МИНОБРНАУКИ РОССИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ
УНИВЕРСИТЕТ
ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»

В. А. Зеленский

ПРОЕКТИРОВАНИЕ СЛОЖНЫХ СИСТЕМ

Электронное учебное пособие

Самара – 2012

УДК 621.396.4
З – 486

Автор: **Зеленский Владимир Анатольевич**

Зеленский, В. А. Проектирование сложных систем [Электронный ресурс]: электрон. учеб. Пособие / В. А. Зеленский; Минобрнауки России. Самар. гос. аэрокосм. ун-т им. С.П.Королева (нац. исслед. ун-т). – Электрон. текстовые и граф. дан. (720 Кбайт). – Самара 2012, 1 эл. опт. диск (CD-RW).

Режим доступа: http: // rtfmoodle.ssau.ru

В учебном пособии рассмотрены вопросы исследования и проектирования систем, имеющих сложную структуру. Функционирование таких систем проанализировано с точки зрения передачи, приема и преобразования информации между подсистемами. Предложены методы и инструментальные средства проектирования сложных систем.

Учебное пособие «Проектирование сложных систем» рекомендуется магистрантам РТФ направления 211000.68 «Конструирование и технология электронных средств», изучающим дисциплину «Проектирование сложных систем» в 9 семестре.

Электронное учебное пособие разработано на кафедре конструирования и производства радиоэлектронных средств радиотехнического факультета Самарского государственного аэрокосмического университета.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Единого, общепринятого определения понятия «система» не существует. Системой обычно называют множество элементов, находящихся в отношениях и связях друг с другом, которые образуют определённую целостность, единство [1]. Добавим к этому определению, что такое единство позволяет решать определенные задачи. Существует также философский аспект образования системы как результат перехода количества входящих в неё элементов в новое качество.

Сложная система — система, состоящая из множества взаимодействующих составляющих (подсистем), вследствие чего сложная система приобретает новые свойства. Таким образом, если в системе можно выделить ряд входящих в неё составляющих (подсистем), её можно назвать сложной. Заметим, что в соответствии с этим подходом практически любая техническая система может считаться сложной.

С понятием системы тесно связано понятие информации. Характерно, что единого определения понятия «информация», как научного термина, в настоящее время также не существует. Сложность данных понятий раскрывается через известные уже студентам определения энтропии, сигнала, иерархии, декомпозиции. Приведены примеры из области функционирования сложных технических, информационных и экономических систем, способствующие лучшему пониманию изложенного в пособии материала.

Учебное пособие предназначено в первую очередь для магистрантов, обучающихся техническим дисциплинам, но может быть полезно также аспирантам и специалистам, работающим в области разработки и исследования сложных систем.

1. ОБЩИЕ СВЕДЕНИЯ О СЛОЖНЫХ СИСТЕМАХ

1.1 Информация, энтропия, сигналы

Понятие системы тесно связано с понятием энтропии. Энтропией в естественных науках называется мера беспорядка системы, состоящей из многих элементов.

Энтропия – это мера неопределенности какого-либо опыта, который может иметь разные исходы.

Пусть Q – некоторый источник информации; N – количество событий, которые могут произойти; p_i – вероятность осуществления i-го события, причем $\sum_{i=1}^{N} p_i = 1$. В этом случае, величина энтропии рассчитывается по формуле:

$$H(Q) = -\sum_{i=1}^{N} p_i \log_2 p_i .$$

Основные свойства энтропии:

1. Энтропия есть величина неотрицательнная: $H \geq 0$.

2. Энтропия есть величина ограниченная снизу: $\lim_{p_i \to 0} p_i \log_2 p_i = 0$.

3. Энтропия равна нулю только тогда, когда одна из $p_i = 1$, а другие вероятности равны нулю.

4. Энтропия достигает своего максимального значения тогда, когда все события равновероятны.

5. Энтропия нескольких независимых источников равна сумме энтропии этих источников.

Принято говорить, что один объект отражает другой или содержит информацию о другом объекте, если состояние одного объекта находится в соответствии с состоянием другого объекта.

Информация есть свойство материи, состоящее в том, что в результате взаимодействия объектов между их состояниями устанавливается определенное соответствие. Чем сильнее выражено это соответствие, тем точнее состояние

одного объекта отражает состояние другого объекта, т.е. тем больше информации один объект содержит о другом.

В настоящее время информация рассматривается как фундаментальное свойство материи.

Сигнал – материальный носитель информации, т.е. средство переноса информации в пространстве и во времени. Один и тот же объект может выступать в роли разных сигналов. В качестве сигналов используются не сами объекты, а их состояния.

Сообщение – совокупность знаков или периодических сигналов, содержащих информацию. Одно и тоже сообщение ведет к разной интерпретации, т.е. к разной информации.

Количество информации I – это мера уменьшения энтропии объекта после совершения некоторого события. Пусть H_0 – энтропия объекта до совершения события, H_1 – энтропия объекта после совершения события, таким образом $H_0 > H_1 \Rightarrow I = H_0 - H_1$.

Единица информации – 1 бит (bit – BInery digiT).

Один бит – это количество информации, получаемое при осуществлении одного из двух равновероятных событий.

Основные производные единицы количества информации: 1 байт = 8 бит; 1Кбайт = 1024 байт; 1 Мбайт = 1024 Кбайт; 1 Гбайт = 1024 Мбайт; 1 Тбайт = 1024 Гбайт.

1.2 Определение сложной системы

Существует множество определений термина система [1, 2].

Первое определение системы: система есть средство достижения цели.

Другое определение: система – это совокупность взаимосвязанных элементов, обособленное от среды и взаимодействующая с ней как единое целое.

Можно дать и обобщенное определение: система – это конечное множество функциональных элементов и отношений между ними, выделяемое из среды

в соответствии с определенной целью в рамках определенного временного интервала.

Элемент системы – некоторый объект, обладающий рядом важных свойств и реализующий в системе определенный закон функционирования, причем, внутренняя структура данного объекта не рассматривается.

Подсистема – это относительно независимая часть системы, которая обладает всеми свойствами системы и, в частности, имеет свою подцель, на достижение которой эта подсистема и ориентирована.

Если же части системы не обладают свойствами системы, а представляют собой просто совокупности однородных элементов, то такие части принято называть компонентами.

Под свойством понимают сторону объекта, обуславливающую его отличие от других объектов или сходство с ними и управляющуюся при взаимодействии с другими объектами.

Под управлением в самом общем виде будем понимать процесс формирования целенаправленного поведения системы посредством информационных воздействий, вырабатываемых человеком или устройством [2].

Система с управлением включает три подсистемы: управляющую систему (УС), объект управления (ОУ) и систему связи (СС). Управляющая система и система связи образуют систему управления (СУ). Основным элементом организационно-технической системы управления является лицо, принимающее решение (ЛПР).

Основными группами функций СУ являются [2]:

1. Функции принятия решений. Эти функции выражаются в создании новой информации в ходе анализа, планирования и оперативного управления. Это связано с преобразованием информации о состоянии ОУ и внешней среды в управляющую информацию при решении задач и выполнении аналитических расчетов, проводимых ЛПР при порождении и выборе альтернатив. Эта группа функций является главной, поскольку обеспечивает выработку информацион-

ных воздействий по удержанию в существующем положении или при переводе системы в новое состояние.

2. Функции обработки информации. Они охватывают учет, контроль, хранение, поиск, отображение, копирование информации. Эта группа функций по обработке информации не изменяет смысл самой информации (рутинные функции).

3. Функции обмена информацией. Эта группа функций связана с доведением выработанных воздействий ЛПР до ОУ.

Совокупность средств информационной техники и людей, объединенных для достижения определенных целей, в том числе и для управления, образуют информационную систему.

Информационная система (ИС) – это организационно-техническая система, использующая информационные технологии в целях обучения, информационно-аналитического обеспечения человеческой деятельности и процессов управления [2].

1.3 История развития сложных систем

Докомпьютерная эпоха (до 50-х гг. XX века). До появления компьютеров ИС представляли собой различные архивы, картотеки и бухгалтерские книги. На карточке информация хранилась в упорядоченном виде, так же строго поддерживалась структурированная система закладок и каталогов для облегчения поиска информации. В больших информационных архивах поиск занимал много времени, а для поддержания строгой иерархической системы требовались значительные усилия.

Появление первых компьютеров (50-е и 60-е гг.). Появление первых компьютеров не оказало значительного влияния на мир ИС, так как первые компьютеры в основном занимались вычислениями и использовались преимущественно в военных целях. Тем не менее, у людей появилась возможность в эффективной автоматизированной обработки информации.

Первые коммерческие ИС (70-е гг.). По мере развития вычислительной техники в это время стали появляться первые коммерческие ИС. Они принадлежали крупным корпорациям и были довольно малочисленны. Это было связано с внушительной стоимостью как вычислительной техники, так и стоимостью создания самой ИС. Типичная ИС того времени работала на большой ЭВМ, к которой подсоединялись множество терминалов ввода данных. С терминалов посылали запросы на информацию и получали ответ на свои запросы. Большая ЭВМ обрабатывала запросы пакетами, и пользователям приходилось ожидать, пока не будет сформирован полный пакет для обработки. Информация вводилась и выводилась в виде текстовой строки, что было неудобно для ее восприятия.

Информация централизованно хранилась в большой ЭВМ в виде базы данных. Каждая ИС требовала написания специализированного программного обеспечения и содержания штата высококвалифицированных специалистов для ее обслуживания. Первые ИС нашли свое применение в банковской сфере, на биржах и в крупных авиакомпаниях. Несмотря на огромную стоимость и неудобство использования, ИС доказали эффективность своего применения.

Появление персональных ЭВМ (с начала 80-х гг. двадцатого века)**.** Как было уже сказано, первые ИС обслуживали интересы крупных корпораций, причем общекорпоративные интересы или интересы руководства. Для конечного пользователя (особенно с увеличением количества подключенных терминалов) работа с ИС была не удобной и не помогала решать часть его задач, выбивавшихся из общекорпоративных шаблонов. Поэтому в начале 80-х годов появляются и быстро набирают популярность персональные ЭВМ. Они стоят гораздо дешевле большой ЭВМ и работают со стандартным программным обеспечением (ПО). В набор такого ПО входили текстовый редактор, электронные таблицы и персональная база данных, которую владелец мог настроить под свои нужды. Эти ЭВМ могли себе позволить уже значительно больший круг покупателей, что в дальнейшем привело к их стремительному развитию. Этот период можно назвать периодом популяризации автоматизированной обработки ин-

формации, и временем, когда компьютеры появились практически в каждом офисе.

Локальные вычислительные сети (со второй половины 80-х гг. двадцатого века). Когда компьютеры появились на каждом рабочем месте, остро встала проблема обмена электронной информацией между сотрудниками одной организации. Решением этой проблемы стало объединение компьютеров одной организации в локальную вычислительную сеть (ЛВС). Сеть позволила сотрудникам обмениваться информацией не вставая с рабочего места и совместно использовать внешнее оборудование. Дальнейшее развитие локальных сетей стали локальные сети с выделенным сервером – отдельным компьютером, предоставляющим какой-либо ресурс, в том числе и информацию, для всех участников сети. Такие компьютеры уже не использовались для обычных целей, они целиком специализировались на предоставлении ресурсов для совместного использования.

Распределенные информационные системы (конец 90-х годов). Развитие локальных сетей с выделенным сервером привело к модели – один мощный сервер и несколько компьютеров-клиентов. Увеличение количества компьютеров-клиентов неизбежно вело к увеличению времени ожидания доступа к совместному ресурсу и конфликтов доступа между компьютерами-клиентами. Решением данной проблемы стало распределение ИС. Данное распределение означало разделение задачи серверного обслуживания на части между иерархической группой серверов, т.е. сервер нижнего уровня обслуживает несколько обычных компьютеров, сервера нижнего уровня обслуживаются сервером более высокого уровня и т.д. Таким образом, один сервер поддерживает небольшое количество клиентов, для обслуживания которых не требуется огромная мощность. При увеличении количества компьютеров-клиентов добавляется еще один сервер на соответствующий уровень, и нагрузка на каждый отдельный сервер этого уровня не увеличивается. Подобная организация ИС позволяет значительно удешевлять их стоимость и сравнительно просто наращивать мощность системы.

Глобализация информационных систем. К концу второго тысячелетия сформировалась тенденция глобализации информационных процессов. Этому способствовало широкое распространение глобальной сети Internet. Процессы глобализации обмена информацией увеличивают интенсивность, оперативность информационного обмена, обеспечивают мобильность сотрудников организации, позволяя работать в любое время суток с любого расстояния от офиса. Основными направлениями развития ИС стали разработка единого способа объединения информации разных форматов и совершенствование систем защиты информации от несанкционированного доступа.

1.4 Особенности современных сложных систем

Комплексный подход. Современные ИС характеризуются понятием комплексности. Это подразумевает целостный подход к автоматизации технологических процессов в организации. Если раньше в каждом отделе была своя, маленькая ИС, то сейчас вся организация работает в единой ИС. Такое построение ИС позволяет использовать информацию одного отдела в работе других подразделений, получать сводную информацию по всей организации и повышать скорость информационных потоков внутри организации.

Оперативность. В современных условиях очень важным параметром в работе организации становится скорость обработки и доступность информации. Поэтому современные ИС проектируются таким образом, чтобы пользователи могли получать максимум информации, доступной на текущий момент. Особое внимание уделяется оперативности информации, то есть процессам получения самой «свежей» информации, так как от этого во многом зависит эффективность принимаемых решений (например, в системах электронной торговли, в навигационных системах и т.д.).

Гибкость. Современные организации вынуждены работать в постоянно изменяющихся условиях, требующих изменений в структуре организации или методах ее работы. Поэтому важным параметром ИС является ее гибкость, т.е.

способность быстро менять конфигурацию или функциональный набор. Наиболее распространенными способами реализации этого принципа являются модульность системы (при необходимости различные функциональные модули могут отключаться или подключаться к системе) и система настроек (т.е. заложена возможность коррекции основных параметров).

Распределенная ИС. Распределенная ИС подразумевает многоуровневую структуру и наличие иерархии серверов [6] (см. раздел 1.3).

Взаимосвязь с другими ИС. Современная организация работает в условиях тесного взаимодействия и интенсивного информационного обмена с другими организациями, поэтому важное значение имеет способность «нашей» ИС взаимодействовать с ИС других организаций. В современных ИС должна быть предусмотрена хотя бы возможность импортировать и экспортировать массивы данных в общепринятых форматах обмена данными (текстовые файлы или электронные таблицы). Также необходима возможность взаимодействия системы с другими программными пакетами в рамках одной организации.

Доступность информации извне. В последнее время значительно увеличилась степень информационной открытости организаций для внешних потребителей (партнеров, клиентов). Современная ИС должна иметь механизмы публикации своих данных в Интернет для внешних пользователей (прайс-листы, перечень услуг, объявления). Естественно, не все данные организация делает общедоступными, поэтому большое внимание уделяется защите ИС от несанкционированного доступа и правильной организации уровней доступа к информации.

1.5 Структура и состав информационной системы

Структура ИС. Организационную структуру любого предприятия условно можно разделить на три части: руководство головного отделения, аналитический отдел и территориальные отделения. В головном отделении находится сервер базы данных (БД), куда стекается информация из всех территориальных

отделений. Аналитический отдел работает со своим, отдельным сервером, на котором выполняются задачи этого отдела. Такая структура позволит разгрузить основной сервер от трудоемких задач аналитического отдела. Результаты работы аналитического отдела помещаются обратно на сервер данных, где их может увидеть руководство предприятия. Сервер базы данных ведет информационный обмен с серверами территориальных отделений. На рабочих местах специальное ПО формирует запросы к серверу приложений и представляет результаты, полученные от сервера БД (рис. 1.1).

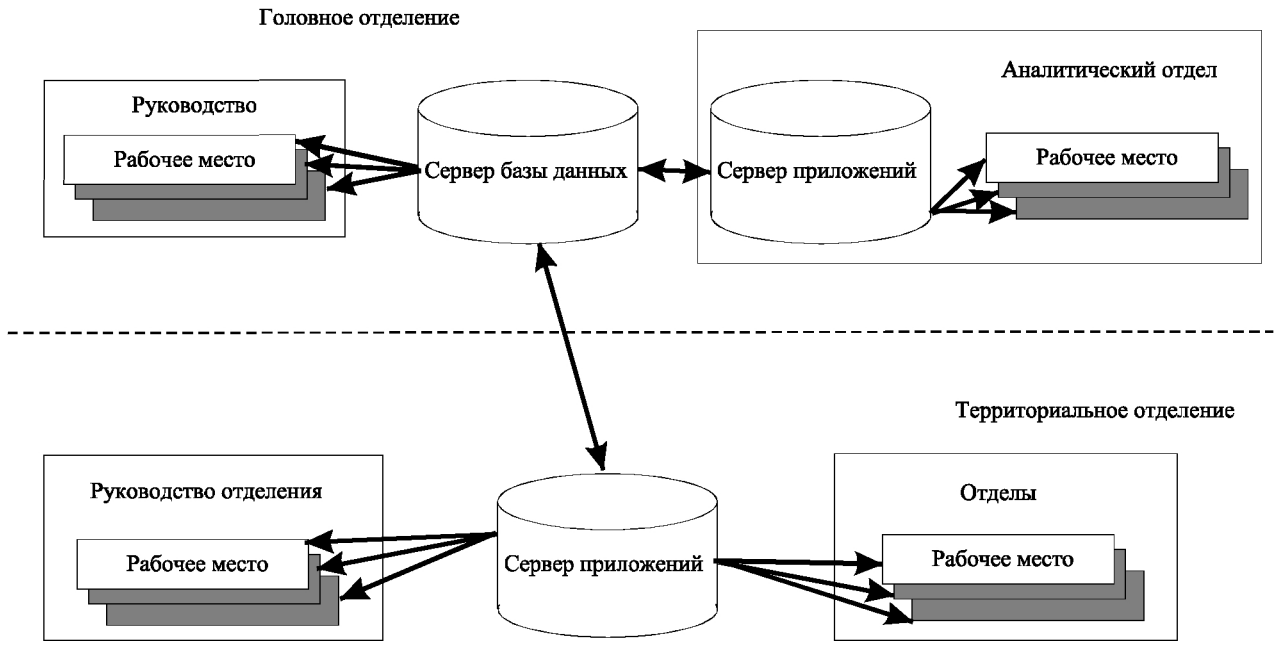

Рис. 1.1 – Структура информационной системы

Состав ИС. ИС состоит из трех основных слоев: слой данных, слой приложений и слой интерфейса. Слой данных обеспечивает хранение данных и передачу данных по запросам слоя приложений. Слой приложений отвечает за обработку данных. Он взаимодействует со слоем данных и слоем интерфейса. Слой интерфейса обеспечивает работу пользователей с графическими формами для формирования запросов и команд слою приложений и представления полученных результатов.

Каждый слой требует различный уровень характеристик компьютера. Для слоя данных необходим мощный компьютер с оптимизированной подсистемой работы с жестким диском (поскольку в основном его работа – запись и чтение данных) и скоростным сетевым адаптером для быстрой передачи данных по сети. Для ускорения операций ввода-вывода рекомендуется также повышенный объем оперативной памяти. Дополнительно требуется устройство резервного копирования для сохранения резервных копий данных.

Слой приложений требует компьютеров с мощным процессором и расширенной оперативной памятью, так как этого требует работа с приложениями. Рабочие места не требуют мощных компьютеров, для них достаточно конфигурации со стандартными характеристиками. Из дополнительного оборудования в этом слое чаще всего используется принтер.

1.6 Классификация сложных систем

В настоящий момент все существующие ИС можно разделить на четыре класса: корпоративные ИС, системы оперативного управления и учета, аналитические ИС, справочные правовые системы [3].

Корпоративные ИС (КИС). КИС обеспечивают интегрированное решение задач управления предприятием как по вертикали (от первичной информации до поддержки принятия решений высшим руководством), так и по горизонтали (все направления деятельности и технологические операции). Весь спектр КИС можно разделить на три группы по степени интеграции: крупные, средние и малые. Они отличаются по набору функций, стоимости и сложности внедрения.

Крупные КИС чаще всего не являются готовыми ИС. Они представляют собой совокупность программных модулей и баз данных, а также технологии их настройки и применения. Эти системы создаются как универсальные и многопрофильные, т.е. достаточно дорогие и сложные в установке и администрировании.

Примеры КИС: R/3 (фирма SAP), Baan 4 (Baan), Oracle application (Oracle), Галактика (Галактика), Парус (Парус), Босс-Корпорация (АйТи), Platinum SQL (Platinum Software Corporation), NS-2000 (Никос-Софт).

Оперативные ИС. Оперативная ИС предназначена для автоматизации оперативной деятельности организации, то есть для повышения эффективности ежедневной текущей деятельности сотрудников предприятия. В зависимости от рода текущей деятельности можно выделить четыре основных блока в оперативной ИС:

1) Блок документооборота. Учитываются входящие и исходящие документы как в бумажном виде, так и в электронном варианте. Блок включает фиксацию маршрута прохождения документа в организации, средства поиска документов и мониторинга их обработки, средства отчетности о документообороте в организации.

2) Блок учета ресурсов. В любой организации существует дефицит ресурсов – денежных, человеческих, транспортных средств, материалов и т.д. Оптимально и наиболее эффективно распределять эти ресурсы и призван данный блок. Он позволяет сотрудникам предприятия оперативно отслеживать состояние этих ресурсов и подавать заявки на их использование.

3) Блок бухгалтерского учета. Ведется полный бухгалтерский учет согласно существующим нормативным документам.

4) Блок кадрового учета. Ведет учет сотрудников организации, выполняет расчет зарплаты и премий, поддерживает штатное расписание и должностную иерархию предприятия. Данный блок позволяет вести мониторинг использования отпусков, больничных, начислять районные коэффициенты и надбавки за вредные условия работы.

Примеры оперативных ИС: 1С:Предприятие (1С), Парус (Парус), БЭСТ (БЭСТ-Программы), Microsoft Business Solution Dynamics.

Аналитические ИС (АИС). Основной целью АИС является накопление информации, необходимой для проведения полного и всестороннего анализа деятельности организации и среды ее функционирования. Для данного анализа не

используется текущая информация, так как она часто меняется. Поэтому анализ основывается на данных законченного временного периода: за прошлый месяц, прошлый квартал, прошлый год. Аналитические данные основаны на агрегировании, суммировании и усреднении информации.

Структура АИС:

1) Блок накопления информации. Чаще всего информация поступает из оперативных ИС. Так как свойства оперативной и аналитической информации различны, то поступающую информацию необходимо преобразовать к виду, необходимому для АИС.

2) Блок поиска информации. За счет того, что данные в АИС агрегируются и упорядочиваются, система позволяет осуществлять более быстрый поиск в больших массивах данных при помощи множества индексов.

3) Блок анализа информации. Так как основной задачей АИС является собственно анализ информации, то система обладает достаточно развитыми средствами ранжирования информации, сравнительного анализа и проведения вычислений. Соответственно, поддерживается широкий набор встроенных статистических, математических и бизнес-функций.

4) Блок конфигурирования и настройки системы. С течением времени часто приходится менять структуру хранимых данных, поэтому АИС должны обладать гибкими средствами настройки. Такие средства включают в свой состав механизмы описания модели данных и дальнейшего использования этой модели при формировании запросов. Для оптимизации хранимых данных используются механизмы, позволяющие оценивать и анализировать качество построения данных с точки зрения скорости вычисления. При отсутствии оптимальности структура данных должна быть изменена.

Виды АИС:

1) Многомерные базы данных (МБД) (MDD – MultiDimensional Database). Информация в многомерной базе данных хранится не в виде индексированных записей в таблицах, а в форме многомерных упорядоченных массивов или кубов. Вид данных похож на данные в электронной таблице, только в таблице

всего два измерения, а МБД-системе измерений может быть гораздо больше. Например, для анализа оказания услуг можно задать многомерный куб, где его измерения будут делить информацию по дате оказания услуги, району отделения, по виду услуги, по исполняющей организации и т.д. Такой способ отображения информации получил название «звезда» (в центре данные, лучи – измерения). Этот способ хранения данных предпочтителен, когда необходима высокая скорость вычислений и выборок, требуются сложные вычисления, четко определена размерность задачи, невелико количество измерений, данные хорошо структурированы. Когда размерность задачи может или должна меняться, применение МБД нежелательно, так как изменение структуры данных связано с большой сложностью изменений. Также противопоказанием для применения МБД является множество невзаимосвязанных измерений. Наиболее известными системами этого класса являются Acumate ES (Kenan Technologies), Essbase (Arbor Software), TM/1 (Sinper), Express (Oracle).

2) Информационные хранилища (ИХ) (DW – Data Warehouse). В ИХ хранение информации похоже на хранение данных в МБД. Схема построения модели в ИХ получила название «снежинка». Здесь, в отличие от «звезды», каждый луч-измерение может иметь в свою очередь свои измерения. Такая схема позволяет более гибко формировать модель данных для сложных предметных областей. Например, в предыдущей ситуации, необходимо разбивать данные еще по наличию выезда на дом, что в МБД потребовало бы создание еще одного измерения, что вызвало бы непропорционально большие расходы вычислительных ресурсов. В ИХ и основного измерения создается подизмерение выезда на дом, что является более рациональным и гибким способом. Плюсы ИХ заключается в более гибкой и рациональной модели данных, позволяющей описывать сложные предметные области, легкость модификации и дополнения модели данных. Наиболее известными представителями данного вида АИС являются Works (Sybase), Axsys (Information Advantage), MetaCube (Stanford Technology Group).

3) Специализированные реляционные БД. Реляционная модель БД накладывает ограничения на скорость выборки и вычислений при сложной модели данных предметной области. Это ограничение послужило поводом для отказа от реляционной модели данных при хранении и обработке больших массивов информации. Позднее производители реляционных СУБД стали выпускать специально оптимизированные СУБД для аналитических данных, получивших название ROLAP-систем (Relation OnLine Analytical Processing). Такие продукты обладают специализированными средствами поиска и выборки данных, оптимизированных для аналитической обработки. Пример: Microsoft SQL Server.

Справочно-правовые системы (СПС). В сфере юридической деятельности и правовой информатизации широко применяется термин «правовая информация». К правовой информации относятся правовые акты, материалы подготовки законопроектов, комментарии и практика применения законов. С помощью СПС специалисту найти различные правовые акты и другие документы стало в сотни раз быстрее и эффективнее, чем при работе с бумажными носителями. У СПС есть две основных функции: 1) хранение и постоянное обновление базы правовой информации, отражающей текущее состояние действующего законодательства, 2) поиск документа в базе. Примеры СПС: Гарант, Консультант плюс, LEXIS-NEXIS.

2. ОСНОВЫ ПРОЕКТИРОВАНИЯ СЛОЖНЫХ СИСТЕМ

2.1 Технология проектирования ИС

Современные информационные технологии предоставляют широкий выбор способов реализации ИС, выбор которых осуществляется на основе требований со стороны предполагаемых пользователей, которые, как правило, изменяются в процессе разработки. Для теории принятия решений процесс проектирования ИС – это процесс принятия проектно-конструкторских решений, направленных на получение проекта системы, удовлетворяющего требования заказчика [4].

Под проектом ИС будет понимать проектно-конструкторскую и технологическую документацию, в которой представлено описание проектных решений по созданию и эксплуатации ИС в конкретной программно-технической среде.

Под проектированием ИС понимается процесс преобразования входной информации об объекте проектирования, о методах проектирования и об опыте проектирования объектов аналогичного назначения в соответствии со стандартами в проект ИС. С этой точки зрения проектирование ИС сводится к последовательной формализации проектных решений на различных стадиях жизненного цикла ИС.

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

В качестве субъекта проектирования ИС выступают коллективы специалистов, которые осуществляют проектную деятельность, как правило, в составе специализированной проектной организации, и организация-заказчик, для которой необходимо разработать ИС. При большом объеме и жестких сроках вы-

полнения проектных работ в разработке системы может принимать участие несколько проектных коллективов. В этом случае выделяется головная организация, которая координирует деятельность всех организаций-соисполнителей.

Осуществление проектирования ИС предполагает использование проектировщиками определенной технологии проектирования, соответствующей масштабу и особенностям разрабатываемого проекта. Технология проектирования – это совокупность методологии и средств проектирования ИС, а также методов и средств организации проектирования (рис. 2.1).

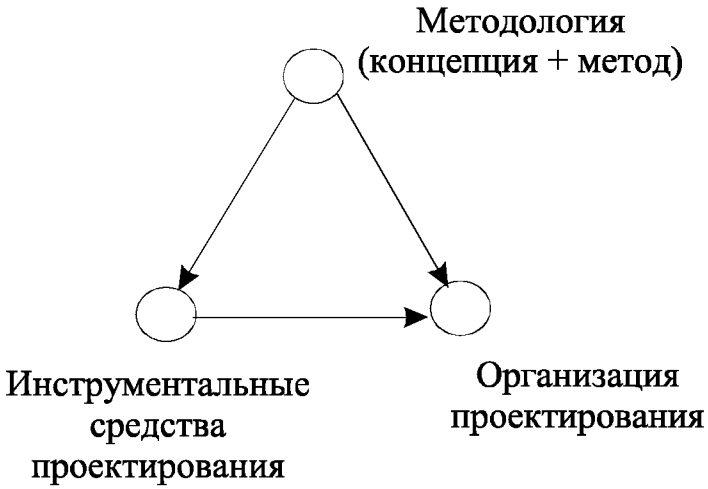

Рисунок 2.1 – Состав компонентов технологии проектирования

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 проектирования ИС в целом делится на совокупность последовательно-параллельных, связанных и соподчиненных цепочек действий. Действия, которые выполняются при проектировании ИС, могут быть определены как неделимые технологические операции или как подпроцессы технологических операций. Все действия могут быть собственно проектировочными, которые формируют или модифицируют результаты проектирования, и оценочными

действиями, 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 и в каком порядке это должно быть сделано.

К основным требованиям, предъявляемым к выбираемой технологии проектирования, относятся следующие:

– созданный с помощью этой технологии проект должен отвечать требованиям заказчика;

– выбранная технология должна максимально отражать все этапы цикла жизни проекта;

– выбираемая технология должна обеспечивать минимальные трудовые и стоимостные затраты на проектирование и сопровождение проекта;

– технология должна быть основой связи между проектированием и сопровождением проекта;

– технология должна способствовать росту производительности труда проектировщика;

– технология должна обеспечивать надежность процесса проектирования и эксплуатации проекта;

– технология должна способствовать простому ведению проектной документации.

Основу технологии проектирования ИС составляет методология, которая определяет сущность, основные отличительные технологические особенности. Методология проектирования предполагает наличие некоторой концепции, принципов проектирования, реализуемых наборов методов проектирования, которые, в свою очередь, должны поддерживаться некоторыми средствами проектирования.

Организация проектирования предполагает определение методов взаимодействия проектировщиков между собой и с заказчиком в процессе создания проекта ИС, которые могут также поддерживаться набором специфических средств.

Для конкретных видов технологий проектирования свойственно применение определенных средств разработки ИС, которые поддерживают выполнение как отдельных проектных работ, этапов, так и их совокупностей. Поэтому перед разработчиками ИС, как правило, стоит задача выбора средств проектирования, которые по своим характеристикам в наибольшей степени соответствуют требованиям конкретного предприятия.

Средства проектирования должны быть:

– в своем классе инвариантными к объекту проектирования;

– охватывать в совокупности все этапы жизненного цикла ИС;

– технически, программно и информационно совместимыми;

– простыми в освоении и применении;

– экономически целесообразными.

2.2 Принципы проектирования сложных объектов

При проектировании сложных объектов используются следующие принципы [5]:

– декомпозиция и иерархичность построения описаний объектов проектирования;

– многоэтапность и итерационность процесса проектирования;

– типизация и унификация проектных решений.

Описания технических объектов должны быть по сложности согласованы: 1) с возможностями восприятия человеком; 2) с возможностями оперирования описаниями в процессе их преобразования с помощью имеющихся средств проектирования.

Выполнить это требование в рамках единого описания удается лишь для простых изделий. Как правило, требуется структурирование описаний и соответствующее разбиение представлений об объекте на иерархические уровни и аспекты. Это позволяет распределить работы по проектированию сложных объектов между подразделениями проектировщиков, что способствует повышению эффективности и производительности труда.

Разделение описаний по степени детализации отображаемых свойств и характеристик объекта лежит в основе блочно-иерархического подхода к проектированию и приводит к появлению иерархических уровней (уровней абстрагирования) в представлениях об объекте.

На уровне 0 (верхнем уровне) сложный объект S рассматривается как система S из n взаимно связанных и взаимодействующих элементов S_i на уровне 1 (рис. 2.2).

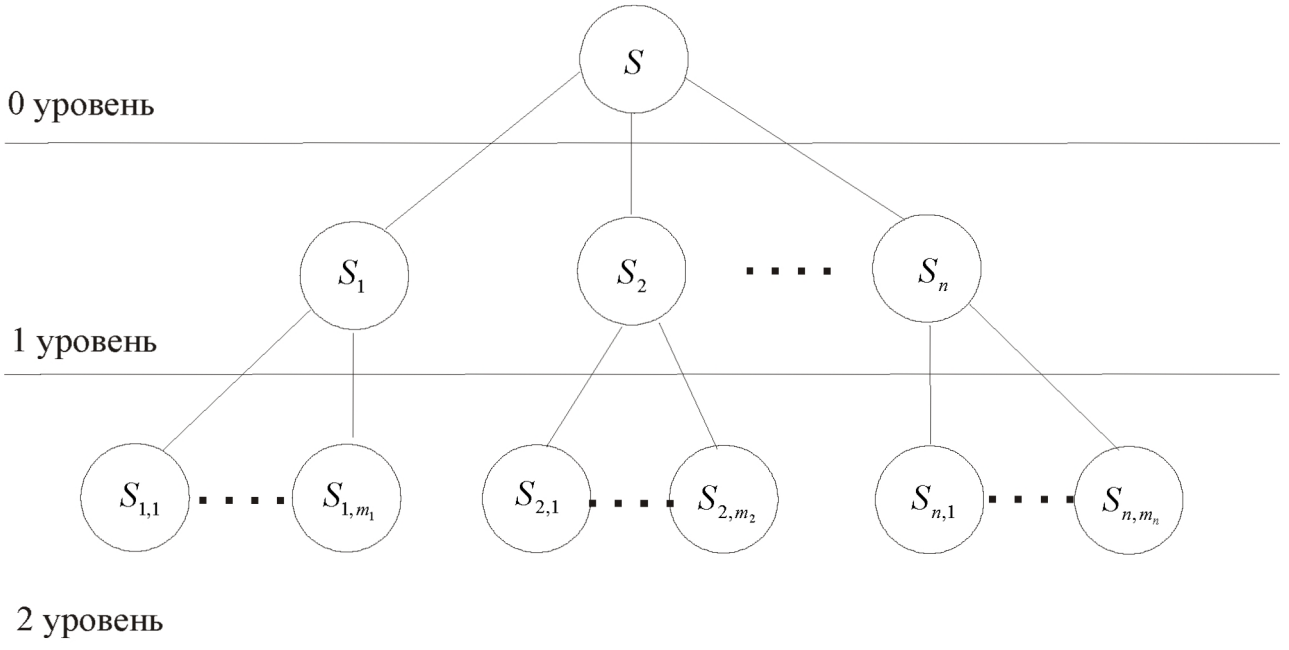

Рис. 2.2 – Иерархические уровни описаний проектируемых объектов

Каждый из элементов в описании уровня 1 представляет собой также довольно сложный объект, который, в свою очередь, рассматривается как описа-

ние системы S_i на уровне 2. Элементами системы S_i являются объекты $S_{i,j}$, $j = 1,2,\bullet\ ,m_i$, где m_i – количество элементов в описании системы S_i. Как правило, выделение элементов $S_{i,j}$ происходит по функциональному признаку.

Подобное разбиение продолжается вплоть до получения на некотором уровне элементов, описания которых дальнейшему делению не подлежат, то есть до элементов, описание которых уже известно. Такие элементы по отношению к объекту S называются ***базовыми*** элементами.

Принцип иерархичности означает структурирование представлений об объекте проектирования по степени детальности описаний (уровни описаний – по вертикали)

Принцип декомпозиции (блочности) означает разбиение представлений каждого уровня на ряд составных частей (блоков) с возможностью раздельного (поблочного) проектирования объектов S_i на уровне 1, объектов $S_{i,j}$ на уровне 2 и т.д.

Кроме разбиения описаний по степени подробности отражения свойств объектов используют декомпозицию описаний по характеру отображаемых свойств объекта. Такая декомпозиция приводит к появлению ряда аспектов описаний. Наиболее крупные аспекты описаний объектов: функциональный; конструкторский; технологический. Решение задач, связанных с преобразованием или получением описаний, относящихся к этим аспектам, называют соответственно функциональным, конструкторским и технологическим проектированием.

Функциональный аспект связан с отображением основных принципов функционирования, характера физических и информационных процессов, протекающих в объекте. Функциональный аспект отображается в принципиальных, функциональных, структурных и других схемах и сопровождающих их документах.

Конструкторский аспект связан с реализацией результатов функционального проектирования, то есть с определением геометрических форм объектов и их взаиморасположением в пространстве.

Технологический аспект относится к реализации результатов функционального и конструкторского проектирования, т.е. связан с описанием методов и средств изготовления объектов.

Внутри каждого аспекта возможно свое специфическое выделение иерархических уровней.

Если решение задач высоких иерархических уровней предшествует решению задач более низких иерархических уровней, то проектирование называют *нисходящим*. Если раньше выполняются этапы, связанные с низшими иерархическими уровнями, то проектирование называют *восходящим*. У каждого из этих двух видов проектирования имеются преимущества и недостатки.

При нисходящем проектировании система разрабатывается в условиях, когда ее элементы еще не определены и, следовательно, сведения о их возможностях и свойствах носят предположительный характер.

При восходящем проектировании, наоборот, элементы проектируются раньше системы, и, следовательно, предположительный характер имеют требования к системе. В обоих случаях из-за отсутствия исчерпывающей исходной информации имеют место отклонения от возможных оптимальных технических результатов.

Поскольку принимаемые предположения могут не оправдаться, часто требуется повторное выполнение проектных процедур предыдущих этапов после выполнения проектных процедур последующих этапов. Такие повторения обеспечивают последовательное приближение к оптимальным результатам и обуславливают *итерационный* характер проектирования. Следовательно, итерационность нужно относить к важным принципам проектирования сложных объектов.

На практике обычно сочетают восходящее и нисходящее проектирование. Например, восходящее проектирование имеет место на всех тех иерархических

уровнях, на которых используются *унифицированные* (стандартные) элементы. Очевидно, что унифицированные элементы, ориентированные на применение в ряде различных систем определенного класса, разрабатываются раньше, чем та или иная конкретная система из этого класса.

Обычно унификация объектов имеет целью улучшение технико-экономических показателей производства и эксплуатации изделий. Использование типовых и унифицированных проектных решений приводит так же к упрощению и ускорению проектирования.

Однако, унификация целесообразна только в таких классах объектов, в которых из сравнительно небольшого числа разновидностей элементов предстоит проектирование и изготовление большого числа систем. Именно эти разновидности элементов подлежат унификации.

Для сложных систем и для элементов, реализующих новые физические принципы или технологические возможности, в каждом конкретном случае приходится заново выполнять многоуровневое иерархическое проектирование.

В этих условиях целесообразно ставить вопрос не об унификации изделий, а об унификации средств их проектирования и изготовления, в частности об унификации проектных процедур в рамках систем автоматизированного проектирования (САПР).

Наличие средств автоматизированного выполнения типовых проектных процедур позволяет оперативно создавать проекты новых изделий, а в сочетании со средствами изготовления в условиях гибких автоматизированных производств осуществлять оперативное изготовление новых оригинальных изделий.

Окончательное описание проектируемого объекта представляет собой полный комплект схемной, конструкторской и технологической документации, оформленной в соответствии с требованиями ГОСТов: ЕСКД (единая система конструкторской документации), ЕСТД (единая система технологической документации), ЕСПД (единая система программной документации). Этот комплект документации предназначен для использования в процессе изготовления и эксплуатации объекта проектирования.

Важное значение в этих описаниях имеют математические модели объектов проектирования, так как выполнение проектных процедур при автоматизированном проектировании основано на оперировании математическими моделями.

Математическая модель (ММ) технического объекта – система математических объектов (чисел, переменных, матриц, множеств и т.п.) и отношений между ними, отражающих некоторые свойства технического объекта.

При проектировании используют математические модели, отражающие свойства объекта, существенные с позиции проектировщика.

Среди свойств объекта, отражаемых в описаниях на определенном иерархическом уровне, в том числе в ММ, различают свойства: систем; элементов систем и внешней среды, в которой должен функционировать объект. Количественное выражение этих свойств осуществляется с помощью величин, называемых параметрами. Величины, характеризующие свойства системы, элементов системы и внешней среды, называют соответственно входными, внутренними и внешними параметрами.

Однако существование ММ не означает, что она известна разработчику и может быть представлена в явном функциональном виде. Типичной является ситуация, когда математическое описание процессов в проектируемом объекте задается моделью в форме системы уравнений, в которой фигурирует вектор фазовых переменных. Фазовые переменные характеризуют физическое или информационное состояние объекта, а их изменения во времени выражают переходные процессы в объекте. Например, состояние некоторой фирмы можно определить такими фазовыми переменными: сырье, материалы, финансовые и трудовые ресурсы.

Выделим следующие особенности параметров в моделях проектируемых объектов:

1. Внутренние параметры в моделях k-го иерархического уровня становятся выходными параметрами в моделях более низкого $(k+1)$-го иерархического уровня. Так, например, трудовые ресурсы являются внутренними при проекти-

ровании производственной фирмы и в то же время выходными при проектировании отдела кадров этой фирмы.

2. Выходные параметры или фазовые переменные, фигурирующие в модели одной из подсистем (в одном из аспектов описаний), часто оказываются внешними параметрами в описании других подсистем (других аспектов). Так, например, выходные параметры подсистемы планирования выпуска продукции некоторой компании являются внешними параметрами подсистемы материально-технического снабжения этой компании.

3. Большинство выходных параметров объекта являются функционалами.

4. В техническом задании на проектирование должны фигурировать величины, называемые техническими требованиями к выходным параметрам (нормами выходных параметров). Данные нормы представляют собой границы допустимых диапазонов изменения выходных параметров.

2.3 Классификация типовых процедур проектирования

Проектная процедура называется *типовой*, если она предназначена для многократного применения при проектировании многих типов объектов. Различают проектные процедуры анализа и синтеза. Синтез заключается в создании описания объекта, а анализ – в определении свойств и исследовании работоспособности объекта по его описанию, т.е. при синтезе создаются, а при анализе оцениваются проекты объектов [5].

Процедуры анализа делятся на процедуры одно- и многовариантного анализа. При одновариантном анализе заданы значения внутренних и внешних параметров, требуется определить значения выходных параметров объекта. Подобная задача обычно сводится к однократному решению уравнений, составляющих математическую модель, что и обуславливает название этого вида анализа. Многовариантный анализ заключается в исследовании свойств объекта в некоторой области пространства внутренних переменных. Такой анализ требу-

ет многократного решения систем уравнений (многократного выполнения одновариантного анализа).

Процедуры синтеза делятся на процедуры структурного и параметрического синтеза. Целью структурного синтеза является определение структуры объекта – перечня типов элементов, составляющих объект, и способа связи элементов между собой в составе объекта. Параметрический синтез заключается в определении числовых значений параметров элементов при заданных структуре и условиях работоспособности на выходные параметры объекта, т.е. при параметрическом синтезе нужно найти точку или область в пространстве внутренних параметров, в которых выполняются те или иные условия (обычно условия работоспособности).

На рис. 2.3 представлена типичная последовательность проектных процедур на одном из этапов нисходящего проектирования. На предыдущем этапе решались задачи $(k-1)$-го иерархического уровня.

Рис. 2.3 – Схема процесса проектирования

Проектирование системы начинается с синтеза исходного варианта ее структуры. Для оценки этого варианта создается: математическая модель (при автоматизированном проектировании); экспериментальная модель или стенд (при неавтоматизированном проектировании).

После выбора исходных значений параметров элементов выполняется анализ варианта, по результатам которого становится возможной его оценка.

Обычно оценка заключается в проверке выполнения условий работоспособности, сформулированных в техническом задании (ТЗ).

Если условия работоспособности выполняются в должной мере, то полученное проектное решение принимается, затем описывается система $(k+1)$-го уровня в принятой форме и формулируется ТЗ на проектирование элементов данного уровня. Если же полученное проектное решение неудовлетворительно, то выбирается один из возможных путей улучшения проекта.

Обычно проще всего изменить числовые значения параметров элементов. Совокупность процедур модификации параметров элементов, анализа и оценки результатов анализа представляет собой процедуру *параметрического синтеза*. Если модификации параметров целенаправленны и подчинены стратегии поиска наилучшего значения некоторого показателя качества, то процедура параметрического синтеза является *процедурой оптимизации*.

Возможно, что путем параметрического синтеза не удастся добиться приемлемой степени выполнения условий работоспособности. Тогда используют другой путь, связанный с модификациями структуры. Новый вариант структуры синтезируется, для него повторяются процедуры формирования модели и параметрического синтеза.

Если не удается получить приемлемое решение и на этом пути, то ставится вопрос о корректировке ТЗ, сформулированного на предыдущем этапе проектирования. Такая корректировка может потребовать повторного выполнения ряда процедур k-го иерархического уровня, что и обуславливает итерационный характер проектирования.

Существует характерная особенность взаимосвязи проектных процедур анализа и синтеза. Эта взаимосвязь имеет характер вложенности процедуры анализа в процедуру оптимизации (параметрического синтеза) и процедуры оптимизации в процедуру синтеза (структурного и параметрического).

Вложенность означает, что:

– анализ входит как составная часть в оптимизацию, а оптимизация – в синтез;

– однократное выполнение процедуры оптимизации требует многократного выполнения процедуры анализа;

– однократное решение задачи синтеза требует многократного решения задачи оптимизации.

Очевидно, такой же характер взаимодействия имеют процедуры анализа: однократный многовариантный анализ основан на многократном одновариантном анализе.

Нетрудно подсчитать, что синтез проектного решения может потребовать чрезмерно большого количества вариантов анализа. Один из путей решения этой проблемы – применение достаточно точных и сложных математических моделей и алгоритмов анализа только на завершающих итерациях синтеза. Для большинства просматриваемых вариантов структуры при этом выполняется лишь ориентировочная оценка на основе косвенных критериев, упрощенных моделей и алгоритмов. Такая оценка позволит без существенных затрат вычислительных ресурсов отсеять большинство неперспективных вариантов и оставить для тщательного анализа малое число вариантов.

2.4 Жизненный цикл информационных систем

Проектирование ИС – трудоемкий, длительный и динамический процесс. Технологии проектирования, применяемые в настоящее время, предполагают поэтапную разработку системы. Этапы по общности целей могут объединяться в стадии. Совокупность стадий и этапов, которые проходит ИС в своем развитии от момента принятия решения о создании системы до момента прекращения функционирования системы, называется *жизненным циклом ИС* [4].

Суть содержания жизненного цикла (ЖЦ) разработки ИС в различных подходах одинакова и сводится к выполнению следующих стадий:

1. Стадия планирования и анализа требований (предпроектная стадия системного анализа): исследование и анализ существующей ИС, определение тре-

бований к создаваемой ИС, оформление технико-экономического обоснования и технического задания на разработку ИС.

2. Стадия проектирования: разработка в соответствии со сформулированными требованиями состава автоматизируемых функций и состава обеспечивающих подсистем, оформление технического проекта ИС.

3. Стадия реализации: разработка и настройка программ, наполнение базы данных, создание рабочих инструкций для персонала.

4. Стадия внедрения: комплексная отладка подсистем ИС, обучение персонала, поэтапное внедрение ИС в эксплуатацию по подразделениям предприятия, оформление акта о приемо-сдаточных испытаниях ИС.

5. Стадия эксплуатации ИС: сбор рекламаций и статистики о функционировании ИС, исправление ошибок и недоработок, оформление требований к модернизации ИС и их выполнение (повторение стадий 2–5).

Важной чертой жизненного цикла ИС является его повторяемость «Системный анализ – разработка – сопровождение – системный анализ». Это соответствует представлению об ИС как о развивающейся, динамической системе. При первом выполнении стадии разработки создается проект ИС, а при повторном выполнении осуществляется модификация проекта для поддержания его в актуальном состоянии.

Другой характерной чертой жизненного цикла является наличие нескольких циклов внутри схемы:

1. Первый цикл – это цикл первичного проектирования ИС.

2. Второй цикл – цикл, который возникает после опытного внедрения, в результате которого выясняются частные ошибки в элементах проекта.

3. Третий цикл возникает после сдачи в промышленную эксплуатацию, когда выявляют ошибки в функциональной архитектуре системы, связанные с несоответствием проекта требованиям заказчика по составу функциональных подсистем, составу задач и связям между ними.

4. Четвертый цикл возникает в том случае, когда требуется модификация системной архитектуры в связи с необходимостью адаптации проекта к новым условиям функционирования системы.

5. Пятый цикл возникает, если проект системы совершенно не соответствует требованиям, предъявляемые к организационно-экономической системе ввиду того, что осуществляется моральное его старение.

Чтобы исключить пятый цикл и максимально уменьшить необходимость выполнение третьего и четвертого циклов, необходимо выполнять проектирование ИС на всех этапах первого, основного цикла разработки ИС, в соответствии со следующими требованиями:

– разработка ИС должна быть выполнена в строгом соответствии со сформулированными требованиями к создаваемой системе;

– требования к ИС должны адекватно соответствовать целям и задачам эффективного функционирования самого объекта;

– созданная ИС должна соответствовать сформулированным требованиям на момент окончания внедрения, а не на момент начала разработки;

– внедренная ИС должна развиваться и адаптироваться в соответствии с постоянно изменяющимися требованиями к ИС.

С точки зрения реализации перечисленных аспектов в технологиях проектирования ИС модели жизненного цикла претерпевали существенные изменения. Среди известных моделей жизненного цикла можно выделить следующие модели:

– каскадная модель (до 70-х годов) – последовательный переход на следующий этап после завершения предыдущего;

– итерационная модель (70 – 80-е годы) – с итерационными возвратами на предыдущие этапы после выполнения очередного этапа;

– спиральная модель (современное время) – модель, предполагающая постепенное расширение прототипа ИС.

Каскадная модель. Для этой модели жизненного цикла характерна автоматизация отдельных несвязанных задач, не требующая выполнения информа-

ционной интеграции и совместимости, программного, технического и организационного сопряжения. В рамках решения отдельных задач каскадная модель жизненного цикла по срокам разработки и надежности оправдала себя. Применение каскадной модели жизненного цикла к большим и сложным проектам вследствие большой длительности процесса проектирования и изменчивости требований за это время приводит к их практической нереализуемости.

Итерационная модель. Создание комплексных ИС предполагает проведение увязки проектных решений, получаемых при реализации отдельных задач. Подход к проектированию «снизу-вверх» обуславливает необходимость таких итерационных возвратов, когда проектные решения по отдельным задачам комплектуются в общие системные решения и при этом возникает потребность в пересмотре ранее сформулированных требований. Как правило, вследствие большого числа итераций возникают рассогласования в выполненных проектных решениях и документации. Запутанность функциональной и системной архитектуры созданной ИС, трудность в использовании проектной документации вызывают на стадиях внедрения и эксплуатации сразу необходимость перепроектирования всей системы. Длительный жизненный цикл разработки ИС заканчивается этапом внедрения, за которым начинается жизненный цикл создания новой ИС.

Спиральная модель. Используется подход к организации проектирования ИС «сверху-вниз», когда сначала определяется состав функциональных подсистем, а затем постановка отдельных задач. Соответственно, сначала разрабатываются такие общесистемные вопросы, как организация интегрированной базы данных, технология сбора, передачи и накопления информации, а затем технология решения конкретных задач. В рамках комплексов задач программирование осуществляется по направлению от головных программных модулей к исполняющим отдельные функции модулям. При этом на первый план выходят вопросы взаимодействия интерфейсов программных модулей между собой и с базой данных, а на второй план – реализация алгоритмов.

3. СТРУКТУРНЫЙ ПОДХОД К ПРОЕКТИРОВАНИЮ СЛОЖНЫХ СИСТЕМ

3.1 Сущность структурного подхода

Сущность структурного подхода к разработке ИС заключается в ее декомпозиции на автоматизируемые функции: т.е. система разбивается на функциональные подсистемы, которые в свою очередь делятся на подфункции, эти подфункции подразделяются на задачи и так далее. Процесс разбиения продолжается вплоть до конкретных процедур. При этом автоматизируемая система сохраняет целостное представление, в котором все составляющие компоненты взаимоувязаны. При разработке системы "снизу-вверх" от отдельных задач ко всей системе целостность теряется, возникают проблемы при информационной стыковке отдельных компонентов [5].

Все наиболее распространенные методологии структурного подхода базируются на ряде общих принципов. В качестве двух базовых принципов используются следующие:

– принцип решения сложных проблем путем их разбиения на множество меньших независимых задач, легких для понимания и решения;

– принцип иерархического упорядочивания, т.е. принцип организации составных частей проблемы в иерархические древовидные структуры с добавлением новых деталей на каждом уровне.

Выделение двух базовых принципов не означает, что остальные принципы являются второстепенными. Основными из этих принципов являются следующие:

– принцип абстрагирования: заключается в выделении существенных аспектов системы и отвлечения от несущественных;

– принцип формализации: заключается в необходимости строгого методического подхода к решению проблемы;

– принцип непротиворечивости: заключается в обоснованности и согласованности элементов;

– принцип структурирования данных: заключается в том, что данные должны быть структурированы и иерархически организованы.

В структурном анализе используются в основном две группы средств, иллюстрирующих функции, выполняемые системой и отношения между данными. Каждой группе средств соответствуют определенные виды моделей (диаграмм), наиболее распространенными среди которых являются следующие:

– SADT (Structured Analysis and Design Technique) модели и соответствующие функциональные диаграммы;

– DFD (Data Flow Diagrams) диаграммы потоков данных;

– ERD (Entity-Relationship Diagrams) диаграммы "сущность-связь".

На стадии проектирования ИС модели расширяются, уточняются и дополняются диаграммами, отражающими структуру программного обеспечения: архитектуру ПО, структурные схемы программ и диаграммы экранных форм.

Перечисленные модели в совокупности дают полное описание ИС независимо от того, является ли она существующей или вновь разрабатываемой. Состав диаграмм в каждом конкретном случае зависит от необходимой полноты описания системы.

3.2 Методология функционального моделирования SADT

Методология SADT представляет собой совокупность методов, правил и процедур, предназначенных для построения функциональной модели объекта какой-либо предметной области. Функциональная модель SADT отображает функциональную структуру объекта, т.е. производимые им действия и связи между этими действиями [5].

Результатом применения методологии SADT является модель, которая состоит из диаграмм, фрагментов текстов и глоссария, имеющих ссылки друг на

друга. Диаграммы - главные компоненты модели, все функции ИС и интерфейсы на них представлены как блоки и дуги. Место соединения дуги с блоком определяет тип интерфейса. Управляющая информация входит в блок сверху, в то время как информация, которая подвергается обработке, показывается с левой стороны блока, а результаты выхода – с правой стороны. Механизм (человек или автоматизированная система), который осуществляет операцию, представляется дугой, входящей в блок снизу (рис. 3.1).

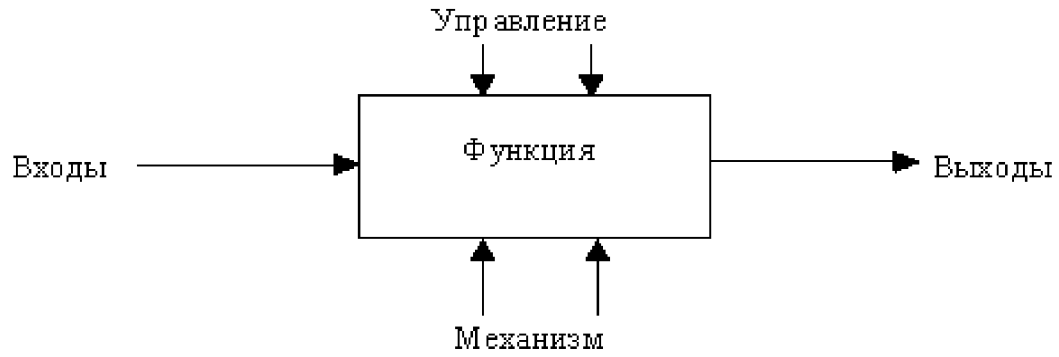

Рис. 3.1 – Функциональный блок и интерфейсные дуги

SADT-модель дает полное, точное и адекватное описание системы, имеющее конкретное назначение. Целью модели является получение ответов на некоторую совокупность вопросов. Эти вопросы неявно присутствуют (подразумеваются) в процессе анализа и, следовательно, они руководят созданием модели и направляют его. Это означает, что сама модель должна будет дать ответы на эти вопросы с заданной степенью точности. Если модель отвечает не на все вопросы или ее ответы недостаточно точны, то мы говорим, что модель не достигла своей цели. Определяя модель таким образом, SADT закладывает основы практического моделирования. Только поняв, насколько хорошо нужно ответить на поставленные вопросы, можно определить, когда процесс моделирования можно считать завершенным (т.е. когда модель будет соответствовать поставленной цели).

Модель является некоторым толкованием системы. Поэтому субъектом моделирования служит сама система. Однако моделируемая система никогда не существует изолированно: она всегда связана с окружающей средой. Причем зачастую трудно сказать, где кончается система и начинается среда. По этой причине в методологии SADT подчеркивается необходимость точного определения границ системы. SADT-модель всегда ограничивает свой субъект, т.е. модель устанавливает точно, что является и что не является субъектом моделирования, описывая то, что входит в систему, и подразумевая то, что лежит за ее пределами. Ограничивая субъект, SADT-модель помогает сконцентрировать внимание именно на описываемой системе и позволяет избежать включения посторонних субъектов. Вот почему мы утверждаем, что SADT-модель должна иметь единственный субъект.

С определением модели тесно связана позиция, с которой наблюдается система и создается ее модель. Поскольку качество описания системы резко снижается, если оно не сфокусировано ни на чем, SADT требует, чтобы модель рассматривалась все время с одной и той же позиции. Эта позиция называется "точкой зрения" данной модели. "Точку зрения" лучше всего представлять себе как место (позицию) человека или объекта, в которое надо встать, чтобы увидеть систему в действии. С этой фиксированной точки зрения можно создать согласованное описание системы так, чтобы в модели не смешивались бы несвязанные описания.

После того как определены субъект, цель и точка зрения модели, начинается первая интеграция процесса моделирования по методологии SADT. Субъект определяет, что включить в модель, а что исключить из нее. Точка зрения диктует автору модели выбор нужной информации о субъекте и форму ее подачи. Цель становится критерием окончания моделирования. Конечным результатом этого процесса является набор тщательно взаимоувязанных описаний, начиная с описания самого верхнего уровня всей системы и кончая подробным описанием деталей или операций системы.

Одной из наиболее важных особенностей методологии SADT является постепенное введение все больших уровней детализации по мере создания диаграмм, отображающих модель.

На рис. 3.2 приведены четыре диаграммы и их взаимосвязи, показана структура SADT-модели. Каждый компонент модели может быть декомпозирован на другой диаграмме. Каждая диаграмма иллюстрирует "внутреннее строение" блока на родительской диаграмме. Дуги, входящие в блок и выходящие из него на диаграмме верхнего уровня, являются точно теми же самыми, что и дуги, входящие в диаграмму нижнего уровня и выходящие из нее, потому что блок и диаграмма представляют одну и ту же часть системы.

Построение SADT-модели начинается с представления всей системы в виде простейшей компоненты - одного блока и дуг, изображающих интерфейсы с функциями вне системы. Поскольку единственный блок представляет всю систему как единое целое, имя, указанное в блоке, является общим. Это верно и для интерфейсных дуг - они также представляют полный набор внешних интерфейсов системы в целом.

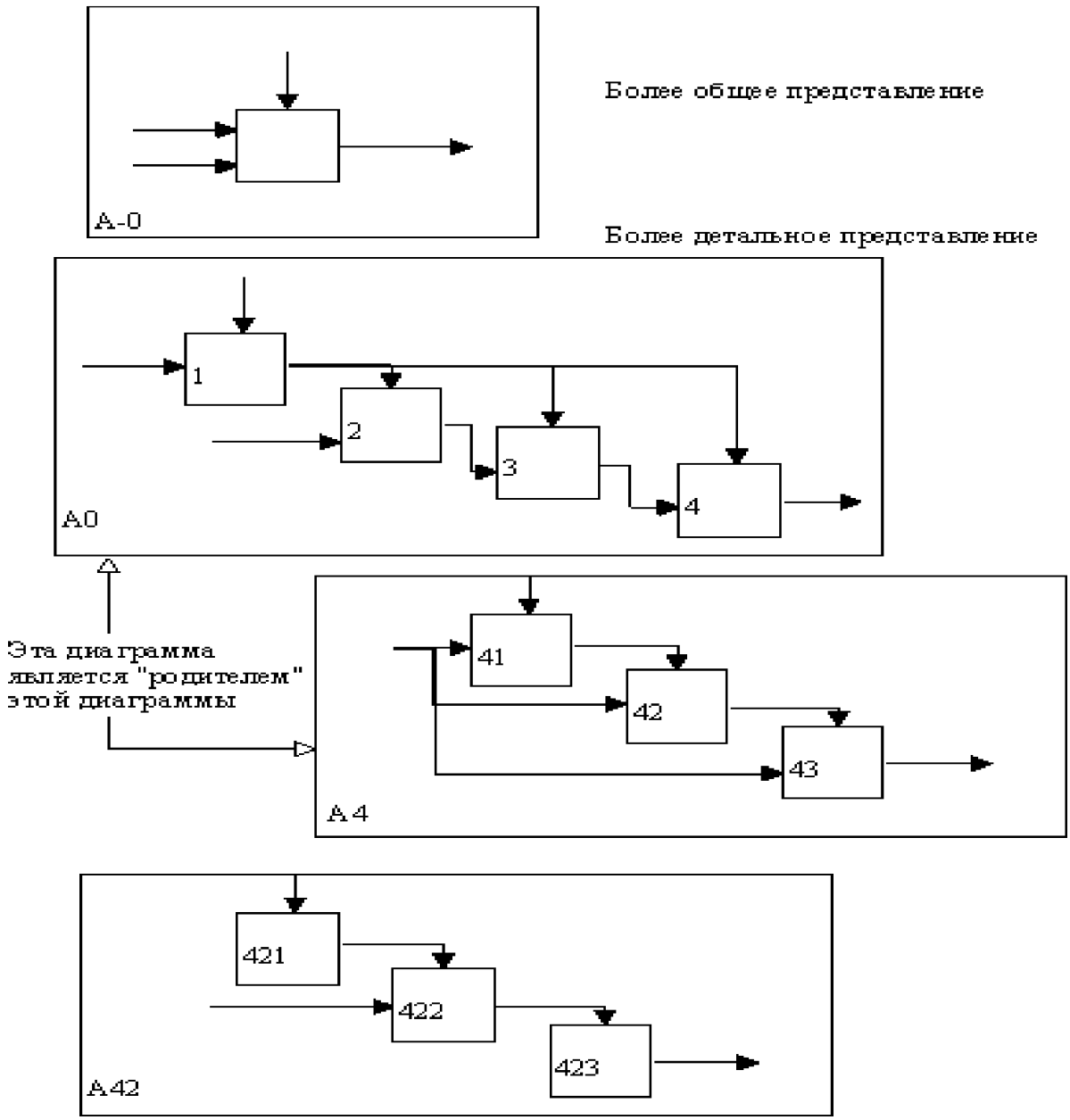

Рис. 3.2 – Структура SADT-модели. Декомпозиция диаграмм

Затем блок, который представляет систему в качестве единого модуля, детализируется на другой диаграмме с помощью нескольких блоков, соединенных интерфейсными дугами. Эти блоки представляют основные подфункции исходной функции. Данная декомпозиция выявляет полный набор подфункций, каждая из которых представлена как блок, границы которого определены интерфейсными дугами. Каждая из этих подфункций может быть декомпозирована подобным образом для более детального представления. Во всех случаях

каждая подфункция может содержать только те элементы, которые входят в исходную функцию.

Модель SADT представляет собой серию диаграмм с сопроводительной документацией, разбивающих сложный объект на составные части, которые представлены в виде блоков. Детали каждого из основных блоков показаны в виде блоков на других диаграммах. Каждая детальная диаграмма является декомпозицией блока из более общей диаграммы. На каждом шаге декомпозиции более общая диаграмма называется *родительской* для более детальной диаграммы.

Некоторые дуги присоединены к блокам диаграммы обоими концами, у других же один конец остается не присоединенным. Не присоединенные дуги соответствуют входам, управлениям и выходам родительского блока. Источник или получатель этих пограничных дуг может быть обнаружен только на родительской диаграмме.

Каждый блок на диаграмме имеет свой номер. Блок любой диаграммы может быть далее описан диаграммой нижнего уровня, которая, в свою очередь, может быть далее детализирована с помощью необходимого числа диаграмм. Таким образом, формируется иерархия диаграмм.

Для того, чтобы указать положение любой диаграммы или блока в иерархии, используются номера диаграмм. Например, A21 является диаграммой, которая детализирует блок 1 на диаграмме A2. Аналогично, A2 детализирует блок 2 на диаграмме A0, которая является самой верхней диаграммой модели.

Блоки SADT никогда не размещаются на диаграмме случайным образом. Они размещаются по степени важности, как ее понимает автор диаграммы. В SADT этот относительный порядок называется доминированием. Доминирование понимается как влияние, которое один блок оказывает на другие блоки диаграммы. Например, самым доминирующим блоком диаграммы может быть либо первый из требуемой последовательности функций, либо планирующая или контролирующая функция, влияющая на все другие функции. Наиболее доминирующий блок обычно размещается в верхнем левом углу диаграммы, а наи-

менее доминирующий - в правом нижнем углу. В результате получается "ступенчатая" схема, подобная представленной на рисунке 3.3. Расположение блоков на странице отражает авторское определение доминирования. Таким образом, топология диаграммы показывает, какие функции оказывают большее влияние на остальные. Чтобы подчеркнуть это, SADT-аналитик может перенумеровать блоки в соответствии с порядком их доминирования.

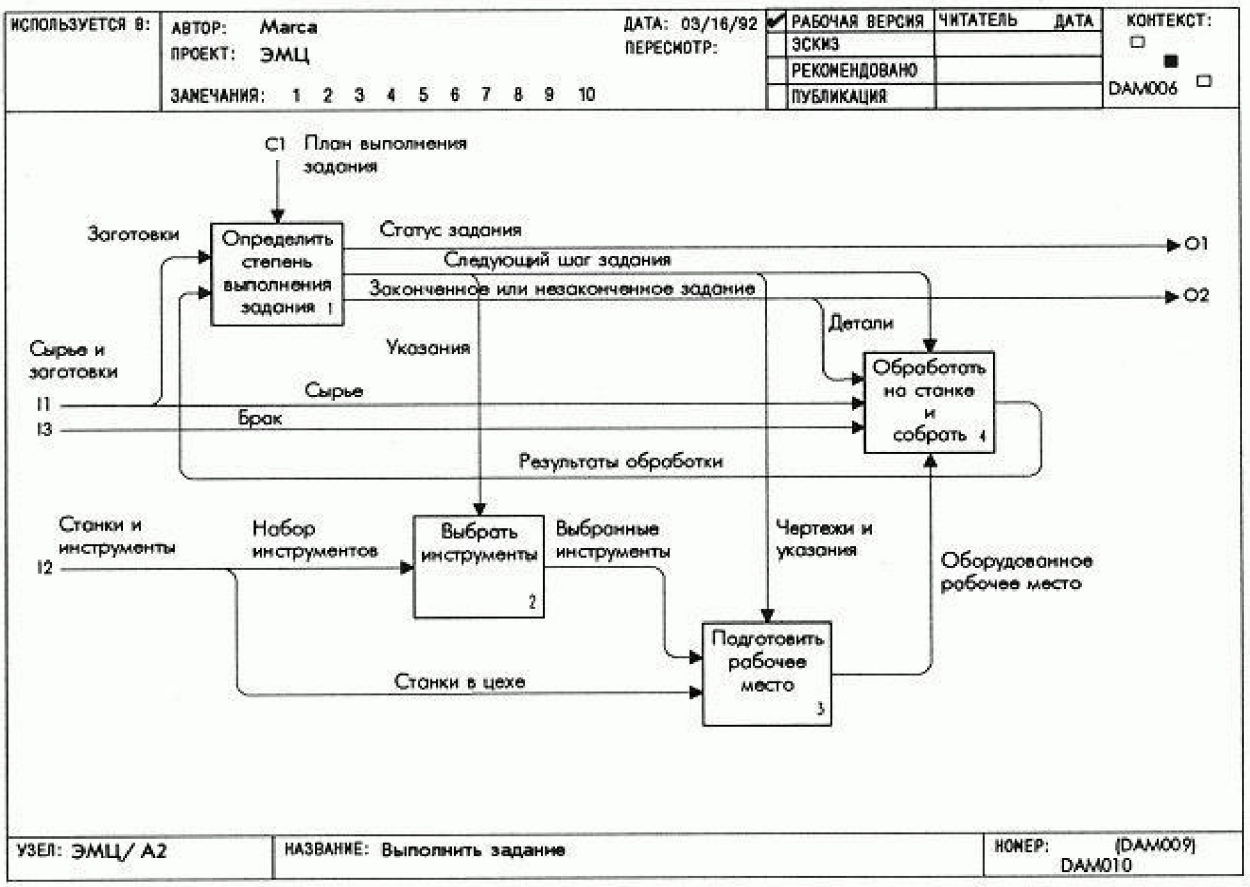

Рис. 3.3 – Пример SADT-диаграммы.

В методологии SADT требуется только пять типов взаимосвязей между блоками для описания их отношений: управление, вход, обратная связь по управлению, обратная связь по входу, выход-механизм. Связи по управлению и входу являются простейшими, поскольку они отражают прямые воздействия, которые интуитивно понятны и очень просты. Отношение управления возникает тогда, когда выход одного блока непосредственно влияет на блок с меньшим доминированием. Например, блок A1 влияет на блоки A2 и A3 (см. рис. 3.3). Отношение входа возникает тогда, когда выход одного блока становится вхо-

дом для блока с меньшим доминированием, например, выход блока А2 становится входом функции А3. Обратная связь по управлению и обратная связь по входу являются более сложными, поскольку они представляют итерацию или рекурсию, т.е. выходы из одной функции влияют на будущее выполнение других функций, что впоследствии влияет на исходную функцию. Обратная связь по управлению возникает тогда, когда выход некоторого блока влияет на блок с большим доминированием. Связь по входной обратной связи имеет место тогда, когда выход одного блока становится входом другого блока с большим доминированием.

Связи "выход-механизм" встречаются нечасто и представляют особый интерес. Они отражают ситуацию, при которой выход одной функции становится средством достижения цели для другой. Например, на рис. 3.3 представлена функция 3 «Подготовить рабочее место», имеющая выход «оборудованное рабочее место», который, в свою очередь, является механизмом для блока 4. Это означает, что оборудованное рабочее место необходимо для того, чтобы начать процесс обработки. В этом случае дуга механизма обозначает строго последовательную взаимосвязь: приготовления должны быть завершены до начала работы. Поэтому связи "выход-механизм" характерны при распределении источников ресурсов (например, требуемые инструменты, обученный персонал, физическое пространство, оборудование, финансирование, материалы).

Разветвления дуг, изображаемые в виде расходящихся линий, означают, что все содержимое дуг или его часть может появиться в каждом ответвлении дуги. Дуга всегда помечается до разветвления, чтобы дать название всему набору. Кроме того, каждая ветвь дуги может быть помечена или не помечена в соответствии со следующими правилами:

– непомеченные ветви содержат все объекты, указанные в метке дуги перед разветвлением (т.е. все объекты принадлежат этим ветвям);

– ветви, помеченные после точки разветвления, содержат все объекты или их часть, указные в метке дуги перед разветвлением (т.е. каждая метка ветви уточняет, что именно содержит ветвь).

Слияние дуг в SADT, изображаемое как сходящиеся вместе линии, указывает, что содержимое каждой ветви идет на формирование метки для дуги, являющейся результатом слияния исходных дуг. После слияния результирующая дуга всегда помечается для указания нового набора объектов, возникшего после объединения. Кроме того, каждая ветвь перед слиянием может помечаться или не помечаться в соответствии со следующими правилами:

– непомеченные ветви содержат все объекты, указанные в общей метке дуги после слияния (т.е. все объекты исходят из всех ветвей);

– помеченные перед слиянием ветви содержат все или некоторые объекты из перечисленных в общей метке после слияния (т.е. метка ветви ясно указывает, что содержит ветвь).

Хорошая методология структурного анализа, позволяющая создавать отдельные диаграммы, должна гарантировать правильное соединение всех диаграмм для образования согласованной модели. SADT-диаграммы имеют внешние дуги – дуги, как бы выходящие наружу и ведущие к краю страницы. Эти дуги являются интерфейсом между диаграммой и остальной частью модели. SADT требует, чтобы все внешние дуги диаграммы были согласованы с дугами, образующими границу этой диаграммы. Другими словами, диаграмма должна быть "состыкована" со своей родительской диаграммой. Обычно это означает, что внешние дуги согласованы по числу и наименованию (но не обязательно по расположению) с дугами, касающимися декомпозированного блока родительской диаграммы. В SADT принята система обозначений, позволяющая аналитику точно идентифицировать и проверять связи по дугам между диаграммами. Эта схема кодирования дуг –"ICOM" - получила название по первым буквам английских эквивалентов слов вход (Input), управление (Control), выход (Output), механизм (Mechanism). Коды ICOM чрезвычайно эффективны, поскольку они позволяют аналитику быстро проверять согласованность внешних дуг диаграммы с граничными дугами соответствующего блока родительской диаграммы. Они также обеспечивают согласованность декомпозиции, поскольку все дуги, входящие в диаграмму и выходящие из нее, должны быть учтены

(см. рис. 3.3). Одним из способов такой стыковки может служить присваивание кодов ICOM внешним дугам новой диаграммы согласно следующим правилам:

– представьте себе рисунок новой диаграммы внутри разлагаемого блока. Продлите внешние дуги почти до края диаграммы. Зрительно соедините каждую внешнюю дугу диаграммы с соответствующей граничной дугой декомпозируемого блока.

– присвойте код каждой зрительной связи. Используйте I для входных дуг, C - для связей между дугами управления, O - для связей между выходными дугами, M - для связей между дугами механизма.

– добавьте после каждой буквы цифру, соответствующую положению данной дуги среди других дуг того же типа, касающихся родительского блока. Причем входные и выходные дуги пересчитываются сверху вниз, а дуги управлений и механизмов пересчитываются слева направо. Теперь запишите каждый код около окончания каждой внешней дуги (см. рис. 3.3).

3.3 Моделирование потоков данных

В основе данной методологии лежит построение модели анализируемой ИС – проектируемой или реально существующей. В соответствии с методологией модель системы определяется как иерархия диаграмм потоков данных, описывающих процесс преобразования информации от ее ввода в систему до выдачи пользователю. Диаграммы верхних уровней иерархии (контекстные диаграммы) определяют основные процессы или подсистемы ИС с внешними входами и выходами. Они детализируются при помощи диаграмм нижнего уровня. Такая декомпозиция продолжается, создавая многоуровневую иерархию диаграмм, до тех пор, пока не будет достигнут такой уровень декомпозиции, на котором процесс становятся элементарными и детализировать их далее невозможно.

Источники информации (внешние сущности) порождают информационные потоки, переносящие информацию к подсистемам или процессам. Те, в свою очередь, преобразуют информацию и порождают новые потоки, которые переносят информацию к другим процессам или подсистемам, накопителям данных или внешним сущностям - потребителям информации. Таким образом, основными компонентами диаграмм потоков данных являются: внешние сущности; системы/подсистемы; процессы; накопители данных; потоки данных.

Внешняя сущность представляет собой материальный предмет или физическое лицо, представляющее собой источник или приемник информации, например, заказчики, поставщики, клиенты. Определение некоторого объекта или системы в качестве внешней сущности указывает на то, что она находится за пределами границ анализируемой ИС. В процессе анализа некоторые внешние сущности могут быть перенесены внутрь диаграммы анализируемой ИС, если это необходимо, или, наоборот, часть процессов ИС может быть вынесена за пределы диаграммы и представлена как внешняя сущность.

Внешняя сущность обозначается квадратом (рис. 3.4), расположенным как бы "над" диаграммой и бросающим на нее тень, для того, чтобы можно было выделить этот символ среди других обозначений.

Рис. 3.4 – Внешняя сущность

При построении модели сложной ИС она может быть представлена в самом общем виде на так называемой контекстной диаграмме в виде одной системы как единого целого, либо может быть декомпозирована на ряд подсистем.

Вид подсистемы (или системы) на контекстной диаграмме изображен на рис. 3.5. Номер подсистемы служит для ее идентификации. В поле имени вво-

дится наименование подсистемы в виде предложения с подлежащим и соответствующими определениями и дополнениями.

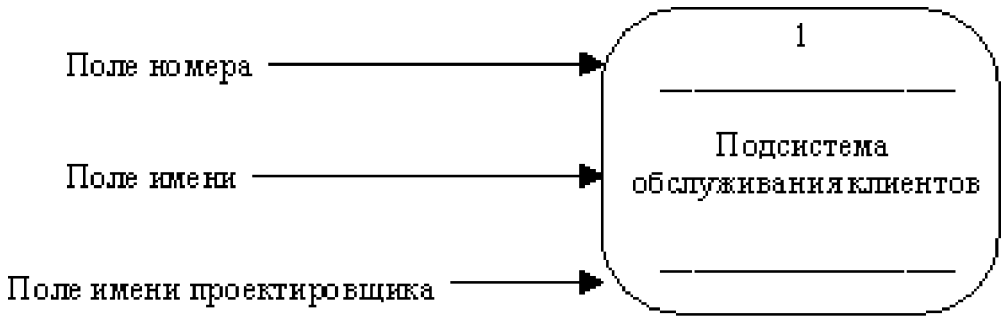

Рис. 3.5 – Подсистема

Процесс представляет собой преобразование входных потоков данных в выходные в соответствии с определенным алгоритмом. Физически процесс может быть реализован различными способами: это может быть подразделение организации, выполняющее обработку входных документов и выпуск отчетов; программа; аппаратно реализованное логическое устройство и т.д. Процесс на диаграмме потоков данных изображается, как показано на рис. 3.6.

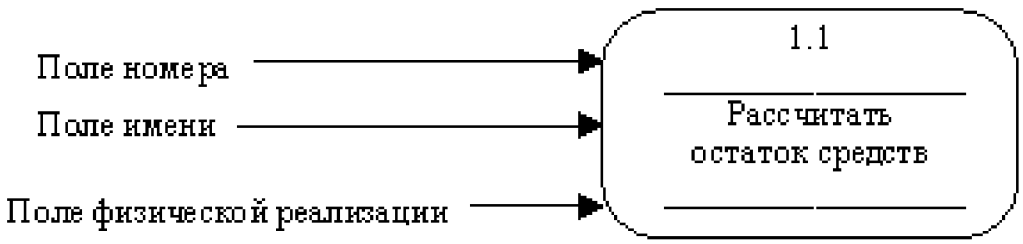

Рис. 3.6 – Процесс

Номер процесса служит для его идентификации. В поле имени вводится наименование процесса в виде предложения с активным недвусмысленным глаголом в неопределенной форме (вычислить, рассчитать, проверить, определить, создать, получить), за которым следуют существительные в винительном падеже, например: "Ввести сведения о клиентах", "Выдать информацию о текущих расходах", "Проверить кредитоспособность клиента".

Информация в поле физической реализации показывает, какое подразделение организации, программа или аппаратное устройство выполняет данный процесс.

Накопитель данных представляет собой абстрактное устройство для хранения информации, которую можно в любой момент поместить в накопитель и через некоторое время извлечь, причем способы помещения и извлечения могут быть любыми.

Накопитель данных может быть реализован физически в виде ящика в картотеке, таблицы в оперативной памяти, файла на магнитном носителе и т.д. Накопитель данных на диаграмме потоков данных изображается, как показано на рис. 3.7.

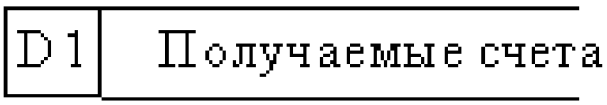

Рис. 3.7 – Накопитель данных

Накопитель данных идентифицируется буквой "D" и произвольным числом. Имя накопителя выбирается из соображения наибольшей информативности для проектировщика. Накопитель данных в общем случае является прообразом будущей базы данных и описание хранящихся в нем данных должно быть увязано с информационной моделью.

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

Поток данных на диаграмме изображается линией, оканчивающейся стрелкой, которая показывает направление потока (рис. 3.8). Каждый поток данных имеет имя, отражающее его содержание.

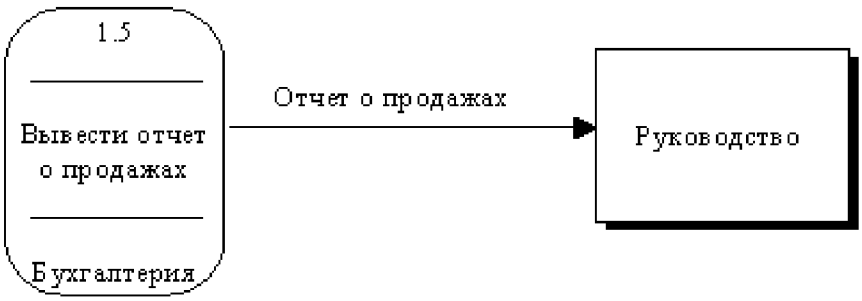

Рис. 3.8 – Поток данных

Первым шагом при построении иерархии диаграмм потоков данных является построение контекстных диаграмм. Обычно при проектировании относительно простых ИС строится единственная контекстная диаграмма в которой находится так называемый главный процесс, соединенный с приемниками и источниками информации, посредством которых с системой взаимодействуют пользователи и другие внешние системы.

Если же для сложной системы ограничиться единственной контекстной диаграммой, то она будет содержать слишком большое количество источников и приемников информации, которые трудно расположить на листе бумаги нормального формата, и кроме того, единственный главный процесс не раскрывает структуры распределенной системы. Признаками сложности (в смысле контекста) могут быть:

– наличие большого количества внешних сущностей;

– распределенная природа системы;

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

Для сложных ИС строится иерархия контекстных диаграмм. При этом контекстная диаграмма верхнего уровня содержит не единственный главный процесс, а набор подсистем, соединенных потоками данных. Контекстные диаграммы следующего уровня детализируют контекст и структуру подсистем.

Иерархия контекстных диаграмм определяет взаимодействие основных функциональных подсистем проектируемой ИС как между собой, так и с внешними входными и выходными потоками данных и внешними объектами (ис-

точниками и приемниками информации), с которыми взаимодействует ИС. Разработка контекстных диаграмм решает проблему строгого определения функциональной структуры ИС на самой ранней стадии ее проектирования, что особенно важно для сложных многофункциональных систем, в разработке которых участвуют разные организации и коллективы разработчиков. После построения контекстных диаграмм полученную модель следует проверить на полноту исходных данных об объектах системы и изолированность объектов (отсутствие информационных связей с другими объектами).

Для каждой подсистемы, присутствующей на контекстных диаграммах, выполняется ее детализация при помощи DFD. Каждый процесс, в свою очередь, может быть детализирован при помощи DFD или мини-спецификации. При детализации должны выполняться следующие правила:

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

– правило нумерации - означает, что при детализации процессов должна поддерживаться их иерархическая нумерация. Например, процессы, детализирующие процесс с номером 12, получают номера 12.1, 12.2, 12.3 и т.д.

Мини-спецификация должна формулировать его основные функции таким образом, чтобы в дальнейшем специалист, выполняющий реализацию проекта, смог выполнить их или разработать соответствующую программу. Мини-спецификация является конечной вершиной иерархии DFD. Решение о завершении детализации процесса и использовании мини-спецификации принимается исходя из следующих критериев:

– наличия у процесса относительно небольшого количества входных и выходных потоков данных (2-3 потока);

– возможности описания преобразования данных процессом в виде известного последовательного алгоритма;

– выполнения процессом единственной логической функции преобразования входной информации в выходную;

– возможности описания логики процесса при помощи мини-спецификации небольшого объема (не более 20-30 строк).

После построения законченной модели системы ее необходимо верифицировать (проверить на полноту и согласованность). В полной модели все ее объекты (подсистемы, процессы, потоки данных) должны быть подробно описаны и детализированы. Выявленные не детализированные объекты следует детализировать, вернувшись на предыдущие шаги разработки.

В согласованной модели для всех потоков данных и накопителей данных должно выполняться правило сохранения информации: все поступающие куда-либо данные должны быть считаны, а все считываемые данные должны быть записаны.

4 ПРИНЦИПЫ КЛАССИФИКАЦИИ ИНФОРМАЦИИ В СЛОЖНЫХ СИСТЕМАХ

4.1 Основные принципы классификации экономической информации

Для того, чтобы приспособить информацию для эффективного поиска, обработки на ЭВМ и передаче по каналам связи, ее необходимо представить в цифровом виде. С этой целью ее нужно сначала упорядочить (классифицировать), а затем формализовать (закодировать) с помощью классификатора.

Классификатор – это объект, с помощью которого осуществляется формализованное описание экономической информации в ИС, содержащей наименования классификационных группировок и их кодовые обозначения. Экономическая информация существует в двух формах: в форме экономических показателей и в форме документов [4].

Экономический показатель является составной единицей информации, отражающей количественную характеристику некоторого процесса предметной области – это реквизит-основание вместе с однозначно определяющими его качество реквизитами-признаками.

Реквизиты-основания подразделяются по типу алгоритмов их получения на количественные, стоимостные, проценты, удельные веса и др. Множество реквизитов-признаков по степени формализации делятся на два подмножества:

– справочные реквизиты-признаки – как правило, наименования предназначены для понимания показателя человеком;

– группировочные реквизиты-признаки – это закодированные аналоги справочных признаков, предназначенные для обработки информации на компьютере.

Основными объектами классификации и кодирования являются справочные реквизиты-признаки, описывающие процессы, место, время выполнения процессов, субъекты и объекты действия, отражаемые в показателе. Например, к числу наименований элементов можно отнести наименования материальных,

трудовых, денежных, энергетических ресурсов, основных средств, готовой продукции и услуг. К числу наименований процессов относятся наименования функций управления, деловых процессов, операций поступления сырья и материалов, расчеты с поставщиками и т.д.

К объектам классификации и кодирования относятся также наименования показателей и документов. Помимо этого к объектам классификации и кодирования относят также наименования компонентов проекта ИС, в том числе файлов, задач, подсистем, программных модулей и др.

Целью разработки классификаторов является установление соответствия между значениями справочных или описательных признаков какого-либо элемента или процесса и значениями группировочных признаков, например, между значением реквизита «Фамилия И.О. рабочего» и значением «Табельный номер рабочего» или между значениями «Наименование материала» и «Код материала».

Для кодирования объектов необходимо их упорядочить по некоторым признакам. Результат упорядоченного распределения объектов заданного множества носит название *классификации*, а совокупность правил распределения объектов множества на подмножества называется *системой классификации*. Процесс распределения объектов классификации в соответствии с принятой системой классификации носит название процесса *классифицирования*. То свойство или характеристика объекта классификации, которое позволяет установить его сходство или различие с другими объектами классификации, называется *признаком классификации*. Множество или подмножество, объединяющее часть объектов классификации по одному или нескольким признакам, носит название *классификационной группировки*.

Основанием классификации называется признак, по которому ведется разбиение множества на подмножества на определенной ступени классификации. *Ступень классификации* – это результат очередного распределения объектов одной классификационной группировки. *Уровень классификации* – это совокупность классификационных группировок, расположенных на одних и тех же

ступенях классификации. *Глубина системы классификации* – это количество уровней классификации, допустимое в данной системе.

Каждая система классификации характеризуется следующими свойствами: гибкостью системы, емкостью системы и степенью заполненности системы. Гибкость системы – это способность допускать включение новых признаков без разрушения структуры классификатора. Емкость системы – это наибольшее количество классификационных группировок, допускаемое в данной системе классификации. Степень заполненности системы $K_{зап}$ определяется как частное от деления фактического количества группировок $Q_ф$ на величину емкости системы P: $K_{зап} = Q_ф / P$.

В настоящее время чаще всего применяются два типа систем классификации: иерархическая и многоаспектная. Характерными особенностями иерархической системы являются:

– наличие в системе неограниченного количества признаков классификации;

– соподчиненность признаков классификации, что выражается разбиением каждой классификационной группировки, образованной по одному признаку, на множество классификационных группировок по подчиненному признаку.

При построении иерархической системы классификации сначала выделяется некоторое множество объектов, подлежащее классифицированию M_o, для которого определяются полное множество признаков классификации G и их соподчиненность друг другу, затем производится разбиение исходного множества объектов на классификационные группировки на каждой ступени классификации (рис. 4.1).

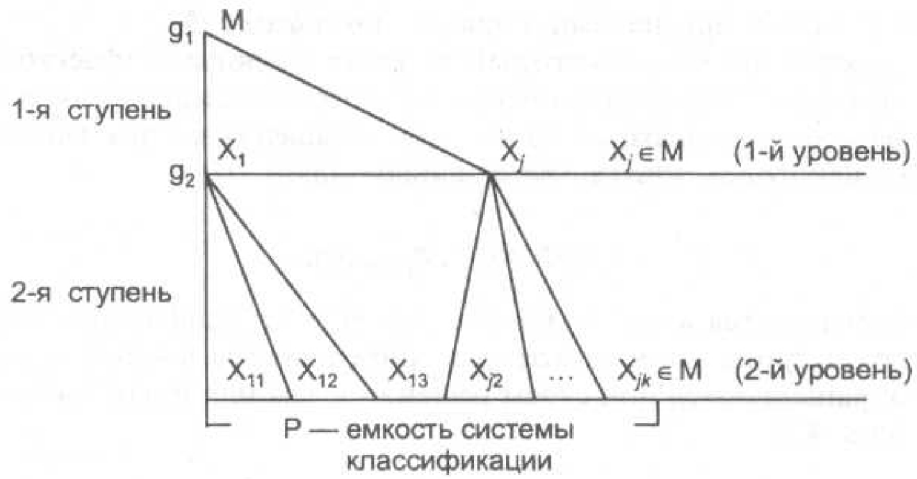

Рис. 4.1 – Схема построения иерархической системы классификации.

$M_0 = \{x_1, x_2, \bullet\ , x_i, \bullet\ , x_n\}$ – мощность классифицируемого множества; $g_1, g_2, \bullet$ – признаки классификации и $g_1 \in G, g_2 \in G$.

При использовании иерархической системы классификации необходимо соблюдать следующие ограничения:

– получающиеся на каждом уровне классификационные группировки должны составлять исходное множество объектов M_0;

– классификационные группировки X_{jk} на каждой ступени не должны пересекаться;

– классификация на каждой ступени должна проводиться только по одному признаку.

К положительным сторонам данной системы следует отнести логичность, простоту ее построения и удобство логической и арифметической обработки.

Однако эта система характеризуется жесткой структурой классификации, не позволяющей вносить новые признаки или изменять их последовательность. Гибкость этой системы обеспечивается только за счет ввода большой избыточности в ветвях, что приводит к слабому наполнению структуры классификатора.

Недостатки, отмеченные в иерархической системе, отсутствуют в других системах, которые относятся к классу многоаспектных систем классификации. Аспект – точка зрения на объект классификации, который характеризуется одним или несколькими признаками. Многоаспектная система – это система классификации, которая использует параллельно несколько независимых признаков (аспектов) в качестве основания классификации.

Существуют два типа многоаспектных систем: фасетная и дескрипторная. Фасет - это аспект классификации, который используется для образования независимых классификационных группировок. Дескриптор - ключевое слово, определяющее некоторое понятие, которое формирует описание объекта и дает принадлежность этого объекта к классу, группе и т.д.

Фасетная система характеризуется следующими особенностями построения (рис. 4.2):

– имеется некоторое множество классифицируемых объектов M_0;

– это множество можно рассматривать в нескольких аспектах, каждый из которых может характеризоваться одним или несколькими признаками, образующими фасет Φ_r;

– устанавливается некоторый порядок следования фасетов с помощью фасетной формулы (при этом последовательность фасетов определяется по частоте обращения к этим фасетам на некотором множестве заданных задач): $F = (\Phi_1, \Phi_2, \bullet\ , \Phi_r, \bullet\ \Phi_R)$;

– определяется количество подмножеств классификационных группировок, число которых определяется числом задач, обращающихся при своем решении к тем или иным фасетам.

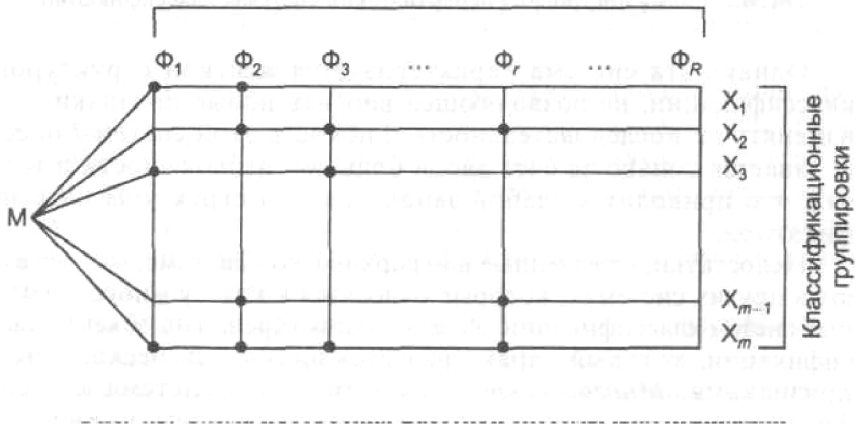

Рис. 4.2 – Схема построения фасетной системы классификации

Внутри фасета значения признаков могут просто перечисляться по некоторому порядку или образовывать сложную иерархическую структуру, если существует соподчиненность выделенных признаков.

К преимуществам данной системы следует отнести большую емкость системы и высокую степень гибкости, поскольку при необходимости можно вводить дополнительные фасеты и изменять их место в формуле. К недостаткам, характерным для данной системы, можно отнести сложность структуры и низкую степень наполненности системы.

Рассмотренные выше системы классификации хорошо приспособлены для организации поиска с целью последующей логической и арифметической обработки информации на ЭВМ и лишь частично решают проблему содержательного поиска экономической информации при принятии управленческих решений. Это объясняется далеко не полным охватом этими системами всех понятий и терминов, используемых для выражения смысла экономических показателей и документов. Помимо этого в этих системах не решается проблема обеспечения однозначности используемой терминологии, идентификации роли отдельных терминов в их общей последовательности при формировании наименований экономических показателей. К недостаткам этих систем классификации можно отнести также и то, что в них не отражаются все отношения между терминами, необходимые для формализации содержания показателей и документов и уста-

новления взаимосвязей между показателями и документами, которые используются на этапе принятия управленческих решений.

Для поиска показателей и документов по набору содержательных признаков используется информационный язык дескрипторного типа, который характеризуется совокупностью терминов, дескрипторов или лексикой и набором отношений между терминами [4]. Эти отношения могут быть двух типов:

– постоянные логические отношения между терминами, вытекающие из отношений между отображаемыми объектами, которые называются *парадигматическими отношениями*;

– переменные отношения между понятиями, возникающие в процессе построения конкретного высказывания, например показателя, называемые *синтагматическими отношениями*.

Парадигматические отношения между терминами отражают статику языка. К ним относятся, например, родовидовые отношения. При этом родовым называется термин или понятие, выражающие существенные признаки класса предметов, в состав которого входят предметы, являющиеся видами этого рода. Видовое понятие выражает существенные признаки подкласса предметов, являющегося видом какого-либо другого класса предметов и входящего в состав этого класса. Например, понятие «машинный носитель» является родовым по отношению к понятиям «жесткий магнитный диск», «гибкий диск», «магнитная лента» и т.д. Отношения этого типа отражаются в классификаторах экономической информации.

Синтагматические отношения составляют грамматику этого языка, т.е. правила построения высказываний из набора терминов или понятий. Такие отношения используются в динамике при вводе данных и формулировании запросов.

В зависимости от того, на каком этапе фиксируются все возможные выражения, языки делятся на *предкоординированные* и *посткоординируемые*. Предкоординированными называются языки, в которых на стадии разработки выделяются все высказывания в терминах этих языков и тем самым заранее опреде-

ляются постоянные отношения между терминами. Для посткоординируемых языков характерна предварительная фиксация лишь постоянных отношений. Все высказывания образуются при использовании лексики данного языка и его грамматики. Языки предкоординированного типа менее гибки при использовании, так как с их помощью можно описывать только те выражения, которые были заранее зафиксированы. Использование посткоординированных языков позволяет образовывать с их помощью значительно большее число высказываний.

К языкам посткоординированного типа относятся дескрипторные языки, основанные на применении метода координатного, или ассоциативного, индексирования.

Согласно идее координатного индексирования предполагается, что содержание документов или показателей можно достаточно полно и точно отразить с помощью списка ключевых слов - дескрипторов. Дескриптор - это термин естественного языка (слово или словосочетание), используемый при описании документов или показателей, который имеет самостоятельный смысл и неделим без изменения своего значения. Например, показатель «Количество продукции, выработанное фактически цехом за смену», записанный на естественном языке, при использовании метода координатного индексирования будет иметь вид: «количество, продукция, выработка, фактический, цех, смена».

Для того чтобы обеспечить точность и однозначность поиска с помощью такого языка, необходимо предварительно определить все постоянные отношения между терминами: родовидовые, отношения синонимии, омонимии, а также ассоциативные отношения.

Синонимия - это отношение между двумя и более различными ключевыми словами, когда они имеют одинаковое значение, обозначают один и тот же предмет или понятие, например, «производство» и «произведено». К синонимам относятся также термины, которые могут существовать как в полном, так и в сокращенном виде, например, «научно-исследовательские работы» и «НИР», «кубические метры» и «куб. м».

Омонимия - это такое отношение между одинаковыми по звучанию и написанию ключевыми словами, когда они имеют разное значение и обозначают разные предметы и понятия. Например, термин «прокат» используется в двух различных смыслах: «прокат тонкой листовой стали» и «сдача предметов во временное пользование».

Большое значение для построения дескрипторного языка имеют выявление и фиксирование ассоциативных отношений между терминами, которые позволяют выдавать более точные ответы на запросы пользователей. К числу ассоциативных отношений относят такие, как отношение части к целому (например, «цех» – «участок»), причинно-следственные отношения (например, «прогул» – «невыполнение»), связи предмета и процесса (например, «план» – «планирование») и др.

Все выделенные отношения явно описываются в систематическом словаре понятий - тезаурусе, который разрабатывается с целью проведения индексирования документов, показателей и информационных запросов.

В свою очередь, дескрипторные языки различаются по семантической силе, которая определяется тем, какой объем сведений может индексироваться с их применением. Семантическая сила языка зависит от числа типов постоянных отношений, фиксируемых в тезаурусе, а также от наличия средств грамматики и степени их сложности. В соответствии с этим признаком дескрипторные языки подразделяются на языки без грамматики, языки с неполной грамматикой и языки с развитой грамматикой. При этом языки первого вида содержат только словари используемых ключевых слов и тезаурусы. В языках с неполной грамматикой, помимо словарей и тезаурусов, имеются правила взаимосвязи только некоторых категорий терминов. Языки с развитой грамматикой позволяют описывать с помощью всех средств сложные высказывания.

В том случае, если объектом поиска в ИС является документ, для этих целей используют информационные языки дескрипторного типа без грамматики. При необходимости хранения и осуществления поиска экономических показателей проектировщики отдают предпочтение языкам второго и третьего типов.

4.2 Единая система классификации и кодирования

Для обеспечения информационной совместимости ИС разных уровней разработана Единая система классификации и кодирования (ЕСКК). ЕСКК предназначена для выполнения следующих функций [4]:

– централизованной разработки общероссийских классификаторов;

– пополнения и обновления, своевременного и систематического оповещения организаций обо всех изменениях, внесенных в классификаторы;

– ответов на разовые запросы;

– оптимизации структуры классификаторов;

– проведения работы по созданию информационно-поисковых языков.

В состав ЕСКК входят три составные части. Первая ее часть «Комплекс нормативно-технических и методологических материалов» включает в себя документы, которые регламентируют:

– состав системы, цели системы, задачи и всю используемую терминологию системы;

– принципы и методы классификации и кодирования;

– категории и сферы действия классификаторов;

– принципы сопряжения и взаимодействия классификаторов;

– структуру работ по созданию и внедрению системы.

Второй частью является «Комплекс общероссийских классификаторов (ОК)», в который входят следующие группы классификаторов:

1. Классификаторы о природных и трудовых ресурсах: профессии рабочих; должности служащих; кадров; специальностей; полезных ископаемых и т.д.

2. Классификаторы о продуктах труда и производственной деятельности: промышленной и сельскохозяйственной продукции; строительной продукции; деталей; услуг: в промышленности, строительстве, сельском хозяйстве, транспорте, материально-техническом снабжении; услуг населению.

3. Классификаторы структуры народного хозяйства и объектов административно-территориального деления: предприятий и организаций; отраслей на-

родного хозяйства; стран; органов государственного управления; объектов административно-территориального деления; пунктов погрузки и разгрузки.

4. Классификаторы управленческой информации и документации: единиц измерения; технико-экономических показателей; управленческой документации; технической документации, обозначений стандартных и технических условий; технологической документации; операций и деталей.

Третьей частью ЕСКК является автоматизированная система ведения общероссийских классификаторов (АСВОК), которая состоит из трех типа подсистем: объектных, функциональных и обеспечивающих.

Объектные подсистемы объединяют предприятия, отрасли, отраслевые институты, которые отвечают за передачу информации об изменениях, происходящих в заданной номенклатуре, число которых может быть равно числу общероссийских классификаторов.

Функциональные подсистемы объединяют однотипные технологические процессы по ведению общероссийских классификаторов и включают в свой состав подсистемы сбора, хранения, внесение коррективов; регулярного обслуживания абонентов; обслуживания по разовым запросам; развития АСВОК, включая оптимизацию структуры классификаторов, устранение недействительных ветвей классификаторов, стандартизацию терминологии.

Обеспечивающие подсистемы состоят из типового набора подсистем, к которым относят программное, техническое, информационное и лингвистическое обеспечение. В состав информационного обеспечения АСВОК входят: тезаурус; сводные эталонные файлы классификаторов; дополнительные эталонные файлы дополнений и исключаемых позиций; файлы поисковых образов позиций классификаторов; файлы незанятых позиций; таблицы сопряжения классификаторов; вспомогательный файл организаций, ответственных за ведение классификаторов; таблицы периодичности оповещения организаций и вспомогательные файлы интересов абонентов.

4.3 Унифицированная система документации

Основной компонентой внемашинного информационного обеспечения ИС является система документации, применяемая в процессе управления экономическим объектом [4]. Под *документом* понимается определенная совокупность сведений, используемая при решении экономических задач, расположенная на материальном носителе в соответствии с установленной формой. Документ рассматривается как специальный знак экономического языка, имеющий единство формы, содержания и материального носителя и обладающий следующими свойствами:

– многофункциональности, поскольку документ может предназначаться для выполнения функций регистрации информации о состоянии элементов и процессов, происходящих в экономической системе, для обработки, хранения этой информации и для передачи ее на расстояние;

– наличия юридической силы, обеспечиваемой присутствием подписей должностных лиц, благодаря которым подтверждается достоверность содержащейся в документе информации.

Система документации – это совокупность взаимосвязанных форм документов, регулярно используемых в процессе управления экономическим объектом.

Существующие системы документации, характерные для неавтоматизированных ИС, отличаются большим количеством разных типов форм документов; большим объемом потоков документов и их запутанностью; дублированием информации в документах и работ по их обработке и, как следствие, низкой достоверностью получаемых результатов. Обработка документов в таких системах занимает более 40% времени работников управления [4]. Для того, чтобы упростить систему документации, используют следующие два подхода:

– проведение унификации и стандартизации документов;

– введение безбумажной технологии, основанной на использовании электронных документов и новых информационных технологий их обработки.

Унификация документов выполняется путем введения единых форм документов в результате осуществления синтаксической и семантической унификации. Таким образом, вводится единообразие в наименования показателей, единиц измерения и терминов, в результате чего получается *унифицированная система документации* (УСД).

Унифицированная система документации – это рационально организованный комплекс взаимосвязанных документов, который отвечает единым правилам и требованиям и содержит информацию, необходимую для оптимального управления некоторым экономическим объектом.

По уровням управления, для которых разрабатываются УСД, они делятся на межотраслевые системы документации, используемые на всех предприятиях стран; отраслевые, применяемые только на предприятиях конкретной отрасли; и системы документации локального уровня, т.е. обязательные для использования в рамках организаций.

В настоящее время разработаны следующие виды УСД на межотраслевом уровне:

– стандарты и технические условия;

– проектно-конструкторская и технологическая документация;

– проектная документация по капитальному строительству;

– плановая документация;

– статистическая отчетность;

– первичная учетная документация;

– финансовая первичная и отчетная документация;

– бухгалтерская документация бюджетных организаций и объединений;

– организационно-распорядительная документация;

– документация по материально-техническому снабжению;

– документация по ценообразованию и торговле.

Любой тип УСД должен удовлетворять следующим требованиям:

– документы, входящие в состав УСД, должны разрабатываться с учетом их использования в системе взаимосвязанных ИС;

– УСД должна содержать полную информацию, необходимую для оптимального управления тем объектом, для которого разрабатывается эта система;

– УСД должна быть ориентирована на использование средств вычислительной техники для сбора, обработки и передачи информации;

– УСД должна обеспечить информационную совместимость ИС различных уровней;

– все документы, входящие в состав разрабатываемой УСД, и все реквизиты-признаки в них должны быть закодированы с использованием международных, общесистемных или локальных классификаторов.

4.4 Проектирование унифицированной системы документации

При разработке системы документации в ИС проектировщик должен решить следующие проблемы: спроектировать и унифицировать новые документы; отобрать документы, которые будут использовать в ИС без изменений; выявить в существующей системе те документы, которые надо унифицировать.

В данном процессе проектирования можно выделить три этапа работ [4]:

– построение новых форм документов;

– унификация всей системы документации;

– разработка инструкций и методических материалов, регламентирующих работу пользователей с системой документации.

Рассмотрим работы, которые выполняются на каждом этапе (идут строго в порядке нумерации):

1. Определение состава результатных показателей.
2. Определение состава первичных показателей.
3. Разбиение показателей по формам документов.
4. Выбор типа формы документа.
5. Определение способа нанесения информации в документы.
6. Проектирование форм документов.

7. Выявление и анализ полной системы документации.

8. Составление перечня документов по функциональным подсистемам.

9. Выявление характеристик документов.

10. Исключение производных и многократно вводимых в компьютер показателей.

11. Введение единой терминологии путем составления словаря.

12. Установление единых единиц измерения.

13. Классификация и кодирование документов и реквизитного состава документов.

14. Уточнение форм и построение единых форм документов.

15. Разработка правил заполнения и использования документов.

16. Построение схем документооборота.

17. Утверждение форм документов, их размножение и составление инструкций для работы с ними.

На первом этапе выполняются работы с 1 по 6, на втором – с 7 по 14 и на третьем – с 15 по 17.

Более подробно рассмотрим выполнение шестой работы: проектирование форм первичных документов и форм документов результатной информации.

Особенности проектирования форм первичных документов. Первичные документы предназначены для отражения процессов в материальной сфере и поставляют всю постоянную и оперативную информацию, необходимую для решения экономических задач и выработки управленческих решений. К числу основных требований, предъявляемых к первичным документам, можно отнести следующие: не избыточность и полноту информации для решения задач, высокую достоверность и своевременность собираемой информации. Кроме того, первичная информация должна быть расположена в документе таким образом, чтобы учитывались требования удобства для последующей обработке данных в компьютере.

При проектировании форм первичных документов должны учитываться следующие принципы:

– отсутствие в первичных документах постоянной информации, для которой необходимо создание самостоятельных файлов;

– отсутствие дублирования показателей в документах;

– выделение реквизитов, имеющих одно или несколько значений на документ, т.е. выделение однозначных и многозначных реквизитов;

– выделение справочных, групповых реквизитов и реквизитов-оснований;

– логичность построения, т.е. старшие по объему понятий признаки должны предшествовать младшим (например, наименование предприятия → наименование цеха → номер участка);

– согласование последовательности реквизитов в документе с макетами размещения информации на экране компьютера и в файлах.

Процесс разработки первичных документов имеет особенности в каждой организации и выполняется в следующей последовательности:

1. Определение полного реквизитного состава каждого документа.

2. Классификация реквизитов: однозначные и многозначные, признаки и основания, справочные и группировочные, переносимые и непереносимые на машинные носители.

3. Установление логической соподчиненности реквизитов первичных документов.

4. Выбор какой-либо формы первичного документа.

5. Осуществление размещения реквизитов по выбранной форме в соответствии с проведенной классификацией.

6. Выполнение расчета размеров документа по вертикали и горизонтали с учетом размера полей.

7. Выбор формата бумажного носителя.

8. Построение эскиза документа соответствующей формы.

9. Выделение толстой линией реквизитов, переносимых на машинный носитель.

10. Редактирование шапок документов в соответствии со словарем.

Особенности проектирования форм документов результатной информации. В результате решения задачи рассчитываются результатные показатели, которые требуется выдать на материальный носитель в виде, удобном для пользователя. Так как результатный документ используется для осуществления процессов управления, то он должен отвечать следующим требованиям:

– полнота информации, т.е. результатные документы должны содержать в себе первичные и результатные показатели;

– количество результатных показателей должно соответствовать количеству группировочных признаков;

– своевременность предоставления информации;

– достоверность предоставляемой информации;

– хорошая читаемость, т.е. логичность построения форм и наличие хорошо отредактированного текста шапок документов;

– отсутствие показателей, рассчитываемых вручную.

Построение результатных документов должно выполнятся в следующей последовательности:

1. Определение полного реквизитного состава документа.

2. Классификация реквизитов-признаков на справочные и группировочные, реквизитов-оснований на первичные и результатные, а результатных оснований по степеням итогов.

3. Выбор формы документа.

4. Размещение реквизитов в форме согласно их логической соподчиненности.

5. Подсчет длины строки $L_{док}$ в табличной зоне с учетом пробелов между реквизитами и разделительных линий граф по формуле $L_{док} = L_1 + L_2 + \bullet \ + L_n + k \cdot d$, где L_i – длина i-го реквизита, k – число колонок в таблице, d – число пробелов между колонками.

Если длина строки документа больше ширины каретки печатающего устройства (с учетом возможного уменьшения размеров шрифта), то перегруппи-

ровка реквизитов таблицы осуществляется с использованием следующих методов:

– вынос итоговых колонок в итоговые строки;

– перенос не уместившихся в листе колонок на новый лист с продолжением нумерации колонок (такие документы затем склеиваются).

При этом осуществляется выделение в верхней правой части документа специальной области для служебных реквизитов (количество листов в документе, номер текущего листа, количество экземпляров, номер экземпляра).

5 МЕТОДОЛОГИИ ПРОЕКТИРОВАНИЯ СЛОЖНЫХ СИСТЕМ

5.1 Методология RAD

Одним из возможных подходов к разработке ИС в рамках спиральной модели ЖЦ является получившая в последнее время широкое распространение методология быстрой разработки приложений RAD (Rapid Application Development) [5]. Под этим термином обычно понимается процесс разработки ИС, содержащий три элемента:

– небольшое количество разработчиков (от 2 до 10 человек);

– короткий, но тщательно проработанный производственный график (от 2 до 6 мес.);

– повторяющийся цикл, при котором разработчики, по мере того, как ИС начинает обретать форму, запрашивают и реализуют требования, полученные через взаимодействие с заказчиком.

Команда разработчиков должна представлять из себя группу профессионалов, имеющих опыт в анализе, проектировании, генерации кода и тестировании ИС с использованием CASE-средств. Члены коллектива должны также уметь трансформировать в рабочие прототипы предложения конечных пользователей.

Жизненный цикл ИС методологии RAD состоит из четырех фаз: фаза анализа и планирования требований; фаза проектирования; фаза реализации; фаза внедрения.

На фазе анализа и планирования требований пользователи системы определяют функции, которые она должна выполнять, выделяют наиболее приоритетные из них, требующие проработки в первую очередь, описывают информационные потребности. Определение требований выполняется в основном силами пользователей под руководством специалистов-разработчиков. Ограничивается масштаб проекта, определяются временные рамки для каждой из последующих фаз. Кроме того, определяется сама возможность реализации данного проекта в установленных рамках финансирования, на данных аппаратных сред-

ствах. Результатом данной фазы должны быть список и приоритетность функций будущей ИС, предварительные функциональные и информационные модели ИС.

На фазе проектирования часть пользователей принимает участие в техническом проектировании системы под руководством специалистов-разработчиков. CASE-средства используются для быстрого получения работающих прототипов приложений. Пользователи, непосредственно взаимодействуя с ними, уточняют и дополняют требования к системе, которые не были выявлены на предыдущей фазе. Более подробно рассматриваются процессы системы. Анализируется и, при необходимости, корректируется функциональная модель. Каждый процесс рассматривается детально. При необходимости для каждого элементарного процесса создается частичный прототип: экран, диалог, отчет. Определяются требования разграничения доступа к данным. На этой же фазе происходит определение набора необходимой документации.

После детального определения состава процессов оценивается количество функциональных элементов разрабатываемой системы и принимается решение о разделении ИС на подсистемы, поддающиеся реализации одной командой разработчиков за приемлемое для RAD-проектов время - порядка 60 - 90 дней. С использованием CASE-средств проект распределяется между различными командами (делится функциональная модель). Результатом данной фазы должны быть:

1) общая информационная модель системы;

2) функциональные модели системы в целом и подсистем, реализуемых отдельными командами разработчиков;

3) точно определенные с помощью CASE-средства интерфейсы между автономно разрабатываемыми подсистемами;

4) построенные прототипы экранов, отчетов, диалогов.

Все модели и прототипы должны быть получены с применением тех CASE-средств, которые будут использоваться в дальнейшем при построении системы. Данное требование вызвано тем, что в традиционном подходе при пе-

редаче информации о проекте с этапа на этап может произойти фактически неконтролируемое искажение данных. Применение единой среды хранения информации о проекте позволяет избежать этой ошибки.

На фазе реализации выполняется непосредственно разработка приложения, т.е. разработчики производят итеративное построение реальной системы на основе полученных в предыдущей фазе моделей, а также требований нефункционального характера. Программный код частично формируется при помощи автоматических генераторов, получающих информацию непосредственно из репозитория CASE-средств. Конечные пользователи на этой фазе оценивают получаемые результаты и вносят коррективы, если в процессе разработки система перестает удовлетворять определенным ранее требованиям. Тестирование системы осуществляется непосредственно в процессе разработки.

После окончания работ каждой отдельной команды разработчиков производится постепенная интеграция данной части системы с остальными, формируется полный программный код, выполняется тестирование совместной работы данной части приложения с остальными, а затем тестирование системы в целом. В итоге данной фазы:

1) определяется необходимость распределения данных;

2) производится анализ использования данных;

3) производится физическое заполнение базы данных;

4) определяются окончательные требования к аппаратным ресурсам;

5) определяются способы увеличения производительности;

6) завершается разработка документации проекта.

Результатом фазы является готовая система, удовлетворяющая всем согласованным требованиям.

На фазе внедрения производится обучение пользователей, организационные изменения и параллельно с внедрением новой системы осуществляется работа с существующей системой (до полного внедрения новой). Так как фаза реализации достаточно непродолжительна, планирование и подготовка к вне-

дрению должны начинаться заранее, как правило, на этапе проектирования системы.

Следует, однако, отметить, что методология RAD, как и любая другая, не может претендовать на универсальность, она хороша в первую очередь для относительно небольших проектов, разрабатываемых для конкретного заказчика.

Методология RAD неприменима для построения сложных расчетных программ, операционных систем или других систем, требующих написания большого объема (сотни тысяч строк) уникального кода. Не подходят для разработки по методологии RAD приложения, в которых отсутствует ярко выраженная интерфейсная часть, наглядно определяющая логику работы системы (например, приложения реального времени) и приложения, от которых зависит безопасность людей (например, управление атомной электростанцией), так как итеративный подход предполагает, что первые несколько версий наверняка не будут полностью работоспособны, что в данном случае исключается.

В качестве итога перечислим основные принципы методологии RAD:

1) разработка приложений итерациями;

2) необязательность полного завершения работ на каждом из этапов жизненного цикла;

3) обязательное вовлечение пользователей в процесс разработки ИС;

4) необходимое применение CASE-средств, обеспечивающих целостность проекта;

5) применение средств управления конфигурацией, облегчающих внесение изменений в проект и сопровождение готовой системы;

6) необходимое использование генераторов кода;

7) использование прототипов, позволяющих полнее выяснить и удовлетворить потребности конечного пользователя;

8) тестирование и развитие проекта, осуществляемые одновременно с разработкой;

9) ведение разработки немногочисленной хорошо управляемой командой профессионалов;

10) грамотное руководство разработкой системы, четкое планирование и контроль выполнения работ.

5.2 Методология DATARUN

Современные методологии и реализующие их технологии поставляются в электронном виде вместе с CASE-средствами и включают библиотеки процессов, шаблонов, методов, моделей и других компонент, предназначенных для построения ИС того класса систем, на который ориентирована методология. Электронные методологии включают также средства, которые должны обеспечивать их адаптацию для конкретных пользователей и развитие методологии по результатам выполнения конкретных проектов.

Процесс адаптации заключается в удалении ненужных процессов, действий жизненного цикла и других компонентов методологии, в изменении неподходящих или в добавлении собственных процессов и действий, а также методов, моделей, стандартов и руководств. Настройка методологии может осуществляться также по следующим аспектам: этапы и операции ЖЦ, участники проекта, используемые модели ЖЦ, поддерживаемые концепции и др.

Электронные методологии и технологии (и поддерживающие их CASE-средства) составляют ядро комплекса согласованных инструментальных средств среды разработки ИС.

Одной из наиболее распространенных в мире электронных методологий является методология DATARUN. В соответствии с методологией DATARUN ЖЦ ИС разбивается на стадии, которые связываются с результатами выполнения основных процессов, определяемых стандартом ISO 12207. Каждую стадию кроме ее результатов должен завершать план работ на следующую стадию.

Стадия формирования требований и планирования включает в себя действия по определению начальных оценок объема и стоимости проекта (условно назовем эти действия бизнес-планом ИС). Должны быть сформулированы тре-

бования и экономическое обоснование для разработки ИС, функциональная модель и исходная концептуальная модель данных, которые дают основу для оценки технической реализуемости проекта. Основными результатами этой стадии должны быть модели деятельности организации, требования к системе, включая требования по сопряжению с существующими ИС.

Стадия концептуального проектирования начинается с детального анализа первичных данных и уточнения концептуальной модели данных, после чего проектируется архитектура системы. Архитектура включает в себя разделение концептуальной модели на обозримые подмодели. Оценивается возможность использования существующих ИС и выбирается соответствующий метод их преобразования. После построения проекта уточняется исходный бизнес-план. Выходными компонентами этой стадии являются концептуальная модель данных, модель архитектуры системы и уточненный бизнес-план.

На стадии спецификации приложений продолжается процесс создания и детализации проекта. Концептуальная модель данных преобразуется в реляционную модель данных. Определяется структура приложения, необходимые интерфейсы приложения в виде экранов, отчетов и пакетных процессов вместе с логикой их вызова. Модель данных уточняется бизнес-правилами и методами для каждой таблицы. В конце этой стадии принимается окончательное решение о способе реализации приложений. По результатам стадии должен быть построен проект ИС, включающий модели архитектуры ИС, данных, функций, интерфейсов (с внешними системами и с пользователями), требований к разрабатываемым приложениям (модели данных, интерфейсов и функций), требований к доработкам существующих ИС, требований к интеграции приложений, а также сформирован окончательный план создания ИС.

На стадии разработки, интеграции и тестирования должна быть создана тестовая база данных, частные и комплексные тесты. Проводится разработка прототипа и тестирование баз данных и приложений в соответствии с проектом. Отлаживаются интерфейсы с существующими системами. Описывается конфигурация текущей версии ИС. На основе результатов тестирования прово-

дится оптимизация базы данных и приложений. Приложения интегрируются в систему, проводится тестирование приложений в составе системы и испытания системы. Основными результатами стадии являются готовые приложения, проверенные в составе системы на комплексных тестах, текущее описание конфигурации ИС, скорректированная по результатам испытаний версия системы и эксплуатационная документация на систему.

Стадия внедрения включает в себя действия по установке и внедрению баз данных и приложений. Основными результатами стадии должны быть готовая к эксплуатации и перенесенная на программно-аппаратную платформу заказчика версия системы, документация сопровождения и акт приемочных испытаний по результатам опытной эксплуатации.

Стадии сопровождения и развития включают процессы и операции, связанные с регистрацией, диагностикой и локализацией ошибок, внесением изменений и тестированием, проведением доработок, тиражированием и распространением новых версий ПО в места его эксплуатации, переносом приложений на новую платформу и масштабированием системы.

Стадия развития фактически является повторной итерацией стадии разработки.

Методология DATARUN базируется на системном подходе к описанию деятельности организации. Построение моделей начинается с описания процессов, из которых затем извлекаются первичные данные (стабильное подмножество данных, которые организация должна использовать для своей деятельности). Первичные данные описывают продукты или услуги организации, выполняемые операции (транзакции) и потребляемые ресурсы. К первичным относятся данные, которые описывают внешние и внутренние сущности, такие как служащие, клиенты или агентства, а также данные, полученные в результате принятия решений, как например, графики работ, цены на продукты.

Основной принцип DATARUN заключается в том, что первичные данные, если они должным образом организованы в модель данных, становятся основой для проектирования архитектуры ИС. Архитектура ИС будет более стабильной,

если она основана на первичных данных, тесно связанных с основными деловыми операциями, определяющими природу бизнеса, а не на традиционной функциональной модели.

Любая ИС представляет собой набор модулей, исполняемых процессорами и взаимодействующих с базами данных. Базы данных и процессоры могут располагаться централизованно или быть распределенными. События в системе могут инициироваться внешними сущностями (такими как клиенты у банкоматов) или временные события (конец месяца или квартала). Все транзакции осуществляются через объекты или модули интерфейса, которые взаимодействуют с одной или более базами данных.

Подход DATARUN преследует две цели:

1) определить стабильную структуру, на основе которой будет строиться ИС. Такой структурой является модель данных, полученная из первичных данных, представляющих фундаментальные процессы организации;

2) спроектировать ИС на основании модели данных.

Объекты, формируемые на основании модели данных, являются объектами базы данных, обычно размещаемыми на серверах в среде клиент/сервер. Объекты интерфейса, определенные в архитектуре компьютерной системы, обычно размещаются на клиентской части. Модель данных, являющаяся основой для спецификации совместно используемых объектов базы данных и различных объектов интерфейса, обеспечивает сопровождаемость ИС.

6 ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ В ПРОЕКТИРОВАНИИ СЛОЖНЫХ СИСТЕМ

6.1 Общие сведения о CASE-средствах и их классификация

Современные CASE-средства охватывают обширную область поддержки многочисленных технологий проектирования ИС: от простых средств анализа и документирования до полномасштабных средств автоматизации, покрывающих весь жизненный цикл [5].

В настоящее время под термином CASE (Computer Aided Software Engineering) понимают автоматизированный процесс проектирования ИС [4].

Наиболее трудоемкими этапами разработки ИС являются этапы анализа и проектирования, в процессе которых CASE-средства обеспечивают качество принимаемых технических решений и подготовку проектной документации. При этом большую роль играют методы визуального представления информации. Это предполагает построение структурных или иных диаграмм в реальном масштабе времени, использование многообразной цветовой палитры, сквозную проверку синтаксических правил. Графические средства моделирования предметной области позволяют разработчикам в наглядном виде изучать существующую ИС, перестраивать ее в соответствии с поставленными целями и имеющимися ограничениями.

В разряд CASE-средств попадают как относительно дешевые системы для персональных компьютеров с весьма ограниченными возможностями, так и дорогостоящие системы для разных аппаратных платформ и операционных сред. Так, современный рынок программных средств насчитывает около 300 различных CASE-средств, наиболее мощные из которых так или иначе используются практически всеми ведущими западными фирмами.

Выделим преимущества CASE технологии по сравнению с традиционной технологией проектирования:

– улучшение качества разрабатываемой ИС за счет средств автоматического контроля и генерации;

– возможность повторного использования компонентов разработки;

– поддержание адаптивности и сопровождения ИС;

– снижение времени создания системы, что позволяет на ранних стадиях проектирования получить прототип будущей системы и оценить его;

– освобождение разработчиков от рутинной работы по документированию проекта, т.к. при этом используется встроенный документатор;

– возможность коллективной разработки ИС в режиме реального времени.

Любое CASE-средство состоит из репозитория, графических средств моделирования, верификатора диаграмм, документатора проекта, администратора проекта и сервиса.

Ядром системы является репозиторий проекта. Он представляет собой специализированную базу данных, предназначенную для отображения состояния проектируемой ИС в каждый момент времени. Объекты всех диаграмм синхронизированы на основе общей информации репозитория. В репозитории хранятся описания следующих объектов: самих проектировщиков и их прав доступа к различным компонентам системы; организационных структур; диаграмм; компонентов диаграмм; связей между диаграммами; структур данных; программных модулей; процедур и т.д.

Графические средства моделирования предметной области позволяют разработчикам ИС в наглядном виде изучать существующие прототип, перестраивать его в соответствии с поставленными целями и имеющимися ограничениями. Все модификации диаграмм, выполняемых разработчиками в интерактивном режиме, вводятся в репозиторий, контролируются с общесистемной точки зрения и могут использоваться для дальнейшей генерации действующих функциональных приложений.

Верификатор диаграмм служит для контроля правильности построения диаграмм в заданной методологии проектирования ИС.

Документатор проекта позволяет получать информацию о состоянии проекта в виде различных отчетов. Отчеты могут строиться по нескольким признакам, например, по времени, по автору, по элементам диаграмм, по диаграмме или по проекту в целом.

Администратор проекта представляет собой инструменты, необходимые для выполнения следующих административных функций: инициализации проекта, задания начальных параметров проекта, назначения и изменения прав доступа к элементам проекта, мониторинга выполнения проекта.

Сервис представляет собой набор системных утилит по обслуживанию репозитория. Эти утилиты выполняют функции архивации данных, восстановления данных и создания нового репозитория.

Современные CASE-системы классифицируются по следующим признакам [4]:

1) по поддерживаемым методологиям проектирования: функционально-ориентированные, объектно-ориентированные и комплексно-ориентированные;

2) по поддерживаемым графическим нотациям построения диаграмм: с фиксированной нотацией, с отдельными нотациями и наиболее распространенными нотациями;

3) по степени интегрированности: tools – отдельные локальные средства; toolkit – набор неинтегрированных средств, охватывающих большинство этапов разработки ИС; workbench – полностью интегрированные средства, связанные репозиторием;

4) по типу и архитектуре вычислительной техники: ориентированные на локальное рабочее место; ориентированные на локальную вычислительную сеть; ориентированные на глобальную вычислительную сеть;

5) по режиму коллективной разработки проекта: не поддерживающие коллективную разработку; ориентированные на режим реального времени разработки проекта; ориентированные на режим объединения подпроектов;

6) по типу операционных систем и аппаратных платформ: работающие только в одной операционной системе; работающие в нескольких операцион-

ных системах, но на одной аппаратной платформе; работающие на разных аппаратных платформах.

6.2 Характеристики CASE-средств

Silverrun.

CASE-средство Silverrun американской фирмы Computer Systems Advisers, Inc. (CSA) используется для анализа и проектирования ИС бизнес-класса и ориентировано в большей степени на спиральную модель ЖЦ. Оно применимо для поддержки любой методологии, основанной на раздельном построении функциональной и информационной моделей (диаграмм потоков данных и диаграмм "сущность-связь").

Настройка на конкретную методологию обеспечивается выбором требуемой графической нотации моделей и набора правил проверки проектных спецификаций. В системе имеются готовые настройки для наиболее распространенных методологий: DATARUN (основная методология, поддерживаемая Silverrun), Gane/Sarson, Yourdon/DeMarco, Merise, Ward/Mellor, Information Engineering. Для каждого понятия, введенного в проекте имеется возможность добавления собственных описателей. Архитектура Silverrun позволяет наращивать среду разработки по мере необходимости.

Silverrun имеет модульную структуру и состоит из четырех модулей, каждый из которых является самостоятельным продуктом и может приобретаться и использоваться без связи с остальными модулями.

Модуль построения моделей бизнес - процессов в форме диаграмм потоков данных (BPM - Business Process Modeler) позволяет моделировать функционирование обследуемой организации или создаваемой ИС. В модуле BPM обеспечена возможность работы с моделями большой сложности: автоматическая нумерация, работа с деревом процессов (включая визуальное перетаскивание ветвей), отсоединение и присоединение частей модели для коллективной разра-

ботки. Диаграммы могут изображаться в нескольких предопределенных нотациях, включая Yourdon/DeMarco и Gane/Sarson. Имеется также возможность создавать собственные нотации, в том числе добавлять в число изображаемых на схеме дескрипторов определенные пользователем поля.

Модуль концептуального моделирования данных (ERX – Entity Relationship eXpert) обеспечивает построение моделей данных "сущность-связь", не привязанных к конкретной реализации. Этот модуль имеет встроенную экспертную систему, позволяющую создать корректную нормализованную модель данных посредством ответов на содержательные вопросы о взаимосвязи данных. Возможно автоматическое построение модели данных из описаний структур данных. Анализ функциональных зависимостей атрибутов дает возможность проверить соответствие модели требованиям третьей нормальной формы и обеспечить их выполнение. Проверенная модель передается в модуль RDM.

Модуль реляционного моделирования (RDM - Relational Data Modeler) позволяет создавать детализированные модели "сущность-связь", предназначенные для реализации в реляционной базе данных. В этом модуле документируются все конструкции, связанные с построением базы данных: индексы, триггеры, хранимые процедуры и т.д. Гибкая изменяемая нотация и расширяемость репозитория позволяют работать по любой методологии. Возможность создавать подсхемы соответствует подходу ANSI SPARC к представлению схемы базы данных. На языке подсхем моделируются как узлы распределенной обработки, так и пользовательские представления. Этот модуль обеспечивает проектирование и полное документирование реляционных баз данных.

Менеджер репозитория рабочей группы (WRM - Workgroup Repository Manager) применяется как словарь данных для хранения общей для всех моделей информации, а также обеспечивает интеграцию модулей Silverrun в единую среду проектирования.

Платой за высокую гибкость и разнообразие изобразительных средств построения моделей является такой недостаток Silverrun, как отсутствие жесткого

взаимного контроля между компонентами различных моделей (например, возможности автоматического распространения изменений между DFD различных уровней декомпозиции). Следует, однако, отметить, что этот недостаток может иметь существенное значение только в случае использования каскадной модели ЖЦ.

Для автоматической генерации схем баз данных у Silverrun существуют мосты к наиболее распространенным СУБД: Oracle, Informix, DB2, Ingres, Progress, SQL Server, SQLBase, Sybase. Для передачи данных в средства разработки приложений имеются мосты к языкам: JAM, PowerBuilder, SQL Windows, Uniface, NewEra, Delphi. Все мосты позволяют загрузить в Silverrun RDM информацию из каталогов соответствующих СУБД или языков четвертого поколения (4GL). Это позволяет документировать, перепроектировать или переносить на новые платформы уже находящиеся в эксплуатации базы данных и прикладные системы. При использовании моста Silverrun расширяет свой внутренний репозиторий специфичными для целевой системы атрибутами. После определения значений этих атрибутов генератор приложений переносит их во внутренний каталог среды разработки или использует при генерации кода на языке SQL. Таким образом можно полностью определить ядро базы данных с использованием всех возможностей конкретной СУБД: триггеров, хранимых процедур, ограничений ссылочной целостности. При создании приложения на языке 4GL данные, перенесенные из репозитория Silverrun, используются либо для автоматической генерации интерфейсных объектов, либо для быстрого их создания вручную.

Для обмена данными с другими средствами автоматизации проектирования, создания специализированных процедур анализа и проверки проектных спецификаций, составления специализированных отчетов в соответствии с различными стандартами в системе Silverrun имеется три способа выдачи проектной информации во внешние файлы:

1) Система отчетов. Можно, определив содержимое отчета по репозиторию, выдать отчет в текстовый файл. Этот файл можно затем загрузить в текстовый редактор или включить в другой отчет.

2) Система экспорта/импорта. Для более полного контроля над структурой файлов в системе экспорта/импорта имеется возможность определять не только содержимое экспортного файла, но и разделители записей, полей в записях, маркеры начала и конца текстовых полей. Файлы с указанной структурой можно не только формировать, но и загружать в репозиторий. Это дает возможность обмениваться данными с различными системами: другими CASE-средствами, СУБД, текстовыми редакторами и электронными таблицами.

3) Хранение репозитория во внешних файлах через ODBC-драйверы. Для доступа к данным репозитория из наиболее распространенных систем управления базами данных обеспечена возможность хранить всю проектную информацию непосредственно в формате этих СУБД.

Групповая работа поддерживается в системе Silverrun двумя способами:

1) В стандартной однопользовательской версии имеется механизм контролируемого разделения и слияния моделей. Разделив модель на части, можно раздать их нескольким разработчикам. После детальной доработки модели объединяются в единые спецификации.

2) Сетевая версия Silverrun позволяет осуществлять одновременную групповую работу с моделями, хранящимися в сетевом репозитории на базе СУБД Oracle, Sybase или Informix. При этом несколько разработчиков могут работать с одной и той же моделью, так как блокировка объектов происходит на уровне отдельных элементов модели.

Имеются реализации Silverrun для трех операционных систем: MS Windows, MacOS и OS/2.

JAM.

Средство разработки приложений JAM (JYACC's Application Manager) – продукт фирмы JYACC (США). Основной чертой JAM является его соответст-

вие методологии RAD, поскольку он позволяет достаточно быстро реализовать цикл разработки приложения, заключающийся в формировании очередной версии прототипа приложения с учетом требований, выявленных на предыдущем шаге, и предъявить его пользователю.

JAM имеет модульную структуру и состоит из следующих компонент:

– ядро системы;

– JAM/DBi - специализированные модули интерфейса к СУБД (JAM/DBi-Oracle, JAM/DBi-Informix, JAM/DBi-ODBC и т.д.);

– JAM/RW - модуль генератора отчетов;

– JAM/CASEi - специализированные модули интерфейса к CASE средствам (JAM/CASE-TeamWork, JAM/CASE-Innovator и т.д.);

– JAM/TPi - специализированные модули интерфейса к менеджерам транзакций (например, JAM/TPi-Server TUXEDO и т.д.);

– Jterm - специализированный эмулятор X-терминала.

Ядро системы (собственно, сам JAM) является законченным продуктом и может самостоятельно использоваться для разработки приложений. Все остальные модули являются дополнительными и самостоятельно использоваться не могут.

Ядро системы включает в себя следующие основные компоненты:

– Редактор экранов. В состав редактора экранов входят: среда разработки экранов, визуальный репозиторий объектов, собственная СУБД JAM - JDB, менеджер транзакций, отладчик, редактор стилей.

– Редактор меню.

– Набор вспомогательных утилит.

– Средства изготовления промышленной версии приложения.

При использовании JAM разработка внешнего интерфейса приложения представляет собой визуальное проектирование и сводится к созданию экранных форм путем размещения на них интерфейсных конструкций и определению экранных полей ввода/вывода информации.

Проектирование интерфейса в JAM осуществляется с помощью редактора экранов. Приложения, разработанные в JAM, имеют многооконный интерфейс. Разработка отдельного экрана заключается в размещении на нем интерфейсных элементов, возможной (но не обязательной) их группировке и конкретизации различных их свойств, включающих визуальные характеристики (позиция, размер, цвет, шрифт и т.п.), поведенческие характеристики (многообразные фильтры, форматы, защита от ввода и т.п.) и ряд свойств, ориентированных на работу с БД.

Редактор меню позволяет разрабатывать и отлаживать системы меню. Реализована возможность построения пиктографических меню (так называемые toolbar). Назначение каждого конкретного меню тому или иному объекту приложения осуществляется в редакторе экранов.

В ядро JAM встроена однопользовательская реляционная СУБД JDB. Основным назначением JDB является прототипирование приложений в тех случаях, когда работа со штатной СУБД невозможна или нецелесообразна.

В JDB реализован необходимый минимум возможностей реляционных СУБД за исключением индексов, хранимых процедур, триггеров и представлений (view). С помощью JDB можно построить БД, идентичную целевой БД (с точностью до отсутствующих в JDB возможностей) и разработать значительную часть приложения.

Отладчик позволяет проводить комплексную отладку разрабатываемого приложения. Осуществляется трассировка всех событий, возникающих в процессе исполнения приложения.

Утилиты JAM включают три группы:

– Конверторы файлов экранов JAM в текстовые. JAM сохраняет экраны в виде двоичных файлов собственного формата. В ряде случаев (например для изготовления программной документации проекта) необходимо текстовое описание экранов.

– Конфигурирование устройств ввода/вывода. JAM и приложения, построенные с его помощью, не работают непосредственно с устройствами вво-

да/вывода. Вместо этого JAM обращается к логическим устройствам ввода/вывода (клавиатура, терминал, отчет). Отображение логических устройств в физические осуществляется с помощью средств конфигурирования;

– Обслуживание библиотек экранов (традиционные операции с библиотеками). Одним из дополнительных модулей JAM является генератор отчетов. Компоновка отчета осуществляется в редакторе экранов JAM. Описание работы отчета осуществляется с помощью специального языка. Генератор отчетов позволяет определить данные, выводимые в отчет, группировку выводимой информации, форматирование вывода и др.

Приложения, разработанные с использованием JAM, не требуют так называемых исполнительных (run-time) систем и могут быть изготовлены в виде исполняемых модулей. Для этого разработчик должен иметь компилятор С и редактор связей. Для изготовления промышленной версии в состав JAM входит файл сборки (makefile), исходные тексты (на языке С) ряда модулей приложения и необходимые библиотеки.

JAM содержит встроенный язык программирования JPL (JAM Procedural Language), с помощью которого в случае необходимости можно написать модули, реализующие специфические действия. Данный язык является интерпретируемым, что упрощает отладку. Существует возможность обмена информацией между средой визуально построенного приложения и такими модулями. Кроме того, в JAM реализована возможность подключения внешних модулей, написанных на каком-либо языке, совместимым по вызовам функций с языком С.

С точки зрения реализации логики приложения JAM является событийно-ориентированной системой. В JAM определен набор событий, включающий открытие и закрытие окон, нажатие клавиши клавиатуры, срабатывание системного таймера, получение и передача управления каждым элементом экрана. Разработчик реализует логику приложения путем определения обработчика каждого события. Например, обработчик события "нажатие кнопки на экране" (мышью или с помощью клавиатуры) может открыть следующее экранное окно. Обработчиками событий в JAM могут быть как встроенные функции JAM,

так и функции, написанные разработчиком на C или JPL. Набор встроенных функций включает в себя более 200 функций различного назначения. Встроенные функции доступны для вызовов из функций, написанных как на JPL, так и на C.

Промышленная версия приложения, разработанного с помощью JAM, включает в себя следующие компоненты:

– исполняемый модуль интерпретатора приложения. В этот модуль могут быть встроены функции, написанные разработчиками на языках 3-го поколения;

– экраны, составляющие само приложение (могут поставляться в виде отдельных файлов, в составе библиотек экранов или же быть встроены в тело интерпретатора);

– внешние JPL-модули. Могут поставляться в виде текстовых файлов или в прекомпилированном виде, причем прекомпилированные внешние JPL-модули могут быть как в виде отдельных файлов, так и в составе библиотек экранов;

– файлы конфигурации приложения - файлы конфигурации клавиатуры и терминала, файл системных сообщений, файл общей конфигурации.

Непосредственное взаимодействие с СУБД реализуют модули JAM/DBi (Data Base interface). Способы реализации взаимодействия в JAM разделяются на два класса: ручные и автоматические. *При ручном способе* разработчик приложения самостоятельно пишет запросы на SQL, в которых как источниками, так и адресатами приема результатов выполнения запроса могут быть как интерфейсные элементы визуально спроектированного внешнего уровня, так и внутренние, невидимые для конечного пользователя переменные. *Автоматический режим*, реализуемый менеджером транзакций JAM, осуществим для типовых и наиболее распространенных видов операций с БД, так называемых QBE (Query By Example - запросы по образцу), с учетом достаточно сложных взаимосвязей между таблицами БД и автоматическим управлением атрибутами экранных полей ввода/вывода в зависимости от вида транзакции (чтение, запись и т.д.), в которой участвует сгенерированный запрос.

JAM позволяет строить приложения для работы более чем с двадцати СУБД: ORACLE, Informix, Sybase, Ingres, InterBase, NetWare SQL Server, Rdb, DB2, ODBC-совместимые СУБД и др.

Отличительной чертой JAM является высокий уровень переносимости приложений между различными платформами (MS DOS/MS Windows, SunOS, Solaris (i80x86, SPARC), HP-UX, AIX, VMS/Open VMS и др.). Может потребоваться лишь "перерисовать" статические текстовые поля на экранах с русским текстом при переносе между средами DOS-Windows-UNIX. Кроме того, переносимость облегчается тем, что в JAM приложения разрабатываются для виртуальных устройств ввода/вывода, а не для физических. Таким образом при переносе приложения с платформы на платформу, как правило, требуется лишь определить соответствие между физическими устройствами ввода/вывода и их логическими представлениями для приложения.

Использование SQL в качестве средства взаимодействия с СУБД также создает предпосылки для обеспечения переносимости между СУБД. При условии переноса структуры самой БД в ряде случаев приложения могут не требовать никакой модификации, за исключением инициализации сеанса работы. Такая ситуация может сложиться в том случае, если в приложении не использовались специфические для той или иной СУБД расширения SQL.

При росте нагрузки на систему и сложности решаемых задач (распределенность и гетерогенность используемых ресурсов, количество одновременно подключенных пользователей, сложность логики приложения) применяется трехзвенная модель архитектуры "клиент-сервер" с использованием менеджеров транзакций. Компоненты JAM/TPi-Client и JAM/TPi-Server позволяют достаточно просто перейти на трехзвенную модель. При этом ключевую роль играет модуль JAM/TPi-Server, так как основная трудность внедрения трехзвенной модели заключается в реализации логики приложения в сервисах менеджеров транзакций.

Интерфейс JAM/CASE подобен интерфейсу к СУБД и позволяет осуществить обмен информацией между репозиторием объектов JAM и репозиторием

CASE-средства аналогично тому, как структура БД импортируется в репозиторий JAM непосредственно из БД. Отличие заключается в том, что в случае интерфейса к CASE этот обмен является двунаправленным. Кроме модулей JAM/CASEi, существует также модуль JAM/CASEi Developer's Kit. С помощью этого модуля можно самостоятельно разработать интерфейс (т.е. специализированный модуль JAM/CASEi) для конкретного CASE-средства, если готового модуля JAM/CASEi для него не существует.

Мост (интерфейс) Silverrun-RDM <-> JAM реализует взаимодействие между CASE-средством Silverrun и JAM (перенос схемы базы данных и экранных форм приложения между CASE-средством Silverrun-RDM и JAM версии 7.0). Данный программный продукт имеет 2 режима работы:

– Прямой режим (Silverrun-RDM->JAM) предназначен для создания объектов CASE-словаря и элементов репозитория JAM на основе представления схем в Silverrun-RDM. В этом режиме мост позволяет, исходя из представления моделей данных интерфейса в Silverrun-RDM, производить генерацию экранов и элементов репозитория JAM. Мост преобразует таблицы и отношения реляционных схем RDM в последовательность объектов JAM соответствующих типов. Методика построения моделей данных интерфейса в Silverrun-RDM предполагает применение механизма подсхем для прототипирования экранов приложения. По описанию каждой из подсхем RDM мост генерирует экранную форму JAM.

– Обратный режим (JAM->Silverrun-RDM) предназначен для переноса модификаций объектов CASE-словаря в реляционную модель Silverrun-RDM.

Режим реинжиниринга позволяет переносить модификации всех свойств экранов JAM, импортированных ранее из RDM, в схему Silverrun. На этом этапе для контроля целостности базы данных не допускаются изменения схемы в виде добавления или удаления таблиц и полей таблиц.

Ядро JAM имеет встроенный интерфейс к средствам конфигурационного управления (PVCS на платформе Windows и SCCS на платформе UNIX). Под управлением этих систем передаются библиотеки экранов и/или репозитории.

При отсутствии таких систем JAM самостоятельно реализует часть функций поддержки групповой разработки.

Использование PVCS является более предпочтительным по сравнению с SCCS, так как позволяет организовать единый архив модулей проекта для всех платформ. Так как JAM на платформе UNIX не имеет прямого интерфейса к архивам PVCS, то выборка модулей из архива и возврат их в архив производятся с использованием PVCS Version Manager. На платформе MS-Windows JAM имеет встроенный интерфейс к PVCS и действия по выборке/возврату производятся непосредственно из среды JAM.

JAM, как среда разработки, и приложения, построенные с его использованием, не являются ресурсоемкими системами, т.к. требования к аппаратуре определяются самой операционной системой.

CASE-средства AllFusion.

Средства AllFusion компании Computer Associates (США) – это семейство интегрированных решений для разработки, развертывания и управления информационными системами на предприятии. Средства моделирования и инструменты управления изменениями и конфигурациями при разработке ИС позволяют организациям моделировать, разрабатывать и внедрять ИС масштаба предприятия.

Под маркой AllFusion существует три комплекта CASE-средств:

а) AllFusion Modeling Suite – комплект интегрированных средств моделирования, в состав которого входят:

1) AllFusion Process Modeler. Это ведущий инструмент визуального моделирования бизнес-процессов. Дает возможность наглядно представить любую деятельность или структуру в виде модели, что позволяет оптимизировать работу организации, спроектировать организационную структуру, снизить издержки, исключить ненужные операции, повысить гибкость и эффективность. Данный программный продукт поддерживает три нотации моделирования: IDEF0, DFD, IDEF3. Совместное использование этих нотаций позволяет описы-

вать предметную область более комплексно и оптимизировать выполнение любых процедур в компании. Данный программный продукт полностью поддерживает методы расчета себестоимости по объему хозяйственной деятельности (функционально-стоимостной анализ), интегрирован со средством имитационного моделирования Arena, содержит собственный генератор отчетов, позволяет эффективно манипулировать моделями, имеет широкий набор средств документирования моделей, проектов и т.д.

2) AllFusion ERWin Data Modeler. Позволяет проектировать, документировать и сопровождать базы данных, хранилища данных и витрины данных. Поддерживается прямое (создание структуры базы данных на основе модели) и обратное (генерация модели по имеющейся базе данных) проектирование для 20 типов СУБД. Поддерживает методологию SADT и нотации IDEF1X, IE, Dimensional. Позволяет переносить структуру базы данных из одной СУБД в другую, позволяет максимально повысить производительность ИС благодаря поддержке работы с базой данных на физическом уровне, учитывая особенности каждой конкретной СУБД.

3) AllFusion Data Model Validator. Это инструмент для проверки структуры баз данных и создаваемых моделей, позволяющий выделять недочеты и ошибки проектирования. Данный программный продукт дополняет функциональность первых двух программных средств, автоматизируя трудоемкую задачу поиска и исправления ошибок, одновременно повышая квалификацию проектировщиков благодаря встроенной системе обучения.

4) AllFusion Model Manager. Это среда для совместной работы группы проектировщиков в первом и/или втором программном продукте над одним проектом. Она защищает хранимые на собственном сервере модели, позволяя задавать для сотрудников различный уровень доступа к ним и координировать весь ход работы над проектом.

5) AllFusion Component Modeler. Это программный продукт для проектирования, визуализации и поддержки ИС. Благодаря обеспечению расширенной поддержки совместного проектирования и многократного использования ком-

понентов модели, продукт можно использовать как при создании новых приложений, так и при изменении или объединении существующих. Данное средство имеет возможность использования функциональной модели вместе с объектной. Продукт поддерживает около десятка стандартных нотаций (UML, Bootch), интегрируется с технологиями COM/DCOM, CORBAPLUS, VisiBroker и др.

6) AllFusion Model Navigator. Это инструмент для просмотра моделей в режиме «только для чтения». Он позволяет пользоваться информацией, содержащейся в моделях, сотрудникам, не занимающимся напрямую разработкой моделей, но использующих их в своей работе. Это позволяет предотвратить несакционированные изменения моделей, но при этом использовать их для создания презентаций, докладов, отчетов и т.д.

б) Change and Configuration Management представляет собой набор средств для всестороннего управления изменениями и конфигурациями. Эти средства позволяют управлять всеми стадиями жизненного цикла ИС и выполнять интегрированные задачи по настройке межплатформенного программного обеспечения всего предприятия, включая разработку Web-приложений. В данный состав входит три средства:

1) AllFusion Harvest Change Manager. Это средство для управления конфигурациями и различными версиями программного обеспечения при разработке сложных корпоративных систем на основе общего репозитория. Оно помогает синхронизировать деятельность разработчиков на различных платформах, на всем предприятии и в течении всего жизненного цикла.

2) AllFusion Change Manager Enterprise Workbench. Это средство создано для упрощения и автоматизации процессов разработки программного обеспечения на различных платформах.

3) AllFusion Endevor Change Manager. Это средство создано для упрощения и автоматизации процессов разработки программного обеспечения для мейнфреймов и суперкомпьютеров.

в) AllFusion Project & Process Management. Это комплект средств интегрированного управления проектами и процессами. Эти средства управления проектами и процессами позволяют эффективно управлять всеми аспектами разработок на предприятии: готовыми наработками, проектами, ресурсами и передаваемыми файлами. Предлагаемые продукты расширяют возможность традиционных систем «календарного» управления проектами. В данный состав входят:

1) AllFusion Process Managment Suite. Это полный набор инструментальных средств для эффективного управления всеми аспектами организации разработки: проверенными методиками, проектами, ресурсами и конечными результатами.

2) Advisor. Это Web-среда для управления информацией и работами. Данный продукт позволяет руководителю проекта и членам его команды быстро получать отчеты и аналитику с центральной консоли или инструментальной панели для лучшего отслеживания проектов по разработке приложений.

ЛИТЕРАТУРА

1. Система // Большой энциклопедический словарь. – М.:БРЭ. – 2003, с. 1437.

2. Информационно-вычислительные системы в машиностроении (CALS-технологии)/ Ю.М. Соломенцев, В.Г. Митрофанов, В.В. Павлов, А.В. Рыбаков – М.: Наука, 2003. - 292 с.

3. Проектирование экономических информационных систем: Учебник/ Галина Николаевна Смирнова, Алексей Алексеевич Сорокин, Юрий Филиппович Тельнов. – М.: Финансы и статистика, 2001. – 512 с.

5. Туголуков Е.Н., Ткачев А.Г., Рухов А.В. и др. Проектирование сложных систем. – Тамбов: Изд-во Тамб. гос. тех. ун- та, 2008. – 32 с.

6. Джо Хабрейкен, Мэтт Хайден. Сетевые технологии за 24 часа. 3-е издание. С.-Петербург: Издательский дом Вильямс, 2007. – 240 с.

7. Машков С.В. Программа Autodesk AutoCAD. Учебное пособие по автоматизированному проектированию. М., Альянс-пресс, 2007 – 448 с.

8. Основы проектирования электронных средств: Учебное пособие/ А.В.Зеленский, В.А.Зеленский, Г.Ф.Краснощекова, А.А. Нюхалов – Самара: «Изд-во Самарского научного центра РАН», 2007.-243 с.