Рисунок 9 - Декомпозиция бизнес процесса возможных действий со страховым полисом «Как-Есть»

Декомпозиция бизнес процесса заключения договоров страхования состоит как из элементов содержащихся в функциональной модели существующих бизнес процессов AS-IS (рис. 8), так и нового элемента модели TO-BE (рис. 10).

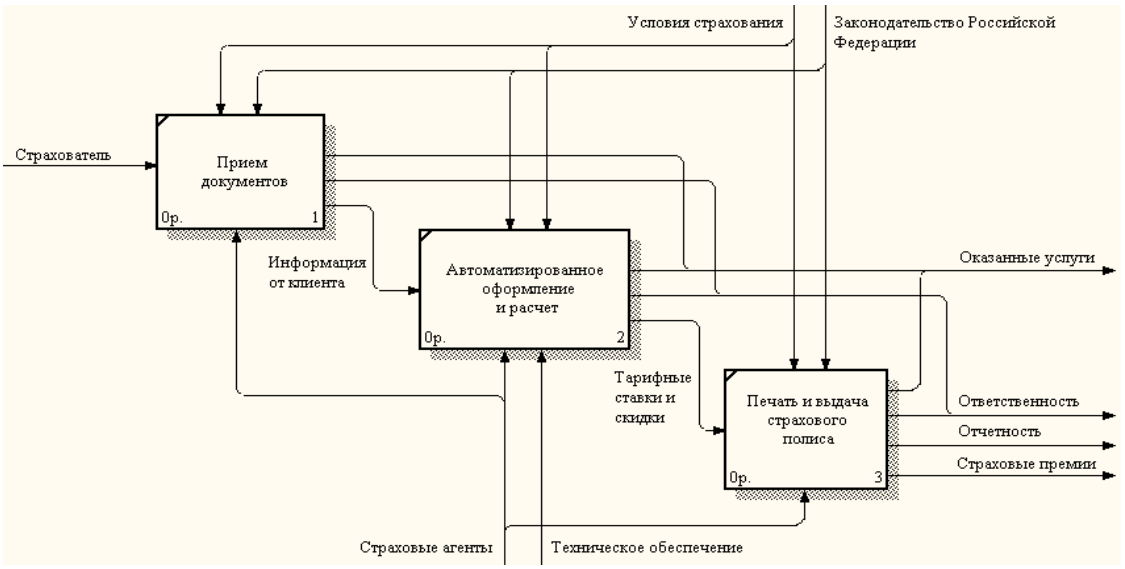


Рисунок 10 - Модель TO-BE декомпозиция бизнес процесса оформления страхового полиса ОСАГО «Как-Будет»

Модель ТО – ВЕ декомпозиции бизнес процесса оформления полиса состоит из следующих бизнес процессов:

- прием документов
- автоматизированное оформление полиса и расчет страховой премии

– печать и выдача страхового полиса

Декомпозиция бизнес процесса движения страхового полиса внутри страховой компании также состоит из элементов существующего бизнес процесса AS-IS (рис. 8), так и нового элемента модели TO-BE (рис. 11).

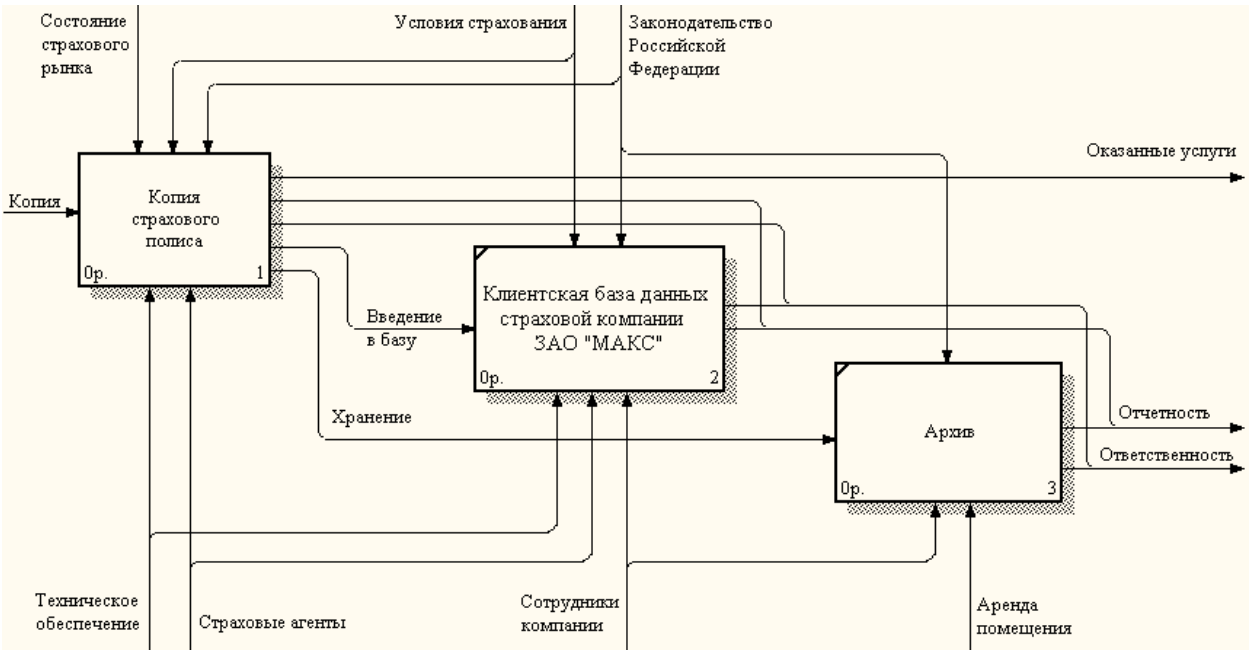


Рисунок 11 - Модель TO-BE декомпозиция бизнес процесса движения страхового полиса внутри страховой компании «Как-Будет».

Модель TO – BE состоит из главного бизнес процесса: создания информационной Клиентской базы данных.

И декомпозиция бизнес процесса возможных действий со страховым полисом состоит из элементов содержащихся в функциональной модели существующих бизнес процессов AS-IS и нового элемента модели TO-BE (рис. 12).

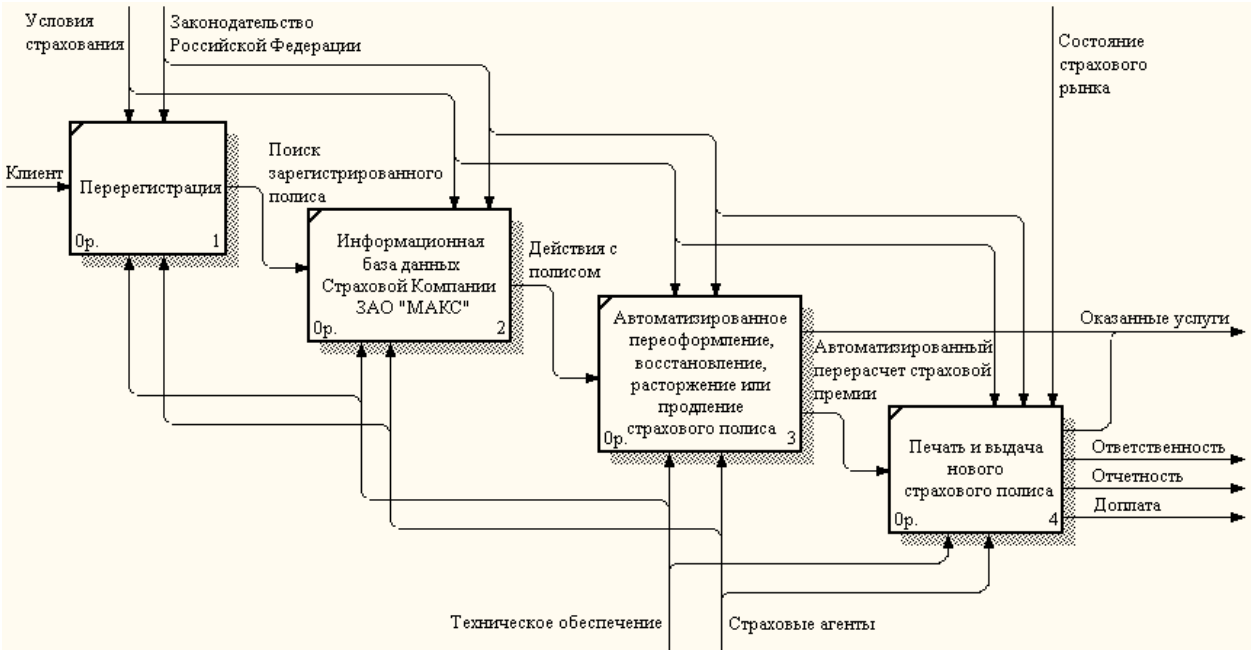


Рисунок 12 - Модель TO-BE бизнес процесса возможных действий со страховым полисом «Как-Будет»

В Модели ТО – ВЕ главным является «Информационная база данных» и «Автоматизированное переоформление, и перерасчет страховой премии», а также печать документа.

Основным достоинством модели ТО-ВЕ является расширенные возможности по получению статистических и аналитических данных.

5 ПРОЕКТИРОВАНИЕ КОНЦЕПТУАЛЬНОЙ МОДЕЛИ БАЗЫ ДАННЫХ

Схема данных состоит из следующих таблиц:

1) Транспортное средство, таблица содержит информацию о ТС;
2) Марка ТС, здесь перечислены все марки транспортных средств;
3) Страховые агенты, таблица содержит данные о страховых агентах, заключающих договора страхования;
4) Калькулятор, данная таблица предназначена для вычисления страховой суммы по договору ОСАГО;
5) Договор ОСАГО, таблица содержит все необходимые данные о страхователе, собственнике, ТС, периоде и сроке страхования, водителях, допущенных к управлению ТС, начисленной страховой сумме и о страховом агенте, который заключил договор;
6) Юридическое лицо, таблица содержит данные о юридическом лице;
7) Водители, допущенные к управлению ТС, таблица содержит информацию о водителях;
8) Страхователь ТС, таблица содержит данные о страхователе ТС;
9) Собственник ТС, таблица содержит данные о Собственнике ТС;
10) Республика, край, область, район, в данной таблице перечислены места проживания физических и юридических лиц;
11) Города, таблица содержит наиболее крупные города РФ;
12) Улицы, таблица содержит улицы Москвы и МО.

Главной таблицей в схеме данных является таблица «Договор ОСАГО». Ключевыми полями в каждой таблице являются:

1) Транспортное средство – «Код_ТС»;
2) Марка транспортного средства – «Код_Марка_ТС»;
3) Страховые агенты – «Код_Агенты»;
4) Калькулятор – «КалькуляторНомер»;
5) Договор ОСАГО – «Код_Договора_ОСАГО»;
6) Юридическое лицо – «Код_Юр_Лица»;
7) Водители, допущенные к управлению ТС – «Код_Водители»;

8) Республика, край, область, район – «Код_Республика_Край_Область_Район»;
9) Города – «Код_Города»;
10) Улицы – «Код_Улиц»;
11) Страхователь транспортного средства – «Код_Страхователя»;
12) Собственник транспортного средства – «Код_Собственника».

Функциональная связь имеется между всеми информационными объектами.

Таблица «Транспортное средство» и «Марка транспортного средства» имеет связь один-ко-многим (1:∞), так как любая модель ТС имеет только одну определенную марку, например марка ВАЗ имеет несколько моделей: 2104, 2105, 2110, 2121 т другие.

Таблица «Марка транспортного средства» и «Договор ОСАГО» имеет связь один-ко-многим (1:∞), так как в каждом договоре ОСАГО указывается только одна определенная марка транспортного средства.

Таблица «Страховые агенты» и «Договор ОСАГО» имеет связь один-ко-многим (1:∞), так как каждый договор ОСАГО заключает только один определенный страховой агент, а каждый страховой агент заключает большое количество договоров.

Таблица «Калькулятор» и «Договор ОСАГО» имеет связь один-ко-многим (1:∞), так как один калькулятор применяется для каждого договора один раз – рассчитывается сумма исходя из данных в договоре, а в каждом договоре различные данные. То есть один калькулятор рассчитывает множество договоров.

Таблица «Договор ОСАГО» и «Водители, допущенные к управлению ТС» имеет связь один-ко-многим (1:∞), так как один страховой договор может иметь много водителей, управляющих транспортным средством, а каждый водитель транспортного средства, имеет только один договор ОСАГО.

Таблица «Договор ОСАГО» и «Юридическое лицо» имеет связь один-ко-многим (1:∞), так как в каждом страховом договоре указывается один страхователь ТС и один собственник ТС, которым может быть либо физическое лицо, либо юридическое лицо. Если страхователем является

юридическое лицо, то его данные указываются в договоре. Юридическое лицо может имеет несколько транспортных средств, а значит и страховых договоров будет несколько.

Таблица «Договор ОСАГО» и «Страхователь ТС» имеет связь один-ко-многим (1:∞), так как в каждом страховом договоре указывается только один страхователь ТС, а каждый страхователь может застраховать несколько транспортных средств, то есть он может заключить несколько договоров ОСАГО.

Таблица «Договор ОСАГО» и «Собственник ТС» имеет связь один-ко-многим (1:∞), так как в каждом страховом договоре указывается только один собственник ТС, а у каждого собственника может быть несколько договоров ОСАГО, если он имеет в собственности несколько транспортных средств.

Таблица «Республика, край, область, район» имеет связь один-ко-многим (1:∞) с таблицами: «Юридическое лицо», «Водители, допущенные к управлению ТС», «Страхователь ТС» и «Собственник ТС», так как каждое лицо (физическое, юридическое, водитель, страхователь либо собственник) проживает в определенной республике, крае, области или районе.

Таблица «Города» имеет связь один-ко-многим (1:∞) с таблицами: «Юридическое лицо», «Водители, допущенные к управлению ТС», «Страхователь ТС» и «Собственник ТС», так как каждое лицо (физическое, юридическое, водитель, страхователь либо собственник) проживает в определенном городе. А в каждом городе живут миллионы людей.

Таблица «Улицы» имеет связь один-ко-многим (1:∞) с таблицами: «Юридическое лицо», «Водители, допущенные к управлению ТС», «Страхователь ТС» и «Собственник ТС», так как каждое лицо (физическое, юридическое, водитель, страхователь либо собственник) проживает на определенной улице (проспекте, проезде, переулке, бульваре). А на каждой улице живет большое количество человек.

ЗАКЛЮЧЕНИЕ

В ходе выполнения научной работы были получены следующие результаты:

1) изучена предметная область автоматизации;

2) рассмотрены основные операции, происходящие при работе страховой компании;

3) сформулирована необходимость автоматизации деятельности компании;

4) рассмотрены аналоги информационных систем автоматизации страхования;

5) произведено обоснование выбора программных сред для реализации информационной системы. Для разработки инфорационной системы была выбрана среда разработки – MS Access;

6) Спроектирована SADT-модель информационной системы;

7) Разработана модель данных, которая включает 21 сущность.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Гражданский кодекс Российской Федерации. Часть 2. – М.: ТК Велби, Изд. Проспект, 2005.

2. Федеральный закон «Об организации страхового дела в РФ» от 27.11.92 г № 4015-1.

3. Сплетухов Ю.А., Дюжиков Е.Ф. «Страхование» – М.:ИНФРА-М, 2014

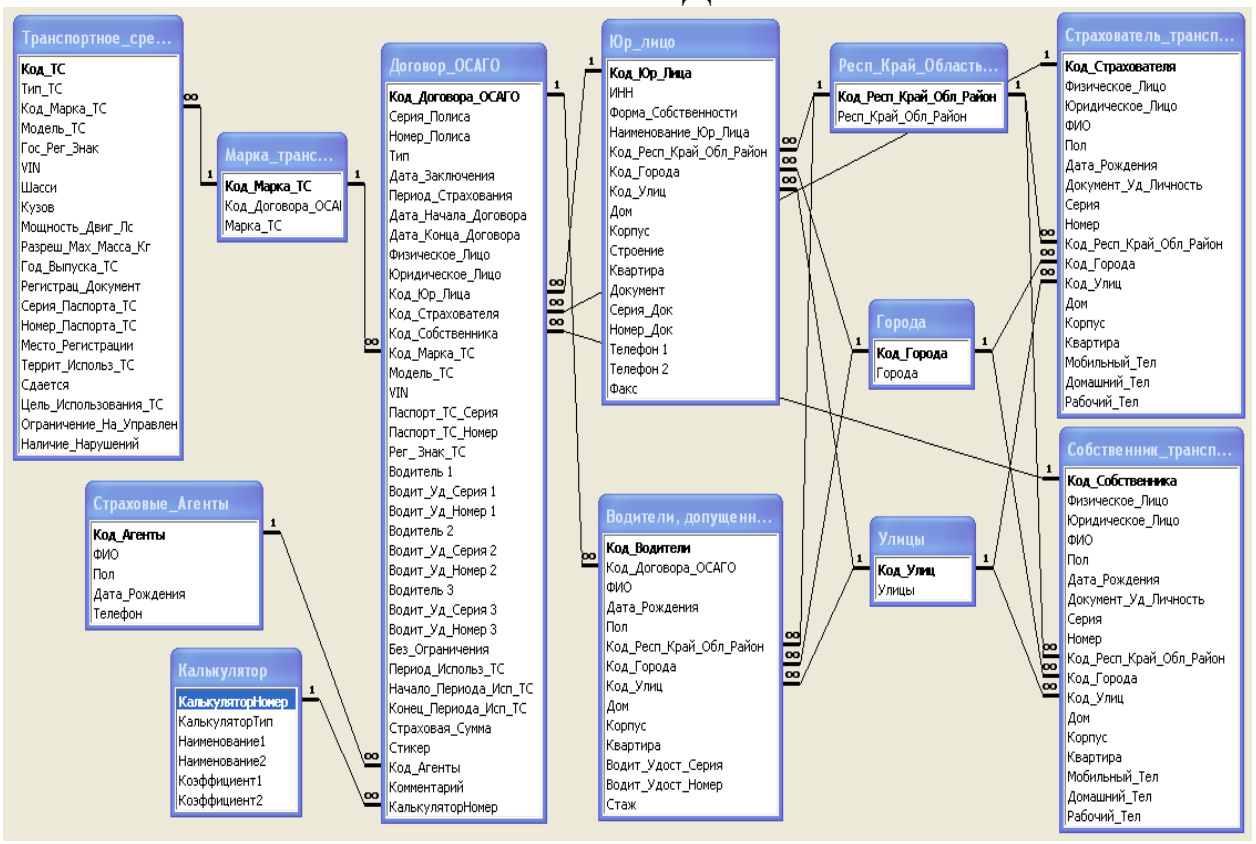
4. Федорова Т.А. «Страхование». Учебник. – 2-е изд., перераб. и доп. – М.: Экономистъ, 2014.

5. Рябикина В.И. «Страхование». Учеб. пособие.– М.: Экономистъ, 2013.

6. Скамай Л.Г., Мазурина Т.Ю. «Страховое дело» – М.:ИНФРА-М, 2012.

7. Ермасов С.В., Ермасова Н.Б.Страхование: Учеб. пособие для вузов. — М.: ЮНИТИ-ДАНА, 2014.

ПРИЛОЖЕНИЕ А
СХЕМА ДАННЫХ



Приложение 2
Пример 2 оформления отчета по научной работе

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)

Кафедра автоматизированных систем управления (АСУ)

СЕРВИСНОГО ОБСЛУЖИВАНИЯ КОМПЬЮТЕРНОЙ ТЕХНИКИ ООО «АПГРЕЙД»

Отчет по дисциплине «Научная работа 1»

Выполнил студент

группа: ___

ФИО

Руководитель

Доцент каф. АСУ,

канд. техн. наук

_____ А. И. Исакова

« » _____ 201_ г.

Томск 2017_____

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность темы исследования заключается в том, что учет использования компьютерного оборудования любого предприятия – одна из современных тенденций в автоматизации управления предприятием.

Рост объемов информации и, соответственно, документов, потребовал внедрения техники для своевременной обработки документов, а с появлением компьютеров – и самой информации. От качества работы такой службы существенно зависят уровень и качество самого управления.

В качестве объекта исследования выступает предприятие ООО «Апгрейд», а предметом исследования - бизнес-процессы предприятия.

Цель научной работы:

- рассмотреть аналоги информационных систем по учету ремонта и обслуживания компьютерной техники;
- выбрать среду разработки ИС;
- выполнить проектирование информационно-логической SADT-модели информационной системы по учету ремонта и обслуживания компьютерной техники ООО «Апгрейд»;
- спроектировать концептуальную модель базы данных.

1 ЦЕЛИ, ЗАДАЧИ И ФУНКЦИИ ИС

Цель в теоретическом аспекте - изучение особенностей учета процессов обслуживания и ремонта компьютерной техники в ООО «Апгрейд».

Цель в практическом аспекте - создание информационной системы учета процессов обслуживания и ремонта компьютерной техники ООО «Апгрейд».

Разработка информационной системы по учету процессов обслуживания и ремонта компьютерной техники должна решить следующие задачи:

1. Устранение ошибок неполноты информации, устранение дублирования информации.
2. Автоматизация расчета показателей.

3. Получение необходимых аналитических отчетов.

Решение перечисленных задач обеспечит следующее:

- увеличение упорядоченности и доступности необходимых документов;
- устранение, где это возможно и оправданно, печатных копий и рукописных документов;
- облегчение создания новых и поиска существующих документов;
- устранение ошибок при создании новых документов;
- повышение достоверности информации;
- сокращение временных затрат на обработку и получение необходимой информации.

Среди задач автоматизации отдельно следует выделить задачу автоматизации ведения базы комплектующих компьютерной и офисной техники, а также ведение базы данных заявок на обслуживание.

Накопление данных о заявках на обслуживание является очень важной частью автоматизации предприятия, которое направлено на обслуживание клиентов, по этой базе можно понять, что нужно клиентам в данный момент, даже если мы не можем в данный момент удовлетворить запрос клиента, информация о том, что поможет в развитии ассортимента услуг компании.

Автоматизация базы данных даст возможность быстрее и полнее реагировать на пожелания клиентов, производить сложные выборки и поиски

по многочисленным параметрам, что существенно превосходит сложившуюся в данный момент времени ситуацию, когда все данных хранятся в табличных документах формата Microsoft Excel.

Основная функция автоматизации - учёт процессов ремонта и обслуживания компьютерной и офисной техники.

При разработке ИС учета работы компании «Апгрейд» необходимо заложить в нее следующие возможности и функциональность:

– возможность просмотра списка обслуживаемого оборудования каждого типа;
– возможность формирование отчета по профилактическим работам и ремонту оборудования;
– возможность вывести на печать состав оборудования и комплектующих;
– просмотр, добавление, удаление данных о заявках;
– просмотр, добавление, удаление данных о сотрудниках.

Основными свойствами ИС для данного предприятия будут являться:

- Малая и легкодоступная. – т.к нет необходимости при текущих задачах в большой и сложной ИС, выполняющих большое количество задач неприменимых в данной сфере деятельности,
- Минимально необходимый уровень защищенность – на данном этапе у предприятия нет сформированных требований к защищенности ИС,
- Человек принимает активное участие в работе ИС – ввод данных, получение данных.
- Снижение воздействия случайных факторов таких как – ошибки технических устройств, ошибки персонала.

2 АНАЛОГИ ИС СЕРВИСНОГО ОБСЛУЖИВАНИЯ КОМПЬЮТЕРНОЙ ТЕХНИКИ

Программа IT Invent позволяет вести инвентаризационный учет компьютеров, программного обеспечения, комплектующих, расходных материалов и хозяйственного инвентаря. Учет ведется в рамках организаций и их филиалов. Используемая база данных - Microsoft Access или MS SQL Server. Программа позволяет получать доступ к необходимым параметрам и отслеживать изменения, производимые с каждой учетной конфигурационной единицей. С помощью программы IT Invent можно легко отследить все основные этапы жизни IT-оборудования в организации: закупку, поступление на склад, установку, перемещение, обслуживание и списание. Наличие модуля инвентаризации позволяет проводить ручную инвентаризацию оборудования с помощью сканера штрих-кодов [1].

Ключевые особенности программы:

- поддержка базы данных MS Access и MS SQL Server;
- многопользовательский режим работы - все филиалы работают с единой базой;
- возможность создания и настройки собственных дополнительных свойств различных типов;
- учет заказов поставщикам на все виды учетных единиц;
- учет выполнения работ любых видов внутри организации;
- поддержка работы со сканером штрих-кодов. Поиск записей в базе по штрих-коду;
- модуль инвентаризации с автоматической обработкой результатов;
- ведение истории изменений ключевых полей объектов учета;
- учет ремонтов и профилактических обслуживаний оборудования и компьютеров;
- логическое связывание программ и комплектующих с оборудованием;
- учет расходных материалов, комплектующих запчастей, канцелярии;
- учет инвентаря и хозяйственных принадлежностей;

- закрепление учетных единиц за сотрудниками организации. Акты приёма-передачи;

- ведение базы поставщиков, сервисных организаций и прочих контрагентов;

- гибкое разграничение прав доступа для пользователей системы;

- настройка E-Mail оповещений по действиям пользователей в программе;

- большое количество встроенных печатных форм и отчетов с возможностью их редактирования;

- импорт и просмотр данных напрямую из Active Directory;

- импорт данных из Excel/CSV файлов. Программа IT Invent позволяет вести инвентаризационный учет компьютеров и оборудования, принтеров, оргтехники, программного обеспечения, комплектующих, расходных материалов и любой другой техники а так же мебели и другого инвентаря.

Внешний вид программы представлен на рисунке 1.

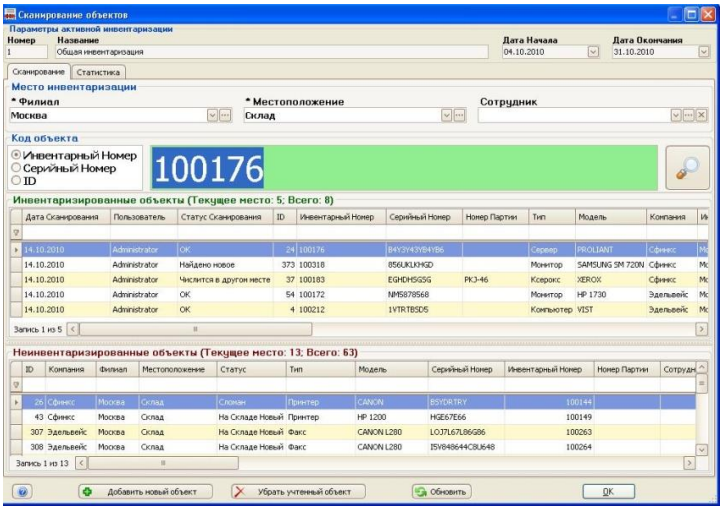


Рисунок 1 – Интерфейс программы «IT Invent»

Total Network Inventory 2 (TNI 2) - программа для инвентаризации компьютеров, оргтехники и сетевого оборудования [2].

Основными функциями программы являются:

- сканирование сети. Сканирование компьютеров на базе Windows, Mac OS X и Linux доступно только администратору системы. Можно сканировать отдельные узлы, диапазоны сетевых адресов или структуру Active Directory.

- учет компьютеров. В централизованной базе данных системы информация об одном компьютере занимает всего несколько десятков

килобайт. Есть возможность группировки устройств, добавления к ним комментариев и прикрепления дополнительной информации.

- формирование отчетов. Существует возможность формирования гибких отчетов по разным категориям данных: можно строить табличные отчеты, используя множество полей модели данных TNI 2. Отчеты можно копировать, экспортировать и распечатывать.

- учет ПО и лицензий. В распоряжении администраторов находится список всех приложений, найденных программой в сети. Таким образом, можно определить количество копий приложения и получить список компьютеров, на которых оно установлено. Также есть возможность фильтрации и группировки информации по программному обеспечению.

- планировщик сканирования. Автоматизация сбора данных подразумевает создание одноразовых отложенных задач или расписания для периодического сканирования компьютеров. Можно составить график сканирования сети, и таким образом постоянно получать свежую информацию.

- хранение информации о пользователях. Существует возможность занесения в базу данных пользователей компьютеров, пароли для разных устройств и протоколов. Можно следить за онлайн-статусом устройств в реальном времени.

Пример рабочего окна системы приведен на рисунке 2.

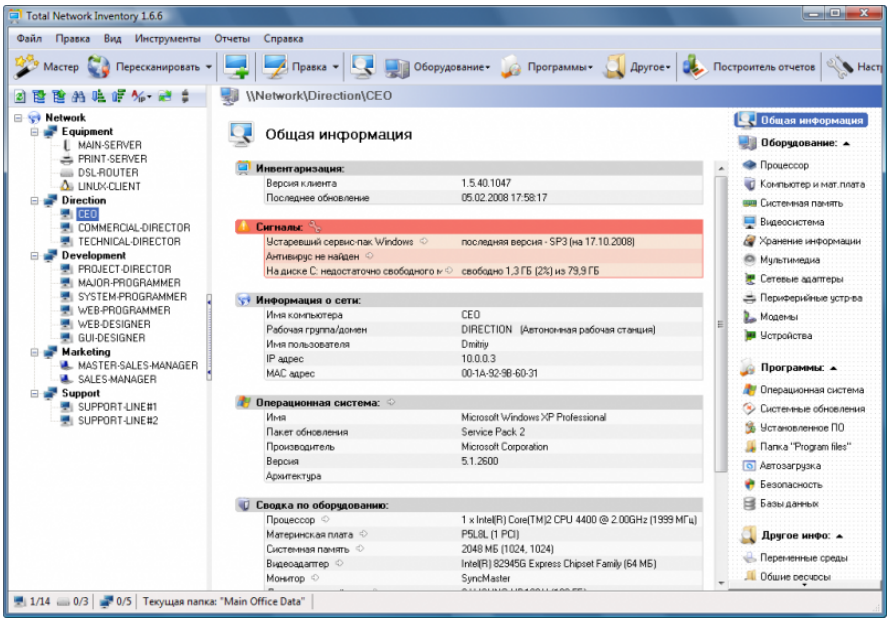


Рисунок 2 – Total Network Inventory 2

Hardware Inspector - программа для инвентаризации компьютеров и оргтехники, учёта расходных материалов, лицензий на ПО, заявок от пользователей, кроссировки сети и автоматизации деятельности сотрудников IT-подразделения [4].

Hardware Inspector позволяет всегда быть в курсе всей информации о вашем компьютерном парке, получать разнообразные отчеты, планировать обслуживание, ремонт и обновление.

Hardware Inspector решает следующие задачи:

- автоматизация инвентарного учета компьютерной техники и комплектующих, с возможностью хранения всей истории перемещений и обслуживания;
- защита компьютеров и комплектующих от хищения и подмены благодаря механизму ревизии рабочих мест;
- осуществление детального контроля за параметрами конфигурации компьютера, обеспечивающее свободу и оперативность действий по планированию, модернизации и перераспределении устройств;
- автоматизация отчетности перед материальной бухгалтерией.

Возможности программы:

Учёт отдельных комплектующих, а не просто описание параметров рабочих станций и текущего состояния параметров компьютера.

На каждое устройство заводится паспорт, в котором отражается информация о его покупке, технических параметрах, истории его перемещений по рабочим местам и обслуживания.

Отслеживание истории перемещения устройств, их ремонта, профилактики и инвентаризации.

Возможность не только ручного ввода данных, но и импорта информации из отчетов программ анализа конфигурации компьютеров AIDA, EVEREST, ASTRA и ASTRA32.

Учет лицензий на программное обеспечение.

На каждую лицензию заводится паспорт, в котором отражается информация о её покупке, параметрах, истории ее перемещений по рабочим местам.

Учёт заявок от пользователей.

Hardware Inspector обеспечивает хранение истории сообщений в заявке, прикрепляемые файлы и прочее. Возможность совместной работы с веб-интерфейсом.

Инвентаризация устройств с использованием штрих-кодов.

Механизм инвентаризации с использованием штрих-кодов показывает реальное наличие устройств на рабочих местах.

Гибкое разграничение доступа к данным.

Обширный перечень прав доступа к функциям и данным позволяет очень гибко настроить возможности каждого оператора базы данных.

Мощные механизмы поиска устройств, лицензий и прочего.

Большой набор настраиваемых отчетов, экспорт в различные форматы.

Поддержка многопользовательской работы с базой данных в сети.

Одна лицензия на программу позволяет совершать инсталляции без ограничений.

Пример рабочего окна системы приведен на рисунке 3.

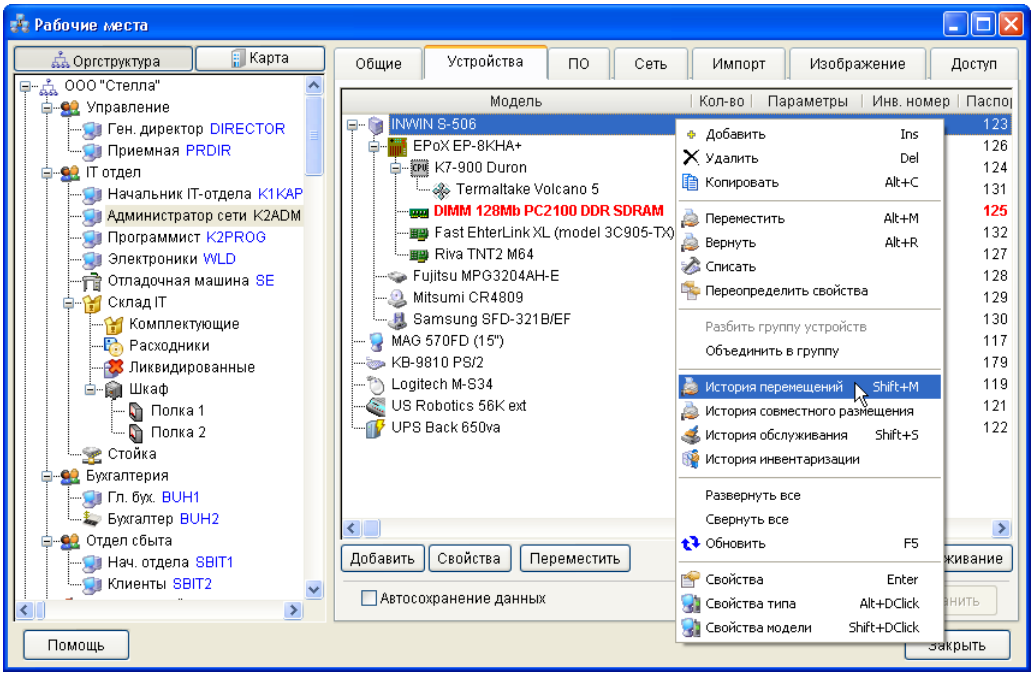


Рисунок 3 – Интерфейс программы «Hardware Inspector»

Таблица 1 – Сравнительная характеристика программ-аналогов учета заявок на обслуживание компьютерной техники

Критерий	«IT Invent»	«Hardware Inspector»	Total Network Inventory 2
Функцио-нальность	Много-функциональна	Многофункциональна	Многофункционал ьна
Интерфейс	Интуитивно понятный	Простой – интуитивно понятный	Максимально удобный
Дизайн	Хороший	Приемлемый	Стандартный
Удобство для пользователя	Удобно	Проста в использовании	Индивидуальные наборы настроек
Достоинства	Работает по сети	Работает по локальной сети Обновление 2 раза в месяц Одну лицензию можно установить на любое количество компьютеров, внутри одной локальной сети, одной организации	Действует бесплатная линия консультаций по электронной почте и ICQ, а в случае необходимости консультации по телефону.
Недостатки	Платная	Платная	Платная

3 ОБОСНОВАНИЕ ВЫБОРА СРЕДЫ РАЗРАБОТКИ ИС

Технология Delphi сделала разработку мощных приложений Windows быстрым процессом, доставляющим удовольствие там, где раньше требовалось большое количество человеческих усилий, поэтому в настоящее время многие приложения могут быть написаны одним человеком с использованием технологии Delphi.

Многооконный интерфейс обеспечивает полную передачу CASE-технологий в интегрированную систему поддержки работ по созданию прикладной системы на всех фазах жизненного цикла проектирования и системы.

Delphi имеет широкий набор функций, начиная от дизайнерских форм и заканчивая поддержкой всех популярных форматов баз данных. Среда устраняет необходимость программировать такие компоненты Windows, общего назначения, как метки, значки и даже диалоговые панели. Работая в Windows, можно увидеть те же «объекты» во многих разнообразных приложениях. Диалоговые окна (например, Выбрать файл и Сохранить файл) являются примерами многократно используемых компонентов, построенных прямо в Delphi, что позволяет применить эти компоненты к существующей задаче так, что они работают именно так, как необходимо, чтобы создавать приложения. Также здесь есть предопределенные визуальные и не визуальные объекты, включая кнопки, объекты данных, меню и диалоговые уже построенные панели. С помощью этих объектов можно, например, обеспечить ввод данных всего несколькими щелчками мыши, без необходимости программирования.

Часть, которая непосредственно связана с программированием интерфейса пользователя, называется системой визуального программирования.

Преимущества проектирования с помощью Delphi:

1) Исключает необходимость повторного ввода данных;

2) Обеспечивает согласованность проекта и его реализации;

3) Повышение производительности разработки и переносимость программ.

Визуальное программирование как бы добавляет новое измерение к созданию приложений, что позволяет представлять эти объекты на экране, чтобы выполнить программу.

Через визуальные инструменты разработчик приложения может работать с объектами, держа их перед глазами и получить результаты практически сразу. Способность видеть объекты так, как они появляются в ходе исполнения программы, снимает необходимость для множества ручных операций, что характерно для среды без визуальных средств - независимо от того, объектно-ориентированные они или нет. Как только объект находится в форме визуальной среды программирования, все его атрибуты сразу отображаются в виде кода, который соответствует объекту как единице, выполненной во время выполнения программы.

Размещение объектов в Delphi происходит в тесной взаимосвязи между объектами и реальным программным кодом. Объекты помещают в форму, код которой и соответствующих объектов автоматически сохраняются в исходном файле. Этот код компилируется, обеспечивая существенно более высокую производительность, чем визуальная среда, которая интерпретирует информацию лишь в ходе выполнения программы.

Три основные части разработки интерфейса следующие: проектирование панели, проектирование и представление диалоговых окон.

Программа DELPHI состоит из файла проекта (файл с расширением DPR) и несколько модулей (файлы с расширением PAS). Каждый из файлов расположен в отдельном модуле программы содержит программу на языке Object Pascal.

Файл проекта программы, также написанный в Object Pascal, обрабатывается компилятором. Эта программа создается автоматически при создании проекта DELPHI и содержит только несколько контрольных строк.

Модуль программы, который составляется независимо и содержит все необходимые компоненты в разделе описаний (типы, константы, переменные, процедуры и функции) и, если необходимо, исполняемые коды.

Самый популярный и широко используемый компонент в модуле DELPHI - форма.

Интерфейс этого модуля содержит объявление нового класса, и она автоматически обновляется при дополнении новыми элементами.

Delphi – структурированный, объектно-ориентированный язык программирования, диалект ObjectPascal. Начиная со среды разработки Delphi 7.0 [3], в официальных документах Borland стала использовать название Delphi для обозначения языка ObjectPascal. Начиная с 2007 года уже язык Delphi (производный от ObjectPascal) начал жить своей самостоятельной жизнью и претерпевал различные изменения, связанные с современными тенденциями (например, с развитием платформы .NET) развития языков программирования: появились classhelpers, перегрузки операторов и другое.

Изначально среда разработки Delphi была предназначена исключительно для разработки приложений MicrosoftWindows, затем был реализован вариант для платформ Linux (как Kylix), однако после выпуска в 2002 годуKylix 3 его разработка была прекращена, и вскоре было объявлено о поддержке Microsoft .NET.

Реализация среды разработки проектом Lazarus (FreePascal, компиляция в режиме совместимости с Delphi) позволяет использовать его для создания приложений на Delphi для таких платформ, как Linux, Mac OS X и Windows CE.

После проведения сравнительной характеристики программных аналогов можно сделать вывод, что необходима разработка собственной информационной системы.

Собственная разработка позволит:

- не нести затраты на покупку и сопровождение программного обеспечения у стороннего разработчика;

- оперативно вносить исправления в программный продукт;

- компания будет являться собственником исходного кода программы, что позволит его передавать вновь принятым специалистам для дальнейшей модернизации.

Рассмотрев различные СУБД по определённым характеристикам, определили, что наиболее подходящей для хранения данных является СУБД Access. При этом небольшие затраты на её внедрение и освоение окупятся

эффективной и надежной работой.

MS Access – СУБД от корпорации Microsoft. Данная СУБД поставляется вместе с пакетом программ MS Office. Крайней версией на данный момент является MS Access 2007. По умолчанию Access 2007 отключает все потенциально небезопасные программы или другие компоненты независимо от версии Access, в которой создавалась эта база данных. Данной СУБД также предоставляются шифрование или дешифрование базы данных с использованием пароля, упаковка, подпись и развертывание базы данных, работа с сертификатами и др.

У такой СУБД есть и недостатки. Защита информации от сбоев реализована сохранением данных после любого действия, что отрицательно влияет на быстродействие системы. Отсутствие встроенного компилятора EXE-файлов, что не позволяет правильно закончить технологический цикл разработки приложения без привлечения других средств программирования. СУБД использует библиотеки ОС Windows, что также отрицательно влияет на быстродействие и безопасность. Но, несмотря на недостатки, MS Access все равно остается достаточно популярной СУБД.

При работе с базой данных в Delphi была использована технология ADO -Microsoft ActiveX Data Objects, которая обеспечивает универсальный доступ к источникам данных из приложений БД. Такую возможность предоставляют функции набора интерфейсов, созданные на основе общей модели объектов COM и описанные в спецификации OLE DB.

Технология ADO и интерфейсы OLE DB обеспечивают для приложений единый способ доступа к источникам данных различных типов (рис. 3). Например, приложение, использующее ADO, может применять одинаково сложные операции и к данным, хранящимся на корпоративном сервере SQL, и к электронным таблицам, и локальным СУБД. Запрос SQL, направленный любому источнику данных через ADO, будет выполнен.

Провайдеры ADO обеспечивают соединение приложения, использующего данные через ADO, с источником данных (сервером SQL, локальной СУБД, файловой системой и т. д.). Для каждого типа хранилища данных должен существовать провайдер ADO.

Провайдер «знает» о местоположении хранилища данных и его содержании, умеет обращаться к данным с запросами и интерпретировать возвращаемую служебную информацию и результаты запросов с целью их передачи приложению.

Список установленных в данной операционной системе провайдеров доступен для выбора при установке соединения через компонент TADOConnection.

Соединение с данными СУБД Access при посредстве технологии ADO обеспечивает Microsoft Jet OLE DB Provider

Компонент TADOConnection

Компонент TADOConnection предназначен для управления соединением с объектами хранилища данных ADO. Он обеспечивает доступ к хранилищу данных компонентам ADO, инкапсулирующим набор данных.

Применение этого компонента дает разработчику ряд преимуществ:

– все компоненты доступа к данным ADO обращаются к хранилищу данных через одно соединение;

– возможность напрямую задать объект провайдера соединения;

– доступ к объекту соединения ADO;

– возможность выполнять команды ADO;

– выполнение транзакций;

– расширенное управление соединением при помощи методов-обработчиков событий.

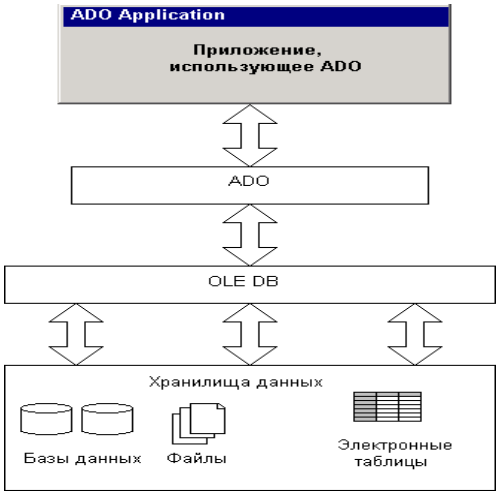


Рисунок 4 – Схема доступа к данным через ADO

На странице ADO Палитры компонентов Delphi, кроме компонентов соединения есть стандартные компоненты, инкапсулирующие набор данных и адаптированные для работы с хранилищем данных ADO (рис. 5). Это компоненты:

TADODataSet – универсальный набор данных;

TADOtable – таблица БД; TADOQuery – запрос SQL;

TADOstoredProc – хранимая процедура.

Как и положено для компонентов, инкапсулирующих набор данных, их общим предком является класс TDataSet, предоставляющий базовые функции управления набором данных.

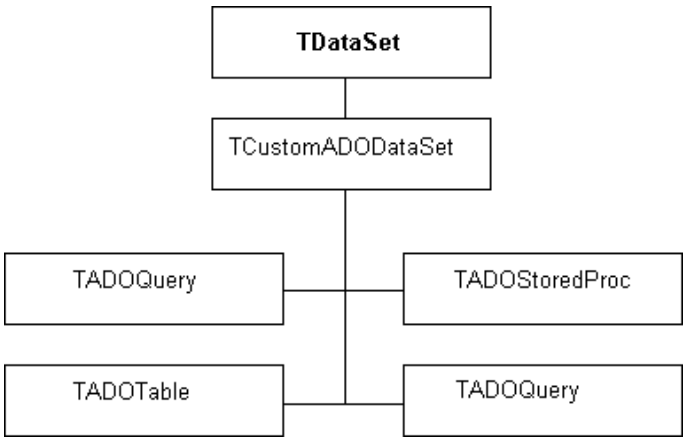


Рисунок 5 – Иерархия классов наборов данных ADO

Компонент TADOTable обеспечивает использование в приложениях Delphi таблиц БД, подключенных через провайдеры OLE DB. По своим функциональным возможностям и применению он подобен стандартному табличному компоненту. В основе компонента лежит использование команды ADO, но ее свойства настроены заранее и изменению не подлежат.

4 ФУНКЦИОНАЛЬНЫЙ СОСТАВ ИС ПО СИНТАКСИСУ МЕТОДОЛОГИИ SADT

Для наглядного отображения существующей на предприятии системы взаимоотношений с клиентами использовано CASE-средство верхнего уровня AllFusion Process Modeler (BPwin). Методология IDEF0 (функциональная модель) предписывает построение иерархической системы диаграмм – единичных описаний системы.

С помощью функционального моделирования (нотация IDEF0), можно провести систематический анализ процессов и систем, сосредоточившись на регулярно решаемых задачах (функциях), свидетельствующих об их правильном выполнении, показателях, необходимых для этого ресурсах, результатах и исходных материалах (сырье).

Рассмотрим основные процессы сервисного обслуживания (рис.6).

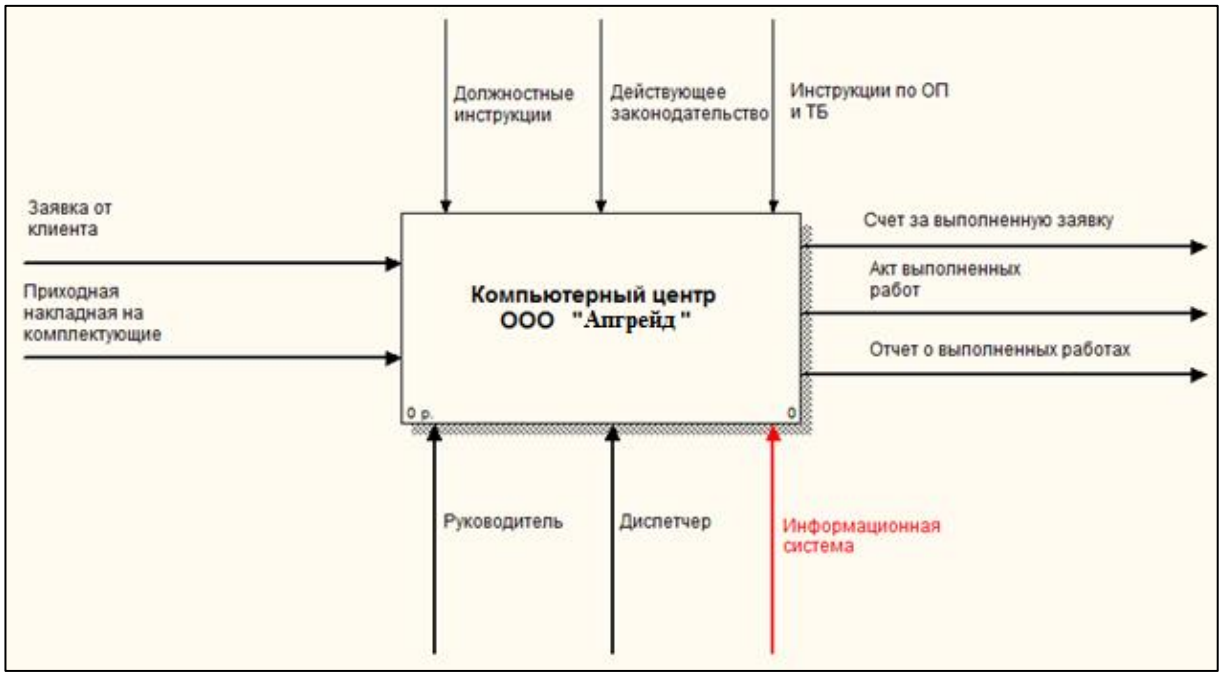


Рисунок 6 – Контекстная диаграмма учета оформления заявок на ремонт и обслуживание компьютерной техники

На рисунке 7 приведена диаграмма декомпозиции контекстной диаграммы.

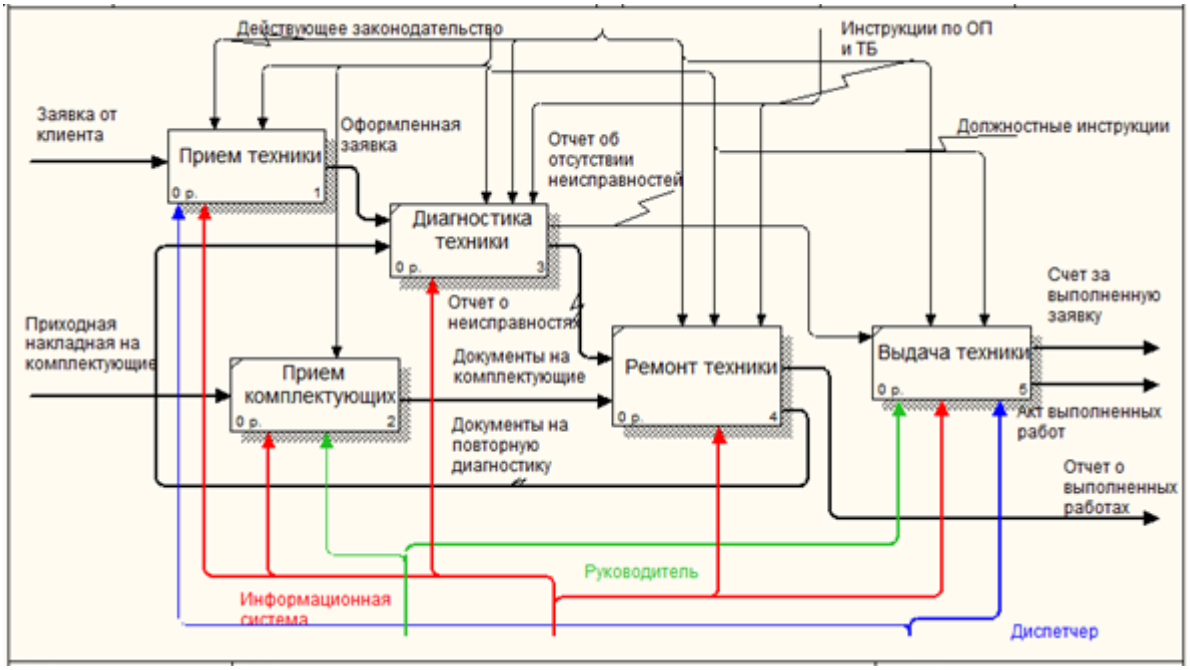


Рисунок 7 – Диаграмма декомпозиции ИС учета оформления заявок на ремонт компьютерной техники

На рисунке 8. приведена диаграмма декомпозиции работы «Прием техники».

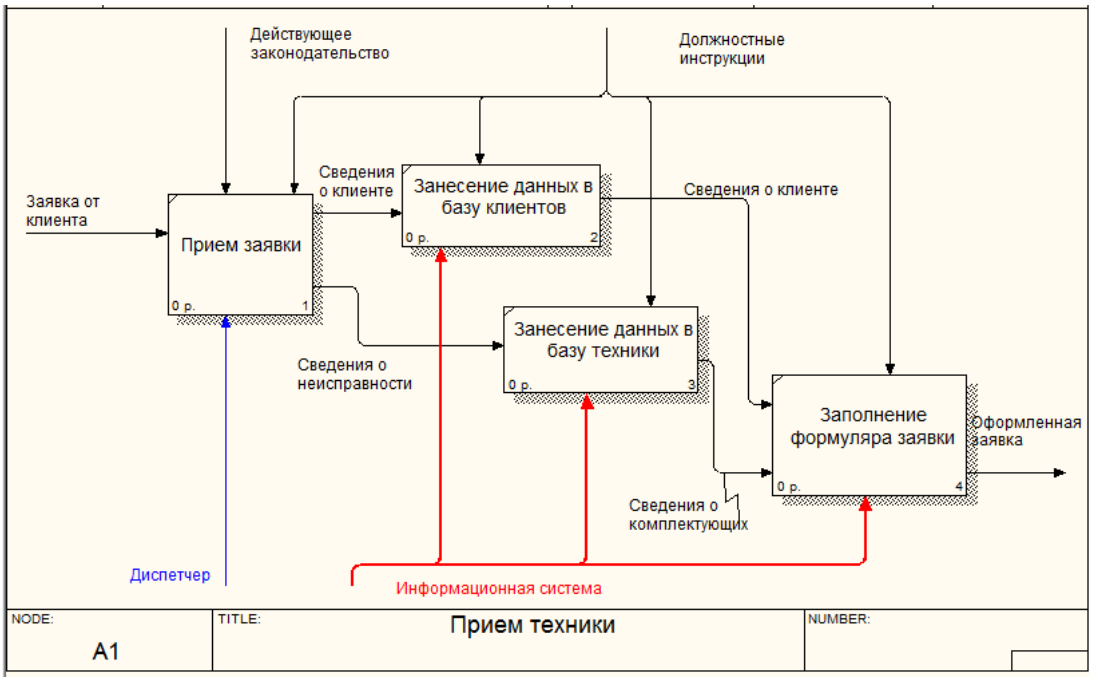


Рисунок 8 – Диаграмма декомпозиции работы «Прием техники»

В декомпозиции работы «Прием техники» определено четыре внутренние работы:

1) Прием заявки – процесс приема заявки от клиента в письменном виде.

2) Занесение данных в базу клиентов – телефонный звонок пользователю-клиенту о дате проведения профилактики.

3) Занесение данных в базу техники телефонный звонок пользователю-клиенту о дате проведения профилактики.

4) Заполнение формуляра заявки - .

Принимаем заявку на ремонт и заносим ее в базу. Собираем сведения о компьютере и месте неисправности. Согласовываем порядок действий по устранению неисправностей и подбираем сотрудника для выполнения этих работ. Запускаем пакет диагностических программ для выявления неисправностей. Подбираем необходимые комплектующие для устранения неисправностей. Затем работник с необходимой квалификацией выполняет все работы.

После выполнения ремонта отправляем данные в базу данных материальных ценностей по заказу для составления Счета-фактуры на использованные комплектующие и в базу данных работ по заказам о проделанной работе для составления Акта выполненных работ.

5 ПРОЕКТИРОВАНИЕ КОНЦЕПТУАЛЬНОЙ МОДЕЛИ БАЗЫ ДАННЫХ

Построение модели данных предполагает определение сущностей и атрибутов, т. е. необходимо определить, какая информация будет храниться в конкретной сущности или атрибуте.

Сбор необходимой информации, ее анализ и структурирование помогли создать модель предметной области. Модель включает в себя объекты, информация о которых храниться в БД.

В программе ERwin была создана логическая модель, которая содержит основные сущности и показывает связи между ними.

Инфологическая модель для базы данных «Учет ремонта и обслуживания» проектировалась, как модель «Сущность-связь» и состоит из тринадцати сущностей:

1) Заказы.
2) ЗаказыПоставщику.
3) Клиенты.
4) МатЦенности.
5) МатЦенностиПоЗаказу.
6) Менеджеры.
7) Прейскурант.
8) Работники.
9) РаботыПоЗаказу.
10) Реквизиты.
11) СоставЗаказаПоставщику.
12) Специализации.
13) Статусы.

Сущность – это класс однотипных объектов. Каждая из сущностей имеет свой набор атрибутов, идентифицирующих данную сущность. Словарь сущностей представлен в таблице 2 [6].

Таблица 2 – Сущности и их определения

Имя сущности	**Определение**
Заказы	Данные о заказах клиентов
ЗаказыПоставщику	Данные о заказе и кто заказал
Клиенты	Данные о клиентах, контактная информация
МатЦенности.	Список комплектующих
МатЦенностиПоЗаказу	Список комплектующих, использованных в заказах
Менеджеры	Данные о менеджерах, контактная информация
Прейскурант	Справочник услуг, единицы измерения, цена
Работники	Данные о работниках, контактная информация, логины и пароли
РаботыПоЗаказу	Список работ, использованных в заказах
Реквизиты	Данные о предприятии для формирования отчетных документов
СоставЗаказаПоставщику	Данные о количестве и стоимости товара включенного в поставки
Специализации	Специализации сотрудников
Статусы	Статус заказа, информация о процессе выполнения

Связи между сущностями отражены в таблице 3.

Таблица 3 – Связи между сущностями

Сущность1	Сущность2	Связь
Заказы	МатЦенностиПоЗаказу, РаботыПоЗаказу	Один-ко-многим
ЗаказыПоставщику	СоставЗаказаПоставщику	Один-ко-многим
Клиенты	Заказы	Один-ко-многим
МатЦенности.	МатЦенностиПоЗаказу, СоставЗаказаПоставщику	Один-ко-многим
Прейскурант	РаботыПоЗаказу	Один-ко-многим
Работники	РаботыПоЗаказу, Заказы, ЗаказыПоставщику	Один-ко-многим
Специализации	Работники	Один-ко-многим
Статусы	Заказы	Один-ко-многим

В результате была сформирована модель предметной области на уровне ER представления данных (рисунок 9).

Рисунок 9 – Диаграмма ER-уровня модели

Отношения, разработанные на стадии формирования инфологической модели данных, дополняются внешними и первичными ключами, образуя модель данных КВ-уровня (рис. 10).

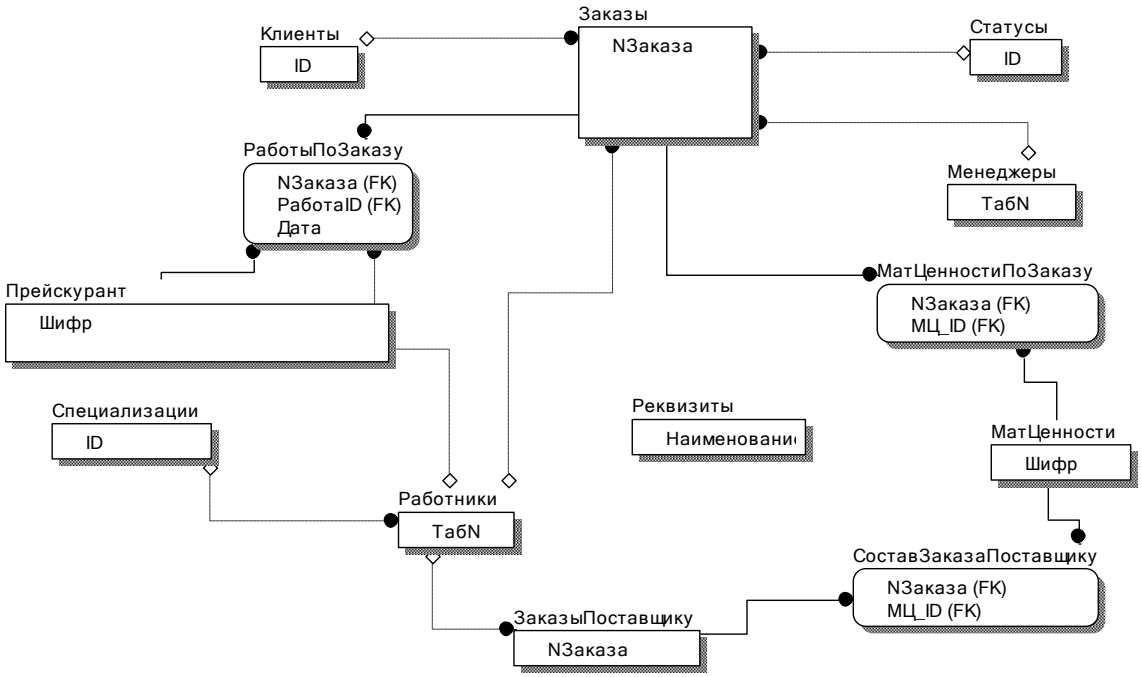


Рисунок 10 – Диаграмма КВ-уровня модели

Диаграмма FA-уровня модели содержит все атрибуты сущностей – ключевые и неключевые (рис. 11).

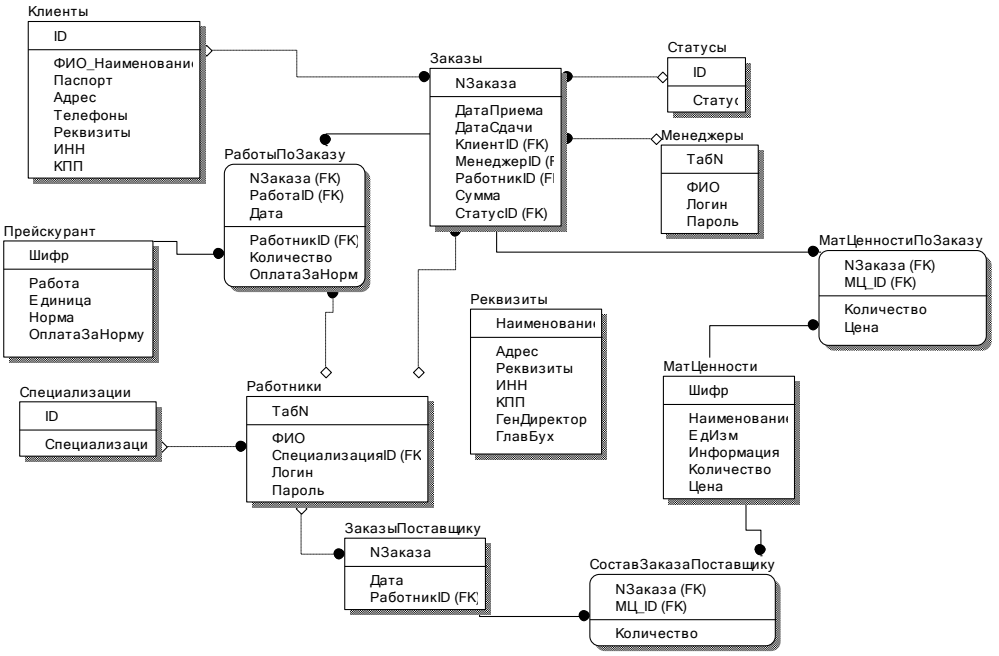


Рисунок 11 – Диаграмма FA-уровня модели на логическом уровне представления

Диаграмма FA-уровня модели на физическом уровне представления дополняется типами данных (рис. 12).

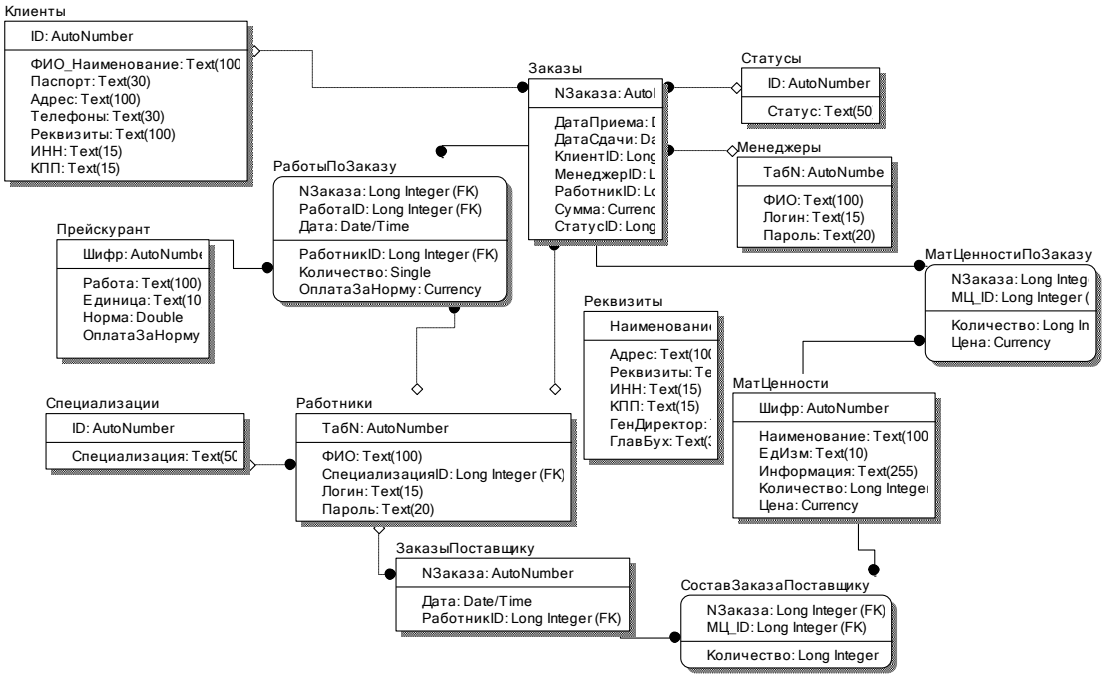


Рисунок 12 – Диаграмма FA-уровня модели на физическом уровне представления

ЗАКЛЮЧЕНИЕ

В ходе выполнения научной работы были получены следующие результаты:

- изучена предметная область автоматизации;
- рассмотрены основные операции учета ремонта и обслуживания компьютерной техники в ООО «Апгрейд»;
- сформулирована необходимость автоматизации учета ремонта и обслуживания компьютерной техники в ООО «Апгрейд»;
- рассмотрены аналоги информационных систем учета ремонта и обслуживания компьютерной техники

Выявлены следующие общие недостатки:

- недостаток необходимых для ООО «Апгрейд» функций; избыточность стандартных функций;

Была спроектирована автоматизированная информационная система «Учет ремонта и обслуживания компьютерной и офисной техники», которая позволяет автоматизировать обработку информации о комплектующих и заказах на комплектацию.

Информационная система обеспечивает работу с данными: ввод, удаление, поиск, печать ответственного лица или пользователя, выборку всех данных о комплектации и также существует возможность просмотра и печати отчетов в виде листов Excel или диаграмм.

Также была спроектирована SADT-модель информационной системы и разработана концептуальная модель (ER-KB-FA-уровней), которая включает 13 сущностей.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Абрамов Г.В., Медведкова И.Е., Коробова Л.А. Проектирование информационных систем – Воронеж: ВГУИТ, 2012 г. - 172 с.
2. Алешин, Л.И. Информационные технологии: Учебное пособие / Л.И. Алешин. - М.: Маркет ДС, 2011. - 384 с.
3. Бабаева З.В. Информационные системы в экономике – М.: МИИТ, 2011. – 29 с.
4. Базы данных / И.Е. Медведкова, Ю.В.Бугаев, С.В.Чикунов – Воронеж: ВГУИТ, 2014. – 105 с.
5. Бекаревич Ю. Самоучитель Microsoft Access 2013 / Ю. Бекаревич, Н. Пушкина – С-Пб.: БХВ-Петербург, 2014. – 465 с.
6. Венделева, М.А. Информационные технологии в управлении: Учебное пособие для бакалавров / М.А. Венделева, Ю.В. Вертакова. - М.: Юрайт, 2013. - 462 с.
7. Голицына, О.Л. Информационные технологии: Учебник / О.Л. Голицына, Н.В. Максимов, Т.Л. Партыка, И.И. Попов. - М.: Форум, ИНФРА-М, 2013. - 608 с.
8. Илюшечкин В. М.Основы использования и проектирования баз данных / Илюшечкин В. М. - М. :Юрайт, 2011. - 213 с.
9. Киселев, Г.М. Информационные технологии в экономике и управлении (эффективная работа в MS Office 2007): Учебное пособие / Г.М. Киселев, Р.В. Бочкова, В.И. Сафонов. - М.: Дашков и К, 2013. - 272 с.
10. Клочкова, Е. Н. Экономика предприятия / Е. Н. Клочкова, В. И. Кузнецов, Т. Е. Платонова. - М.: Юрайт, 2014. - 448 с.
11. Леонтьев В.П. Работа на компьютере 2014. Windows 8.1. Office 2013. Office 365 – М.: Олма Медиа Групп, 2014. – 643 с.
12. Майер-Шенбергер В. Большие данные. Революция, которая изменит то, как мы живем, работаем и мыслим /В. Майер-Шенбергер, К. Кукьер – М.: Изд-во Манн, Иванов и Фербер, 2014. – 222 с.
13. Нестеров С.А. Базы данных. – СПб.: Изд-во Полит. Ун-та, 2013. – 150 с.
14. http://www.interface.ru/logworks/sp2.htm (дата обращения: 22.11.2017)