ВЫСШЕЕ ОБРАЗОВАНИЕ – БАКАЛАВРИАТ
серия основана в 1996 г.

В.В. Коваленко

ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННЫХ СИСТЕМ

УЧЕБНОЕ ПОСОБИЕ

Рекомендовано Учебно-методическим объединением учебных заведений Российской Федерации по образованию в области прикладной информатики в качестве учебного пособия для студентов (бакалавров и специалистов) высших учебных заведений, обучающихся по направлению 09.03.03 «Прикладная информатика»

Москва ФОРУМ 2018
ИНФРА-М

УДК 004(075.8)
ББК 32.973-02я73
К56

Рецензенты:
Пылькин А.Н. — доктор технических наук, профессор, заведующий кафедрой вычислительной и прикладной математики Рязанского государственного радиотехнического университета;
Нечаев Г.И. -- доктор технических наук, профессор, заведующий кафедрой автоматизированных систем управления Рязанского государственного радиотехнического университета

Коваленко В.В.

К56 Проектирование информационных систем : учеб. пособие / В.В. Коваленко. — М. : ФОРУМ : ИНФРА-М, 2018. — 320 с. — (Высшее образование: Бакалавриат).

ISBN 978-5-00091-628-5 (ФОРУМ)
ISBN 978-5-16-014434-4 (ИНФРА-М, print)
ISBN 978-5-16-101923-8 (ИНФРА-М, online)

В пособии рассмотрены особенности проектирования ИС, участвующих в реализации CALS-технологий: MRP/MRPII/ERP-систем, систем электронной коммерции (B2B), управления цепочками поставок (SCM), управления взаимоотношениями с клиентами (CRM), а также систем поддержки принятия решений (OLAP). Обсуждены вопросы выбора технологии проектирования, программного инструментария для разработки проекта, построения функциональных и информационных моделей в среде пакетов All Fusion Modeling Suite и Oracle Designer 10g, а также разработки технической и эксплуатационной документации. Рассмотрены характеристики CASE-технологий и их реализация в среде Oracle Designer 10. Выполнен сравнительный анализ стандартов на организацию жизненного цикла создания и использования ИС, даны практические рекомендации по разработке профилей стандартов, приведены примеры и разработки проекта ИС по каскадной модели жизненного цикла. Подробно обсуждены современные подходы к выбору готовых ИС и их внедрению на автоматизируемых предприятиях.

Учебное пособие предназначено для студентов (бакалавров и специалистов) и магистров высших учебных заведений, обучающихся по направлению «Прикладная информатика». Рекомендуется преподавателям и специалистам, работающим в области информационных технологий.

УДК 004(075.8)
ББК 32.973-02я73

ISBN 978-5-00091-628-5 (ФОРУМ)
ISBN 978-5-16-014434-4 (ИНФРА-М, print)
ISBN 978-5-16-101923-8 (ИНФРА-М, online)

Введение

У вас в руках книга, в которой автор шаг за шагом вводит читателя в увлекательный мир проектирования современных информационных систем по технологиям известных российских и зарубежных фирм. Обычно в учебниках по проектированию приводятся всевозможные классификации информационных систем, иногда почти вся книга состоит из перечисления классификаций для всевозможных критериев. С точки зрения автора, это только затрудняет понимание читателями содержания учебного материала. По этой причине вместо классификации класс изучаемых информационных систем определен, исходя из тех систем, которые обеспечивают управление ресурсами предприятия по изготовлению продукции или оказанию услуги на определенных этапах жизненного цикла изделия в структуре CALS-технологий.

Наиболее подробно рассмотрены вопросы проектирования и внедрения информационных экономических систем классов MRP/MRPII/ERP/CSRP. Для систем, которые обеспечивают взаимоотношения с клиентами и поставщиками — системы электронной коммерции (B2B), управления цепочками поставок (SCM), управления взаимоотношениями с клиентами (CRM), а также систем поддержки принятия решений (OLAP) — изучается их назначение, функциональность, особенности проектирования и внедрения.

В первой главе описывается стратегия CALS, структура CALS-технологий и информационные системы, которые эти технологии реализуют на различных этапах жизненного цикла изделий. Из всего многообразия систем для более подробного изучения выбраны те, которые связаны с управлением производством, снабжением ресурсами и сбытом продукции (MRP/MRPII/ERP-системы, электронной коммерции типа B2B, управления взаимоотношениями с клиентами CRM и другие).

Эти системы имеют одинаковые технологии проектирования, разработки и сопровождения, а также архитектуру построения — «клиент—сервер». Для этих видов систем рассмотрены в укрупненном плане этапы проектирования, разработки и внедрения разработанной или адаптации покупной системы.

Во второй главе осуществляется знакомство с наиболее популярными моделями жизненного цикла ИС, описываются технологии проектирования: Oracle CDM, MSF фирмы Microsoft, экстремальное программирование и другие.

Наиболее подробно, до уровня выполняемых работ и задач, рассматриваются технология на базе российских стандартов ГОСТ 34 и технология Oracle CDM. Читатель получает возможность увидеть весь набор задач, последовательность их выполнения и принципы управления проектом.

В третьей главе дается характеристика CASE-технологии и CASE-средств, производится их классификация. Информационные системы классифицируются по определенному набору характеристик на малые, средние и крупные. С учетом этой классификации выдаются рекомендации по выбору программных инструментариев для проектирования ИС, которые, в свою очередь, классифицируются на три группы: малые, средние и крупные интегрированные средства моделирования.

Описывается реализация CASE-технологии на базе пакета Oracle Designer/2000. Приводятся типичные признаки CASE-технологии, объясняется схема применения утилит этого пакета при автоматизированном проектировании информационных систем.

Здесь же происходит знакомство с уровнями технологической зрелости предприятия-разработчика, которые позволяют оценить способность разработчика гарантировать успех проекта.

В четвертой главе рассмотрены наиболее популярные методологии описания функциональных и информационных моделей: Swim Lane diagram, IDEF0, IDEF3, DFD, IDEF1X, Oracle Process Modeler. Описаны их возможности и особенности применения. Знакомство с этими методологиями осуществляется на примере пакетов All Fusion Process Modeler (BPwin), All Fusion ERwin Data Modeler (ERwin) и Oracle Designer 10g.

Исследованы вопросы интеграции IDEF0- и IDEF1X-моделей, связывания объектов модели данных с функциональной моделью, а также генерации базы данных физического уровня в среде СУБД Access.

В пятой главе проводится сравнительный анализ международного стандарта ISO/IEC 12207:1995-08-01, ведомственного стандарта Oracle CDM и российских стандартов ГОСТ 34 с точки зрения их адаптивности к проектируемой ИС, их общей структуры, степени ориентации на заказчика и разработчика.

В связи с отсутствием универсального, всеобъемлющего стандарта большое внимание в главе уделено описанию профилей стандартов и примера создания профиля стандартов для конкретного проекта с подробным описанием содержания технологической и эксплуатационной документации на всех этапах жизненного цикла создаваемой системы.

Шестая глава полностью посвящена рассмотрению реальных действий системных аналитиков и проектировщиков при разработке проекта на основе программных инструментариев All Fusion Modeling Suite и Oracle Designer 10g. Рассматриваются вопросы построения моделей «AS IS» и «TO BE» проектируемой системы.

Седьмая глава знакомит читателя с функциональным наполнением систем класса ERP, систем электронной коммерции типа B2B, управления цепочками поставок SCM, управления взаимоотношениями с клиентами CRM, а также особенностями их проектирования и внедрения.

В восьмой главе изучаются принципы работы систем поддержки принятия решений на примере OLAP-систем, определяется их функциональность и особенности построения многомерной базы данных. Подробно рассматриваются шаги проектирования многомерной базы данных с использованием программного инструментария Oracle Express. Приводится классификация OLAP-систем.

Девятая глава посвящена вопросам выбора и внедрения информационных систем. Типовое внедрение ИС рассмотрено на базе пакетированных решений и модельно-ориентированного проектирования. Также рассмотрено типовое внедрение готовых программных продуктов на основе настройки их опций под автоматизируемую предметную область.

В приложении А детально описаны все работы, выполняемые при проектировании ИС по каскадной модели ЖЦ.

Данное учебное пособие позволит сформировать у студентов знания о системном подходе, технологиях и этапах проектирования ИС, проектной документации, автоматизированном проектировании информационных систем, практические навыки проектирования.

Оно может быть также полезно аспирантам и специалистам, работающим в сфере проектирования или внедрения ИС.

Для понимания содержания учебного пособия предполагается наличие у читателя некоторых навыков работы с пакетом All Fusion Modeling Suite.

Автор приносит благодарность начальнику отдела конструкторского бюро Рябцову Юрию Васильевичу за ценные замечания по вопросам стандартизации и особую признательность инженеру Кузиной Елене Николаевне за помощь при подготовке рукописи.

Глава 1
СТРАТЕГИЯ CALS И КОМПЬЮТЕРНЫЕ СИСТЕМЫ ДЛЯ ЕЕ РЕАЛИЗАЦИИ

1.1. Стратегия CALS как средство повышения конкурентоспособности предприятий

Несмотря на стремительные изменения в окружающем мире, динамику развития бизнеса, основной задачей любого предприятия остается **повышение конкурентоспособности** своих изделий и услуг. Условно конкурентоспособность можно представить в виде простой дроби, в числителе которой находится степень удовлетворения потребностей заказчика изделия, а в знаменателе — издержки предприятия при удовлетворении потребностей заказчика. Таким образом, повышать конкурентоспособность изделия можно путем увеличения числителя и/или уменьшения знаменателя этой дроби.

Под повышением степени удовлетворения потребностей заказчика подразумевается не только создание изделия с требуемыми характеристиками, но и соответствующими потребностями по поставке, эксплуатации, обслуживанию, ремонту и модернизации изделия.

Основными путями **снижения издержек производства** являются сокращение времени выхода изделия на рынок (сокращение временных издержек) и сокращение затрат на создание и эксплуатацию изделия (сокращение материальных издержек).

Главным способом повышения конкурентоспособности изделия или услуги является повышение эффективности процессов его жизненного цикла (ЖЦ). Согласно международным стандартам качества продукции серии ISO 9000 ЖЦ изделия представляет собой совокупность процессов, выполняемых от момента выявления потребностей общества в определенном продукте до момента удовлетворения этих потребностей и утилизации продукта. Можно выделить одиннадцать этапов ЖЦ продукта:

- маркетинг и изучение рынка;
- проектирование и разработка продукта;

- планирование и разработка процессов (технологий производства, эксплуатации и т. п.);
- закупки;
- производство или предоставление услуг;
- упаковка и хранение;
- реализация;
- установка и ввод в эксплуатацию;
- техническая помощь и обслуживание;
- послепродажная деятельность или эксплуатация;
- утилизация и переработка в конце полезного срока службы.

В первую очередь эффективность процессов ЖЦ изделия зависит от эффективности управления ресурсами, используемыми в течение ЖЦ: материальными, финансовыми, кадровыми и информационными.

Существует множество всевозможных методик повышения эффективности управления тем или иным видом ресурсов. Наиболее известными из них являются управление потребностью в материалах (MRP-Material Requirements Planning), управление производственными ресурсами (MPR II — Manufacturing Resource Planning), управление ресурсами предприятия (ERP — Enterprise Resource Planning).

Основными выгодами, полученными от применения CALS (Continuous Acquisition and Life Cycle Support), являются не только сокращение временных и материальных издержек, но и повышение качества изделия. Пути достижения этих выгод заложены в **стратегии CALS**: создание единого информационного пространства (ЕИП) для всех участников ЖЦ изделия, включая и потребителя изделия (рис. 1.1) [19].

Стратегия CALS предусматривает: однократный ввод данных, их хранение в стандартных форматах, стандартизацию интерфейсов и электронный обмен информацией между всеми участниками ЖЦ изделия.

Создание ЕИП позволяет преодолеть информационный хаос и коммуникационные барьеры между участниками ЖЦ изделия. Это приводит к повышению эффективности процессов ЖЦ и улучшению взаимодействия между его участниками. Результатом такого повышения является снижение временных и материальных издержек в течение ЖЦ изделия и повышение степени удовлетворения потребностей заказчика, а это, в свою очередь, неизбежно принесет повышение конкурентоспособности изделия.

ЕИП предполагает отказ от прямого взаимодействия и передачи данных между участниками ЖЦ. Все коммуникации между ними

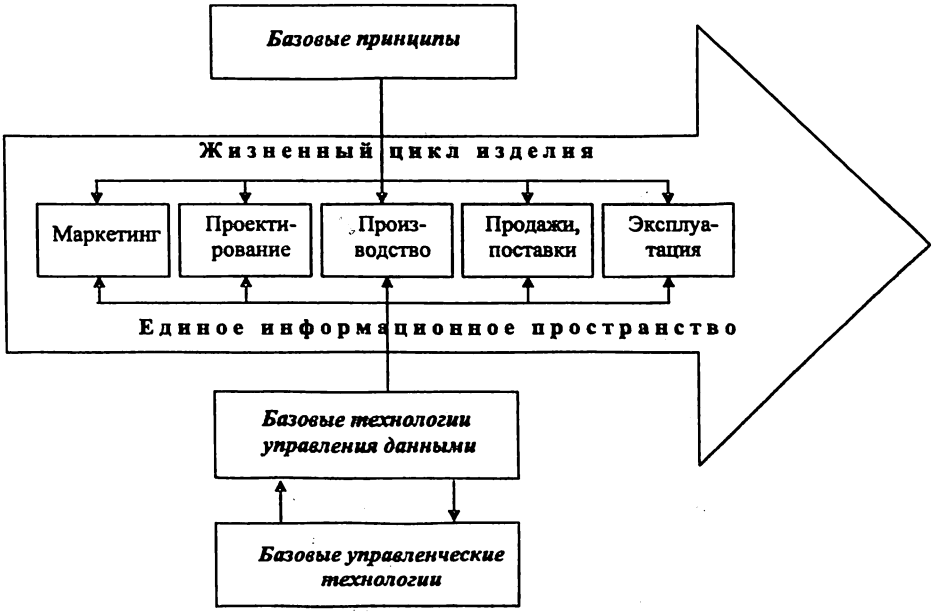


Рис. 1.1. Концептуальная модель CALS-технологий

должны осуществляться через ЕИП, основными свойствами которого являются следующие:

- информация представлена в электронном виде, преимущества которого перед бумажным способом представления информации очевидны: большая эффективность создания, хранения, изменения и доступа к данным;
- ЕИП охватывает всю информацию об изделии, созданную любым участником ЖЦ на любом этапе ЖЦ;
- ЕИП выступает единственным источником данных для любого участника ЖЦ, предоставляя (в соответствии с правами доступа) нужную информацию в нужное время в нужном виде (рис. 1.2).

Использование ЕИП дает предприятию следующие преимущества:

- данные, созданные на начальных этапах ЖЦ изделия, не теряются и могут быть использованы на последующих этапах;
- изменения данных видны всем и сразу: если один из участников ЖЦ изменяет информацию о своей части изделия, то эти изменения становятся доступными для других участников немедленно, что исключает ситуации, при которых возможна работа над устаревшей информацией;

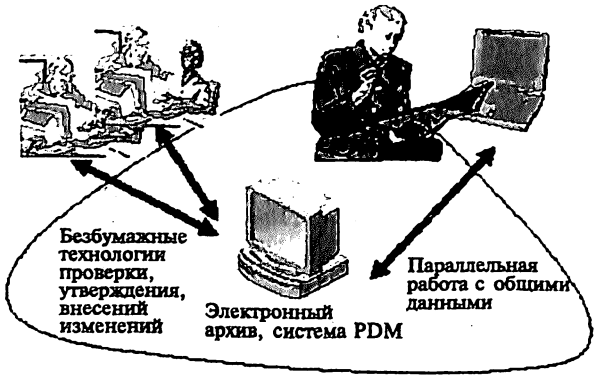


Рис. 1.2. Схема потоков данных в ЕИП

• повышение скорости поиска и доступа к данным — электронное представление данных позволяет значительно сократить время, затрачиваемое на поиск необходимой информации и доступ к ней, по сравнению с бумажным документооборотом.

При этом для создания ЕИП используются существующие на предприятиях программно-аппаратные средства, т. е. предприятиям не нужно отказываться от уже используемых прикладных систем и терять, таким образом, сделанные в них инвестиции. Вопрос стоит только об адаптации этих систем к работе в рамках ЕИП.

1.2. CALS-технологии

Стратегии CALS реализуются с помощью набора CALS-технологий, которые можно представить в виде трех групп:

• технологии реинжиниринга бизнес-процессов;

• технологии представления данных об изделии в электронном виде;

• технологии интеграции данных об изделии в рамках ЕИП.

Преобразование ЖЦ изделия в высокоавтоматизированный процесс должно начинаться с перестройки составляющих ЖЦ бизнес-процессов. Для решения этой задачи необходимо использовать **технологии реинжиниринга бизнес-процессов**, представляющие собой набор методов реструктуризации функционирования предприятия с целью повышения его эффективности. В использовании реинжиниринга бизнес-процессов в CALS-технологиях выделяют три основных аспекта.

Во-первых, бизнес-процессы предприятия могут быть неоптимальными с точки зрения наличия среди них дублирующих процессов или, наоборот, отсутствия формализованных процедур выполнения той или иной операции.

Во-вторых, для реализации стратегии CALS, предполагающей создание ЕИП и представление всех данных в электронном виде, потребуется изменение способа функционирования предприятия. Это связано с тем, что бизнес-процессы, оптимальные для создания и использования бумажной информации, не подходят для создания и использования электронных данных.

В-третьих, преодоление коммуникационных барьеров между участниками ЖЦ изделия предполагает повышение эффективности их взаимодействия между собой, причем такое взаимодействие также является бизнес-процессом.

Технологии представления данных являются набором методов для представления в электронном виде данных об изделии. Эти технологии предназначены для автоматизации отдельных процессов ЖЦ изделия, что является одним из пунктов стратегии CALS по созданию ЕИП. Технологии представления данных включают также технологии перевода данных с бумажных носителей в электронный вид. Рассматриваемая группа CALS-технологий состоит из более или менее известных методов, которые можно расположить по трем группам (в соответствии с укрупненными этапами ЖЦ) (рис. 1.3):

- **проектирование изделия** — технологии автоматизации проектирования изделия, в частности компьютерные системы автоматизированного проектирования (CAD — Computer Aided Design), системы автоматизированной подготовки производства (CAM — Computer Aided Manufacturing), системы инженерных расчетов (CAE — Computer Aided Engineering) и т. п.;
- **производство изделия** — технологии автоматизации производственных процессов. Сюда относятся следующие типы информацион-

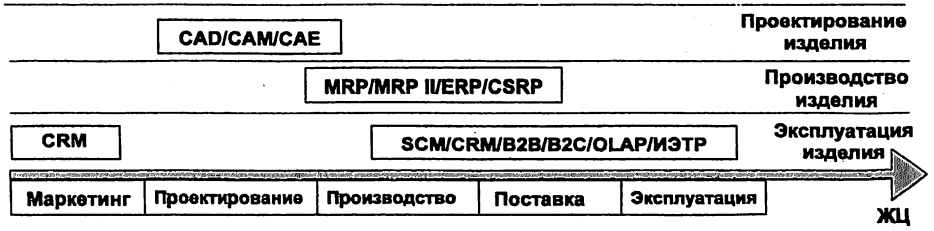


Рис. 1.3. Распределение информационных систем по этапам ЖЦ изделия

ных систем: MRP (Material Requirements Planning — планирование потребностей в материалах), MRP II (Manufacturing Resource Planning — планирование ресурсов производства), ERP (Enterprise Resource Planning — планирование ресурсов предприятия), CSRP (Customer Synchronized Resource Planning — планирование ресурсов, синхронизированное с заказчиком);

• **поставка и эксплуатация изделия** — технологии автоматизации процессов поставки и эксплуатации изделия, которые реализуются в виде следующих систем:

— ***системы логистической поддержки изделия*** — для автоматизации обслуживания и ремонта изделия на этапе эксплуатации, заказа комплектующих к изделию, поставки изделий и комплектующих, в частности, SCM-системы (Supply Chain Management — управление цепочкой поставок);

— ***системы электронной коммерции*** — используются как отдельные блоки ERP-систем, а также в виде самостоятельных систем, предназначенных для проведения коммерческих операций в электронном виде. Среди этих бурно развивающихся в настоящее время систем можно выделить системы типа B2B (business-to-business: взаимодействие предприятий между собой) и системы типа B2C (business-to-customer: взаимодействие поставщика и покупателя, в том числе Internet-магазины);

— ***интерактивные электронные технические руководства (ИЭТР)*** — автоматизированные системы, предоставляющие пользователю эксплуатационную информацию по конкретному изделию, а также возможности по диагностике изделия, поиску и устранению неисправностей, обучению, взаимодействию с поставщиком и т. п.

Технологии интеграции данных представляют собой набор методов для интеграции автоматизированных процессов ЖЦ и относящихся к ним данных, представленных в электронном виде, в рамках ЕИП. Эти технологии реализуют заключительный этап стратегии CALS по созданию ЕИП для всех участников ЖЦ изделия. Технологии интеграции данных реализуются с помощью класса автоматизированных систем, называемых системами управления данными об изделии (Product Data Management — PDM).

С одной стороны, такие системы выступают в качестве ***хранилища всех данных об изделии*** и взаимодействуют с прикладными программами, создающими или использующими данные об изделии. Данные,

созданные любой прикладной программой, передаются на хранение в PDM-систему и становятся доступными любому участнику ЖЦ изделия, имеющему соответствующие права доступа. Если необходимо изменить или обработать данные об изделии, прикладная система запрашивает и получает эти данные из PDM-системы.

С другой стороны, PDM-системы должны решать задачу ***повышения эффективности работы*** отдельного пользователя. В этом случае они должны выступать в качестве рабочей среды пользователя, предоставляя ему нужные данные в нужное время и в нужной форме.

В результате технологии интеграции данных обеспечивают создание ЕИП, которое содержит хранилище данных, реализованное на основе PDM-системы, и прикладные пакеты.

В целом **CALS-технологии** позволяют создавать виртуальные предприятия, взаимодействующие между собой в едином информационном пространстве и обменивающиеся стандартизованной информацией. Стратегия CALS в современной обстановке рассматривается как стратегия выживания в конкурентной среде за счет расширения сферы деятельности предприятия, повышения эффективности бизнес-процессов, повышения привлекательности и конкурентоспособности изделий, обеспечения заданного качества продукции в интегрированной системе поддержки жизненного цикла путем электронного документирования выполняемых процессов.

1.3. Компьютерные системы для реализации CALS-технологий

Для реализации CALS-технологий на предприятиях широко используются средства автоматизации проектирования (**CAD** — Computer-aided design), которые вначале представляли собой простую электронную чертежную доску, а затем по мере развития технологии значительно усовершенствовались. В связи с этим конструкторы получили в свое распоряжение инструментарий для двумерного, а потом трехмерного параметрического моделирования. Они представляют необходимые конструкторские данные об изделии: сведения о составе изделия, о геометрических моделях изделия, его компонентах и технических характеристиках, об их отношениях в структуре изделия, о результатах расчетов и моделирования, о допусках на изготовление и т. д.

Современные системы автоматизации инженерных расчетов (**CAE** — Computer-aided engineering) являются набором разнообразных программных продуктов, позволяющих при помощи расчетных методов (метод конечных элементов, метод конечных разностей, метод конечных объемов) оценить, как поведет себя компьютерная модель изделия в реальных условиях эксплуатации. CAE-системы помогают убедиться в работоспособности изделия, без привлечения больших затрат времени и средств, обычно интегрируются совместно с CAD-системами.

CAM-системы (Computer-aided manufacturing) обеспечивают технологическую подготовку производства изделий, ориентированную на использование ЭВМ. Под термином понимаются как сам процесс компьютеризированной подготовки производства, так и программно-вычислительные комплексы, используемые инженерами-технологами.

Они выдают в электронном виде сведения о способах изготовления и контроля изделия в процессе производства. Кроме этого, формируются описания маршрутных и операционных технологий, нормы времени и расхода материалов, управляющие программы для станков с ЧПУ, а также данные для проектирования приспособлений и операционного режущего и мерительного инструмента и т. д.

Начиная с 90-х годов активно начинают внедряться системы управления ресурсами предприятия (**MRP/MRPII/ERP/CSRP**), которые обеспечивают поставку производственных и эксплуатационных данных об изделии в электронном виде. Это сведения об изделии и его компонентах, содержащие информацию о производственном цикле и степени его соответствия: техническим требованиям, техническим условиям, требованиям стандартов, а также сведения, необходимые для организации обслуживания и ремонта изделия.

Данные системы существенно повысили эффективность работы, значительно ускорив проектирование и изготовление продукции, но при этом на предприятии образовались изолированные участки автоматизации, слабо связанные друг с другом. Каждая из этих систем содержит огромные объемы данных, но из-за отсутствия интеграции их сложно использовать даже внутри одной организации, не говоря уже о том, чтобы поделиться ими с поставщиками и заказчиками.

Поэтому потребовалось решение, которое бы объединило отдельные участки автоматизации в едином информационном пространстве и реализовало сквозной конструкторский, технологический, производственный и коммерческий цикл, от подготовки проекта до утилизации в том виде, как показано на рис. 1.4 [13].

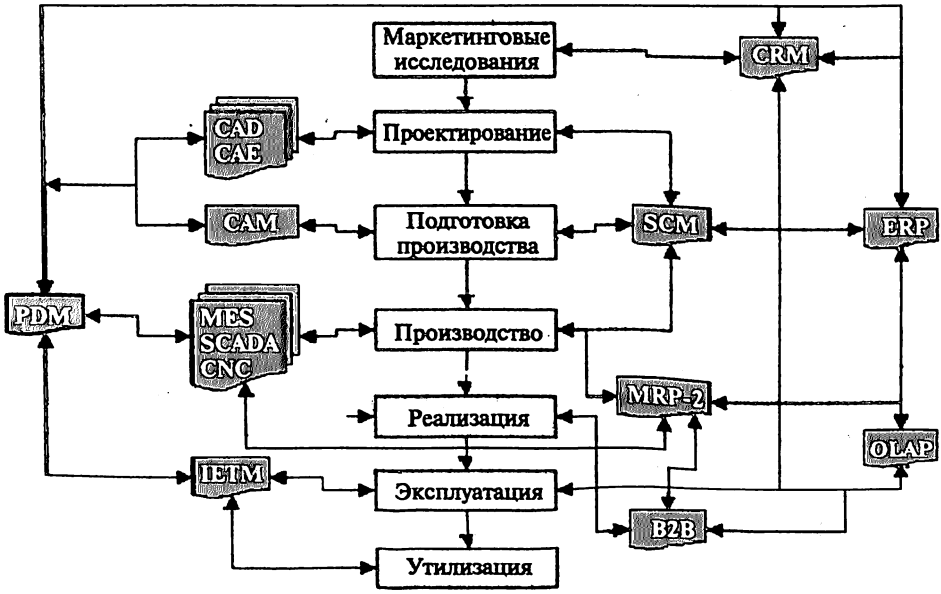


Рис. 1.4. Этапы жизненного цикла изделия и используемые информационные системы

Это объединение было реализовано с помощью **методов CALS**, которые вначале расшифровывались как **Computer-Aided Logistics Support** (компьютерное обеспечение материально-технического обеспечения), а затем по мере расширения функциональности — **Continuous And Life Cycle Support** (непрерывное развитие и поддержка жизненного цикла).

Остановимся подробнее на тех информационных системах, которые обеспечивают управление производством и представляют в целом функции АСУП (они располагаются на рис. 1.4 справа от этапов модели ЖЦ изделия).

На этапе **маркетинговых исследований** находит применение система **CRM** (Customer Relationship Management — управление взаимоотношениями с клиентами) для анализа состояния рынка, прогноза спроса на планируемые изделия и развития их технических характеристик. На вход системы CRM поступают данные, связанные с клиентами компании, а на выходе появляется информация, влияющая на поведение компании в целом или на поведение ее отдельных подразделений, вплоть до конкретного сотрудника компании. В целом CRM-система представляет собой базу данных с информацией о клиентах и набор приложений, которые позволяют собирать информа-

цию о клиенте, обрабатывать ее и принимать управленческие решения на базе этой информации.

На **большинстве этапов жизненного цикла** требуются услуги системы управления цепочками поставок — **Supply Chain Management (SCM)**, которая обеспечивает планирование, выполнение, контроль материальных потоков и информационное сопровождение всех участников на протяжении всего жизненного цикла продукта — от заказа на разработку до послепродажного сервиса и утилизации.

Цепочка поставок начинается с приобретения сырья у поставщиков и заканчивается продажей готовых товаров и услуг клиенту и представляет собой множество звеньев, связанных между собой информационными, денежными и товарными потоками. В цепочке поставок от поставщиков к клиенту движутся товары и услуги, а в обратном направлении — информация о потребности клиентов (рис. 1.5).

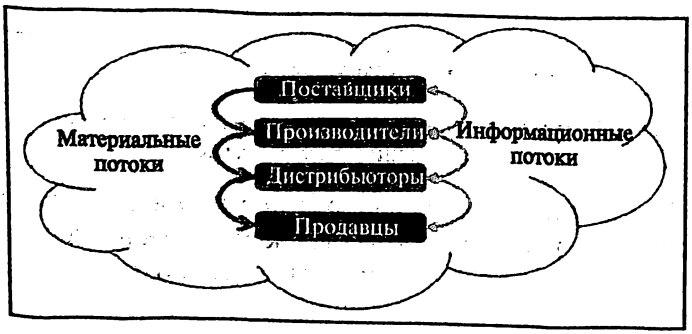


Рис. 1.5. Единое информационное поле пользователей SCM-системы

SCM-система создает единое информационное пространство для всех компаний, участвующих в производстве продукта, его транспортировке, продаже и послепродажном обслуживании.

На этапе производства используется информационная система одного из классов: **MRP, MRP II, ERP или CSRP** [28].

Исходным стандартом планирования производства, разработанным еще в 70-х годах, был **MRP (Material Requirements Planning)**, который включал только планирование материалов для производства (рис. 1.6). Главной сущностью, которой оперирует MRP-система, является объект материального учета: сырье и материалы, сборочные единицы, полуфабрикаты, т. е. практически все необходимое для сборки конечного продукта. Она постоянно отслеживает состояние каждого материала: наличие на складе, его цена, данные о его поставщиках, информация о регулярности поставок материалов.

Рис. 1.6. Взаимосвязь стандартов и их информационных платформ

Таким образом, MRP-система, получая на входе данные о наличии материалов на складе, их свойствах и зная, что именно требуется для производства конечного продукта, позволяет оптимально с точки зрения сроков закупки и производства спланировать процесс производства.

Результатом работы системы является **глобальный план заказов поставщикам на производство**, в котором должно быть расписано, что, когда и у кого необходимо заказать. Но MRP-система обладает серьезным недостатком: необходимые материалы и комплектующие планируются без учета необходимых для превращения их в готовую продукцию ресурсов: производственных мощностей, людских и финансовых ресурсов, складских площадей и т. д.

В результате возникли системы класса **MRP II (Manufacturing Resource Planning)**, которые обеспечивали планирование всех вышеназванных ресурсов предприятия, т. е. обеспечивали управление финансо-

во-хозяйственной деятельностью и материально-техническим снабжением предприятия. Причем аббревиатура MRP II расшифровывается иначе, чем MRP, а «двоечку» добавили потому, что аббревиатуры случайным образом совпали.

Пользователи систем MRPII получают: оперативную информацию о предприятии вплоть до подробностей, которые касаются отдельных заказов; оптимизацию всего цикла работы с материальными ресурсами (уменьшение загруженности складских помещений, автоматизацию заказов необходимых материалов и т. п.); оптимизацию работы по контролю платежей, отгрузки готовой продукции и так далее. Кроме того, руководители предприятия получают уникальный шанс увидеть все производственные и финансовые потоки подконтрольной им структуры. Все это приводит к значительному сокращению затрат.

В начале 1990-х гг. была предложена концепция **ERP-системы (Enterprise Resource Planning)** планирования ресурсов в масштабе предприятия, которая позволила объединить все ресурсы предприятия. Исторически технология MRP была нацелена на решение производственных задач, однако в настоящее время системы класса MRP и ERP широко применяются не только в производстве, но и для управления проектной деятельностью (конструкторские бюро), коммерцией, социальными задачами, эксплуатацией сложной техники (авиакомпании) и т. п.

В настоящее время MRP II и ERP практически полностью вытеснили термин АСУП и используются специалистами для обозначения класса интегрированных ИС, предназначенных для управления производственно-хозяйственной деятельностью предприятия.

Типовой набор задач, решаемых системами класса MRP II, приведен в работе [37]: управление финансовыми ресурсами, управление персоналом, ведение портфеля заказов, расчет потребностей в материалах, оперативно-производственное планирование, управление закупками и продажами, оперативное управление производством, расчет себестоимости продукции и затрат, управление сервисным обслуживанием и некоторые другие.

Для современного предприятия развертывание ERP-системы оказывается недостаточным для целей конкурентоспособности. Поэтому в состав ERP-систем в виде блоков или автономно вводятся:

1) системы управления взаимоотношениями с клиентами **CRM** (Customer Relationship Management);

2) системы управления цепочками поставок **SCM** (Supply Chain Management);

3) системы поддержки принятия решений **OLAP** (On-Line Analytical Processing);

4) системы электронной коммерции **B2B** и **B2C**;

5) приложения для гарантийного и сервисного обслуживания после продажи **ИЭТР** (интерактивное электронное техническое руководство).

Используя их в едином комплексе, получают систему нового типа — **CSRP (Customer Synchronized Resource Planning)**, в которой реализована новая парадигма планирования — планирование ресурсов, **синхронизированное с получателем (заказчиком)**. Поэтому широко используется тезис, что стандарт CSRP «вышел за ворота» отдельного предприятия.

На рис. 1.6 представлены взаимосвязи стандартов и информационных платформ, на которые они опираются [28]. При этом каждый последующий в цепочке эволюции стандарт полностью поглощает платформу предыдущего и требует дополнительного информационного слоя, разрастаясь от конструкторских спецификаций до всего информационного пространства предприятия. Поэтому, например, системы класса MRP и MRP II можно рассматривать как отдельные модули ERP-системы.

В свою очередь, одним из ключевых преимуществ ERP-системы является комплектность решения — автоматизированы и тесно взаимодействуют основные процессы организации: маркетинг и сбыт, планирование и управление производством, планирование материальных запасов, планирование закупок, управление финансами, учет и т. д.

Но ERP-системы сфокусированы исключительно на внутренних процессах: оптимизируют прием заказов, планирование производства, закупку, производство, доставку и управление, т. е. все внутренние операции. CSRP-системы, используя интегрированную функциональность ERP-систем, расширяют планирование от производства к покупателю. Системы подобного класса уже появляются в России. Так, корпорация «Парус» предлагает следующую схему решения для машиностроительного предприятия (рис. 1.7), которая включает в себя пять контуров. В соответствии с этой структурой основной бизнес-процесс предприятия выглядит следующим образом.

Службы маркетинга ищут потребителей, заключают с ними договоры, принимают заказы, после чего к этому процессу подключаются подразделения, занимающиеся разработкой продукции и подготовкой производства. Если речь идет о новой продукции, то она проекти-

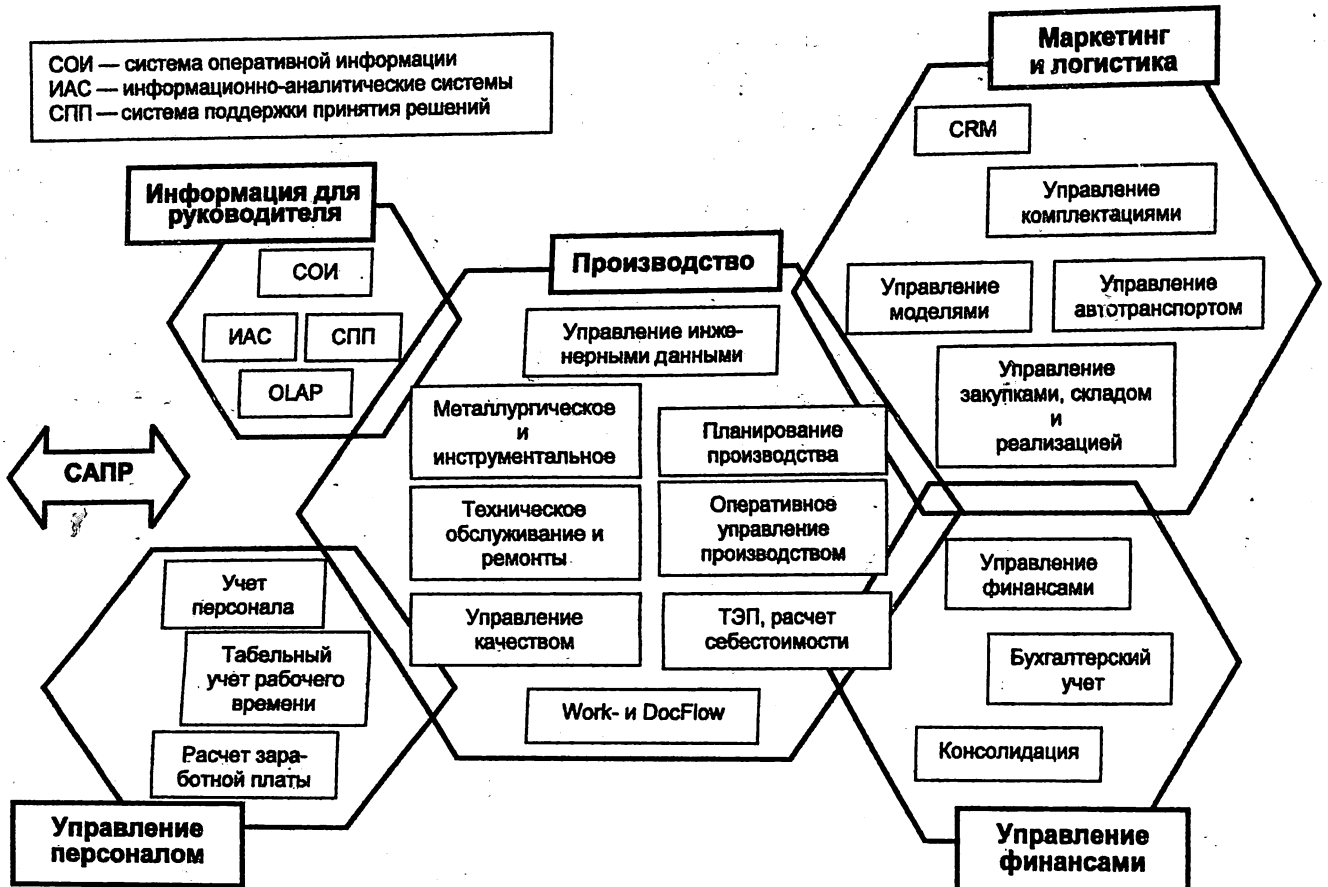


Рис. 1.7. Структура ИС корпорации «Парус» для машиностроительного предприятия

руется с нуля, в других случаях требуется улучшение конструкции или создание той или иной модификации. Затем готовится необходимая технология (определяются маршрутный и операционный технологические процессы, нормы выработки, расхода материалов и пр.).

Отдел закупок (или отдел комплектации) обеспечивает заказ и доставку нужных материалов и комплектующих; инструментальный цех готовит оснастку, металлургическое производство — необходимые заготовки, затем подключается основное производство, осуществляющее процесс изготовления и сборки готового изделия.

После отгрузки продукции возникают процедуры, связанные с учетом труда и расчетом заработной платы сотрудников предприятия, а также (в ряде случаев) обязательства по гарантийному и сервисному сопровождению. Большинство этих процедур сопровождается соответствующими финансовыми потоками, которые затем отражаются в бухгалтерском и налоговом учете и, наконец, обрабатываются в планово-экономическом отделе, который определяет основные показатели работы предприятия (рентабельность, прибыль и пр.) и составляет планы на будущее.

Решение призвано обеспечить и поддерживать весь жизненный цикл изделия, а это, прежде всего, управление его документацией. О любом объекте системы, используя контекстные меню, можно получить любые сведения — технологические, конструкторские, данные о планировании, себестоимости изделия, его количестве на складе и пр. Важно также, что система может поддерживать несколько вариантов документации для одного и того же изделия, используемых при разработке конструкции, подготовке производства и т. д.

В системе реализовано ведение единой базы данных, что позволяет в любой момент определить комплектацию конкретного изделия, выпущенного даже несколько лет назад, увидеть всю его техническую документацию — это важно для технического обслуживания, обеспечения запчастями и пр.

Таким образом, практическая реализация CSRP-системы предполагает использование в едином комплексе систем управления взаимоотношениями с клиентами (CRM), управления производством (MRP/ERP), поддержки принятия решений (OLAP) и электронной коммерции (B2B). Если эту цепочку дополнить системами CAD/CAM/CAE, управления цепочками поставок SCM , интерактивным электронным техническим руководством (ИЭТР) и добавить сверх этого систему управления данными об изделии (PDM), то получим реализацию всех **трех технологий концепции CALS**, обеспечивающих

единое электронное пространство (ЕЭП) жизненного цикла (ЖЦ) изделия и доступ к нему всех участников ЖЦ изделия. Эти различные по функциональному наполнению системы объединяют два очень важных свойства: они имеют во многом похожие технологии проектирования, разработки и сопровождения (за исключением CAD/CAM/CAE-систем) и реализуются в соответствии с архитектурой «клиент—сервер».

Поэтому в дальнейшем будут рассмотрены вопросы проектирования только информационных систем, реализующих CALS-технологии, с учетом особенностей их проектирования.

1.4. Основные этапы автоматизации предприятия

Теперь, когда нам понятны цели построения ИС и известны типы систем, попробуем разобраться, кто и какую выбирает функциональность системы и как затем разрабатывают, внедряют и сопровождают ИС.

Несмотря на очевидные плюсы от внедрения ИС (повышение оперативности управленческой информации, сокращение бумажного документооборота и непроизводительных затрат и, самое главное, повышение степени управляемости предприятия), не все категории работающих заинтересованы в положительном результате автоматизации.

В результатах автоматизации оказываются заинтересованными только три категории [38]:

- владелец (руководитель) предприятия — автоматизация принесет дополнительную прибыль и порядок на предприятии;
- потребитель услуг предприятия — благодаря автоматизации будет расти качество продукции и обслуживания;
- коллектив, непосредственно занимающийся автоматизацией, — он не останется без работы.

Не заинтересованы в положительном результате автоматизации, хотя на практике будут утверждать обратное:

- менеджеры среднего звена — они теряют свою значимость, им придется перепрофилироваться, осваивать работу в компьютерной системе, и их могут даже уволить, так как их работу будет выполнять компьютер;
- менеджеры высшего звена и руководящий персонал — они попадают в информационную зависимость от отдела автоматизации и получают сильных конкурентов в лице этого отдела.

Таким образом, коллектив, занимающийся автоматизацией предприятия, должен быть подотчетен только его владельцу. Если владелец или руководитель лично не возглавляет процесс разработки и внедрения ИС, то отрицательный результат автоматизации гарантирован.

Приступая к автоматизации предприятия, надо четко определить, какая система в большей степени соответствует данному предприятию, как с наименьшими затратами ее внедрить и как обеспечить в дальнейшем ее работоспособность.

Функциональная часть системы может быть исключительно бухгалтерской с элементами документооборота и производственного учета, а может быть с реализацией методологий MRP, MRP II, ERP, фрагментами CRM, SCM и т. д.

Большинство отечественных ИС выросло из бухгалтерских систем или создано с большой оглядкой на бухгалтерию. Бухгалтерия, особенно на российских предприятиях, является одним из основных подразделений, которое, занимаясь финансовым учетом и отчетностью, постоянно нуждается в современной и точной информации.

Но не следует забывать о том, что бухгалтерия — прежде всего обслуживающее подразделение, не являющееся производственным. Никто не будет спорить, что данные, которыми приходится оперировать при решении задач оперативного управления, существенно отличаются от данных, используемых для составления баланса. Если ИС рассчитает для бухгалтера точные остатки по счетам на начало и конец периода, покажет обороты, то бухгалтер будет доволен.

Однако строить ИС всего предприятия в интересах бухгалтерии неправильно — любая ИС предназначена для работы с нею менеджеров, основная функция которых — это принятие управленческих решений. Поэтому и главным объектом ИС должно быть управленческое решение. В случаях, когда необходимо получить управленческое решение, следует опираться не на отчеты, а на конкретные проекты решений, подготавливаемые в системе управления предприятием, т. е. информационная система должна обеспечить «прозрачность» производства для управления.

Например, при принятии заказа менеджеру недостаточно информации из отчета о текущих остатках продукции на складе, система должна в качестве проекта решения представлять данные о возможных датах готовности и отгрузки заказа. Подготовка подобных типовых решений, используемых в стандартных ситуациях, позволяет обеспечить стабильное качество управления на предприятии.

По оценкам специалистов, сейчас на российском рынке представлено более 150 информационных систем корпоративного масштаба отечественных и зарубежных разработчиков. Стоимость внедрения проектов колеблется от 10 до 250 тыс. долларов, а иногда и более. Также надо предполагать довольно значительные затраты на техническое сопровождение ИС и внедрение новых версий.

Некоторые предприятия, имеющие специфическую структуру или выпускаемую продукцию, разрабатывают свои собственные ИС, привлекая для этого фирмы-разработчики. Какой бы вариант ни был выбран (покупная ИС или заказная разработка), начальные этапы ЖЦ проекта содержат практически одинаковый перечень работ и начинать следует с анализа того, что уже автоматизировано на предприятии.

В большинстве случаев результат автоматизации «снизу вверх» представляет лоскутное одеяло из тиражных систем или систем собственного производства (службы сбыта и материально-технического снабжения, бухгалтерский и налоговый учет, учет кадров, труда и зарплаты и др.) (рис. 1.8).

Из-за этого во многих местах возникают информационные разрывы (крестики на рисунке), и чем их больше, тем менее достоверны данные о деятельности предприятия и более вероятен риск принятия ошибочных управленческих решений. Выходом в такой ситуации может быть комплексная (бесшовная) автоматизация «сверху вниз», реализованная с учетом тех данных и функций, которые присутствуют в обособленных друг от друга автоматизированных системах.

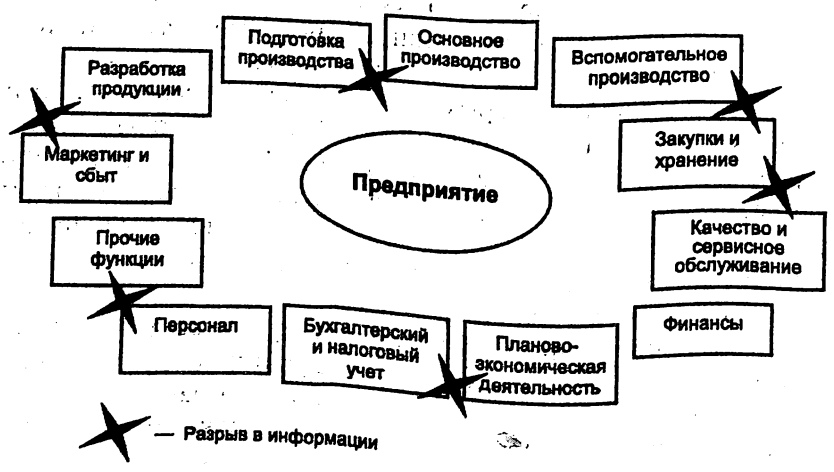


Рис. 1.8. «Кусочная» автоматизация предприятий

Автоматизация «сверху вниз» на предприятии проводится в несколько четко выраженных этапов [22]. Первым из них является этап ***обследования предприятия***, для которого предполагается создать или купить систему. Обследование предприятия производится разработчиками совместно с представителями заказчика, которые хорошо разбираются в процессах управления на предприятии.

На этом этапе необходимо решить следующие вопросы: какие подразделения следует автоматизировать, какая документация поступает в каждое подразделение, какие функции и в какой последовательности выполняются в рамках подразделения, кто является ответственным за выполнение каждой из этих функций, чем руководствуется исполнитель при выполнении каждой из функций, что является результатом работы подразделения (выходная документация).

Результатом обследования являются функциональная и информационная модели организации (модель «AS IS» — «Как есть»), которые в дальнейшем служат в качестве исходных данных для проектирования. Модель «AS IS» отражает существующее на момент обследования положение дел в организации и позволяет увидеть, как управляют предприятием сегодня, прежде чем перейти к тому, как это будут делать после внедрения ИС.

На стадии **стратегического планирования и анализа** выполняется анализ функциональной модели, который позволяет понять, где находятся наиболее слабые места, в чем будут состоять преимущества новых бизнес-процессов и насколько глубоким изменениям подвергнется существующая структура организации производства. Детализация процессов позволяет выявить недостатки организации даже там, где функциональность кажется очевидной.

Признаком малоэффективной деятельности могут быть бесполезные, неуправляемые и дублирующиеся работы, неэффективный документооборот (нужный документ не оказывается в нужном месте в нужное время), отсутствие обратных связей по управлению (на проведение работы не оказывает влияние ее результат) и входу (объекты или информация используются нерационально) и т. д.

Найденные в модели «AS IS» недостатки можно исправить при создании модели «TO BE» («Как должно быть») — модели новой организации управления производством, отражающей представление о новых технологиях работы организации. Подобная модель нужна для анализа альтернативных путей выполнения операций и документирования того, как компания будет вести бизнес в будущем.

Как правило, моделей «ТО BE» строят несколько и по определенному критерию выбирают лучшую. Проблема состоит в том, что таких критериев много и непросто найти важнейший. Для того чтобы определить качество созданной модели с точки зрения эффективности бизнес-процессов, необходима система количественной оценки. Пакет BPwin, например, предоставляет аналитику инструмент для оценки модели: функционально-стоимостный анализ, основанный на работах (**Activity Based Costing, ABC**).

По итогам проведенных исследований и анализа разрабатывается **техническое задание (ТЗ)**, которое является основным юридическим документом во взаимоотношениях между разработчиком и заказчиком. В ТЗ задаются функциональные требования к системе (состав задач, решаемых ИС), указываются ссылки на нормативные документы, которые регламентируют порядок выполнения функций и операций. Кроме этого, в ТЗ указывается целый ряд требований к ИС: по надежности, документированию, безопасности, защите информации от несанкционированного доступа, эргономике, а также порядок приемки системы.

После разработки ТЗ, когда уже достаточно информации о сложности и трудоемкости проекта, обычно заключается контракт между разработчиком и заказчиком с указанием стоимости и продолжительности работ. До этого момента все работы обычно выполняются без оплаты.

Стадия проектирования ИС связана с обоснованием и принятием принципиальных проектных решений по каждому из трех компонентов системы: программным модулям, таблицам базы данных и элементам пользовательского интерфейса. Проектировщики должны в **техническом проекте** описать каждый программный модуль, каждую таблицу базы данных и каждый элемент пользовательского интерфейса настолько конкретно, чтобы у программистов было достаточно информации при написании программ. На этом этапе выбираются технические средства, программное обеспечение, СУБД, решается проблема информационной безопасности и проблема связи с филиалами, если они есть.

Выбору программно-аппаратных средств и СУБД предшествует выбор фирмы, которая не только будет поставлять заказанные компоненты, а фактически станет партнером по бизнесу. От того, насколько внимательно и оперативно фирма-поставщик будет реагировать на нужды предприятия-заказчика, настолько будет гарантирован успех проекта.

В настоящее время эти функции выполняют фирмы, так называемые системные интеграторы, которые не только помогают выбрать нужную конфигурацию программных инструментариев, покупных компонентов ИС, аппаратных средств, но и обеспечивают их установку и техническое сопровождение в соответствии с заключенными договорами на это сопровождение. Обычно эти фирмы не только продают, но и сами разрабатывают и внедряют ИС. Поэтому контакты с ними полезны в плане получения всевозможного рода консультаций по вопросам, которые возникают в большом количестве на всех стадиях создания ИС.

Стадия разработки связана с программированием и отладкой компонентов приложения, которые создаются заново для данной системы. Разработка приложений производится с помощью инструментальных средств, отвечающих требованиям выбранной технологии. Обычно это средства быстрой разработки типа Delphi или программные средства СУБД, например Oracle Developer 10g.

На стадии интеграции и тестирования производится интеграция покупных и разработанных заново компонентов и комплексная проверка созданной системы: клиентских приложений; служб, выполняемых серверами; сетевой инфраструктуры. Устанавливается соответствие компонентов системы и всей ИС в целом требованиям ТЗ и желаниям заказчика с помощью тестов соответствия.

Для этих целей создается комиссия, которая при испытаниях руководствуется следующими документами:

- утвержденным заказчиком и согласованным с разработчиком ТЗ на ИС;
- действующими стандартами на проектирование и испытание программ и на их техническую документацию;
- программой испытаний по всем требованиям ТЗ;
- методиками испытаний по каждому разделу требований ТЗ;
- комплектом сопроводительной документации на комплекс программ.

Программа испытаний, методики их проведения и оценки результатов, разработанные совместно с заказчиком и разработчиком, должны быть согласованы и утверждены.

Результаты испытаний фиксируются в журналах испытаний, на основании которых формируются протоколы.

Выводы из протоколов по всей программе испытаний обобщаются в акте, в котором делается заключение о степени соответствия системы требованиям заказчика и ТЗ.

На стадии ввода в действие производится перенос разработанной ИС с инструментальной платформы разработчика на реальную платформу ИС, т. е. инсталляцию ИС на аппаратно-программном комплексе заказчика. После адаптации и настройки ИС на реальные условия проводятся приемочные испытания, которые заключаются в комплексной проверке реально функционирующей в полном объеме ИС на соответствие ТЗ по разработанной программе и методикам испытаний. Производится анализ результатов испытаний и устранение недостатков, выявленных при испытаниях.

В случае положительных результатов испытаний оформляется акт о приемке ИС в постоянную эксплуатацию. Подписание этого акта является документальным подтверждением того, что разработчик полностью реализовал все пункты ТЗ и выполнил все условия контракта.

На этапе сопровождения анализируется функционирование системы, выявляются отклонения эксплуатационных характеристик от проектных значений, устраняются причины этих отклонений, подготавливаются и выпускаются новые версии ИС с соответствующими изменениями в документации.

Контрольные вопросы

1. Какие CALS-технологии используются для реализации стратегии CALS?
2. Назначение и содержание CALS-технологий.
3. Дать характеристику информационным системам, реализующих CALS-технологии.
4. Проанализировать взаимосвязи стандартов управления MRP, MRP II, ERP и CSRP.
5. Назначение и функциональный состав ИС, обеспечивающий выход ERP-системы «за ворота предприятия».
6. Содержание этапов проектирования ИС.
7. Какие задачи решаются на этапе обследования предприятия?
8. Какие документы оформляются при проведении испытаний ИС?
9. Какие достоинства в применении проектирования ИС «снизу вверх»?
10. Дать характеристику структуры ИС для машиностроительного предприятия.

Глава 2
ТЕХНОЛОГИИ ПРОЕКТИРОВАНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ

2.1. Общие сведения о технологии проектирования ИС

Технологии и инструментальные средства проектирования составляют основу проекта любой ИС. Каждая технология поддерживается конкретными стандартами, методиками и инструментальными средствами, которые обеспечивают выполнение процессов жизненного цикла (ЖЦ) информационной системы.

Для реализации указанных технологий в обязательном порядке используется CASE-технология, которая входит в разряд наиболее стабильных информационных технологий и которая в явном или неявном виде присутствует в каждой технологии с различными уровнями автоматизации выполняемых действий. В результате получаем современный подход к проектированию ИС на основе CASE-технологии — технологии автоматизированной разработки систем.

Технология проектирования определяется как совокупность **трех составляющих:**

- пошаговой процедуры, определяющей последовательность технологических операций проектирования;
- критериев и правил, используемых для оценки результатов выполнения технологических операций;
- нотаций (графических и текстовых средств), используемых для описания проектируемой системы [23].

Технология проектирования, разработки и сопровождения ИС должна удовлетворять следующим требованиям:

- технология должна поддерживать полный жизненный цикл ИС;
- технология должна обеспечивать гарантированное достижение целей разработки ИС с заданным качеством и в установленное время;
- технология должна обеспечивать возможность выполнения крупных проектов в виде подсистем;

• технология должна предусматривать возможность управления конфигурацией проекта, ведение версий проекта, возможность автоматического выпуска проектной документации;

• технология должна обеспечивать независимость выполняемых проектных решений от средств реализации ИС (СУБД, ОС, языков программирования);

• технология должна быть поддержана комплексом согласованных CASE-средств, обеспечивающих автоматизацию процессов, выполняющихся на всех стадиях ЖЦ.

Реальное применение любой технологии проектирования, разработки и сопровождения ИС предполагает использование стандартов, которые должны соблюдаться всеми участниками проекта:

• стандарты проектирования;

• стандарты оформления проектной документации;

• стандарты пользовательского интерфейса.

Стандарты проектирования должны устанавливать набор необходимых моделей (диаграмм) на каждой стадии проектирования; правила фиксации проектных решений на диаграммах (правила именования объектов, набор атрибутов для объектов и правила их заполнения и т. д.); механизм обеспечения совместной работы над проектом.

Стандарт оформления проектной документации должен устанавливать комплектность, состав и структуру документации на каждой стадии проектирования; требования к ее оформлению (включая требования к содержанию разделов, подразделов, таблиц и т. д.); правила подготовки, рассмотрения, согласования и утверждения документации с указанием предельных сроков для каждой стадии.

Стандарт интерфейса пользователя должен устанавливать правила оформления экранов (шрифты и цветовая палитра), состав и расположение окон и элементов управления; правила оформления текстов помощи; перечень стандартных сообщений и правила обработки реакций пользователя.

Каждая из технологий ориентируется на определенные модели ЖЦ ИС.

2.2. Модели жизненного цикла ИС

В основе деятельности по созданию и использованию ИС лежит понятие ее жизненного цикла (ЖЦ). ЖЦ отражает различные состояния ИС, начиная с момента возникновения необходимости в данной

системе и заканчивая моментом ее снятия с эксплуатации. ЖЦ ИС представляет собой некоторую последовательность этапов, для каждого из которых определяются состав и последовательность выполняемых работ, получаемые результаты, методы и средства, необходимые для выполнения работ, роли и ответственность участников и т. д. Такое формальное описание ЖЦ ИС позволяет спланировать и организовать процесс коллективной разработки и обеспечить управление этим процессом.

Существует несколько моделей ЖЦ ИС, которые отличаются различным количеством этапов и их содержанием: каскадная, спиралевидная, непрерывной разработки, быстрого прототипирования и т. п. Выбор модели определяется сложностью ИС, ее масштабом, желанием заказчика, предпочтениями разработчика и т. д. Каждая из ведущих фирм — разработчиков CASE-средств и СУБД имеет свой набор моделей ЖЦ, выполнение этапов которых обеспечивается программным инструментарием.

Традиционной для российских стандартов является «каскадная» («классическая» или «водопадная») модель ЖЦ ИС (рис. 2.4). В общем случае считается, что в процессе развития проекта над ним работают различные команды специалистов и при завершении этапа проекта готовится весь необходимый комплект проектной документации, фиксирующий принятые решения.

При таком подходе оформляется большое количество документов, которые, как правило, не нужны конечному пользователю, но их создание занимает время и ресурсы. Нередко проектные группы, работающие на этапах, не имеют четкого представления об общих целях и задачах данного проекта, его особенностях и нюансах.

Основной характеристикой «каскадной» модели является разбиение всей разработки на этапы и последовательное выполнение всех этапов проекта в строго фиксированном порядке. Переход с одного этапа на следующий происходит только после завершения всех работ на текущем этапе (см. рис. 2.4). Каждый этап завершается выпуском полного комплекта документации, достаточной для того, чтобы разработка могла быть продолжена другой командой разработчиков.

Основной недостаток «каскадной» модели ЖЦ в том, что реальный процесс создания ИС никогда полностью не укладывается в такую жесткую схему: **«Все работы должны выполняться на каждом этапе сразу и за один раз».**

В процессе создания ИС изменяются требования к системе из-за изменения законодательства, структуры предприятия, желаний заказ-

чика и т. п., что требует возвращения на предыдущие этапы для уточнения или пересмотра ранее полученных решений. А поскольку требования к системе «заморожены» в виде технического задания на все время ее создания, то заказчик может внести свои изменения только после полного завершения работы над системой либо в результате заключения с разработчиком частного технического задания. В результате пользователь получает систему, не удовлетворяющую его потребностям.

На практике обычно применяют эту модель в виде **циклической**, когда можно вернуться на предыдущие этапы и выполнить необходимые доработки с учетом измененных требований или желаний заказчика. Эти дополнительные возможности модели отображены на рис. 2.4 пунктирными стрелками снизу вверх. Документально это может оформляться в виде частных ТЗ.

Для преодоления этих проблем была предложена ***спиральная (спиралевидная)*** модель ЖЦ (рис. 2.1), которая делает основной упор на начальные этапы ЖЦ: анализ и проектирование. На каждом витке спирали выполняется создание очередной версии продукта, уточняются требования проекта, определяется его качество и планируется содержание работы следующего витка. Один виток спирали при этом представляет собой законченный проектный цикл по типу «каскадной» модели.

Каждый виток спирали соответствует созданию фрагмента или версии ИС, на нем уточняются цели и характеристики проекта, планируются работы следующего витка спирали. Фактически на каждом

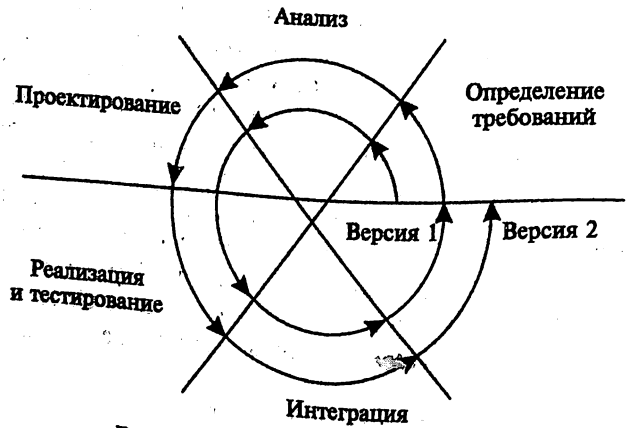


Рис. 2.1. Спиральная модель ЖЦ ИС

витке спирали создается прототип будущей ИС, который углубляется и конкретизируется на последующих витках до тех пор, пока не будет доведен до полной реализации системы.

При этом неполное завершение работ на каждом этапе спирального цикла создания системы позволяет переходить на следующий этап, не дожидаясь полного завершения работы на текущем этапе. Итеративный способ разработки позволяет выполнить недостающую работу на следующей итерации. Главная задача для этой модели ЖЦ — как можно быстрее показать заказчику работоспособный продукт, активизируя тем самым процесс уточнения и дополнения требований.

В настоящее время достаточно популярна **модель непрерывной разработки («продолжающейся разработки»)** (рис. 2.2). Особенностью такого подхода является непрерывный процесс разработки и развития больших ИС с планируемыми точками передачи в эксплуатацию новых версий и новых функциональных блоков (подсистем, задач).

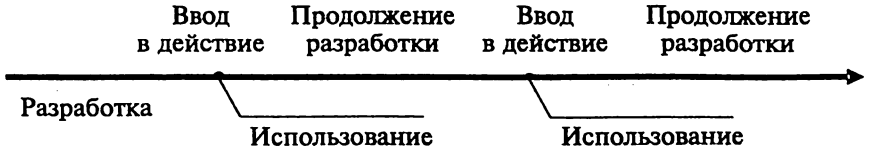


Рис. 2.2. Модель непрерывной разработки ИС

В последнее время модель ЖЦ «продолжающейся разработки» с целью еще более ускоренного внедрения трансформировалась в несколько моделей ЖЦ, получивших название **«быстрого прототипирования»**: fast track, lite и другие. Они содержат небольшое количество этапов, например, в модели fast track (Oracle CDM) их всего три: моделирование требований; проектирование и создание системы; внедрение.

Принцип работы итеративной модели аналогичен спиральной модели ЖЦ, но отличие в том, что здесь результатом итерации является макет-прототип фрагмента будущей системы. В процессе разработки системы выполняются итерации до тех пор, пока макет-прототип не приобретет все необходимые свойства фрагмента системы в соответствии с ТЗ.

Эти модели быстрого прототипирования применимы для ИС средней и малой сложности, наряду с быстрым эффектом дают снижение управляемости проекта в целом и приводят к появлению проблем при стыковке различных фрагментов ИС.

Для правильного выбора модели ЖЦ ИС обычно руководствуются четырьмя критериями. В табл. 2.1 представлены эти критерии и цели, которые достигаются на их основании. Эти критерии взаимосвязаны, нельзя достичь всех четырех целей одновременно (рис. 2.3). Например, если стоимость работ ограниченна, а это бывает почти всегда, то мы имеем две альтернативы: или не будет четкой даты окончания работы, или нужно соглашаться с менее качественной системой.

Рис. 2.3. Графическое представление критериев выбора модели ЖЦ ИС

Таблица 2.1. **Критерии и цели выбора модели ЖЦ**

Критерий	Цель
Стоимость	Низкая стоимость
Время, затрачиваемое на создание ИС	Быстрое завершение разработки
Качество (характеристики, ошибки)	Высокое качество
Степень риска сбоев	Низкая степень риска сбоев

2.3. Технология проектирования на базе комплекса российских стандартов ГОСТ 34

В ГОСТе 34.601—90 «Автоматизированные системы. Стадии создания» определяются стадии и этапы создания ИС (рис. 2.4).

1. ***Формирование требований:***
   - обследование объекта и обоснование необходимости создания ИС;
   - формирование отчета о выполненной работе и заявки на разработку тактико-технического задания.

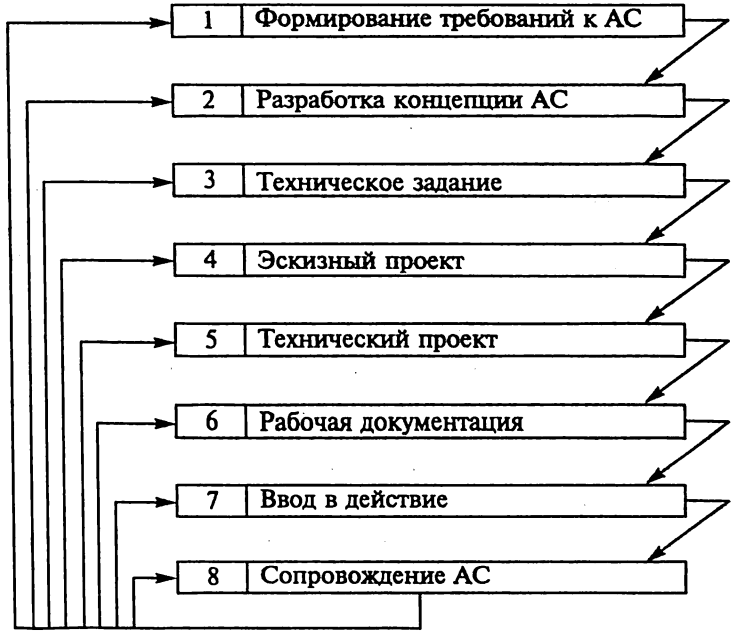


Рис. 2.4. Этапы жизненного цикла ИС

2. ***Разработка концепции АС:***
   - изучение объекта (модель «AS IS» («Как есть»));
   - разработка вариантов концепции ИС и выбор варианта (модели «TO BE» («Как должно быть»));
   - оформление отчета о выполненных работах (в частности, технико-экономическое обоснование).
3. ***Техническое задание:***
   - разработка и утверждение ТЗ.
4. ***Эскизный проект:***
   - разработка предварительных проектных решений по системе и ее частям (определяются функции ИС, функции подсистем; состав комплексов задач и интерфейсов; концепция БД, ее укрупненная структура; функции СУБД; состав вычислительной системы; функции и параметры основных аппаратных средств);
   - разработка документации на ИС.
5. ***Технический проект:***
   - разработка проектных решений по системе и ее частям (разработка общих решений по системе и ее частям, функционально-алгоритмической структуре системы по функциям персона-

ла, спецификаций на программные модули и таблицы БД, организационной структуре, по структуре технических средств, по алгоритмам решаемых задач, применяемым языкам, по организации и ведению информационной базы, по программному обеспечению);

- разработка документации на ИС (полное описание совокупности принятых проектных решений и достаточного для дальнейшего выполнения работ по созданию ИС (ГОСТ 34.201—89)).

6. ***Рабочая документация:***
   - разработка программной и эксплуатационной документации на систему и ее части (для обеспечения работы по вводу ИС в эксплуатацию, а также для поддержания уровня эксплуатационных характеристик системы (ГОСТ 34.201—89));
   - разработка или адаптация (покупных) программ.
7. ***Ввод в действие:***
   - подготовка объекта автоматизации к вводу ИС в действие (организационная подготовка);
   - подготовка персонала;
   - строительно-монтажные работы;
   - пусконаладочные работы (автономная наладка технических и программных средств, загрузка информации в БД; комплексная наладка всех средств системы);
   - проведение предварительных испытаний;
   - проведение опытной эксплуатации;
   - проведение приемочных испытаний.
8. ***Сопровождение АС:***
   - выполнение гарантийных обязательств;
   - анализ функционирования ИС, ее модификация и выпуск новых версий.

Таким образом, последовательность этапов ЖЦ строится в соответствии с принципами нисходящего проектирования и, как правило, носит итерационный характер: ранее выполняемые этапы циклически повторяются в связи с изменениями требований и внешних условий, введением ограничений и т. п.

На каждом этапе ЖЦ генерируется определенный набор документов и технических решений. При этом для каждого этапа исходными являются документы и решения, полученные на предыдущем этапе. Каждый этап завершается верификацией созданных документов и решений с целью проверки их соответствия стандартам и ТЗ.

Главной особенностью «каскадной» модели является концентрация сложности на начальных этапах ЖЦ при относительно невысокой сложности и трудоемкости последующих этапов. Более того, качество работ, выполненных на начальных этапах, определяет успех разработки в целом.

На начальных этапах **системный аналитик** обычно встречается с рядом взаимосвязанных проблем, основной из которых является чрезмерное количество подробных сведений о предметной области. Поэтому ему сложно получить исчерпывающую информацию для оценки требований к системе со стороны заказчика. А **заказчик** со своей стороны не имеет достаточных знаний о проблеме обработки данных, чтобы судить о том, что является выполнимым, а что нет. К тому же для заказчика часто непонятна спецификация ИС из-за технических терминов.

Решение этой проблемы взаимного недопонимания можно существенно облегчить, применяя современные структурные методы, среди которых центральное место занимает ***структурный анализ***.

Структурным анализом принято называть исследование, которое начинается с общего обзора системы и затем детализируется, приобретая иерархическую структуру с увеличивающимся числом уровней (на каждом уровне от трех до семи элементов). В основе структурного анализа используются два базовых принципа: принцип «разделяй и властвуй» и принцип иерархического упорядочения.

Первый принцип системного анализа позволяет решение трудных проблем выполнять путем разбиения их на множество более мелких независимых задач, легких для понимания и решения. При этом свою работу системный аналитик начинает методом «сверху вниз», реализуя первый принцип системного подхода: движение от общего к частному в различных аспектах. Это, прежде всего, движение мысли по иерархической структуре организации (как системы) сверху вниз, рассмотрение отдельных подсистем.

Процессы функционирования департаментов и подразделений делятся на подпроцессы, а те — на отдельные действия. Сложные свойства организации в целом, ее департаментов, подразделений и отдельных сотрудников декомпозируются на менее сложные и далее — на квазипростые (если декомпозировать далее нет необходимости) и простые (если декомпозировать далее уже невозможно) свойства.

Второй принцип в дополнение к первому декларирует, что эти независимые задачи должны быть организованы в виде древовидных иерархических структур. Реализация второго принципа предусматри-

вает единство анализа и синтеза; т. е. предполагается, что каждый шаг по анализу организации выполняется с постоянным прицелом на ее последующую реструктуризацию, на синтез более эффективной структуры и разработку более эффективного алгоритма функционирования организации.

Результаты применения структурного анализа при нисходящем проектировании ИС очень наглядно представлены на диаграмме «Дерево узлов» (см. рис. 4.12). Реализация этой технология на базе инструментария All Fusion Modeling Suite представлена в гл. 4 с использованием методологий Swim Lane, IDEF0, IDEF3, DFD и IDEF1X.

2.4. Методология Oracle Custom Development Method (CDM)

2.4.1. Модели ЖЦ Oracle CDM

Методология Oracle Custom Development Method (CDM) по разработке прикладных ИС под заказ — конкретный материал, детализированный до уровня заготовок проектных документов, рассчитанных на прямое использование в проектах ИС с опорой на инструментарий фирмы Oracle.

Методология Oracle CDM — это совокупность точно определенных процессов заказной разработки с разными режимами управления. Методология, в основе которой лежит CASE-технология, обеспечивает точное определение бизнес-требований в самом начале процесса разработки и их сохранение на протяжении всего процесса разработки. Методология CDM радикально повышает возможность успешной реализации проекта.

Первоначально методология CDM предназначалась для крупных и средних проектов, но в принципе ее можно использовать и для небольших. Она определяет задачи и проектные решения, которые должны включаться в полный жизненный цикл любого проекта.

Общая структура CDM определяется методологией системной разработки на базе процессов. ***Процесс*** — это связанная совокупность задач, отвечающая одной из конкретных целей проекта. Результатом одного процесса является одно или несколько ключевых проектных решений.

Модель ЖЦ методологии Oracle CDM имеет два измерения (рис. 2.5). **Первое измерение** связано с тем, какой процесс должен

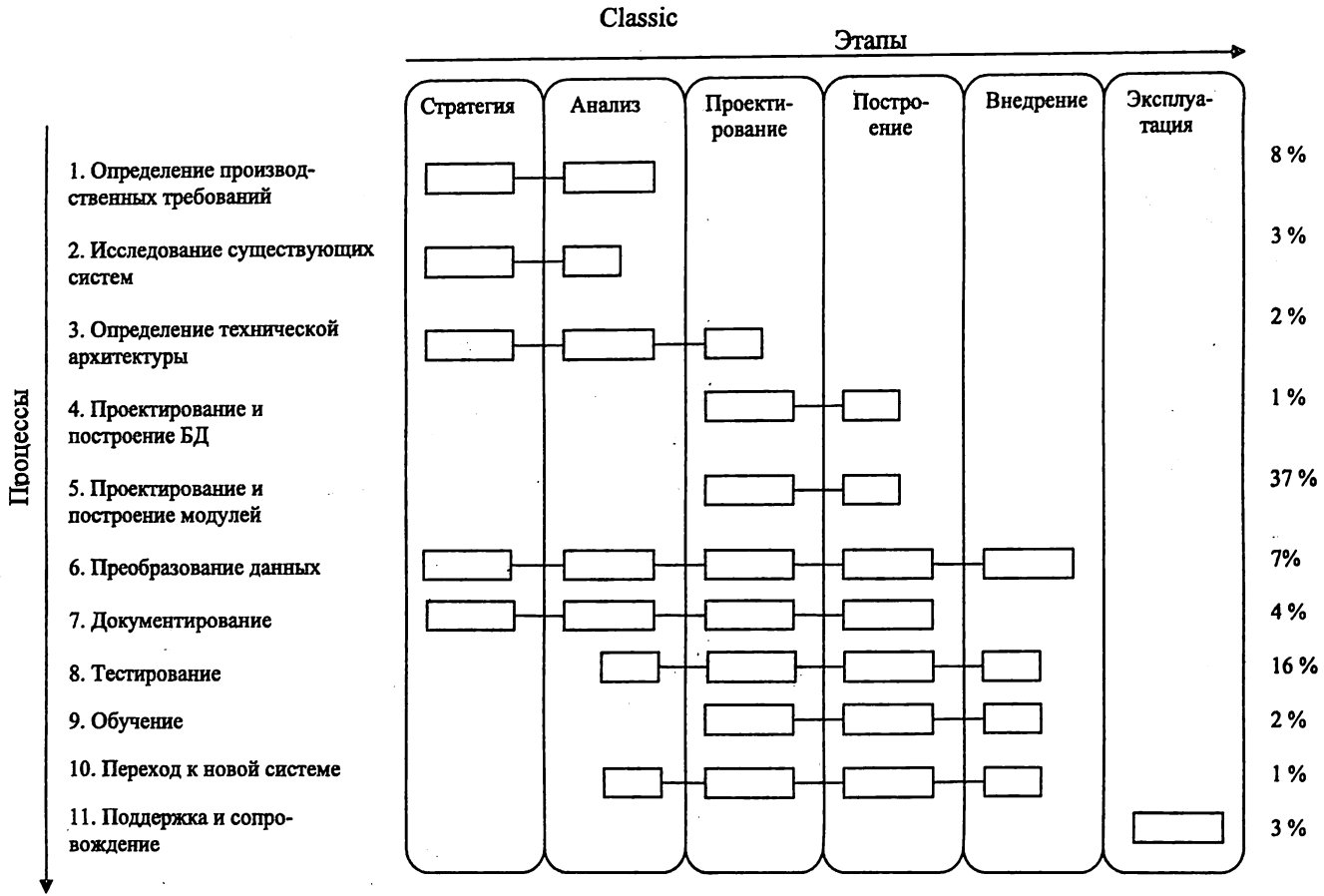


Рис. 2.5. Структура Oracle CDM для модели ЖЦ classic

быть выполнен для разработки проекта. Это измерение определяется процессами в рамках CDM. **Второе измерение** связано с тем, когда должны выполняться процессы в ЖЦ проекта. Это измерение определяется этапами ЖЦ. В методике CDM используются **три модели ЖЦ**, которые обеспечивают деление проекта на этапы: ***классическая (Classic)***, ***быстрой разработки (Fast Track)***, ***облегченной разработки (Lite)*** (рис. 2.6).

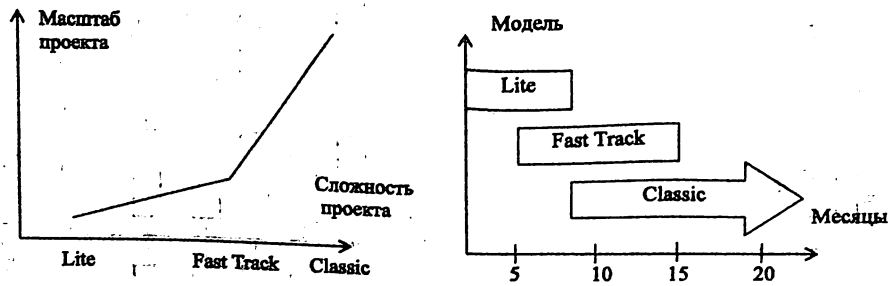


Рис. 2.6. Выбор моделей ЖЦ для конкретных проектов ИС

При этом для каждой модели ЖЦ существует свой набор этапов и процессов. Например, модель ЖЦ **Classic** имеет шесть этапов и одиннадцать процессов (см. рис. 2.5). **Классический подход** в CDM применяется для крупных разработок продолжительностью более восьми месяцев и содержит шесть этапов: определение требований (стратегия), анализ, проектирование, реализация, внедрение и эксплуатация.

Модель Fast Track предназначается для заказных разработок среднего объема (продолжительностью 6—8 месяцев) и содержит только три этапа (моделирование требований, проектирование и создание системы, внедрение) и все 11 процессов, как и в модели Classic.

Модель Lite (облегченная разработка) применяется для небольших проектов продолжительностью до полугода и использует всего два этапа (прототипирование и построение; внедрение) и девять процессов (кроме 6-го и 3-го).

2.4.2. Процессы и этапы ЖЦ модели Classic в Oracle CDM

Классическая (Classic) модель ЖЦ Oracle CDM (рис. 2.5) включает в себя наибольшее по сравнению с моделями Fast Track и Lite количество этапов и процессов, соответственно шесть и одиннадцать. Поэтому рассмотрим более подробно содержание процессов и этапов

модели Classic. Длина прямоугольника на пересечении процесса и этапа означает продолжительность его выполнения на данном этапе.

Процесс *определение производственных требований (постановка задачи)* заключается в определении бизнес-требований прикладной системы. Группа анализа сначала создает модель бизнес-процессов, затем модель бизнес-данных для представления информационных потребностей данного предприятия и модель бизнес-функций, в которой детально определены все бизнес-функции, указанные в модели процессов.

Затем в эти модели вводят технологические требования, такие как пользовательский интерфейс, время ответной реакции и т. п.

Процесс *исследование существующих систем* — основным требованием многих заказных разработок является замена функциональных возможностей существующей системы или работа с применением существующей технической архитектуры.

Процесс *определение технической архитектуры* заключается в определении элементов технической базы данных разработки. Аналитики начинают с начального плана возможностей и разрабатывают начальную техническую архитектуру. После получения более детальной информации группа анализа трансформирует ее в два проектных решения:

- определение аппаратной и программной базы;
- определение распределенной архитектуры.

Процесс *проектирование и построение базы данных* начинается с создания логического проекта БД и заканчивается созданием DDL-скрипта для эксплуатационной БД.

Процесс *проектирование и реализация модулей* представляет ядро проекции CDM. На основе модели системных процессов, модели системных данных и модели системных функций, а также технической архитектуры вначале разрабатывают проект системной архитектуры и модель модульных процессов, а затем специфицируют функциональные и технические детали каждого модуля. После этого программисты используют проектную документацию и/или прототипы для создания прикладного кода.

Целью **процесса *преобразование данных*** является миграция, преобразование и тестирование всех **наследуемых данных**, необходимых для тестирования и работы нового приложения. Процесс включает проектирование, кодирование, тестирование модулей, создаваемых разработчиком для преобразования наследуемых данных, а также выполнение всех преобразований самими разработчиками.

Процесс ***документирование*** — создается вся пользовательская, техническая и учебная документация по данному проекту.

Процесс ***тестирование*** включает: тестирование модулей, ориентированное на функциональные возможности; тестирование модулей в целом на соответствие бизнес-требованиям; системные испытания; приемосдаточные испытания.

Назначением процесса ***обучение*** является обучение группы пользователей и администраторов для выполнения задач, связанных с применением новой прикладной системы.

Проектная группа также может проводить обучение персонала технического обслуживания и персонала для проведения приемосдаточных испытаний.

Процесс ***внедрение или переход на новую систему*** начинается на начальных стадиях проекта при определении конкретных требований для перехода на новую прикладную систему. Затем в этот процесс включаются такие задачи, как план инсталляции, подготовка эксплуатационной среды, выполнение перехода на новую систему, вывод из эксплуатации старой системы.

Процесс ***поддержка и сопровождение*** имеет четыре цели: текущий контроль и ответная реакция на все проблемы, связанные с системой; наращивание приложений для устранения ошибок и проблем, связанных с производительностью; оценка системы в условиях эксплуатации; планирование модернизации.

Модель Classic в CDM имеет **шесть этапов**, которые определяют последовательность выполнения процессов (см. рис. 2.5).

Этап ***определение требований (стратегия)*** — определение бизнес-требований и требований для информационной системы на верхнем уровне, необходимых для выполнения набора определенных бизнес-задач. Результатом этапа является точное работоспособное определение масштаба проекта.

Этап ***анализ*** — производится исследование предметной области, определенной на этапе стратегии. Члены проектной группы достигают полного понимания предметной области в результате создания точных моделей и описаний, которые определяют, что делается в предметной области и какая информация используется.

Этап ***проектирование*** — трансляция требований, полученных на этапе анализа, в детальные системные спецификации, с учетом технической архитектуры и имеющихся технологий.

Этап ***построение*** — кодирование и тестирование приложения с применением соответствующих методов.

Этап ***внедрение*** — инсталляция новой ИС, подготовка персонала заказчика к использованию и администрированию системы с последующим вводом в эксплуатацию.

Этап ***эксплуатация*** — обеспечение поддержки приложения, текущий контроль за приложением, планирование дальнейших функциональных расширений.

2.4.3. Обзорная диаграмма этапа определения требований (стратегия) модели Classic CDM

В методике Oracle CDM для каждого этапа создается обзорная диаграмма (рис. 2.7), которая конкретизирует:

- **предпроцессы** (какие документы и проектные решения должны быть в наличии перед началом выполнения работ на очередном этапе);
- **ключевые проектные решения** (какие документы и технические решения должны быть получены на выходе каждого процесса данного этапа в результате его выполнения).

Кроме этого, для каждого этапа определяется его цель, критические факторы успеха, наиболее вероятные риски, методы снижения рисков и рекомендации по комплектованию штатов.

Рассмотрим выполнение этих действий на примере **этапа определения требований (стратегия)**.

Цель этапа: определение бизнес-требований и требований верхнего уровня иерархии для ИС, необходимых для выполнения определенных бизнес-целей. Результатом этапа является точное определение функциональности проекта.

Критические факторы успеха:

- точное определение бизнес-целей, для достижения которых должен выполняться проект;
- активное участие наиболее знающих пользователей и технических представителей участков производственной деятельности, связанных с проектными задачами;
- обеспечение для проектной группы доступа к информации, относящейся к существующим бизнес-процессам и системам, связанным с проектными задачами;
- эффективное управление функциональным масштабом и решение спорных вопросов руководителем проекта.

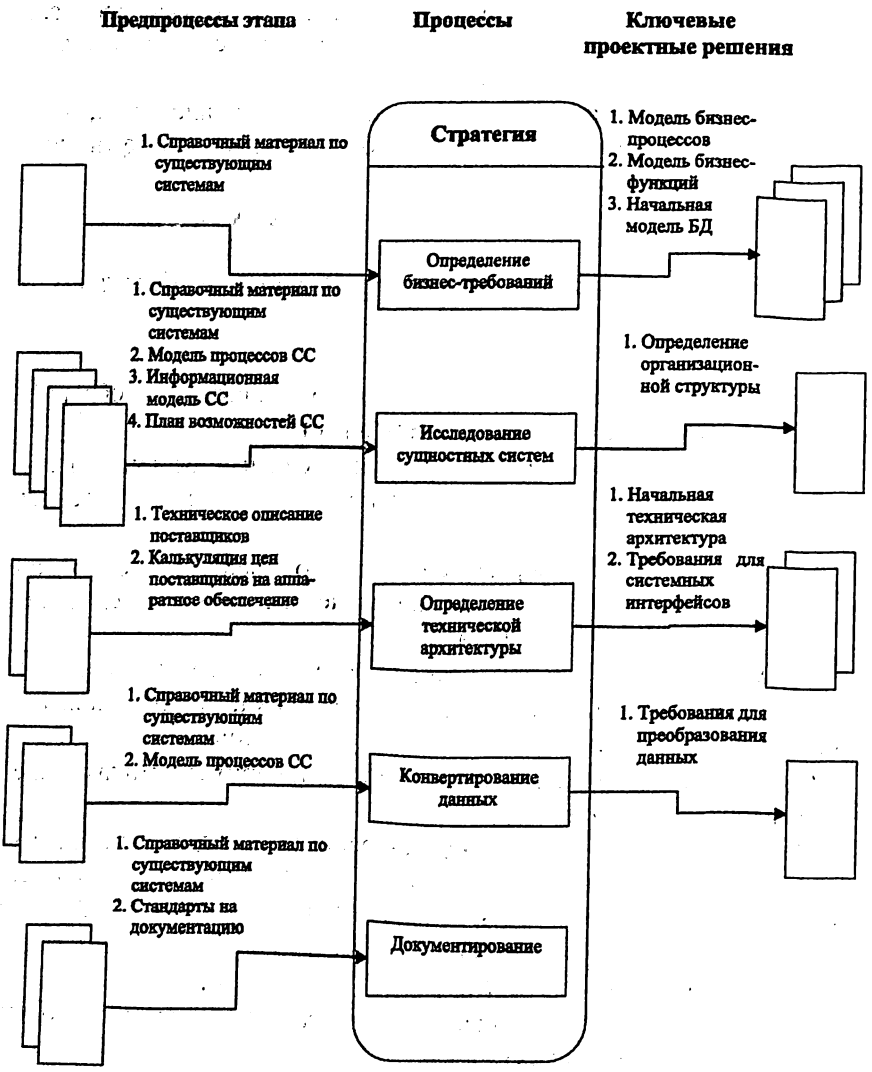


Рис. 2.7. Обзорная диаграмма этапа «Определение требований»

Управление рисками. Наиболее вероятные риски на этапе определения стратегии:

- неточное определение масштаба проекта;
- отсутствие финансирующего заказчика;
- отсутствие опыта выполнения подобных проектов у руководителя проекта или ведущих членов проектной группы;

• отсутствие опыта работы с инструментальными средствами или методами моделирования CDM у ведущих членов проектной группы.

Методы снижения рисков:

• применение Oracle PJM (Project Management Method) для документирования масштаба работ;

• включение в организационную схему проекта финансирующего заказчика, частое проведение совещаний с финансовым заказчиком по вопросам, связанным с проектом;

• обучение проектной группы работе с инструментальными средствами CDM.

Рекомендации по комплектованию штатов для выполнения различных процессов:

• ***определение бизнес-требований*** — в группу должно войти большинство аналитиков проекта;

• ***исследование существующих систем*** — лучше использовать специалистов по старым системам и пользователей, работающих на них;

• ***определение технической архитектуры*** — использовать опытного специалиста по разработке технической архитектуры, знающего аппаратное обеспечение, сети и операционные системы;

• ***преобразование данных*** — специалисты в области программирования, тестирования, настройки производительности и внедрения;

• ***документирование*** — использовать профессиональных сотрудников по технической документации (технических писателей).

2.4.4. Обзорное представление этапа определения требований модели Classic CDM

Обзорное представление этапа «Определение требований» реализуется в виде графика распределения задач в хронологическом порядке их выполнения (рис. 2.8).

Эти задачи получены в результате декомпозиции процессов, входящих в обзорную диаграмму для данного этапа (см. рис. 2.7). Каждая задача имеет свой порядковый номер и аббревиатуру имени процесса на английском языке.

Как следует из схемы обзорного представления, некоторые задачи различных процессов выполняются параллельно, а другие — в строгой последовательности. Следует отметить, что методология CDM не допускает удаление или введение хотя бы одной задачи. Фактически

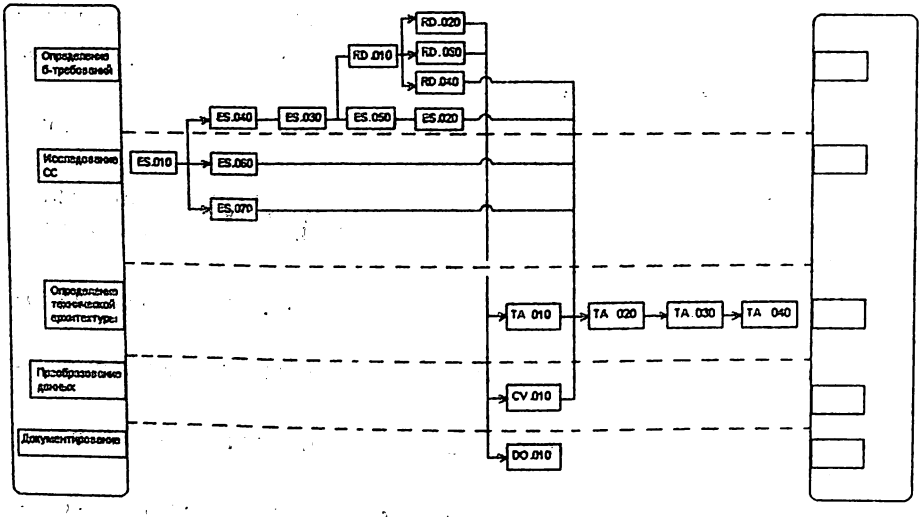


Рис. 2.8. Обзорное представление этапа определения требований

эта методология представляет собой шаблон действий, строгое выполнение которого гарантирует успех проекта.

В обзорном представлении этапа «Определение требований» присутствуют следующие задачи:

- **для процесса определения бизнес-требований:**
  - RD.010 — создать модель бизнес-процессов;
  - RD.020 — получить описание предметной области (верхний уровень);
  - RD.030 — создать модель бизнес-функций (верхний уровень);
  - RD.040 — создать начальную модель бизнес-функций;
- **для процесса исследования существующих систем (СС):**
  - ES.010 — получить справочный материал по СС;
  - ES.020 — документировать организационные структуры;
  - ES.030 — создать информационную модель СС (верхний уровень);
  - ES.040 — создать модель процессов СС;
  - ES.050 — документирование интерфейсов СС;
  - ES.060 — получить техническую архитектуру СС;
  - ES.070 — получить план возможностей СС;
- **для процесса определения технической архитектуры:**
  - TA.010 — документировать требование для системных интерфейсов;

— TA.020 — подготовить начальный план возможностей;
— TA.030 — подготовить начальную техническую архитектуру;
— TA.040 —документировать эксплуатационные требования верхнего уровня для ИС;

- **для процесса преобразования данных:**
  — CV.010 — документировать требования для преобразования данных;
- **для процесса документирования:**
  — DO.010 — подготовить глоссарий.

Таким образом, на примере этапа «Определение требований» модели ЖЦ Classic получено полное представление о составе процессов и этапов метода Oracle CDM (см. рис. 2.5), составе документов и технических решений на входах и выходах процессов каждого этапа (см. рис. 2.7), а также о составе задач для процессов каждого этапа (см. рис. 2.8). Они позволяют в полном объеме получить представление о методологии Oracle CDM и особенностях ее применения.

2.5. Общие понятия о методе управления проектом заказной разработки Oracle PJM

Метод Oracle CDM включает стандартный подход к управлению проектами — Project Management Method (PJM). Целью метода управления проектом Oracle PJM является создание базы для планирования, оценки, контроля и слежения за проектами всех типов в согласованном режиме.

Модель ЖЦ Oracle PJM также имеет два измерения: пять процессов и пять этапов (рис. 2.9).

Первое измерение связано с тем, какая работа должна быть выполнена для управления проектом и поддержки проекта. Это измерение определяется процессами в рамках PJM.

Второе измерение связано с тем, когда должны выполняться задачи управления и поддержки в ЖЦ проекта. Это измерение определяется категориями (этапами) ЖЦ.

Процессы метода управления проектами Oracle PJM:

- ***контроль и отчетность*** — включает задачи, обеспечивающие определение масштаба проекта и подхода к проекту, осуществление управления изменениями и контроля над рисками. Этот процесс со-

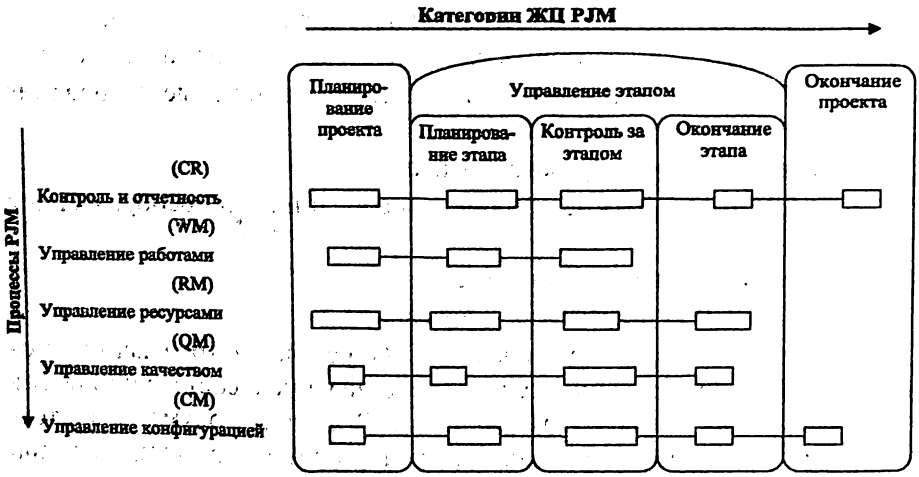


Рис. 2.9. Жизненный цикл управления проектами

держит инструкции по внешней отчетности, по достигнутому прогрессу и контролю над ***планом обеспечения качества***;

• ***управление работами*** — включает задачи, обеспечивающие определение всех видов работ, осуществление текущего контроля и руководство всеми работами при выполнении проекта. В этот процесс также входит ведение финансового учета проекта;

• ***управление ресурсами*** — этот процесс обеспечивает проект персоналом необходимой квалификации, а также рабочую среду для поддержки проекта;

• ***управление качеством*** — определяет меры, необходимые для реализации качества и обеспечивающие соответствие проекта целям и ожиданиям заказчика на протяжении ЖЦ проекта;

• ***управление конфигурацией*** — включает задачи, обеспечивающие хранение, организацию, слежение и контроль над всеми элементами, создаваемыми при выполнении проекта и вводимые в проект.

Так же как и в CDM, процессы в PJM разбиваются на последовательность задач, которые в процессе разработки проекта выполняют представители управленческих структур организации-разработчика (руководители проектов, управлений, отделов, подразделений и т. п.). **Необходимо понять, что задачи CDM — это для разработчиков, а задачи PJM — для управленцев.**

Для каждой задачи в рамках PJM определяется соответствующая категория (этап) ЖЦ, что однозначно определяет, когда должны выполняться те или иные задачи управления проектом и поддержки.

Категории ЖЦ PJM определяют второе измерение метода (см. рис. 2.9):

- ***планирование проекта*** — задачи этой категории включают определение проекта по следующим аспектам: масштаб, качество, время и стоимость. Задачи планирования проекта также определяют соответствующую организацию ресурсов и обязанностей для выполнения проекта;
- ***планирование этапа*** — к этой категории относятся задачи, связанные с корректировкой планов проекта и процедур для определенного этапа Oracle CDM;
- ***контроль над этапом*** — задачи этой категории выполняются одновременно с выполнением этапа Oracle CDM, и они связаны с функциями текущего контроля за проектом, руководства и отчетности в процессе выполнения этапа CDM;
- ***окончание этапа*** — задачи связаны с окончанием и гарантированным завершением этапа CDM;
- ***окончание проекта*** — результатом выполнения задач на этом этапе является удовлетворительное окончание проекта и решение всех нерешенных вопросов до закрытия проекта.

На рис. 2.10 представлена по аналогии с методологией CDM обзорная диаграмма для категории «Планирование этапа» жизненного цикла PJM.

Целями «Планирования этапа» являются:

- внесение корректировок в область применения проекта, которые отражали бы изменения в области применения, согласованные сторонами на предшествующем этапе;
- разработка подробного рабочего плана реализации этапа;
- получение подтверждения от заказчика выделения требуемых ресурсов и выполнения необходимых обязательств;
- определение изменений в инфраструктуре, которые нужно произвести для поддержки реализации этапа;
- обеспечение необходимых ресурсов для реализации этапа.

Критическими факторами успеха при проведении «Планирования этапа» являются:

- изменения в области определения в полном объеме рассмотрены и учтены в планах на этап;
- выявлены новые риски, определены ограничения и меры по сдерживанию рисков;
- стандарты и процедуры, необходимые для реализации этапа и получения качественных результатов, должны быть ясно и четко разъяснены персоналу;

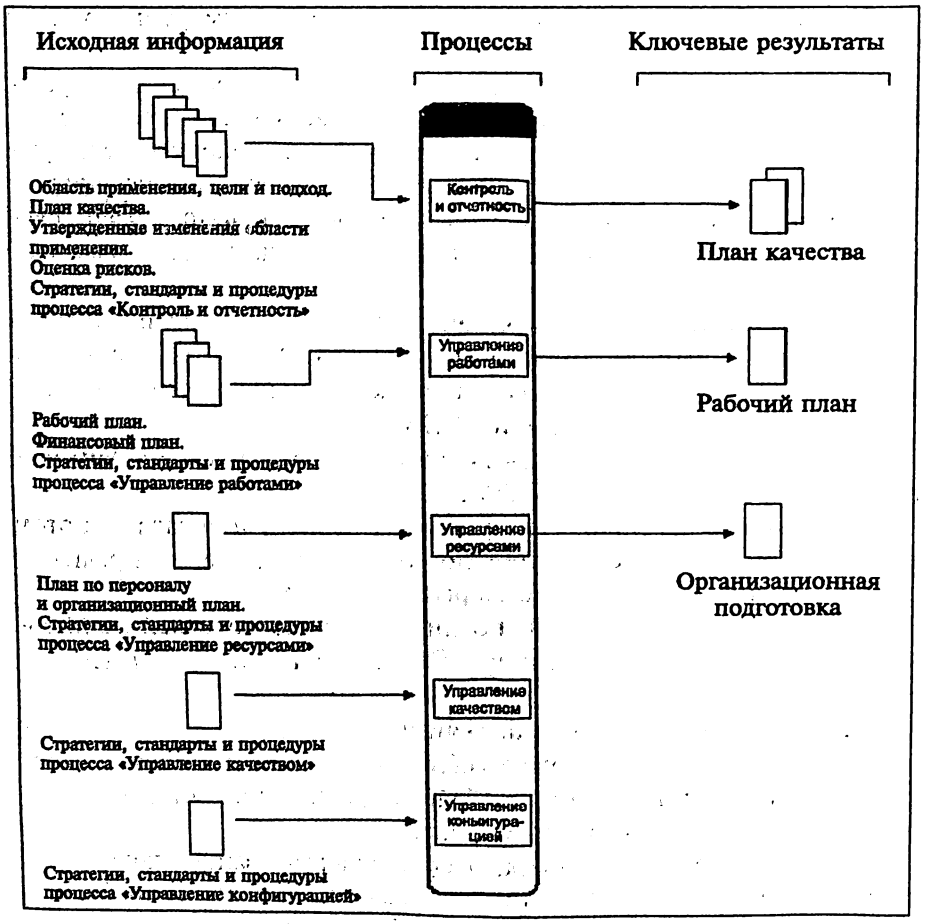


Рис. 2.10. Обзорная диаграмма «Планирование этапа»

- вовремя установленная инфраструктура;
- каждый член команды проекта осознает свою роль в выполнении рабочего плана и получении необходимых результатов.

Управление рисками: среди наиболее вероятных областей риска при «Планировании этапа» можно отметить следующие:

- выделенные время и ресурсы недостаточны для того, чтобы обеспечить этап нужным персоналом и обучить персонал, занятый на этапе;
- выполнение графика проекта требует ресурсов персонала, ранее не планировавшихся для использования в проекте;
- физические ресурсы, необходимые на этапе, не предоставляются вовремя для поддержки реализации задач этапа;

• оценка рисков проекта не корректируется до ревизии (сверки) планов проекта на этап;

• процедуры приемки результатов этапа не согласованы с заказчиком.

Для смягчения этих рисков можно использовать следующие пути:

• нужно как можно раньше, в ходе предшествующего этапа, установить состав критичного для данного этапа персонала, определить потребности в обучении и затем согласовать эти вопросы с управлением исполнителя. Бизнес-менеджеры исполнителя, обеспечивающие этап необходимым персоналом, должны предоставить поименный утвержденный список;

• элементы конфигурации, требующие длительного времени ввода, должны быть идентифицированы и заказаны заранее. Новое программно-аппаратное обеспечение должно быть протестировано;

• провести оценку рисков с управленческим персоналом заказчика и исполнителя, внести в рабочий и финансовый планы меры по сдерживанию рисков;

• провести совместно с заказчиком анализ планируемых проектных результатов, а также провести ревизию плана качества на предмет организации приемки результатов проекта.

Так же как и для методологии CDM, строится обзорное представление для каждого этапа с указанием перечня задач.

Как следует из рис. 2.9, методологии PJM и CDM жестко связаны между собой. Дело в том, что категории *планирование*, *контроль* и *окончание этапа*, объединенные в одну **категорию *управление проектом***, содержат процессы управления для каждого из этапов методологии CDM. То есть содержание процессов этих трех категорий изменяется при переходе от одного этапа CDM к другому. А категории планирования и окончания проекта выполняются только один раз — в начале и конце проекта и с этапами CDM не связаны.

2.6. Методология Microsoft Solutions Framework (MSF)

2.6.1. Три модели MSF

Корпорация Microsoft предлагает технологию Microsoft Solutions Framework (MSF) для управления работой команды разработчиков информационных систем уровня предприятия. Эта технология содер-

жит комплект моделей, методик и руководств по управлению проектами разработки ИС [http://msdn.microsoft.com].

Управление проектами в MSF рассматривается как управление изменениями и рисками. Существует множество причин, приводящих к неудачам: ошибки в прогнозах, постоянно изменяющиеся требования, нечетко поставленные проектные спецификации и т. п. Поэтому очень важно научиться управлять изменениями и рисками, что и обеспечивает MSF, который объединяет наиболее успешный опыт разработчиков Microsoft, опыт заказчиков и партнеров.

В состав MSF входят **три модели:** модель команды, модель проектной группы и модель процесса проектирования. Эти модели можно использовать как самостоятельно, так и в различных комбинациях, в зависимости от задач и потребностей, стоящих перед разработчиками.

Модель команды отвечает на вопрос: как должна быть построена и структурирована проектная группа, для того чтобы можно было оптимизировать процесс проектирования по срокам, качеству и затратам. Структура команды должна позволять отслеживать постоянно изменяющиеся требования в проекте и создавать качественный продукт в условиях ограничений на время и ресурсы. Чтобы быть эффективной, группа, как правило, должна быть небольшой по численности. Предлагаются следующие характеристики такой команды:

- каждый общается с каждым и каждый делает реальную работу;
- общие для всех членов группы цели и планы;
- каждый понимает как проблемы конечного пользователя, так и проблемы разработчика;
- каждый несет ответственность за свою работу, в том числе и перед группой.

Первое утверждение в этом списке — **«каждый общается с каждым»** — отрицает иерархический способ структурирования проектной группы. Для проектной группы MSF вопрос «кто кому подчинен?» не имеет смысла. Модель команды существенно отличается правилами формирования и составом от традиционных проектных групп. Как правило, в традиционную проектную группу не входят такие традиционные для проекта роли, как конечные пользователи и структуры, выполняющие контроль качества и обучение пользователей. Их отсутствие в составе группы повышает риски, удлиняет сроки проектирования, снижает качество продукта.

Таким образом, модель команды представляет собой самоуправляемые малые команды, члены которой полностью понимают и принимают свои роли, четко понимают цели и задачи проекта, несут пол-

ную ответственность перед заказчиком и членами команды. Недостатком такой модели команды является ограниченный контроль высшего руководства за внутренними взаимоотношениями и процессами в проектной группе.

Целью такой команды является создание качественного продукта, который:

- удовлетворяет ожиданиям и заказчика и конечных пользователей;
- удовлетворяет проектным ограничениям;
- соответствует реальным потребностям заказчика;
- эргономичен в использовании конечным пользователем;
- просто внедряется.

Модель проектной группы создана на основе характеристик модели команды, рассмотренных выше. Она определяет основные роли, закрепленные за членами проектной группы (рис. 2.11). В основу модели проектной группы положены следующие идеи.

Рис. 2.11. Модель проектной группы MSF

Для каждого члена группы определяется роль и зоны ответственности на весь период работы над проектом. То есть каждый сфокусирован на успехе проекта и настроен на работу в течение всего цикла проекта. Ключевым фактором успеха являются коммуникации между членами проектной группы. При этом реализуется параллельная работа всех участников группы над проектом, в состав которого в обязательном порядке включаются пользователи и обучающий персонал.

Модель проектной группы MSF практически не связана с организационной структурой, в одной проектной группе могут работать специалисты из различных подразделений. За каждым членом проектной группы закрепляется конкретная роль, для которой строится план работы, являющийся составной частью общего плана проекта. При

этом каждая роль в модели проектной группы имеет свою особую компетенцию.

Рассмотрим подробно каждую из ролей, указанных на рис. 2.11.

Менеджер проекта обеспечивает коммуникационный канал между заказчиком и проектной группой, управляет ожиданиями заказчика, разрабатывает и поддерживает бизнес-контекст проекта. Его задача — определить и обеспечить удовлетворение заказчика. Обычно для этой роли выбирают квалифицированного пользователя, сотрудника коммерческого отдела или другого представителя заказчика, если он понимает задачи и механизм бизнеса.

Менеджер программы управляет коммуникациями и взаимоотношениями в проектной группе, является фактическим координатором, разрабатывает функциональные спецификации и управляет ими, ведет график проекта и отчитывается по состоянию проекта, инициирует принятие критичных для хода проекта решений.

Разработчик принимает технические решения, которые могут быть реализованы и использованы, создает продукт, удовлетворяющий спецификациям и ожиданиям заказчика, консультирует другие роли в ходе проекта. Он участвует также в создании функциональных спецификаций, отслеживает и исправляет ошибки за приемлемое время.

В контексте конкретного проекта роль разработчика может подразумевать установку программного обеспечения, настройку продукта и др.

Следует иметь в виду, что разработка клиент-серверных систем требует детального знания высокоуровневых языков программирования, визуального программирования, сетевых технологий и проектирования баз данных. Поэтому в группу разработчиков обычно включают технических специалистов из всех этих областей, а в качестве руководителя группы разработчиков назначают специалиста, который знает и понимает ключевые моменты каждой из этих технических областей.

Тестер обеспечивает не только проверку кода, но и тестирование функциональных спецификаций, пользовательских интерфейсов, системы обеспечения производительности, планов внедрения и используемую терминологию. Эта роль обеспечивает выявление всех особенностей и задач еще до выпуска версии продукта, разрабатывает стратегию тестирования и планы тестирования для каждой базы проекта.

Инструктор отвечает за снижение затрат на дальнейшее сопровождение продукта и обеспечение максимальной эффективности поль-

зователя. Следует особо отметить, что речь идет **о производительности пользователя, а не системы**. Для обеспечения максимальной эффективности работы пользователей инструктор собирает статистику по производительности пользователей и создает решения для повышения производительности с использованием таких технологий, как мультимедиа, видео, HTML, встроенные системы подсказки, тренажеры и т. п. Инструктор принимает участие во всех обсуждениях пользовательского интерфейса и архитектуры продукта.

Логистик обеспечивает простое внедрение и развитие продукта, его задача — исключить ситуацию, когда внедрение продукта стоит дороже его разработки. Логистик должен обеспечить готовность заказчика к внедрению, чтобы вовремя были выполнены все подготовительные работы, создана необходимая инфраструктура.

Помимо перечисленных ролей можно выделить еще «роли поддержки». Это специалисты и эксперты в ключевых точках инфраструктуры. Они привлекаются к работе, когда это необходимо, но не принимают решений.

Если численность проектной группы меньше шести человек, то часть ролей может совмещать и выполнять один человек. При численности проектной группы больше шести человек одна роль может быть закреплена за несколькими специалистами, среди которых назначается лидер группы.

Параметры различных ролей в составе проектной группы можно формализовать следующим образом (табл. 2.2).

Модель процесса проектирования MSF (модель ЖЦ) является спиральной. Весь жизненный цикл проекта протекает в виде последовательности итераций выпуска версий. Выпуск нескольких версий продукта предпочтительнее, чем попытка сделать все сразу. Границы проекта могут уточняться тогда, когда это необходимо. Потребность в изменении границ проекта может вызываться изменением требований заказчика или рисками, которые трансформировались в проблемы.

MSF рекомендует вкладывать в первую версию продукта только базовую функциональность и затем наращивать ее в следующих версиях (рис. 2.12). Для малых проектов иногда бывает достаточно одной итерации.

Пересмотр функциональности, сетевых графиков работ, планов, спецификаций, требований и других проектных артефактов не прекращается до конца проекта и производится после каждой итерации. Такой подход позволяет планировать возможности для последующих

Таблица 2.2. **Параметры ролей проектной группы**

Роль	Ответственность	Приоритеты
Менеджер проекта	Определение проблемы. Продвижение продукта	Удовлетворенность заказчика
Менеджер программы	Управление спецификациями. Координация работ. Отслеживание состояния проекта	Разработка качественного продукта в срок и в рамках выделенного бюджета. Поиск и решение проблем
Разработчик	Проектирование функциональности. Создание продукта. Тестирование продукта	Надежный и полный продукт
Тестер	Определение стратегии тестирования. Проведение тестирования. Отслеживание результатов тестирования	Согласованный и надежный проект
Инструктор	Разработка документации. Ведение глоссария. Тестирование. Обучение пользователей	Продукт, который можно использовать и сопровождать
Логистик	Прогнозирование ситуации. Подготовка внедрения. Сопровождение продукта на этапе внедрения. Обеспечение адекватной инфраструктуры	Обеспечение гладкого внедрения и развития продукта

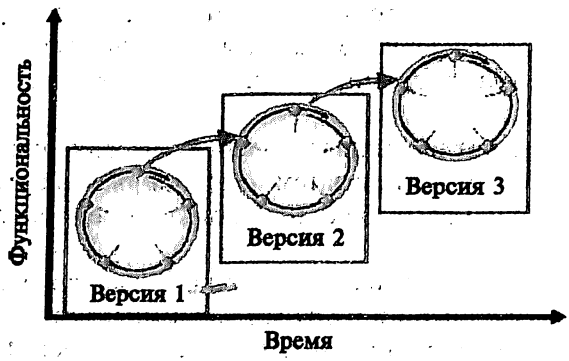


Рис. 2.12. Реализация спиральной модели ЖЦ в версиях ИС

версий, делать учет опыта предыдущих циклов, обеспечивая гибкость и устойчивость к переменам требований.

Таким образом, обеспечивается ведение «живой» документации, которая изменяется по мере эволюции проекта. В рамках одной итерации все документы (равно как и программный код) тоже развиваются итеративно. Так, все изменения в уже принятые документы в рамках одной итерации (например, утвержденное ТЗ) вносятся посредством компромиссов проектной группы и согласно технологии управления изменениями, и часто имеет смысл отложить эти изменения до следующей версии, чтобы не допускать разрастания рамок проекта и срыва сроков сдачи текущей версии.

Создание базовой версии всех проектных документов на самых ранних этапах дает возможность всем членам проектной группы осмыслить свои задачи и цели проекта, а также приступить к работе с минимальными задержками [42].

В рамках одной итерации жизненный цикл выпуска версии разбивается на пять фаз: выработка концепции, планирование, разработка, стабилизация (тестирование), внедрение (рис. 2.13). Однако существуют проекты, в которых может отсутствовать фаза внедрения или разработки (например, разработка игры, или адаптация 1С).

Рис. 2.13. Модель процесса проектирования с вехами

Каждая фаза цикла заканчивается **вехой (контрольной точкой)**. В этом принципиальное отличие MSF от традиционных моделей ЖЦ. Соответственно вехи будут иметь названия: концепция проекта утверждена, планы проекта утверждены, разработка завершена, готовность решения утверждена, внедрение завершено. Каждая веха определяет набор документов, которые должны быть созданы и согласованы с заказчиком. Они отражают текущую ситуацию и ее текущее понимание проектной группой и заказчиком.

Принципиальным моментом является то, что **каждая веха — это точка синхронизации**, когда команда заново пересматривает ожидания заказчика и риски. Это точки обсуждения и ревизии, а не точки фиксации принятых решений. Они позволяют проектной группе и заказчикам сравнить границы проекта и ожидания пользователей. В этой контрольной точке всплывают все противоречия и коллизии, возникшие за период фазы проекта. В решении о закрытии очередной фазы должны принимать участие ответственные представители всех ролей (менеджер проекта, менеджер программы, разработчик, тестер и т. д.).

Ориентация на вехи определяет, что небольшая, заранее определенная часть общего решения будет получена и оттестирована вовремя и риски планирования и качества будут известны заранее — следовательно, будет время на них среагировать.

Обычно работы на первой фазе проекта **«Выработка концепции»** (**«Анализ»**) проводятся до того, как сформированы требования. Все работы до заключения договора осуществляются обычно бесплатно для заказчика и длятся одну-две недели. Эта фаза необходима для того, чтобы команда разработчиков получила данные и оценила усилия, необходимые для создания функциональной спецификации, которая впоследствии будет использована при разработке.

Начинается она с формирования ядра проектной группы, если это первая итерация продукта. Основным результатом первой фазы «Анализ» является составление документа **«Образ и границы проекта»**. Это документ объемом пять—семь страниц, он составляется менеджером проекта (отвечает за правильное отображение потребностей заказчика) и менеджером программы (отвечает за соответствие задачи ожиданиям заказчика) и предназначен для четкого и ясного определения следующего:

- цели проекта, ожидания заказчика, база для исходной оценки рисков проекта;
- критерии для применения модели процесса разработки MSF, для совершенствования характеристик проекта, формирования команды и определения организаций, которые будут принимать участие в проекте;
- ожидаемые затраты, требуемые для формирования функциональной спецификации, которая должна быть создана на следующей фазе проекта.

И что характерно, на успех проекта больше влияет не подробность рассмотрения этих вопросов, а то, насколько всесторонне они были рассмотрены.

После анализа данных, моделирования различных процессов и планирования процесса разработки данный документ должен быть дополнен с учетом лучшего понимания потребностей заказчика, обнаруженных технических решений или рисков, достигнутого компромисса между объемами и сроками работ. Изменения, вносимые в документ «Образ и границы проекта» по мере его разработки, должны становиться все менее и менее значительными, так как они все больше начинают влиять на риски, связанные с временем исполнения проекта.

После утверждения функциональной спецификации изменения в документе «Образ и границы проекта» **запрещаются**, так как он является первичным, основополагающим документом для планирования проекта и управления процессом разработки. Вехой этой фазы будет событие **«Концепция проекта утверждена»**. К этому моменту у заказчика и проектной группы уже сформированы устойчивые представления о задачах, функциональности и ограничениях проекта, ядро проектной группы сформировано, черновой вариант концепции проекта составлен.

Фаза **«Планирование»** включает в себя подготовку проектной группой функциональной спецификации, разработку дизайнов, подготовку рабочих планов, оценку проектных затрат и сроков разработки различных составляющих проекта.

Результаты проектирования документируются в функциональную спецификацию, которая детально описывает вид и поведение всех составляющих решения. Функциональные спецификации описывают, какими возможностями должен обладать результирующий продукт, и являются одним из основных результатов этой фазы. Этот документ представляет собой как бы договор между проектной группой и заказчиком. За составление функциональных спецификаций отвечает менеджер программы.

На основании спецификации команда разработчиков производит оценивание работ, достигается четкое соглашение с заказчиком о том, что должно быть сделано, для каждой роли создается детальный план. Примерами планов могут быть: план внедрения, план тестирования, план обучения, план мер безопасности и т. п. Затем эти планы объединяются в сводный план проекта и сводный сетевой график работ. На этом же этапе создается также план управления рисками.

Завершается фаза вехой **«Планы проекта утверждены»**, на момент которой уже существуют документы: функциональная спецификация, план управления рисками, сводный план и сводный календар-

ный график работ, который определяет, когда будет готово то, что описано в функциональных спецификациях.

Фаза **«Разработка»** предполагает активное и деятельное участие всех ролей в соответствии с их обязанностями в тестировании, выявлении дефектов, анализе удовлетворения нужд заказчика и других задачах. К моменту завершения этой фазы разработка всех компонент завершена и решение готово к комплексному тестированию.

Пользователи могут апробировать продукт и определить, все ли их потребности нашли в нем отражение. Кроме того, это — первое тестирование процедур внедрения и поддержки продукта. На этой фазе разрабатывается стратегия внесения изменений в работающий продукт. Она будет поддерживать и выпуск последующих версий.

Обычно разработка продолжается и на фазе стабилизации, и даже на фазе внедрения иногда приходится что-то править.

Фаза завершается вехой **«Разработка завершена»**. Эта веха достигается тогда, когда получена первая бета-версия полного продукта, содержащая полный код, который должен быть тщательно оттестирован.

На фазе **«Стабилизация»** фокус смещается в область тестирования и документирования решения, а также создания пилотного внедрения. Результатами этой фазы являются: окончательный продукт, документация выпуска, материалы поддержки решения, результаты тестирования, проектная документация и анализ пройденной фазы.

Фаза завершается вехой **«Готовность решения утверждена»**, что означает работоспособность продукта и его передачу группам поддержки и сопровождения. Эффективность продукта проверяется во время бета-тестирования. К моменту выпуска версии продукта этими группами уже накоплен необходимый опыт по сопровождению, собран материал об имеющихся особенностях и типовых трудностях, с которыми сталкиваются пользователи при работе с продуктом.

Не менее важно выполнение работ по продвижению продукта, информированию о его возможностях, особенностях и «цене» заказчика и конечных пользователей, для того чтобы создать положительное восприятие нового продукта или следующей версии.

На фазе **«Внедрение»** выполняется установка и отладка системы в реальных условиях эксплуатации, передача системы персоналу поддержки и сопровождения, получение окончательного одобрения результатов проекта со стороны заказчика. Достижение вехи **«Внедрение завершено»** означает получение следующих результатов:

- работающие процедуры и процессы;
- базы знаний, отчеты, журналы протоколов;

- версии проектных документов, массивы данных и программный код, разработанные во время проекта;
- отчет о завершении проекта;
- окончательные версии всех проектных документов;
- показатели удовлетворенности заказчика и потребителей.

Следует иметь в виду, что после внедрения первой версии продукта может последовать следующая итерация жизненного цикла. На любой итерации в первую очередь следует реализовывать рискованные нововведения, минимизируя за счет этого влияние риска на весь проект.

Управление рисками. Планирование с учетом рисков — это прием, когда компоненты проекта, связанные с наибольшим риском, разрабатываются первыми. Особенно важно планировать с учетом рисков в проекте, связанном с разработкой программного обеспечения или созданием инфраструктур.

MSF поощряет разработчиков создавать макеты и прототипы как можно раньше. Это снижает риск получения неполноценного продукта, нереального графика или краха всего проекта.

Разрабатывается план по предотвращению наиболее вероятных и опасных ситуаций и, на всякий случай, план «ликвидации катастрофы», т. е. план возврата к последнему работоспособному варианту без потери данных.

Обязательно пересматривается проектный план для включения в него «смягчающего» риски плана. Как правило, это потребует ресурсов и времени, а также выполнения дополнительных задач.

Если компоненты, связанные с высокой степенью риска, потребуют больше времени, чем предполагалось, то планирование с учетом рисков предоставит дополнительное время для реагирования.

Следует также учитывать, что одним из ключевых факторов успешного бизнес-решения является вовлечение пользователей в процесс проектирования на всех его фазах. «Лучший способ приобрести сторонников — это сделать их причастными».

К сожалению, вовлечением пользователей в процесс проектирования сложно управлять. Основные соображения, почему организации и команды разработчиков не привлекают пользователей к работам над проектом, следующие:

- пользователи плохо формулируют то, что им нужно;
- как правило, они сопротивляются изменениям. Им не нужны новые технологии, и, соответственно, они не хотят работать над их проектированием;

• пользователи придерживаются различных мнений о том, какие возможности должна включать в себя новая система, и имеют широкий спектр предпочтений цветов и расположения элементов на экране;

• пользователи просят реализовать ту или иную функцию, но, когда это сделано, они хотят чего-нибудь другого.

Выпуск версий продукта. Выпуск нескольких версий продукта означает, что не вся функциональность реализуется сразу, процесс разработки итерационен и проектная группа принимает решения о включении той или иной функциональности по мере необходимости, ориентируясь на версии и даты выпуска.

При последовательном выпуске версий проектная группа может начать с реализации функций, наиболее критичных для заказчика, и планомерно получать от пользователей обратную связь, которая потребуется при разработке следующих версий.

Первый выпуск продукта содержит базовый набор функций, а последующие — включают все больше и больше дополнительных функций, вплоть до финального продукта. Последующие версии позволяют проектной группе в очередной раз проверить или изменить образ продукта (например, если изменились требования бизнеса).

Механизм выпуска последовательных версий позволяет управлять взаимоотношениями с заказчиком и достигать оптимального баланса между конкурирующими целями. Когда этот баланс найден, удовлетворенность заказчика и качество продукта являются максимальными.

Особо следует отметить тот факт, что выпуск последовательных версий продукта требует меньших маркетинговых затрат, чем выпуск совершенно нового продукта. Одну из версий продукта проще позиционировать и продвигать на рынок.

Полученная версия может быть названа качественным продуктом, если она:

• удовлетворяет ожиданиям заказчика и пользователей;
• удовлетворяет проектным ограничениям;
• соответствует реальным потребностям;
• обеспечивает его гладкое внедрение.

2.6.2. Технология и инструменты разработки решений

Решение задачи организации **процесса разработки** в среде MSF нуждается в серьезной технологической поддержке. Технологическое обеспечение должно, во-первых, поддерживать весь спектр необходи-

мых операций, во-вторых, быть достаточно интеллектуальным, чтобы допускать автоматизацию типовых процедур, и, в-третьих, быть взаимосогласованным и расширяемым.

Microsoft Repository представляет собой механизм хранения и совместного использования объектов различными приложениями и средствами разработки программного обеспечения (ПО). Примерами объектов, которые можно хранить в Microsoft Repository, являются надстройки (Add-Ins) Visual Basic, специализированные, повторно используемые компоненты (ActiveX-элементы управления и серверы), текстовые документы, готовые проекты и т. д.

Visual Basic 6.0 является клиентом этого репозитария. В его состав входит ядро хранилища Microsoft Repository (база данных в формате Microsoft Access), содержащее информационную модель инструментария (Tool Information Model, TIM), которая описывает проект Visual Basic. Кроме того, Visual Basic содержит надстройку (Add-Ins) для работы с Microsoft Repository и средство просмотра его данных. Кроме того, услугами Microsoft Repository пользуются **Visual Modeler** и **Visual Component Manager**, которые являются компонентами Visual Basic 6.0.

Благодаря механизму Microsoft Repository появились или были существенно улучшены:

- повторное использование средства для классификации и поиска необходимых разработчику компонентов, кода и сервисов;
- отслеживание зависимостей: средства создания отношений между объектами и выяснения наличия таких отношений;
- управление ресурсами: библиотека доступных ресурсов для работы с ними, включая сервисы и компоненты;
- поддержка коллективной разработки: средства поддержки параллельного проектирования и разработки нескольких версий и конфигураций ПО.

Информационная модель инструментария (Tool Information Model, TIM) — это описание систем или процессов, опубликованных в репозитарии. В нем может находиться любое число таких моделей, каждая с описанием своей системы или процесса. Модель определяет объекты, интерфейсы, отношения и связи, образующие систему. С другой стороны, TIM можно рассматривать и как своего рода шаблон для систем конкретного типа. Стандартизация моделей позволяет разным средам и средствам разработки взаимодействовать с данными в хранилище.

Программа просмотра Repository Browser, которая поставляется в составе Visual Basic, позволяет просматривать содержимое репозита-

рия. Большинство средств разработки чаще всего пользуется моделями конкретного репозитария, относящегося к конкретной системе или процессу, тогда как программу просмотра хранилища можно рассматривать как инструмент общего назначения для просмотра моделей и хранилищ. Это означает, что с его помощью вы сможете просматривать любые модели в любых репозиториях.

Интерфейс программы просмотра аналогичен интерфейсу проводника Windows, в левой панели отображаются данные в виде дерева, а в правой — их детализированное представление в виде списка. Панель инструментов открывает доступ к функциям программы просмотра.

Визуальный диспетчер компонентов (Visual Component Manager) — это составная часть Visual Basic Professional Edition и Enterprise Edition. Ее основное назначение — обеспечить разработчику возможность сохранять программные компоненты в репозитарии и тем самым упростить повторное использование кода. Таким образом, появляется возможность повысить производительность труда коллектива разработчиков за счет классификации и публикации компонентов и обеспечения доступа к ним.

Работая с диспетчером, программист может найти необходимый фрагмент кода и воспользоваться им, загрузить шаблон проекта, запустить мастер или добавить в проект управляющий элемент ActiveX. Кроме того, репозитарий может играть роль центра хранения документации, включая принятые стандарты программирования, функциональные спецификации или архитектурные диаграммы.

Разработка модели сложной программной системы непосредственно перед ее реализацией является неотъемлемой частью всего проекта, подобно тому, как чертеж является основой для построения большого здания. Хорошая модель является основой для гладкого взаимодействия в команде разработчиков, гарантирует общий успех проекта.

Существует большое число факторов, влияющих на общий успех разработки, но наличие строгого стандарта на язык моделирования является первостепенным фактором. Этим объясняется огромный интерес к промышленному объектно-ориентированному стандарту языка моделирования, которым является **унифицированный язык моделирования** — UML. В настоящее время имеется целый ряд инструментальных средств, производители которых заявляют о поддержке UML. Можно выделить такие инструментальные средства, как Rational Rose, Select Enterprise, Platinum и Visual Modeler.

Visual Modeler корпорации Microsoft позволяет разработчикам создавать повторно используемые компоненты, взаимодействующие со службами пользовательского интерфейса и схемами баз данных. В совокупности с Visual Basic и другими инструментами Visual Studio© пакет Visual Modeler поддерживает весь жизненный цикл разработки и сопровождения сложных клиент-серверных приложений, начиная от концептуального проектирования и заканчивая модификацией готовых приложений.

2.7. Методология экстремального программирования

2.7.1. Общие сведения об экстремальных методологиях

К настоящему времени разработано несколько десятков методологий и подходов к организации процессов создания программного продукта или ИС. Условно их можно разделить на тяжелые (CDM, MSF, RUP компании Rational и др.) и легкие, которые чаще называются экстремальными [7].

Тяжелые методологии дают наибольший эффект в крупных компаниях, занятых промышленным выпуском ИС. Такие подходы дают хорошие результаты, но процесс внедрения растягивается на несколько лет. Классическая тяжелая методология состоит из шести этапов: анализ требований, проектирование, кодирование, сборка, тестирование, внедрение и сопровождение. На ее основе разработано несколько популярных моделей ЖЦ: каскадная, последовательная, спиралевидная и т. д.

Например, **каскадная модель** предусматривает последовательный переход от фазы к фазе, поочередно выполняются определение системных требований, анализ требований к продукту, предварительное и детальное проектирование, кодирование, тестирование, объединение модулей, проверка работы всей системы, комплексное и системное тестирование, внедрение и сопровождение.

Подобная модель предполагает разработку большого количества документов после завершения каждого этапа, которые после окончания разработки ИС практически не используются. Другим серьезным недостатком является недостаточная гибкость к изменяющимся условиям окружающей среды и требованиям заказчика.

Каскадная модель становится неприемлемой, если требования за время разработки системы меняются настолько, что результат разра-

ботки становится непригодным для применения. В ряде случаев заказчик не в состоянии адекватно сформулировать свои требования, не видя работающего прототипа системы.

Эти недостатки смягчаются в таких моделях, как **спиралевидные** или **быстрого прототипирования**, но многократное повторение цикла «анализ—проектирование—кодирование—тестирование» занимает достаточно много времени, чтобы оперативно учесть все изменения требований. В них всегда присутствует главный недостаток — экспоненциальный рост времени на устранение выполняемых на последних этапах ошибок, особенно если они связаны с неверно спроектированной архитектурой системы или с плохо сформулированными требованиями на начальных этапах.

На фоне этих моделей ЖЦ тяжелой методологии выгодно смотрится **экстремальная методология**, основная идея которой заключается в нетрадиционном процессе создания приложений: вместо отдельных фаз планирования, анализа и проектирования продукта с расчетом на долгосрочную перспективу все эти работы выполняются постепенно в ходе разработки. Работа над проектом сводится к частому выпуску отдельных версий продукта с постепенно нарастающим набором функций. Для каждой версии определяется необходимый набор функций, и уже в процессе работы над версией осуществляются и планирование, и проектирование, и реализация, и тестирование.

Легкие методологии предназначены для использования небольшими динамическими компаниями, работающими в условиях очень сжатых сроков, быстро меняющихся требований и необходимости обеспечения достаточно высокого качества. Внедрение таких методологий не требует серьезных инвестиций и перестройки структуры фирмы — сотрудникам достаточно договориться о новом способе работы.

Эта методология особенно актуальна сейчас, когда успешно работают множество виртуальных и оффшорных команд, когда резко возросла конкуренция за выгодные заказы, когда в ходе выполнения работ возможны существенные изменения требований заказчиком и т. п.

Экстремальные методологии (ЭМ) представляют собой наборы рекомендаций, которые по отдельности выглядят противоречащими здравому смыслу и классическим схемам — отсюда и пошло название — экстремальные методологии. Но эти рекомендации, собранные в правильной комбинации, превращаются в эффективно работающий инструмент.

В первую очередь ЭМ нацелены на скорейшее получение конкретного результата, удовлетворение главных требований пользователя и более медленную доработку системы в соответствии с дополнительными требованиями. При этом новые требования компонуются в виде набора и реализуются в течение одной итерации цикла, продолжительность которой обычно не более недели. Направление развития системы рассматривается, прежде всего, с точки зрения потенциальных рисков: **сложные и сомнительные функциональные возможности откладываются на более позднее выполнение.**

Несомненным достоинством легких методологий является значительное уменьшение документации. Практически вся документация включается в код и представляет собой списки правил, принципов и уточнений заказчика. В принципе суть легких методологий заключается в постоянном стремлении минимизировать реальные объемы работ и делать это везде, где только возможно.

Однако работать ЭМ будут только при наличии **высокопрофессиональной команды**, в которой отлажен механизм взаимодействия, а каждый сотрудник способен переключаться на разные виды работ.

Рассмотрим более подробно одну из наиболее популярных экстремальных методологий — **экстремальное программирование** (ЭП, XP, eXtreme Programming).

2.7.2. Экстремальное программирование (XP)

eXtreme Programming (**XP**) — это еще одна гибкая методика разработки программного обеспечения, позволяющая команде (разработчикам, пользователям и руководителям проекта) быстро адаптировать продукт к изменяющимся требованиям. Методика XP имеет следующие достоинства:

- пользователи быстро получают первую версию продукта и могут приступать к ее апробации на ранних стадиях проекта, что позволяет тестировать продукт в реальном рабочем окружении;
- разработчики в первую очередь реализуют наиболее важные компоненты продукта, указанные заказчиками;
- важнейшие функции и компоненты продукта проходят наиболее тщательное тестирование;
- конечный продукт оптимально удовлетворяет требованиям заказчика.

Для создания XP его автор Kent Beck исходил из следующих соображений [7].

Если в классических методиках считается, что тестирование — это хорошо, то тестировать следует все, что можно, и при этом непрерывно.

Если полезно проектирование системы, то анализ и синтез структуры продукта будут выполняться ежедневно.

Если важна простота при построении системы, то целью разработки ставится скорейшее создание реальной работающей системы, реализующей минимум самой главной функциональности.

Если эффективны комплексные сборки, позволяющие выявить проблемы с нестыковкой нескольких модулей, и итерационное тестирование, то они будут выполняться несколько раз в день.

Наконец, если хорошо работают короткие итерационные циклы «анализ—проектирование—разработка—тестирование», то их продолжительность будет составлять не недели и месяцы, а часы или минуты.

Главная идея XP заключается в том, что после того, как сформулирована задача, программист готовит **тестовый пример**, проверяющий базовую функциональность будущей программы. Затем эта функциональность реализуется в виде программы с ориентацией на подготовленный тест. Если проверка с его помощью выполнена успешно, начинается следующая итерация — готовится новый тест, уточняющий и расширяющий требования предыдущего, под новые требования дописывается программный код и т. д. Главное в таком цикле разработки — стараться плавно расширять требования тестов, чтобы удавалось быстро создать новое, правильно работающее приложение.

Любой новый код интегрируется в существующее приложение не позднее чем через несколько часов после создания, после чего приложение собирается в единое целое, прогоняются все тесты, и если хотя бы один из них не выполняется корректно, то внесенные изменения отменяются.

При таком подходе гарантируется, что ненужные или излишние требования, вошедшие в первоначальный проект (например, на уровне описания классов), не будут реализованы. Таким образом, система разрабатывается максимально короткими, частыми и гибкими итерационными циклами, которые выполняются на фоне непрерывного улучшения качества этих циклов, при непрерывной связи с заказчиком и избавлении от избыточной документации.

Рассмотрим теперь более подробно принципы XP, которые определяют его достоинства.

Разработка через тестирование. На протяжении всего проекта усилия сосредоточены на проверке создаваемой системы. Программисты на основе требований заказчика предварительно (и в этом основной смысл) создают тесты, а затем пишут программное обеспечение, которое должно соответствовать этим тестам. Если разработанные таким образом программы будут выполняться на соответствующих тестах, то дополнительной отладки уже не потребуется, гарантируется соответствие продукта заказанной функциональности, и разработчикам будет заранее известно, как будет выглядеть ИС (схема взаимодействия с пользователем, реагирование на различные типы входных данных, структура пользовательского интерфейса и т. д.).

Парное программирование. Этот важный принцип XP заключается в организации одновременной работы двух программистов над одним кодом: один пишет код, другой контролирует его работу или работает над архитектурой программы, периодически меняясь местами. Таким образом обеспечивается контроль над чужим исходным текстом и получение обратной связи в реальном времени. При одинаковом уровне квалификации программисты будут постоянно поправлять друг друга, перехватывать инициативу, не давать друг другу отвлекаться в сторону наращивания функциональности. При разном уровне подготовки программистов менее опытный программист намного быстрее освоится при наблюдении за работой более опытного коллеги.

Простота проектирования. Создаваемая программа реализует только текущие требования заказчика, в нее не закладываются возможности на будущее. При этом требуемая заказчиком функциональность должна быть очень просто реализована с помощью имеющегося набора средств, повторного использования готового кода (рефакторинга).

Функциональные требования к проекту (каждое из них называется story) записываются на обычных липких листочках. Каждая story (история) записывается пятью словами с помощью активных глаголов. Если заказчик не укладывается в этот лимит, то полагают, что он сам не знает, чего хочет от конкретной возможности системы. При этом история должна быть сформулирована так, чтобы по ее описанию было легко построить тест.

Планирование релиза (версии). Заказчик определяет, какие задачи наиболее важны или имеют высокий приоритет. С учетом этого выбора задач заказчик и разработчик вместе выбирают карточки с историями (story), которые будут составлять следующий релиз. Отобран-

ные карточки сортируются по приоритету, выбирается наиболее приоритетная, уточняются задачи, пишутся все необходимые тесты и начинается фаза кодирования. Затем на готовом коде запускаются все написанные тесты, и если они проходят успешно, то выбирается следующая по приоритету карточка и процесс повторяется.

Такой подход, основанный на выборе приоритетов, гарантирует, что разработчик всегда занимается самой приоритетной задачей по проекту, что облегчает быстрое завершение итерации. Для заказчика такое планирование также удобно: он практически безболезненно для хода проекта может изменять требования к системе (что-то улучшить, что-то поменять и что-то вообще выбросить из системы). Итерацией в XP называется фаза разработки, в результате которой создается рабочая версия проекта (релиза).

Небольшие релизы. Разработчики выпускают работоспособную версию очень рано и расширяют ее возможности очень часто. Чем короче итерации, тем чаще заказчик может оценить текущее состояние продукта, опробовать его в рабочих условиях, протестировать готовую функциональность.

Если команда разработчиков делает продукт долго и без связи с заказчиком, то такая ситуация считается большим риском — наверняка делается то, что не надо, или не так, как надо. Рекомендуется выпуск релизов не реже раза в 1—2 недели.

Непрерывная интеграция. Сборка продукта выполняется много раз в день. Такой подход снимает множество проблем, связанных со стыковкой модулей, разработанных разными программистами, а также позволяет избежать ситуации, когда новая функциональность оказывается несовместимой с остальным проектом. Для сведения к минимуму риска нежелательных последствий интеграции рекомендуется выполнять ее на отдельном компьютере парой программистов.

При этом обязательны регулярные совещания, на которых принимаются решения о том, что именно следует реализовывать дальше и как распределять работу в группе, а также желательно наличие представителя заказчика в составе группы.

Непротиворечивая совокупность принципов XP способна ввести в процесс разработки интеллектуальный резонанс, заметно повысив качество продукта и приблизив время его выпуска. Основное достоинство XP — прогнозируемость и сведение к минимуму затрат на разработку; предоставление заказчику продукта, который он желает получить на момент выпуска; общение и обучение разработчиков в процессе разработки проекта.

В XP четко описаны роли, которые могут быть совмещены в одном человеке. Со стороны заказчика рекомендуют **три роли:** составитель историй, приемщик и большой босс.

Составитель историй должен быть специалистом предметной области, способным доступно изложить и описать требования к разрабатываемой системе. Этот человек или группа людей ответственны за написание историй (story) пользователя и консультации с программистами.

Приемщик контролирует правильность функционирования системы, хорошо разбирается в предметной области, в его обязанности входит также написание приемочных тестов.

Большой босс следит за работой всех звеньев от разработчиков до конечных пользователей, контролирует внедрение системы и сопутствующие организационные моменты. Может быть инвестором проекта.

Со стороны разработчика вводятся **четыре роли:** программист, инструктор, наблюдатель и дипломат.

Программист занимается кодированием и проектированием на низком уровне, достаточно компетентен для решения текущих задач разработки.

Инструктор — опытный разработчик, хорошо владеющий всем процессом разработки и его методиками. Несет ответственность за обучение команды, контролирует правильность выполнения методик.

Наблюдатель — член команды разработчиков, пользующийся доверием всей группы, который следит за прогрессом разработки. Он сравнивает предварительные оценки трудозатрат и реально потраченные трудозатраты, выводя количественные показатели работы команды.

Дипломат — коммуникабельная личность, инициирующая общение между членами команд и обеспечивающая передачу опыта внутри команды. Дипломат регулирует и упрощает общение между заказчиками и разработчиками.

Можно также отметить, что XP позволяет небольшим командам разработчиков выжить в условиях современного бизнеса со всеми его рисками и проблемами. Но перед обращением к **eXtreme Programming** надо всегда объективно определить, будет ли сам проект экстремальным, например сроки сжаты, а требования к системе не определены, или же можно обойтись классическим подходом.

Контрольные вопросы

1. Каким требованиям должна удовлетворять технология проектирования, разработки, сопровождения ИС?
2. Какие категории стандартов используются в технологии проектирования ИС?
3. Сравнить модели ЖЦ информационных систем: каскадную, спиральную и быстрого прототипирования.
4. Принципы структурного анализа.
5. Состав двумерной модели ЖЦ Oracle CDM.
6. Выполнить сравнительный анализ моделей ЖЦ Oracle CDM.
7. Проанализировать обзорную диаграмму этапа «Определение требований».
8. Назначение обзорного представления этапа «Определение требований».
9. Назначение метода управления проектом Oracle PJM.
10. Содержание трех моделей методологии MSF.
11. Особенности модели ЖЦ в методологии MSF.
12. Принципы экстремального программирования.

Глава 3
CASE-ТЕХНОЛОГИИ И ТЕХНОЛОГИЧЕСКАЯ ЗРЕЛОСТЬ IT-ПРЕДПРИЯТИЙ

3.1. CASE-технологии и CASE-средства: характеристика и классификация

Современные информационные технологии создают фундамент для постоянного роста сложности и масштабности информационных систем в различных предметных областях. Для успешной реализации проекта предметная область должна быть адекватно описана с помощью полных и непротиворечивых функциональных и информационных моделей будущей ИС. Это логически сложная, трудоемкая и длительная по времени работа, требующая высокой квалификации участвующих в ней специалистов: системных аналитиков, проектировщиков, представителей заказчика.

Во многом эти трудности устраняются благодаря появлению программно-технологических средств специального класса — CASE-средств, реализующих CASE-технологию создания и сопровождения ИС. Значение термина CASE (**Computer Aided *Software* Engineering**) первоначально ограничивался вопросами автоматизации разработки только лишь программного обеспечения. В настоящее время этот термин используется в более широком смысле, охватывая процесс разработки сложных ИС в целом. Теперь под термином **CASE** (**Computer Aided *System* Engineering**) понимаются программные средства, поддерживающие процессы создания и сопровождения ИС, включая анализ и формулировку требований, проектирование прикладного ПО и баз данных, генерацию кода, тестирование, документирование, обеспечение качества, конфигурационное управление и управление проектом, а также другие процессы.

CASE-средства вместе с системным программным обеспечением и техническими средствами образуют полную среду разработки ИС.

Появлению CASE-средств и CASE-технологии предшествовали исследования в области методологии программирования. Программирование обрело черты системного подхода с разработкой и внедрением языков высокого уровня, методов структурного и модульного программирования, языков проектирования и средств их поддержки, формальных и неформальных языков описаний системных требований и спецификаций и т. д. Кроме того, появлению CASE-технологии способствовали и такие факторы, как:

- подготовка аналитиков и программистов, восприимчивых к концепциям модульного и структурного программирования;
- широкое внедрение и постоянный рост производительности компьютеров, позволившие использовать эффективные графические средства и автоматизировать большинство этапов проектирования;
- внедрение сетевой технологии, предоставившей возможность объединения усилий отдельных исполнителей в единый процесс проектирования путем использования разделяемой базы данных, содержащей необходимую информацию о проекте.

Таким образом, **CASE-технология** представляет собой методологию проектирования ИС, а также набор инструментальных средств, позволяющих в наглядной форме моделировать предметную область, анализировать эту модель на всех этапах разработки и сопровождения ИС и разрабатывать приложения в соответствии с информационными потребностями пользователей.

CASE-технологии имеют ряд характерных особенностей:

- обладают графическими средствами для проектирования и документирования модели информационной системы;
- имеют организованное специальным образом хранилище данных, содержащее информацию о версиях проекта и его отдельных компонентах;
- расширяют возможности для разработки систем за счет интеграции нескольких компонент CASE-технологий.

Современные CASE-средства поддерживают также множество технологий моделирования информационных систем, начиная от простых методов анализа и регламентации и заканчивая инструментами полной автоматизации процессов всего жизненного цикла программного обеспечения (http://www.realcoding.net).

CASE-технологии можно классифицировать по функциональной направленности:

- средства моделирования предметной области;
- средства анализа и проектирования;

- технологии проектирования схем баз данных;
- средства разработки приложений;
- технологии реинжиниринга программного кода и схем баз данных.

Современные CASE-средства в основном базируются на методологиях **структурного** (функционально-модульного) или **объектно-ориентированного** анализа и проектирования, использующих спецификации в виде диаграмм или текстов для описания внешних требований, связей между моделями системы, динамики поведения системы и архитектуры программных средств.

Функционально-модульный подход основан на принципе алгоритмической декомпозиции с выделением функциональных элементов и установлением строгого порядка выполняемых действий.

Объектно-ориентированный подход основан на объектной декомпозиции с описанием поведения системы в терминах взаимодействия объектов.

В большинстве CASE-систем применяются методологии структурного анализа и проектирования, которые основаны на наглядных диаграммах. При этом для описания модели проектируемой системы используются графы, диаграммы, таблицы и схемы. Такие методологии обеспечивают строгое и наглядное описание проектируемой системы. Оно начинается с общего обзора системы, затем детализируется, приобретая иерархическую структуру.

Структурный системный анализ является основой методологий, положенных в основу большинства CASE-систем, которые появились во второй половине 80-х годов на рынке и стали быстро завоевывать популярность. Основные положения этих методологий можно сформулировать следующим образом:

- основополагающей концепцией является построение логической (не физической) модели системы при помощи графических методов, которые дают возможность пользователям, аналитикам и проектировщикам получить ясную и общую картину системы, выявить, как сочетаются между собой компоненты системы и как будут удовлетворены потребности пользователя;
- эта методология предполагает построение системы «сверху вниз» за счет последовательной детализации: вначале получают кон-

• хорошая разработка включает итерацию, т. е. следует быть готовыми уточнить логическую модель и физический проект с учетом информации, получаемой при использовании первой версии модели или проекта.

В целом функциональные методики лучше дают представление о существующих функциях в организации, о методах их реализации. При этом чем выше степень детализации исследуемого процесса, тем лучше они позволяют описать систему.

Главный недостаток функциональных моделей заключается в том, что процессы и данные существуют отдельно друг от друга: помимо функциональной декомпозиции существует структура данных, находящаяся на втором плане. Кроме того, не ясны условия выполнения процессов обработки информации, которые динамически могут изменяться.

Перечисленные недостатки функциональных моделей отсутствуют в **объектно-ориентированных моделях**, в которых главным структурообразующим компонентом выступает **класс объектов** с набором функций, которые могут обращаться к атрибутам этого класса. Для объектно-ориентированного подхода разработаны графические методы моделирования предметной области, обобщенные в языке унифицированного моделирования UML.

Однако по наглядности представления модели пользователю-заказчику объектно-ориентированные модели явно уступают функциональным моделям. К недостаткам объектно-ориентированного подхода относятся высокие начальные затраты. Этот подход не дает немедленной отдачи. Эффект от его применения сказывается после разработки двух-трех проектов и накопления повторно используемых компонентов.

В связи с наличием двух подходов к проектированию программного обеспечения существуют CASE-технологии, ориентированные на структурный подход, объектно-ориентированный подход, а также комбинированные. Идея синтетической методики заключается в последовательном применении функционального и объектного подхода с учетом возможности реинжиниринга существующей ситуации.

Если посмотреть с позиций пользователя, то CASE-средства позволяют ему с помощью простых средств визуального моделирования вводить сведения о предметной области в единственное хранилище (репозитарий) для всех участников проекта. Затем с помощью процедур автоматической генерации введенная в виде моделей информация преобразуется в таблицы базы данных и программные модули, го-

товые к выполнению на компьютере. Таким образом, средства визуального проектирования позволяют разработчикам создавать ИС, используя исключительно визуальные объекты, а затем автоматически генерировать программный код по этим построениям с помощью средств языков программирования четвертого поколения (4GL).

Используемые в CASE-средствах модели (функциональные, информационные, событийные и другие) представляют собой визуальную составляющую CASE-инструментов и являются общим языком для всех участников проекта. Если на каком-либо этапе необходимо внести изменение в проект, то достаточно внести изменение в соответствующие модели и затем в автоматическом режиме произвести перегенерацию системы.

Таким образом, CASE-технология позволяет отделить проектирование системы от собственно программирования и отладки. Системные аналитики и проектировщики занимаются проектированием на более высоком уровне, не отвлекаясь на мелкие детали. Это дает возможность избежать множества ошибок на ранних стадиях проекта и получить более совершенные программные продукты. Ошибки, допущенные на ранних этапах анализа и проектирования и не обнаруженные в процессе тестирования, выливаются в итоге в трудноразрешимые проблемы, которые способны привести к неудаче всего проекта.

Другим достоинством CASE-средств является их способность быстро строить прототип (или макет) ИС — упрощенную черновую версию системы, которая дает возможность заказчику увидеть приложение в действии, поэкспериментировать с ней и выдать разработчикам соответствующие коррективы.

При этом CASE-средства используются не только как комплексные технологические конвейеры для производства ИС, но и как мощный инструмент решения исследовательских и проектных задач, связанных с начальными этапами разработки, таких как анализ предметной области, разработка проектных спецификаций, выпуск проектной документации, планирование и контроль разработок, управление ресурсами и т. д.

Нередко применение CASE-средств выходит за рамки проектирования и разработки ИС. Они дают возможность оптимизировать модели организационных и управленческих структур предприятий, позволяют более эффективно решать задачи планирования, финансирования и управления производством.

И кроме этого, CASE-средства обеспечивают правильное ведение работ коллективом разработчиков, обеспечивают коммуникацию уча-

стников проекта на всех этапах и с разных позиций (как между командами заказчика и разработчика, так и внутри рабочей группы).

Единый пользовательский интерфейс означает, что CASE-инструментарий — это программные средства коллективного пользования для совместной реализации проекта. Коллективная работа команды разработчиков в сетевой среде предполагает обмен информации, контроль выполнения задач, отладку изменений и версий, планирование, взаимодействие и управление.

В этом плане CASE-технологии аналогичны **CALS-технологиям**, если изделием считать саму информационную систему: разработка ведется в соответствии с этапами ЖЦ в едином информационном пространстве (репозитарии), в котором информация обо всех этапах представлена в электронном виде и к которой имеют доступ все участники ЖЦ изделия (ИС).

Всегда следует иметь в виду, что CASE-средства становятся весьма эффективными только при их комплексном применении на всех этапах жизненного цикла. Поэтому производители CASE-средств стремятся к тому, чтобы их программный инструментарий имел средства разработки и сопровождения проекта одинаково эффективный на всех стадиях проекта.

Применение CASE-технологии для разработки системы комплексной автоматизации предприятия предполагает выполнение комплексного анализа, который требует использования множества разных типов моделей, отображающих разные стороны деятельности системы. При этом для обеспечения целостности процесса моделирования и анализа необходимо иметь возможность интеграции результатов моделирования в рамках общего проекта или одной модели.

Поэтому от выбора инструментальных средств моделирования существенно зависят объем и сроки выполнения, глубина и качество анализа при создании системы. Хотя формальный перечень работ, который необходимо выполнить на начальных этапах анализа системы, практически не зависит от того, какие методологии и инструменты будут использованы для моделирования и анализа. От выбора программного инструментария зависит только результат.

В процессе разработки ИС обычно выполняются три уровня анализа, каждый из которых соответствует трем этапам жизненного цикла: определение требований, формирование спецификаций и внедрение [40].

Определение требований начинается со сбора информации о предметной области, которая после предварительного экспресс-ана-

лиза отображается в виде моделей текущего состояния объекта проектирования (модели «AS IS»). Анализ этих моделей позволяет изучить особенности функционирования объекта, выявить имеющиеся узкие места, определить недостатки в организации процессов, структур и т. п.

Следующим шагом на этом этапе является создание концептуальных моделей будущей ИС (модели «TO BE»).

Результатом первого уровня анализа обычно становится техническое задание на разработку ИС.

Формирование спецификаций сопровождается выпуском технического проекта ИС, составной частью которого являются модели. На этом шаге принимаются во внимание ограничения, которые необходимо учитывать в моделях.

Третий уровень анализа — внедрение — связан с конкретной реализацией проекта ИС на предприятии.

При выполнении работ по моделированию на каждом из трех представленных выше уровней могут быть использованы различные инструментальные средства. Однако решающее значение при выборе инструмента моделирования имеет класс, к которому относится проектируемая система [39].

Специфика решаемых с помощью ИС задач, различная сложность их создания, модификации, сопровождения, интеграции с другими ИС позволяют разделить информационные системы на следующие классы:

- малые информационные системы;
- средние информационные системы;
- крупные информационные системы (корпоративные информационные системы — системы уровня федеральных организаций).

К классу малых информационных систем относятся системы уровня небольшого предприятия. К основным признакам таких систем следует отнести:

- непродолжительный жизненный цикл;
- ориентацию на массовое использование;
- невысокую цену;
- практическое отсутствие средств аналитической обработки данных;
- отсутствие возможности незначительной модификации без участия разработчиков;
- использование в основном настольных СУБД (Clarion, FoxPro, Clipper, Paradox, Access и др.);

• однородность аппаратного и системного программного обеспечения (широкое использование в качестве аппаратного обеспечения недорогих персональных компьютеров);

• практическое отсутствие средств обеспечения безопасности.

В отличие от предыдущего класса, признаками **средних информационных систем** являются:

• длительный жизненный цикл (возможность роста до крупных систем);

• наличие аналитической обработки данных;

• наличие штата сотрудников, осуществляющих функции администрирования аппаратных и программных средств;

• наличие средств обеспечения безопасности;

• тесное взаимодействие с фирмами-разработчиками программного обеспечения по вопросам сопровождения компонентов ИС.

И наконец, к характерным признакам **корпоративных информационных систем** следует отнести:

• длительный жизненный цикл;

• миграцию унаследованных систем;

• разнообразие используемого аппаратного обеспечения, жизненный цикл которого меньше, чем у создаваемой системы;

• разнообразие используемого программного обеспечения;

• масштабность и сложность решаемых задач;

• пересечение множества различных предметных областей;

• ориентацию на аналитическую обработку данных;

• территориальную распределенность, что особенно характерно для России.

В зависимости от класса создаваемой ИС, целей и объема моделирования, функциональности средств, их интеграции с другими инструментами и приложениями выбираются соответствующие программные инструментарии. В настоящее время известны более трехсот CASE-средств, которые классифицируются на **малые**, **средние** и **крупные интегрированные средства моделирования**.

Малые интегрированные средства моделирования поддерживают несколько типов моделей и методов, среди которых наиболее известными является пакет **All Fusion Modeling Suite**. Пакет имеет в своем составе программные инструментарии ***All Fusion Process Modeler*** (ранее BPwin) и ***All Fusion ERwin Data Modeler*** (ранее ERwin).

Пакет BPwin предназначен для описания, анализа и моделирования бизнес-процессов: позволяет строить организационные диаграммы, диаграммы плавательных дорожек, интегрированные функцио-

нальные модели на основе методологий IDEF0, IDEF3 и DFD. В BPwin имеется возможность указать стоимость, время и частоту его выполнения для любого процесса. Эти характеристики в дальнейшем могут быть просуммированы по всем функциональным блокам с целью вычисления общей стоимости затрат — таким образом выявляются узкие места технологических цепочек, определяются затратные центры. Кроме этого, BPwin интегрируется с пакетом ERwin для согласования функциональной и информационной моделей.

Пакет ERwin предназначен для моделирования баз данных с использованием методологии IDEF1X, основанной на ER-диаграммах. Она имеет два уровня представления модели — логический и физический, что позволяет строить одно из представлений на основе другого.

Кроме этого, пакет имеет средства для реализации многомерных хранилищ данных типа «звезда» и «снежинка», обеспечивает коллективную работу над проектом (All Fusion Model Manager), для выбранной СУБД автоматически создает базу данных физического уровня (таблицы, индексы, хранимые процедуры, триггеры и другие объекты).

Характерными особенностями этой категории является наличие в инструментальном средстве независимых компонентов (All Fusion Process Modeler, All Fusion ERwin Data Modeler и др.) и интеграция моделей путем экспорта и импорта.

Малые интегрированные системы практически не позволяют выполнить комплексный анализ систем, который в большей или меньшей степени необходим для создания средних и крупных ИС. С их помощью можно разрабатывать малые ИС или небольшие подсистемы, предназначенные для автоматизации отдельных бизнес-цепочек, например систем электронной коммерции, управления взаимоотношениями с клиентами, а также для решения задач так называемой«кусочной» автоматизации предприятия в соответствии с принципом «снизу вверх».

Средние интегрированные средства моделирования представлены продуктами, которые имеют единую среду для разработки всех поддерживаемых типов моделей и которые изначально были ориентированы на комплексное использование различных методов и типов моделей: Designer 10g (Oracle), Rational Rose (Rational Software), Paradigm Plus (CA/Platinum).

В состав **Designer 10g** входят Process Modeler и System Modeler. **Process Modeler** предназначен для построения иерархических моделей процессов, которые по структуре аналогичны IDEF0-моделям, а по

форме соответствуют диаграммам плавательных дорожек. Модель процессов позволяет в графической форме описать бизнес-процессы автоматизируемого предприятия и представить их более наглядно за счет анимации и использования мультимедийных файлов. Модели процессов используются на всех этапах разработки ИС.

System Modeler содержит в своем составе диаграммы для построения моделей иерархии функций (Function Hierarchy Diagrammer), в том числе на основе модели процессов в автоматическом режиме, моделей потоков данных (Data Flow Diagrammer) и информационных моделей типа сущность—связь (ER-Diagrammer).

Rational Rose и **Paradigm Plus** основаны на объектно-ориентированном подходе к моделированию на базе языка **UML** (Unified Modeling Language). Они предназначены для автоматизации этапов анализа и проектирования программного обеспечения, а также для генерации кодов на различных языках и выпуска проектной документации. Отличия между ними состоят в основном в доступных пользователю типах диаграмм и методов. Последние версии Rational Rose позволяют строить восемь типов диаграмм: диаграммы прецедентов, диаграммы классов, диаграммы последовательности, диаграммы сотрудничества, диаграммы состояний, диаграммы действий, компонентов и диаграммы развертывания.

Основным типом диаграмм являются диаграммы классов.

Средства моделирования среднего класса предназначены для выполнения комплексного анализа систем и могут быть использованы при создании малых и средних систем, особенно с этапа анализа спецификаций.

Крупные интегрированные средства моделирования ориентированы на проектирование крупных ИС, например систем управления предприятием класса ERP. Таким инструментальным средством является семейство **ARIS** (ARIS Toolset, ARIS Easy Design) компании IDS Sheer AG. Программные инструментарии ARIS (**Architecture of Integrated Information Systems**) ориентированы на описание бизнес-процессов организации. Методология ARIS рассматривает предприятие как совокупность взглядов на организационную структуру, структуру функций, структуру данных и структуру процессов. Использует нотации EPC (event-driven process chain), ERM (Entity-Relationship Model), UML.

В ARIS особое внимание уделено первому уровню анализа — анализу требований, и в нем обеспечивается четыре различных «взгляда» на моделирование и анализ: процессы, функции, данные и организа-

ция. При этом для каждого «взгляда» поддерживаются три уровня: требования, спецификации и внедрение. Каждый из уровней анализа состоит из своего комплекта моделей различных типов, в том числе и диаграмм UML.

Процессный взгляд является характерной особенностью этого инструментария — для моделирования процессов предназначено 57 типов моделей из 85 (рис. 3.1). Для конкретных разработок количество используемых типов моделей может быть ограничено с помощью специальных фильтров.

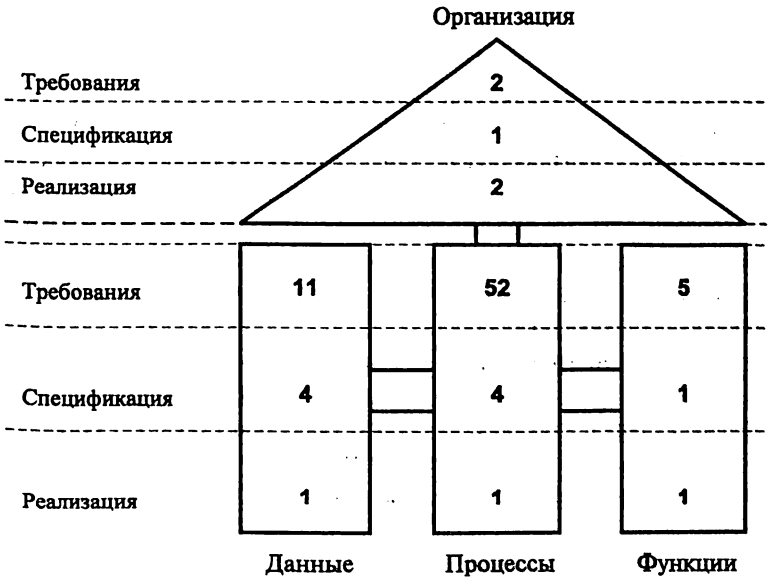


Рис. 3.1. Количество типов моделей ARIS для разных «взглядов» и уровней моделирования

Система ARIS имеет уникальные возможности для моделирования и анализа систем, моделирование может выполняться как «сверху вниз», так и «снизу вверх».

Система позволяет контролировать процесс разработки моделей, определять условия для функционально-стоимостного анализа, имитационного моделирования, определять цели и критические факторы, производить оценку рисков и конкурентов.

Наряду с классификацией инструментальных средств на малые, средние и крупные в настоящее время используется классификация ERP и не-ERP. Принадлежность к категории ERP для средства моде-

лирования означает, что оно предназначено для выполнения комплексного анализа на всех стадиях разработки ERP-системы: требования, спецификации, внедрение. Из рассмотренных выше инструментальных средств к категории ERP можно отнести только ARIS. Естественно, что такое средство может быть использовано при создании любых ИС.

3.2. Реализация CASE-технологии в инструментальной среде Oracle Designer 10g

Основные компоненты CASE-технологии рассмотрим на примере программного инструментария **Designer 10g** фирмы Oracle [10].

Основу CASE-технологии в инструментальной среде фирмы Oracle составляет **методология структурного проектирования** «сверху вниз», при которой разработка системы представляется в виде последовательности четко определенных этапов модели ЖЦ (рис. 3.2). Обеспечивается **поддержка всех этапов ЖЦ системы**, начиная с самых общих описаний предметной области до получения и сопровождения готового программного продукта.

ИС строится **по архитектуре «клиент—сервер»** с использованием всех особенностей современных серверов БД, включая декларативные ограничения целостности, хранимые процедуры, триггеры баз данных, и с поддержкой в клиентской части всех современных стандартов и требований к графическому интерфейсу конечного пользователя.

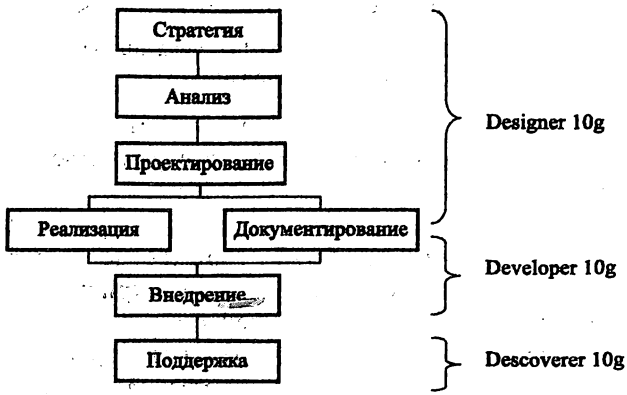


Рис. 3.2. Этапы построения ИС по технологии фирмы Oracle

Обязательным является использование **репозитария** для хранения спецификаций проекта разрабатываемой системы на всех этапах ее разработки и обеспечения доступа к ним всех участников разработки с учетом их прав доступа. Такой репозитарий представляет собой базу данных специальной структуры, которая работает под управлением СУБД. В настоящее время заказчики получают в свое распоряжение не только прикладные программы и базу данных разработанной ИС, но весь репозитарий, что существенно упрощает процесс модификации и сопровождение системы.

Для доступа к репозитарию и управления имеется навигатор, позволяющий просматривать и модифицировать объекты, хранящиеся в нем, а также осуществлять административные функции: удаление, управление доступом, экспорт/импорт и т. п., и является местом, в котором создается сама информационная система.

В репозитории обеспечивается централизованное хранение проекта системы и управление **одновременным доступом** к нему всех участников разработки, поддерживается согласованность действий разработчиков и не допускается ситуация, когда каждый разработчик или программист работает со своей версией проекта и модифицирует ее независимо от других.

С помощью специальных утилит реализуется **автоматизация перехода** от одного этапа разработки к следующему. Эти утилиты позволяют по спецификациям концептуального уровня (модели предметной области) автоматически получить первоначальный вариант спецификаций уровня проектирования (описание структуры базы данных и состава программных модулей), а затем на их основе с учетом необходимых уточнений и дополнений **автоматически генерировать готовые к выполнению программы** (рис. 3.3).

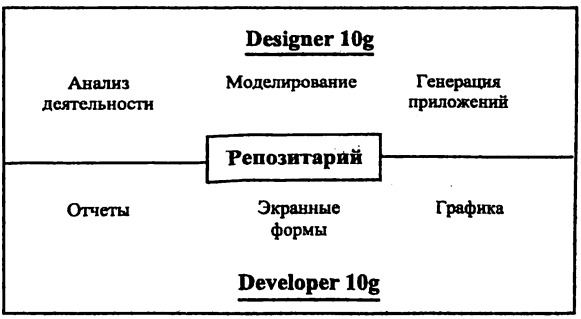


Рис. 3.3. Среда создания и сопровождения прикладных систем (фирма Oracle)

На основе содержимого репозитария предусматривается **генерация многочисленных отчетов**, обеспечивающих полное документирование текущей версии системы на всех этапах ее разработки. С помощью специальных процедур предоставляется возможность проверки спецификаций на полноту и непротиворечивость.

В соответствии с общей архитектурой CASE-системы Designer 10g можно выделить следующие этапы процесса разработки системы: моделирование и анализ деловой деятельности, разработка концептуальных моделей предметной области, проектирование прикладной системы и реализация (см. рис. 3.2).

Первый этап связан с моделированием и анализом процессов, описывающих деятельность организации, технологические особенности работы. Целью является построение моделей существующих процессов, выявление их недостатков и возможных источников усовершенствования.

Поддерживающие средства позволяют строить наглядные представления процессов и их взаимосвязей, а также анализировать их с использованием средств мультимедиа. Эти действия выполняются в режиме постоянного диалога с заказчиком с помощью инструментария для построения различных диаграмм, детализированных информационных и функциональных моделей проектируемой системы.

Подход, реализованный в Designer 10g, предполагает первым шагом в разработке системы моделирование бизнес-процесса в среде инструментария **Process Modeller** (см. рис. 3.3). Полученная иерархическая **модель процессов** затем используется в качестве основы для разработки концептуальных моделей и проектирования системы (см. рис. 6.4, 6.5). Затем применяют диаграммеры, входящие в **Systems Modelling: Function Hierarchy Diagrammer** — для построения функциональной модели (см. рис. 6.6), **Entity Relationship Diagrammer** — для построения логической модели базы данных в виде ER-диаграммы и **Data Flow Diagrammer** — для графического представления документооборота.

На втором этапе разрабатываются детальные концептуальные модели предметной области, описывающие особенности функционирования системы, ее информационные потребности и т. п. Этот этап фактически является этапом выявления, анализа и формализации требований к будущей системе. Для описания используются диаграммы «сущность—связь», диаграммы иерархии функций и диаграммы потоков данных. Оценка завершенности и полноты моделей производится с использованием множества матричных диаграмм.

Матричная диаграмма (Matrix Diagrammer) является средством анализа и модификации связей объектов различных типов. Для любого типа объектов репозитария Matrix Diagrammer автоматически выделяет подмножество связанных с ним типов и после указания требуемого (парного) типа строит соответствующую диаграмму в виде электронной таблицы. Например, описание модулей и используемых в них таблиц, описание модулей и используемых в них колонок таблиц.

Результатом являются модели двух типов: функциональные, описывающие особенности решаемых задач, и информационные, отображающие структуру и общие закономерности предметной области.

На этапе проектирования на основании концептуальных моделей вырабатываются технические спецификации будущей системы — специфицируется набор программных модулей, определяется структура и состав базы данных. Первоначальный вариант проектных спецификаций может быть получен автоматически с помощью специальных утилит на основании данных концептуальных моделей (рис. 3.4).

Утилита чернового проектирования **Database Design Wizard** на основе ER-модели выполняет генерацию описаний объектов базы данных (таблиц, внешних ключей, последовательностей и др.), а затем **Data Diagrammer** строит соответствующие диаграммы, отображающие взаимосвязи таблиц. Утилита представляет информационную модель ИС в виде множества объектов БД (таблиц, ключей, последовательностей и т. д.), определенных разработчиком в процессе проектирования.

Генерация описания прикладных программных модулей выполняется утилитой **Application Design Wizard.** Для этих целей используется информация, содержащаяся как в функциональной, так и в ER-модели проектируемой системы.

Формирование и просмотр структуры проектируемой системы (построение иерархии из полученных программных модулей) осуществляется с помощью **Module Structure Diagrammer**. Это средство описывает структуру прикладной системы на уровне взаимодействия модулей меню, экранных форм, отчетов, процедур и функций на языке PL/SQL.

Module Structure Diagrammer также предоставляет следующие возможности: создание новых модулей; включение в диаграмму модулей, полученных утилитой **Application Design Wizard**; изменение описания модуля (тип, язык и т. п.); изменение описания использования данных в модуле; копирование, слияние и расщепление модулей.

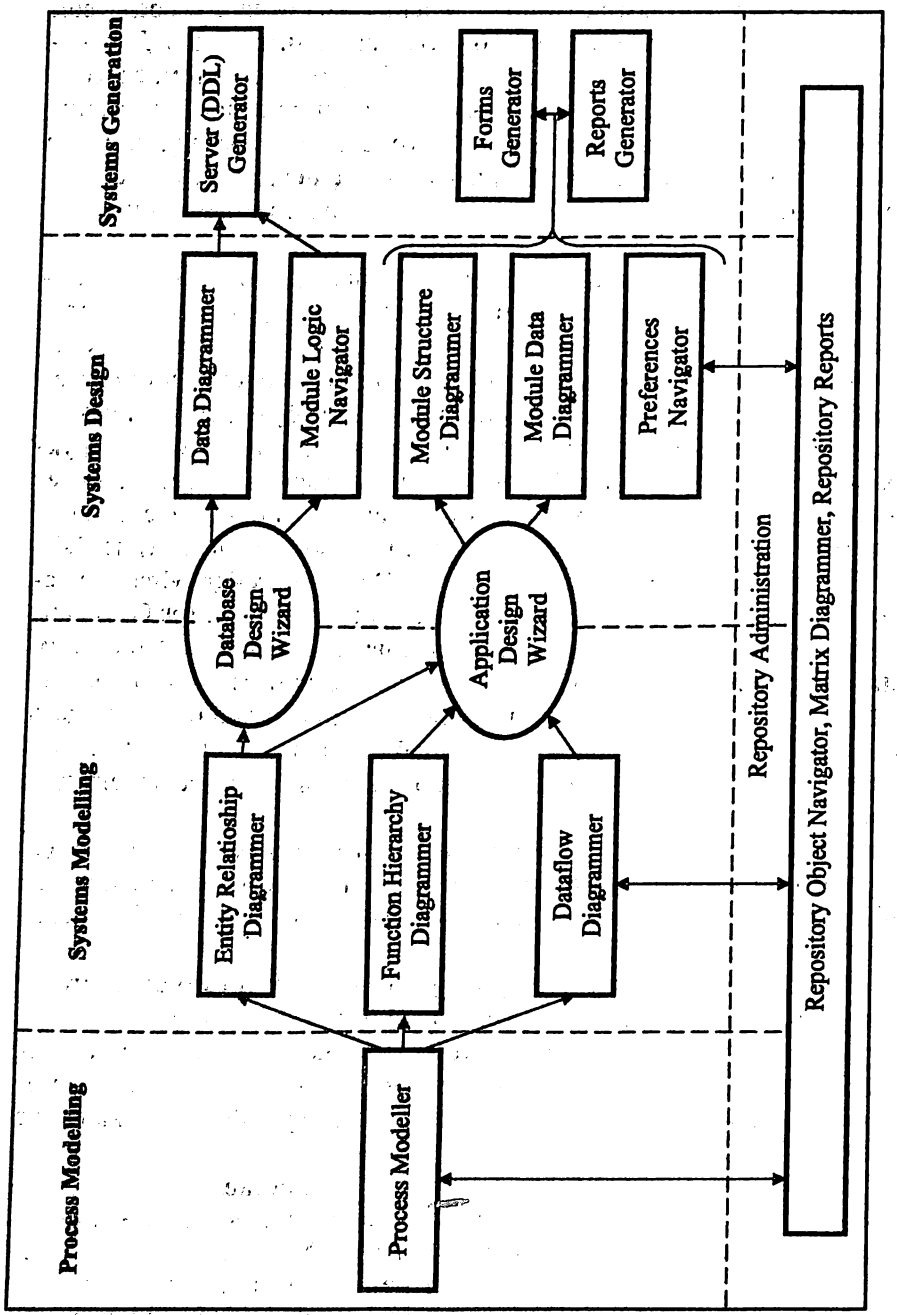


Рис. 3.4. Общая архитектура CASE-системы Designer 10g

Формирование детального описания использования данных каждым конкретным модулем осуществляется **Module Data Diagrammer**. Например, таблицы представляются отдельными прямоугольниками, со списками всех используемых полей. Связи между таблицами, соответствующие внешним ключам, отображаются в виде линий и используются для описания отношений типа родитель—потомок.

Здесь же разработчик имеет возможность специфицировать предполагаемый режим работы с таблицей. Например, будет ли в данном модуле разрешена модификация данных в таблице, или она используется только для просмотра. Средство позволяет также быстро и просто определить вид представления данных на экране (для диалоговых модулей).

Сложная логика обработки данных (хранимые процедуры, триггеры, функции и т. д.) описываются на языке PL/SQL, построенного на базе языка программирования Ada и языка структурированных запросов SQL. Инструментом для их создания и отладки служит **Module Logic Navigator**. Генерация модулей выполняется утилитой **Server DDL Generator**.

И наконец, ***на этапе реализации*** (генерация приложений) создается первая рабочая версия прикладной системы. Для этого необходимо на основе информации, хранящейся в репозитарии, выполнить генерацию объектов базы данных и исполняемых программных модулей, работающих с ней.

Генераторы, входящие в состав Designer10g, разбиваются на две группы: генератор серверной части **Server (DDL) Generator** и генераторы клиентской части (или генераторы приложений) **Forms Generator и Reports Generator**.

Server Generator по спецификациям БД автоматически генерирует тексты программ на языке SQL (DDL-скрипты), после исполнения которых на сервере будет создана база данных. Server Generator позволяет строить распределенные системы хранения и обработки данных, закрепляя объекты БД за конкретными узлами распределенной базы.

После того как объекты базы данных будут созданы на сервере, разработчик может сгенерировать исполняемые модули на клиентской части для ведения диалога с оператором (экранные формы) и подготовки отчетов. Эта работа выполняется с помощью средств генерации экранных форм (**Forms Generator**) и отчетов (**Reports Generator**).

Эти средства могут быть вызваны непосредственно из **Module Structure Diagrammer**, **Module Data Diagrammer**, **Preferences Navigator** и применены для построения экранных форм и отчетов соответственно.

Наличие средств предварительной настройки позволяет изменять правила формирования экрана по умолчанию, что позволяет на этой стадии сгенерировать экранные формы максимально соответствующими требованиям заказчика. При необходимости сгенерированные тексты могут быть доработаны с помощью пакета **Oracle Developer 10g**.

Все действия по генерации экранных форм и отчетов выполняются автоматически на основе описаний структур таблиц данных и описаний программных модулей, хранящихся в репозитарии.

Таким образом, в идеальном случае концепция CASE-технологии выглядит следующим образом.

Системные аналитики совместно с представителями заказчика определяют желаемые возможности будущей системы и требования к ней, общаясь на понятном обеим сторонам формальном языке с использованием наглядных визуальных диаграмм. Затем построенные таким образом диаграммы, объединившие в одно целое все требования к системе, уточняются на техническом уровне проектировщиками, к ним добавляются детали реализации, после чего они автоматически транслируются в готовые исходные тексты программ.

Программисты их немного правят, настраивая различные нюансы поведения системы, и получают готовый продукт, в котором ошибок гораздо меньше, чем в приложениях, созданных традиционным ручным кодированием.

3.3. Технологическая зрелость IT-предприятий

Все рассмотренные выше технологии проектирования относятся к CASE-технологиям, опыт освоения которых в очередной раз подтвердил принцип, утверждающий, что нельзя автоматизировать хаос. На одних предприятиях новая технология осваивалась быстро и давала положительные результаты, на других — приживалась с трудом.

Специфика различных CASE-технологий оказалась совершенно ни при чем — трудности освоения в основном обусловлены технологической незрелостью некоторых организаций-разработчиков ИС и

заключаются в неспособности организаций управлять технологическим процессом разработки информационной системы.

Понятие технологической зрелости было введено в США в середине 1980-х годов для формализации процедуры предварительного отбора достойных кандидатов для участия в тендерах на разработку дорогостоящих проектов ИС военного назначения и решения фундаментальной проблемы «хронического кризиса программного обеспечения».

Модель технологической зрелости является описанием стадий эволюции, которые проходят организации-разработчики по мере того, как они определяют, реализуют, измеряют, контролируют и совершенствуют процессы создания ИС. Эта модель помогает организации выбрать адекватную стратегию усовершенствования этих процессов, предоставляя методическую основу для определения текущего уровня их совершенства и выявления проблем, критичных для качества разрабатываемых ИС.

Для достижения устойчивых результатов в процессе развития технологии и организации управления жизненным циклом ИС в стандарте ГОСТ Р ИСО/МЭК 15504-1—2009 «Информационные технологии. Оценка процессов» рекомендуется методология обеспечения качества сложных программных средств CMM (Capability Maturity Model). Это система и модель оценки зрелости комплекса применяемых технологических процессов. Модель основана на формализации и использовании **пяти уровней зрелости** технологий поддержки ЖЦ ПС, которые определяют потенциально возможное качество и безопасность создаваемых комплексов программ.

Эти уровни технологической зрелости характеризуются степенью формализации, адекватностью измерения и документирования процессов и продуктов ЖЦ ПС, широтой применения стандартов и инструментальных средств автоматизации работ, наличием и полнотой реализации функций, системой обеспечения качества технологических процессов и их результатов. Они, в некоторой степени, подобны семи оценочным уровням доверия в стандарте **ISO 15408-3** [24].

Для оценки технологической зрелости организации-разработчика ИС существует **специальная методика**, в основу которой положено пять вопросников — по одному на каждый уровень зрелости. Вопросы касаются всех аспектов технологии разработки ИС, включая организацию управления проектом, формализацию технологического процесса, контроль качества, регистрацию промежуточных и конечных результатов и порядок их использования, а также квалификацию участников процесса создания ИС.

В результате были определены пять уровней технологической зрелости: ***первоначальный***, ***повторяемый***, ***определенный***, ***управляемый*** и ***оптимизируемый***. Эти уровни схематически показаны на рис. 3.5, где справа от названия соответствующего уровня перечислены основные меры, которые следует принять, чтобы сделать возможным переход на следующую, более высокую ступень.

Уровень 1. ***Начальный.*** Наиболее массовые разработки проектов ПС характеризуются относительно небольшими объемами программ в несколько тысяч строк, создаваемых несколькими специалистами. Они применяют простейшие неформализованные технологии с использованием типовых инструментальных компонентов операционных систем. Основные процессы ЖЦ ИС на этом уровне не регламентированы, выполняются не совсем упорядоченно, в некоторых случаях даже хаотически и зависят от нескоординированных индивидуальных усилий специалистов.

Процессы на первом уровне характеризуются своей непредсказуемостью по срокам в связи с тем, что их состав, назначение и последовательность выполнения могут меняться случайным образом в зависимости от текущей ситуации. Успех проекта, как правило, зависит от энергичности, таланта и опыта отдельных программистов. При увольнении этих сотрудников проект останавливается.

Уровень 2. ***Повторяемый.*** Для сложных проектов ИС объемом в десятки и сотни тысяч строк, в которых участвуют десятки специалистов разной квалификации, необходимы организация, регламентирование технологии и унификация процессов деятельности каждого из них. Процессы на этом уровне заранее планируются, их выполнение контролируется, чем достигается предсказуемость результатов и времени выполнения этапов, компонентов и проекта в целом.

Основной особенностью второго уровня является наличие формализованных и документированных процессов управления проектами, которые пригодны для модернизации, а их результаты поддаются количественной оценке. Основное отличие от первоначального уровня состоит в том, что выполнение процесса планируется и контролируется.

Необходимая дисциплина соблюдения установленных процессов имеет место и обеспечивает возможность ***повторения успеха предыдущих проектов*** в той же прикладной области. Успех проекта зависит от опыта и интуиции руководителя, для крупномасштабных проектов ПС с гарантированным качеством, риск провала остается еще достаточно большим.

Рис. 3.5. Уровни технологической зрелости организации-разработчика

Уровень 3. ***Определенный.*** Процессы ЖЦ ПС на этом уровне должны быть стандартизированы и представлять собой единую технологическую систему, обязательную для всех подразделений. Каждый проект использует утвержденную, адаптированную к особенностям данного проекта версию этой технологии. Полагают, что третий уровень технологической зрелости соответствует требованиям CASE-технологии.

Описание каждого процесса должно включать условия его выполнения, входные данные, рекомендации стандартов и процедуры выполнения, механизмы проверки качества результатов, выходные данные, условия и документы завершения процессов. В описания процессов включаются сведения об инструментальных средствах, необходимых для их выполнения, роль, ответственность и квалификация специалистов.

Основное отличие от повторяемого уровня заключается в том, что элементы процесса уровня 3 планируются и управляются на основе единого стандарта компании. Качество разрабатываемого проекта уже не зависит от способностей отдельных личностей.

Уровень 4. ***Управляемый.*** Для реализации проектов крупномасштабных, особенно сложных ИС в жестко ограниченные сроки и с высоким гарантированным качеством, необходимы активные меры для предотвращения и выявления дефектов и ошибок на всех этапах ЖЦ ИС. На этом уровне должна применяться система детального поэтапного оценивания характеристик качества как технологических процессов ЖЦ, так и самого создаваемого программного продукта и его компонентов. Должны разрабатываться и применяться универсальные методики количественной оценки реализации процессов и их качества.

Одновременно с повышением сложности и требований к качеству ПС следует совершенствовать управление проектами за счет сокращения текущих корректировок и исправлений дефектов при выполнении процессов. Результаты процессов становятся предсказуемыми по срокам и качеству в связи с тем, что они измеряются в ходе их выполнения и реализуются в рамках заданных ресурсных ограничений.

Основное отличие от определенного уровня состоит в более объективной, количественной оценке продукта и процесса.

Уровень 5. ***Оптимизируемый.*** Дальнейшая последовательная модернизация и совершенствование технологических процессов ЖЦ ИС для повышения качества их выполнения и расширения глубины контроля над их реализацией с использованием математических моделей процесса разработки.

Одна из основных целей этого уровня — сокращение проявлений и потерь от случайных дефектов и ошибок путем выявления сильных и слабых сторон используемых процессов. При этом приоритетным является анализ рисков, дефектов и отклонений от заданных требований заказчика. Эти данные также используются для снижения себестоимости ЖЦ особо сложных ИС в результате внедрения новых технологий и инструментария, а также для планирования и осуществления модернизации всех видов процессов.

Основное отличие от управляемого уровня заключается в том, что технология создания и сопровождения программных продуктов планомерно и последовательно совершенствуется.

Организация **считается соответствующей** данному уровню технологической зрелости, если все ключевые области процессов этого и всех нижестоящих уровней удовлетворены. Организация **не считается соответствующей** данному уровню технологической зрелости, если хотя бы одна ключевая область процессов этого или любого нижестоящего уровня не удовлетворяется.

По последним данным, в мире насчитывалось 139 организаций-разработчиков ИС высокого уровня зрелости (73 — четвертого и 66 — пятого уровня). Из общего числа высокоуровневых организаций 80 находятся за пределами США, в том числе в Индии 72 организации (!), из них 29 (4) и 43 (5). В России только две организации сертифицированы по пятому уровню: Центр разработок фирмы Motorola в Санкт-Петербурге и компания Люксофт.

Опыт работы с моделями технологической зрелости позволяет сделать следующие выводы.

- Следует избавиться от заблуждения, что организация с первого уровня зрелости может совершить «большой» скачок. Статистика показывает, что переход с одного уровня технологической зрелости на следующий уровень является продолжительным процессом. Он занимает для небольших коллективов от одного года и до трех лет для крупных.
- Для освоения современных технологий (выход на третий уровень) необходимо разработать 2—3 реальных проекта по новой технологии.
- Для уровней зрелости выше третьего только одно накопление необходимых статистических сведений и их обработка требует несколько лет.
- Подъем на каждую очередную ступеньку требует целой системы мероприятий, целенаправленно проводимых на протяжении продолжительного времени (см. рис. 3.5).

• Применение CASE-средств в организациях первого и второго уровней не дает желаемых результатов из-за отсутствия коллективной технологической культуры, которая должна создаваться годами. Обычно разработчики полагают, что строгие правила технологии, поддерживаемые CASE-средствами, слишком ограничивают их творческую свободу. В результате CASE-средства перестают использоваться по своему прямому назначению.

• С 1994 года в США к тендерам на военные заказы допускаются организации, имеющие уровень зрелости не ниже третьего.

Контрольные вопросы

1. Содержание понятия CASE-технологии.
2. Методологии современных CASE-средств: функционально-модульный (структурный) и объектно-ориентированный подходы.
3. Классификация информационных систем на малые, средние и крупные ИС.
4. Классификация CASE-средств на малые, средние и крупные интегрированные средства моделирования.
5. Проанализировать этапы модели ЖЦ на базе общей архитектуры CASE-системы Designer 10g (см. рис. 3.4).
6. Понятие технологической зрелости IT-предприятия.
7. Уровни технологической зрелости IT-предприятия.

Глава 4
МЕТОДОЛОГИИ ПАКЕТА ALL FUSION MODELING SUITE ДЛЯ ПОСТРОЕНИЯ ФУНКЦИОНАЛЬНЫХ И ИНФОРМАЦИОННЫХ МОДЕЛЕЙ

4.1. Разработка организационной диаграммы и Swim Lane Diagram

Рассмотрим более подробно CASE-технологию структурного моделирования как наиболее распространенную и доступную при первичном знакомстве с проектированием ИС. Для этого воспользуемся пакетом **All Fusion Modeling Suite**, который также является достаточно популярным в учебном процессе и при разработке проектов простых систем.

В последних версиях пакета появились диаграммы нового типа — **Swim Lane**, использующие нотацию IDEF3. Диаграммы плавательных дорожек Swim Lane иллюстрируют несколько параллельных потоков, что позволяет отобразить несколько взаимосвязанных процессов в виде плавательных дорожек на одной диаграмме (рис. 4.1). Кроме того, на диаграммах Swim Lane можно указать роли исполнителей работ, тем самым более качественно задокументировать роли и ответственность.

Организационные диаграммы (**Organization charts**) позволяют описать структуру предприятия и создаются на основе предварительно собранной информации о сотрудниках предприятия. Благодаря организационным диаграммам можно отобразить как структуру организации, так и любую другую иерархическую структуру (рис. 4.2).

Для построения этих диаграмм (см. рис. 4.1, 4.2) необходимо сформировать четыре словаря: словарь изображений, словарь ресурсов, словарь ролей и словарь групп ролей [25].

Словарь изображений содержит условные обозначения различных ролей для улучшения внешнего вида диаграмм. Изображения импор-

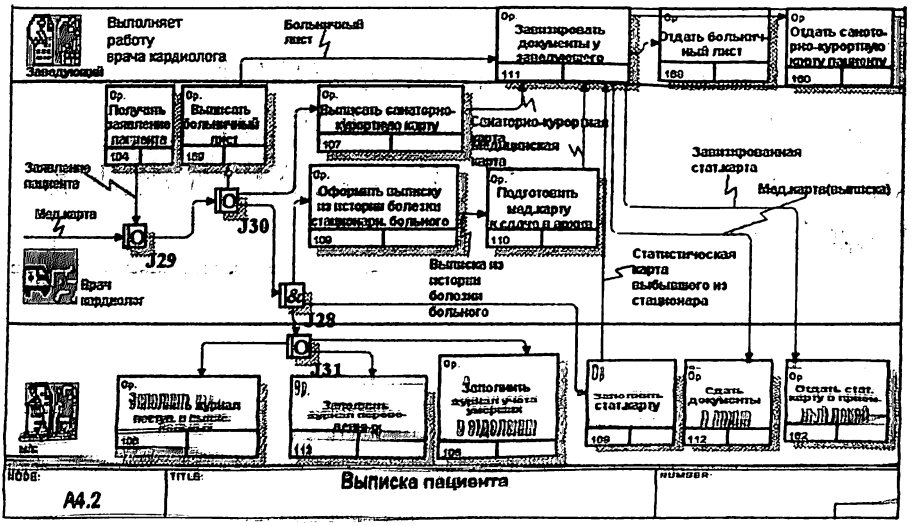


Рис. 4.1. Диаграмма Swim Lane

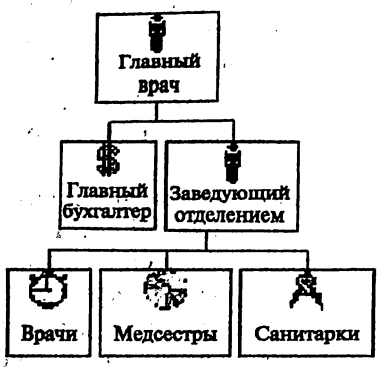


Рис. 4.2. Организационная диаграмма отделения больницы

тируются в словарь, для чего следует выполнить команду **Dictionary/Bitmaps**, в появившемся диалоговом окне **Bitmap Dictionary** щелкнуть по кнопке **Import** и найти файл формата **.bmp**.

В словаре групп ролей (Role Group Dictionary) создаются и определяются свойства групп ролей (необходимо выполнить команду **Dictionary/Role Group**) (рис. 4.3).

В качестве значения группы ролей может быть название предприятия, отдела, цеха и т. д., но удобнее в качестве группы ролей представлять различные виды деятельности: производственной, конструк-

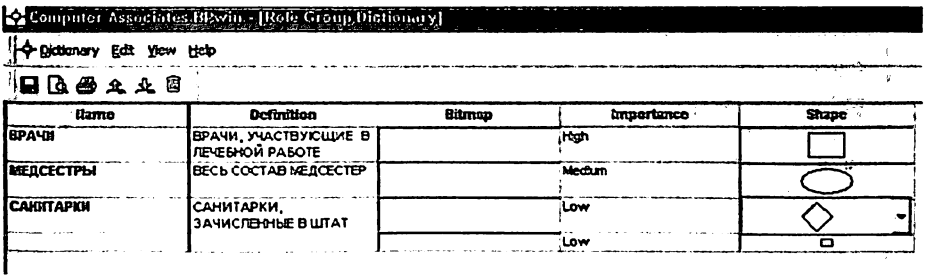


Рис. 4.3. Окно для создания словаря групп ролей

торской, технологической, финансовой, бухгалтерской, снабженческой и т. п.

Формирование групп ролей ведется системными аналитиками на основе выявленных ролей, которые заносятся в **словарь ролей (Role Dictionary)** по команде **Dictionary/Role** (рис. 4.4). Ролью может быть должность или профессия конкретного исполнителя.

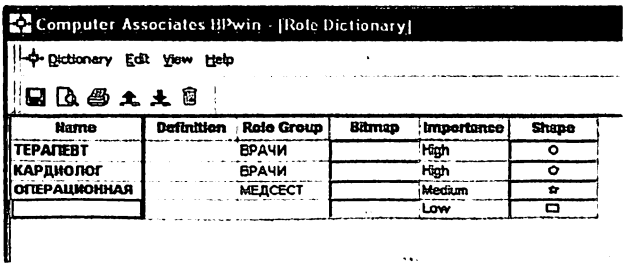


Рис. 4.4. Окно для создания словаря ролей

Каждой роли может соответствовать одна или несколько групп ролей, которые заносятся в словарь ролей. Кроме этого, в словарь ролей для каждой роли заносится определение (**Definition**), изображение для данной роли (**Bitmap**) и геометрическая фигура (**Shape**), а также указывается важность роли (**Importance**).

Затем формируется **словарь ресурсов (Resource Dictionary)** по команде **Dictionary/Resource**. Ресурсом для роли является конкретный исполнитель (фамилия, имя, отчество), который связывается с конкретной комбинацией «группа ролей/роль» (рис. 4.5). То есть в словарь ресурсов заносятся имя исполнителя, описание исполнителя и из списка выбираются его группа ролей и роль.

В результате этой работы системные аналитики получают точное представление о структуре и составе сотрудников предприятия. Сле-

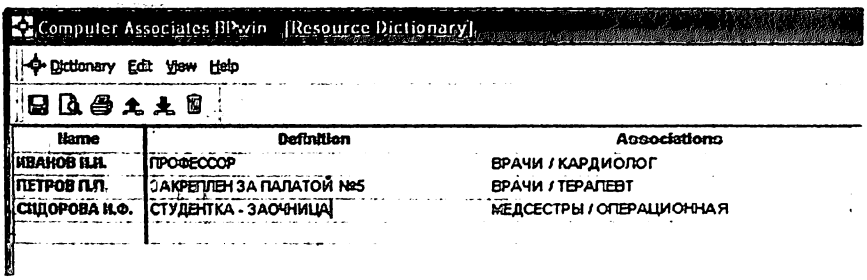


Рис. 4.5. Окно для создания словаря ресурсов

дующим шагом является построение самой организационной диаграммы, которая не только представит в графическом виде все группы ролей и соответствующие им роли, но и упорядочит их в виде иерархической структуры.

Создание диаграммы выполняется с помощью **мастера построения (Organization Chart Wizard)**, который запускается командой **Diagram/ Add Organization Chart** (рис. 4.6).

На первом шаге вносятся имя диаграммы, имя автора диаграммы, группа ролей и роль для верхнего уровня иерархии диаграммы.

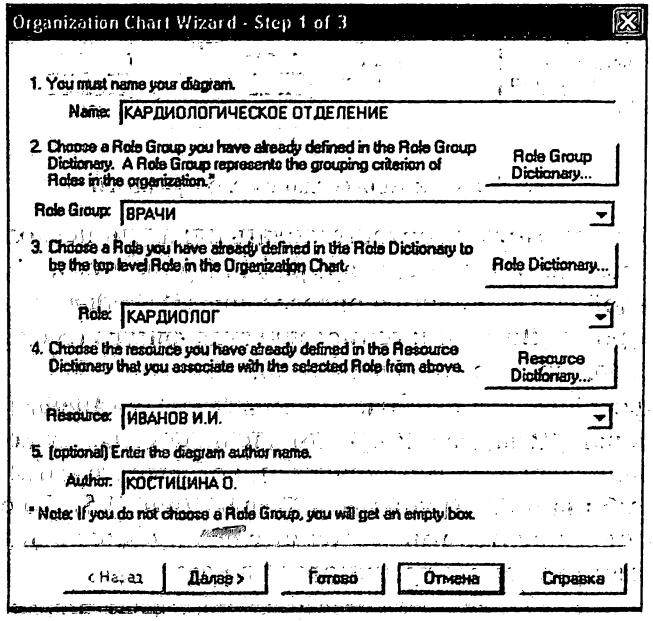


Рис. 4.6. Окно мастера построения организационной диаграммы

На втором шаге создается второй уровень иерархии с помощью диалогового окна, которое в верхнем окне содержит все доступные роли и ресурсы. Из этого списка переносятся в нижний список с помощью кнопки **Add** роли и ресурсы второго уровня.

Третий шаг предназначен для изменения свойств организационной диаграммы: в группе **drawing** можно выбрать ту информацию, которая будет отображаться на блоках диаграммы (наименование блока, имя группы ролей, роль, ресурс). После щелчка по иконе **Готово (Finish)** создается два уровня организационной диаграммы.

Для построения оставшихся уровней иерархического дерева используется контекстное меню, которое вызывается щелчком правой клавиши мыши по соответствующему блоку диаграммы. Оно содержит следующие элементы: **edit subroutine list** (редактирование блока), **add subordinates** (добавить нижний уровень), **add sibling on left** (добавить блок на текущий уровень слева от редактируемого блока), **add sibling on right** (добавить блок на текущий уровень справа от редактируемого блока).

Информация для заполнения блоков вызывается из словарей, для чего предварительно в контекстном меню необходимо щелкнуть по строке **Role name**.

На основе полученной организационной диаграммы и сформированных словарей можно приступить к графическому описанию основных сценариев предметной области с помощью **диаграмм плавательных дорожек (Swim Lane Diagram)**. Эти диаграммы представляют собой горизонтальные полосы (плавательные дорожки) для каждой роли. На этих плавательных дорожках с помощью графических объектов нотации IDEF3 (единиц работы, связей и перекрестков) отражается участие той или иной роли в рассматриваемом сценарии (см. рис. 4.1). Диаграмма может быть построена для одной или нескольких групп ролей, при этом из данной группы выбираются те роли, для которых необходимо сформировать плавательные дорожки. Для создания диаграммы Swim Lane следует выполнить команду **Diagram/Add Swim Lane Diagram**, которая запускает мастер построения плавательной диаграммы (**Swim Lane Diagram Wizard**) (рис. 4.7).

На первом шаге следует внести название диаграммы, имя автора и группу ролей, из которой можно будет выбрать роли, связанные с диаграммой. Если диаграмма строится на основе IDEF3-диаграммы, то следует указывать ее имя и номер.

На втором шаге следует выбрать роли, на основе которых будет создана диаграмма (рис. 4.8). В диаграмме будет столько полос (пла-

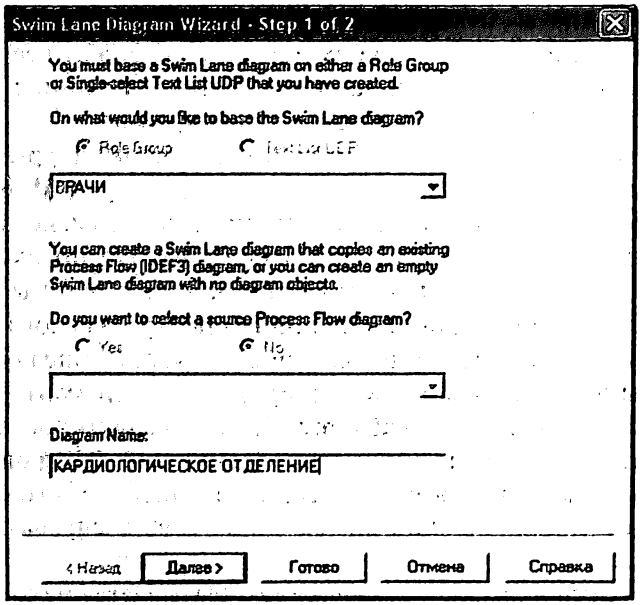


Рис. 4.7. Окно мастера построения диаграммы Swim Lane (1-й шаг)

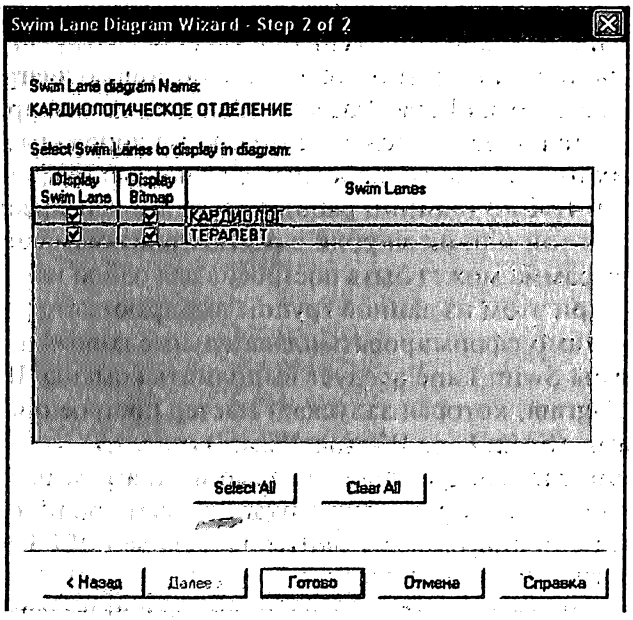


Рис. 4.8. Окно мастера построения диаграммы Swim Lane (2-й шаг)

вательных дорожек), сколько ролей будет отмечено «галочкой» в колонке **Display Swim Lane**.

После щелчка по иконе **Готово** создается новая диаграмма. Если она создавалась на основе IDEF3-диаграммы, то все объекты этой диаграммы будут располагаться по полосам произвольно и их надо разместить по ним в соответствии с ролями вручную. При создании диаграммы Swim Lane без использования готовой IDEF3-диаграммы она будет содержать только полосы, в которых необходимо вручную построить все необходимые объекты в соответствии со сценарием.

При необходимости можно добавлять новые полосы или удалять ненужные путем выбора опций в столбце **Display Swim Lane** (см. рис. 4.8).

Содержание диаграмм Swim Lane интуитивно понятно неподготовленному пользователю, и они чрезвычайно полезны на начальном этапе формализации предметной области в виде диаграмм и обычно используются в качестве модели «AS IS».

4.2. Методология IDEF0

Методология структурного моделирования предполагает построение функциональной модели «TO-BE» («Как должно быть»), т. е. модели, которая должна использоваться при построении информационной системы для предприятия. Для построения функциональных моделей обычно используется методология IDEF0, которая хорошо представлена в пакетах Design/IDEF и All Fusion Process Modeler (BPwin) [25, 26].

Построение IDEF0-моделей в среде этих двух пакетов практически не отличается, но в пакете BPwin возможно построение интегрированной функциональной модели, объединяющей диаграммы трех методологий: IDEF0, IDEF3 и DFD (Data Flow Diagramm).

Методология IDEF0, более известная как методология SADT (Structure Analysis and Design Technique), предназначена для представления функций системы и анализа требований к системам. Она является одной из самых известных и широко используемых методологий проектирования ИС.

В IDEF0 реализованы идеи системного анализа, под которыми понимают исследования, начинающиеся с общего обзора системы, а затем детализируют ее в виде иерархической структуры с определенным числом уровней, на каждом из которых допускается не более 8

элементов. В результате система разбивается на функциональные части, дается их описание, исследуются информационные потоки и формализуется структура данных.

Нотация IDEF0 позволяет наглядно представить бизнес-процессы и легко выявить такие недостатки, как недостаточно эффективное управление, ненужные, дублирующие, избыточные или неэффективные работы, неправильно использующиеся ресурсы и т. д. Признаком неэффективной организации работ является, например, отсутствие обратных связей по входу и управлению для многих критически важных работ.

В нотации IDEF0 основными элементами диаграмм являются функциональные блоки и дуги, которые представляются соответственно прямоугольниками и стрелками. Все дуги и функциональные блоки должны быть поименованы. Дуги представляют собой информацию, в которой блоки нуждаются или ее производят. Поэтому имена дуг — существительные, имена блоков должны начинаться с глагола или отглагольного существительного, поскольку блоки на диаграмме представляют функции, которые показывают, что должно выполняться: «выполнить расчет зарплаты», «сформировать накладную», «оформление договора». Каждый блок имеет свой номер, который размещается обычно в нижнем правом углу.

В методологии IDEF0 функциональный блок, который на самом верхнем уровне иерархии представляет систему в качестве единого блока (контекстная диаграмма), детализируется на другой диаграмме с помощью нескольких блоков, соединенных между собой интерфейсными дугами. Эти блоки представляют основные подфункции (подсистемы) единого исходного модуля. Каждый из этих подмодулей может быть декомпозирован подобным образом для более детального представления. Количество уровней иерархии не ограничивается, процесс декомпозиции блоков заканчивается тогда, когда каждый из модулей самого нижнего уровня декомпозиции может быть реализован в проектируемой системе одним программным модулем.

Результатом применения IDEF0-методологии является функциональная модель, которая состоит из диаграмм, фрагментов текста и глоссария, имеющих ссылки друг на друга. Диаграммы являются главными компонентами модели.

В IDEF0 используется **четыре типа дуг**: входные (INPUT), управления (CONTROL), выходные (OUTPUT) и механизма (MECHANIZM), представляющие собой **ICOM**-коды (аббревиатура из первых букв английских названий дуг). В качестве иллюстрации приведем

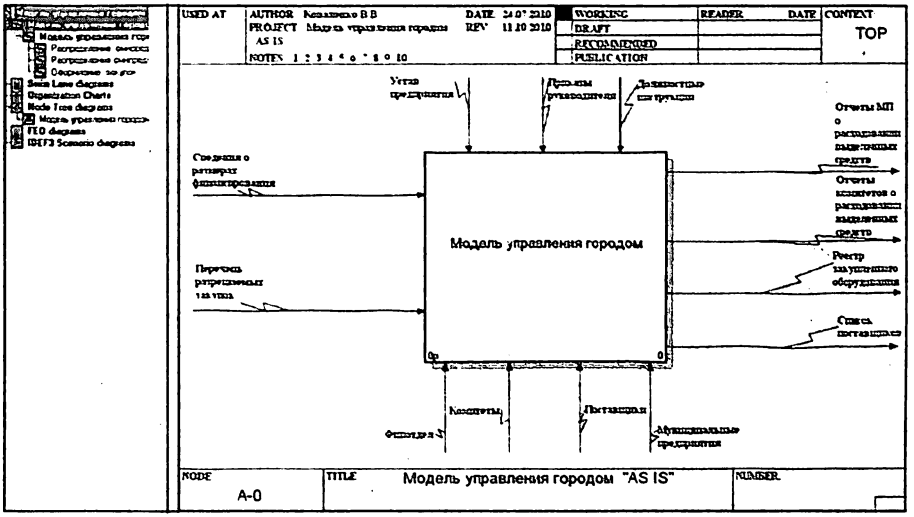


Рис. 4.9. Контекстная диаграмма методологии IDEF0

контекстную диаграмму функциональной модели управления городом (рис. 4.9).

Входы (INPUT) — это информация, которая используется или преобразуется функциональным блоком для получения результата. Они входят в функциональный блок слева (медицинская карта, заявления клиентов, накладные на поступивший товар и т. д.), допускается отсутствие входных дуг у блока.

Управление (CONTROL) — это правила, стратегии, процедуры или стандарты, которые указывают, чем руководствуется исполнитель при выполнении каждого из функциональных блоков (приказы Минздрава России, нормативная документация для выполнения данной работы в виде стандартов, законов, устава предприятия, инструкций, технических условий, должностных обязанностей и т. п.). Дуги управления входят в функциональный блок сверху, должна быть обязательно хотя бы одна дуга.

Механизмы (MECHANIZM) — это ресурсы, которые выполняют работу в функциональном блоке (персонал, подразделения, предприятия и т. п.). Они определяют, кто является ответственным за выполнение каждой из функций. Эти дуги входят в функциональный блок снизу. По усмотрению системного аналитика дуги ресурсов могут не изображаться графически, а представляться в виде текстового описания на диаграмме.

Дуги механизмов или управления могут быть входами других блоков.

Выходы (OUTPUT) — это данные, которые получаются в результате выполнения функции, т. е. указывают, что является результатом выполнения работы (всевозможные выходные формы, отчеты, договоры, больничный лист и т. д.). Выходные дуги выходят справа от функционального блока.

Обязательно должна быть хотя бы одна дуга, так как работа без результата не имеет смысла.

Контекстная диаграмма является верхним уровнем иерархии функциональной модели и представляет собой самое общее описание системы и ее взаимодействия с внешней средой. Для ее декомпозиции необходимо после выделения функционального блока щелкнуть левой клавишей мыши (ЛКМ) по ярлыку в панели инструментов ▼. В появившемся диалоговом окне (рис. 4.10) следует выбрать методологию, в нотации которой будет выполняться его декомпозиция, и количество блоков декомпозиции.

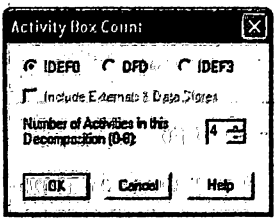


Рис. 4.10. Окно выбора методологий и количества блоков декомпозиции

Щелкнув по клавише OK, получают диаграмму декомпозиции первого уровня, которая содержит: выбранное выше количество блоков и ICOM-коды, т. е. названия всех дуг контекстной модели. Необходимо дать названия блокам, подсоединить к ним по смыслу ICOM-коды и построить требуемые соединительные (внутренние) дуги, присвоив им названия (рис. 4.11). В результате получают представление о том, какие функции и в какой последовательности выполняются в рамках декомпозированной родительской функции.

Если ICOM-объекты не используются на каком-либо уровне иерархии, то дуги, соединяющие эти объекты, можно поместить в **туннель**. Причём если в туннель помещен конец дуги (конец дуги помещается в квадратные скобки), то дуга и соответствующий ей ICOM-объект отсутствует на диаграмме-родителе. Если в туннель помещено начало дуги, то дуга и соответствующий ей ICOM-объект отсутствуют на диаграмме-потомке.

Для того чтобы извлечь дугу из туннеля, необходимо курсор поместить между квадратных скобок, а затем вызвать щелчком ПКМ контекстное меню и выполнить команду **Arrow Tunnel**. В появившемся диалоговом окне извлечь дугу из туннеля.

Дуги могут разветвляться и соединяться, образовывать обратные связи, итерации. Обратные связи этого типа на диаграмме отображаются снизу, т. е. обходят функции. С помощью этой обратной связи на диаграмме отображают тот факт, что регулярно анализируется выполнение плана и при необходимости формируют информацию, необходимую для корректировки плана на следующий период. То есть обратные связи по информации позволяют отобразить на диаграммах информационные потоки, необходимые для корректировки действий, выполняемых по ходу бизнес-процесса.

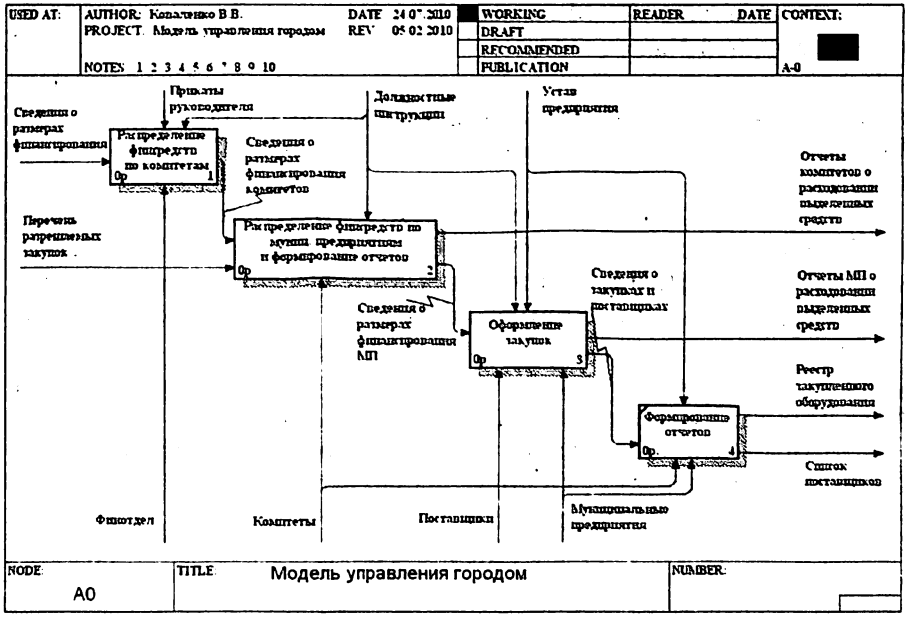


Рис. 4.11. IDEF0-диаграмма первого уровня декомпозиции

Обратная связь по управлению изображается в виде дуги, выходящей из правой стороны блока и входящей в верхнюю сторону блока, огибая диаграмму сверху функций. Обратная связь по управлению означает, что при анализе выполнения плана могут быть приняты оперативные управленческие решения, регулирующие выполнение по производству готовой продукции.

Для перемещения объекта необходимо вначале его выделить. Затем при нажатой ЛКМ переместить указатель объекта, который появится вместо курсора, в нужную точку.

При удалении объекта его необходимо выделить, а затем нажать на клавишу <Delete>. Для редактирования графических параметров объекта его следует выделить. После этого установить указатель мыши на необходимый маленький квадратик и, удерживая ЛКМ, перемещать курсор до получения желаемого графического состояния редактируемого объекта.

У выделенного объекта можно изменять тип линий, толщину и цвет, а также его фон. Это делается с помощью верхней инструментальной панели стандартным для Windows-приложений способом.

Для редактирования имен объектов необходимо с ним совместить курсор, с помощью ПКМ вызвать контекстное меню и воспользоваться командой Name.

Для удаления какой-либо из диаграмм ее необходимо открыть, выполнить команду **Edit/Delete diagram** и щелкнуть по кнопке **Delete**. Другой вариант удаления — выполнить команду **Diagram/Diagram manager**.

Для создания в автоматическом режиме отчета по IDEF0-модели необходимо находиться на любой IDEF0-странице и выполнить команду **Tools/Report**. После этого открывается меню, в котором следует выбрать один из семи отчетов, а также тип, стиль и размер шрифта.

Иерархия IDEF0-модели может быть представлена в виде «дерева узлов» (рис. 4.12). Для этого необходимо воспользоваться командой **Node Tree diagrams** в навигаторе, запускающей мастер построения диаграммы (**Node Tree Wizard**), который позволяет задать необходимые опции для выбора нужного типа диаграммы.

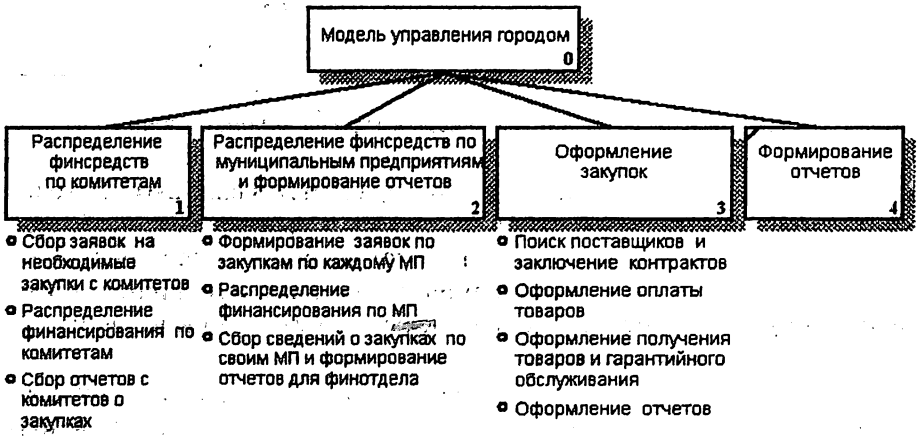


Рис. 4.12. «Дерево узлов»

IDEF0-методология в большей степени ориентируется на описание процессов верхних уровней. Ее применение характеризуется полнотой описания предметной области, которая достигается за счет наличия средств, отображающих управляющие воздействия, обратные связи по управлению и информации. Эта методология обеспечивает связанность диаграмм различных уровней декомпозиции в составе единой модели предметной области за счет использования механизма мигрирования (ICOM-кодов) и туннелирования дуг. Такой механизм обеспечивает связанность создаваемых диаграмм между собой и делает модель предметной области наглядной.

На более низких уровнях иерархии можно использовать применение методологий IDEF3 и DFD для создания диаграмм в составе единой модели. Основными признаками для применения методологии IDEF3 является наличие логики в описании процессов, для применения методологии DFD — наличие хранилищ для промежуточного хранения данных.

4.3. Методология IDEF3

Нотация **IDEF3 (Workflow diagramming)** является второй важнейшей категорией после IDEF0 и ориентирована на описание логики взаимодействия информационных потоков. Особенно удобно применять IDEF3 на нижних уровнях иерархии функциональных моделей при описании работ, выполняемых в подразделениях и на рабочих местах (рис. 4.13). С помощью диаграмм IDEF3 удобно описывать сценарии действий работников подразделения, содержащих логику: когда процессы выполняются в определенной последовательности, задаваемой соответствующими логическими условиями (есть ли товар на складе, подписан ли документ, заключен ли договор с поставщиком и т. п.).

IDEF3 предполагает создание двух типов моделей: модель может отражать некоторые процессы в их логической последовательности, позволяя увидеть, как функционирует предприятие (процессно-ориентированный подход), или же модель может показывать сеть переходных состояний объекта, предлагая вниманию системного аналитика последовательность состояний, в которых может оказаться объект при прохождении через определенный процесс. Обычно используется первый тип модели.

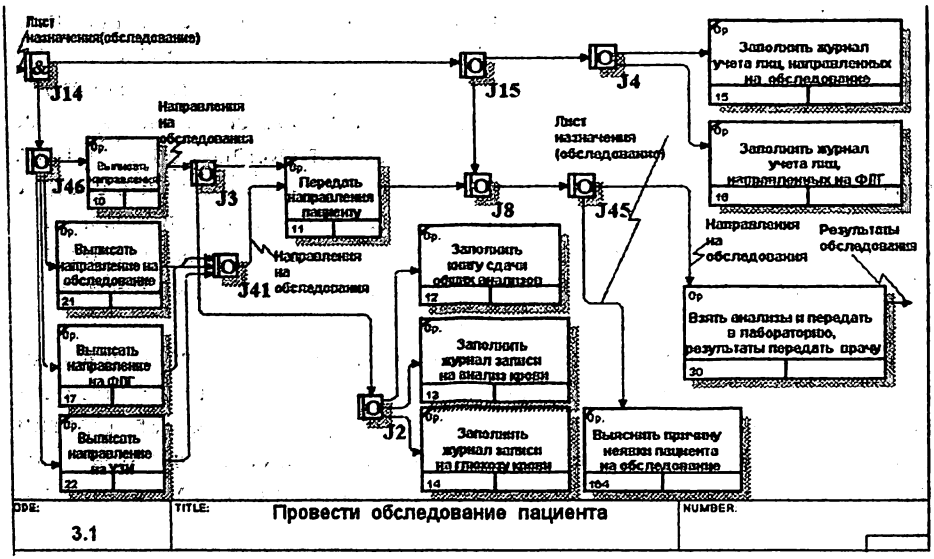


Рис. 4.13. IDEF3-диаграмма сценария «Провести обследование пациента»

Центральным компонентом модели является **единица работы**, близкая по смыслу к работе IDEF0, которая изображается прямоугольником и имеет имя в виде отглагольного существительного в составе фразы, обозначающего процесс действия (изготовление изделия, генерация выходной формы и т. п.).

Взаимоотношения между работами реализуются с помощью **стрелок**, которые рисуются слева направо или сверху вниз.

Старшая стрелка (связь предшествования) изображается сплошной линией и означает, что работа-источник должна закончиться прежде, чем работа-цель начнется.

Связь отношения обозначается пунктирной линией и показывает связь между двумя работами или между работой и объектом ссылки.

Объект **Ссылка (References)** в IDEF3 выражает данные или определенную идею, которые нельзя связать со стрелкой, перекрестком или работой (клиент, заказы клиента, склад и т. п.). Объекты ссылок должны быть связаны с единицами работ или перекрестками пунктирными линиями.

Характерным объектом IDEF3 является **Перекресток (Junction)**, который отображает не только логику взаимодействия стрелок при слиянии и разветвлении, но и для отображения множества событий, которые могут быть завершены перед началом следующей работы.

В диаграммах IDEF3 любое разветвление или объединение стрелок происходит только с помощью перекрестков для разветвления (Fan-out Junction) или слияния (Fan-in Junction) соответственно. Имеется пять наименований перекрестков, которые обеспечивают любую логику в сценариях (табл. 4.1).

Таблица 4.1. **Описание перекрестков IDEF3**

Обозначение	Наименование	Смысл в случае слияния стрелок	Смысл в случае разветвления стрелок
	Asynchronus AND	Все предшествующие процессы должны быть завершены	Все следующие процессы должны быть запущены
	Synchronus AND	Все предшествующие процессы завершены одновременно	Все следующие процессы запускаются одновременно
	Asynchronus OR	Один или несколько предшествующих процессов должны быть завершены	Один или несколько следующих процессов должны быть запущены
	Synchronus OR	Один или несколько предшествующих процессов завершены одновременно	Один или несколько процессов запускаются одновременно
	XOR (Exclusive OR)	Только один предшествующий процесс завершен	Только один следующий процесс запускается

Для внесения перекрестка в диаграмму служит кнопка в палитре инструментов (добавить в диаграмму перекресток — Junction), тип перекрестка выбирается из диалогового окна **Junction Type Editor**. Все перекрестки на диаграмме нумеруются, каждый номер имеет префикс J.

Работы в диаграммах IDEF3 также декомпозируются, как и в диаграммах IDEF0, но при этом не происходит мигрирования и туннелирования стрелок. Поэтому системный аналитик должен сам заботиться о связанности моделирования процесса и корректности декомпозиции.

Эту особенность необходимо учитывать при декомпозиции работы из диаграмм IDEF0 в диаграмму IDEF3. Как уже было отмечено, согласно нотации IDEF3-диаграмма не должна иметь граничных стрелок, все стрелки должны заканчиваться на работах, перекрестках или объектах ссылки. Поэтому все стрелки на родительском блоке IDEF0-диаграммы автоматически помещаются в тоннель, а в дочер-

ней диаграмме вместо граничных стрелок следует создать объекты ссылки и затем внутренними стрелками соединить их с соответствующими работами.

4.4. Методология диаграммы потоков данных (DFD)

Диаграммы потоков данных (**Data Flow Diagramming, DFD**) применяются для документирования механизма передачи и обработки информации в проектируемой системе, они удобны для наглядного изображения текущей работы системы документооборота организации. Обычно DFD используют в качестве дополнения IDEF0-модели или в ее составе на нижних уровнях иерархии для более наглядного отображения текущих операций документооборота в корпоративных системах обработки информации [25, 32].

DFD строится на основе четырех элементов: работа, хранилище данных, внешние ссылки, дуги. Наличие в диаграммах DFD элементов для описания источников, приемников и хранилищ данных позволяет более эффективно и наглядно описать процесс документооборота.

Работы (функции обработки информации) обозначают процессы, которые обрабатывают и изменяют ситуацию. Они представляются на диаграммах в виде прямоугольников со скругленными углами. Стрелки идут от объекта-источника к объекту-приемнику, обозначая информационные потоки в системе документооборота. В DFD каждая сторона работы не имеет четкого назначения, как в IDEF0, поэтому стрелки могут подходить и выходить из любой грани прямоугольника работы, они могут сливаться и разветвляться.

Хранилища данных изображают объекты в покое, например в очереди на обработку, представляя собой данные, к которым осуществляется доступ и которые могут быть созданы или изменены работами (бумажный документ, файл, база данных и т. п.).

Если поставщик или потребитель информации представляет процесс сохранения или запроса информации, то вводится хранилище данных, для которого данный процесс является интерфейсом. Фактически хранилище представляет «срезы» потоков данных во времени. Информация, которую оно содержит, может использоваться в любое время после ее получения, при этом данные могут выбираться в любом порядке.

В отличие от стрелок, описывающих объекты в движении, хранилища данных изображают объекты в покое. Имя хранилища должно определять его содержимое и быть существительным. На одной диаграмме может присутствовать несколько копий одного и того же хранилища, чтобы исключить наличие длинных и запутанных стрелок.

Внешние ссылки (сущности) указывают на организацию или человека, которые участвуют в процессе обмена информацией с системой, но располагаются за пределами данной диаграммы. То есть они изображают входы в систему и/или выходы из нее. Внешние ссылки изображаются в виде прямоугольника с тенью и располагаются по краям диаграммы. Одна внешняя ссылка может быть использована многократно на одной или нескольких диаграммах, чтобы исключить наличие длинных и запутанных стрелок.

В отличие от стрелок IDEF0, которые представляют собой жесткие взаимосвязи, стрелки DFD показывают, как объекты (включая данные) двигаются от одной работы к другой. Это представление потоков совместно с хранилищами данных и внешними сущностями делает модели DFD более похожими на физические характеристики системы — движение объектов, хранение объектов, поставка и распространение объектов.

В DFD также применяются двунаправленные стрелки для описания диалогов типа команды-ответа между работами, между работой и внешней сущностью и между внешними сущностями.

Процесс построения DFD начинается с создания так называемой основной диаграммы типа «звезда», на которой представлен моделируемый процесс и все внешние сущности, с которыми он взаимодействует.

С помощью нотации DFD можно создать описание реально существующих в организации потоков данных, как по **процессному**, так и по **функциональному** признаку. В первом случае получают модели бизнес-процессов в формате DFD, во втором — схему обмена данными между подразделениями.

Диаграмма потоков данных при функциональном подходе для кардиологического терапевтического отделения представлена на рис. 4.14. Пациент при поступлении в терапевтическое отделение представляет медицинскую и статистическую карты, проходит обследование — все это является входным потоком документов. На выходе диаграммы данного уровня пациент может получить следующий перечень выходных документов: медицинскую и статистическую карты с результатами обследования и лечения, санаторно-курортную карту

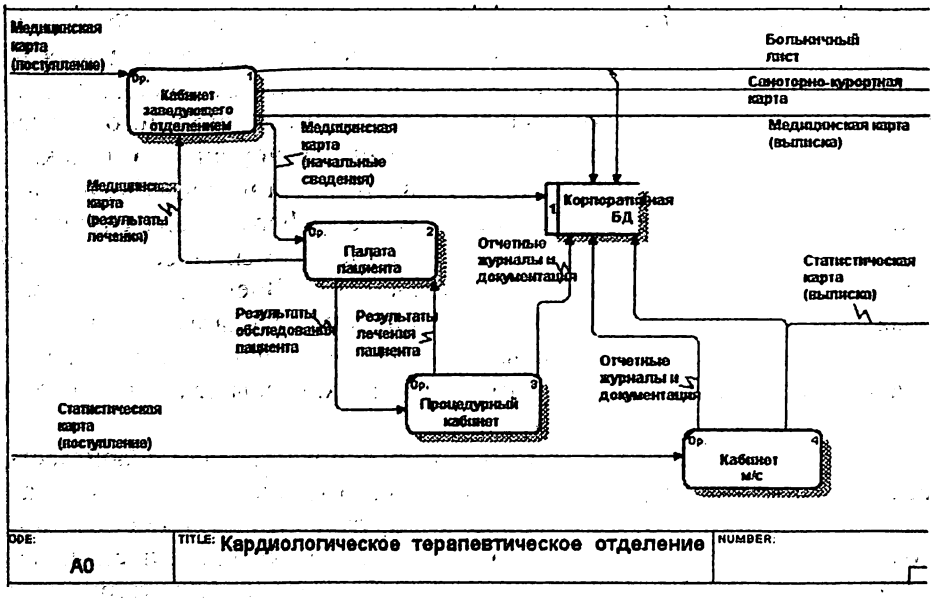


Рис. 4.14. Диаграмма потоков данных при функциональном подходе (DFD-методология)

и больничный лист. На диаграмме присутствует хранилище данных, в котором будут размещаться копии документов и отчетные журналы.

В тех случаях, когда организационная структура терапевтического отделения не представляет интереса, более удобно использовать «объектную» диаграмму потоков данных (рис. 4.15). Основным объектом на диаграмме является терапевтическое отделение, а пациент представляется диаграмме как внешняя ссылка.

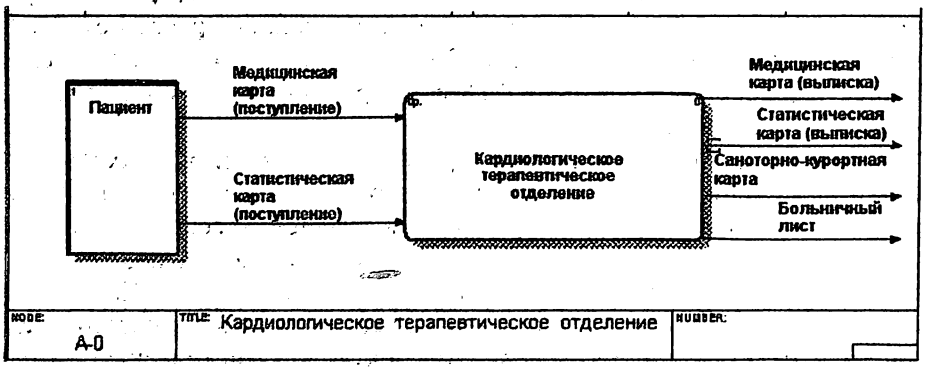


Рис. 4.15. Диаграмма потоков данных при объектном подходе (DFD-методология)

Подобная реализация диаграммы потоков данных позволяет более наглядно представить потоки документов между различными объектами.

На практике обычно применяют смешанные модели: на верхних уровнях иерархии используют нотацию IDEF0 для выявления функций, входящих в процесс. Затем при достижении некоторого уровня иерархии становится целесообразно сформировать диаграммы в формах DFD (для отображения потоков данных и материалов) и IDEF3 (для отображения логики в сложных сценариях).

Применение универсальных графических языков моделирования IDEF0, IDEF3 и DFD обеспечивает логическую целостность и полноту описания предметных областей.

Пакет BPwin обладает удобным инструментом для навигации по уровням иерархии модели **Model Explorer**, который по организации подобен обычному проводнику Windows (см. рис. 4.9). Функциональные IDEF0-блоки в Model Explorer показываются зеленым цветом, DFD — желтым и IDEF3 — синим. Щелкнув мышкой по любой из работ, представленных в навигаторе, пользователь может переходить на диаграмму, которая содержит данную работу. Однако следует помнить об определенных правилах декомпозиции работы одной нотации в диаграмму другой. В пакете BPwin допускаются только следующие переходы с одной нотации на другую: **IDEF0 → DFD; IDEF0 → IDEF3; DFD → IDEF3**.

Согласно нотации DFD-диаграмма не должна иметь граничных стрелок, т. е. все стрелки должны начинаться и заканчиваться на работах, хранилищах данных или внешних ссылках. Поэтому при строгом следовании правилам нотации необходимо на дочерней DFD-диаграмме удалить все граничные стрелки и вместо них создать соответствующие внешние ссылки и хранилища данных. Затем вместо удаленных граничных стрелок создать внутренние стрелки, начинающиеся с внешних ссылок или хранилищ данных.

В тех случаях, когда на смешанных моделях сложно придерживаться этих правил нотации, BPwin позволяет создавать граничные дуги на DFD-диаграммах, не идентифицируя их как синтаксическую ошибку.

При декомпозиции работы IDEF0 или DFD в диаграмму IDEF3 дуги не мигрируют на дочернюю диаграмму, поскольку дуги на диаграммах IDEF3 могут показывать только последовательность выполнения работ и, следовательно, имеют другой смысл, чем в IDEF0 и DFD.

Пакет BPwin позволяет во всех нотациях использовать для работы не прямоугольники, а практически **любые геометрические фигуры**.

При этом на работах можно поместить изображение, импортированное в словарь Bitmap Dictionary (рис. 4.16).

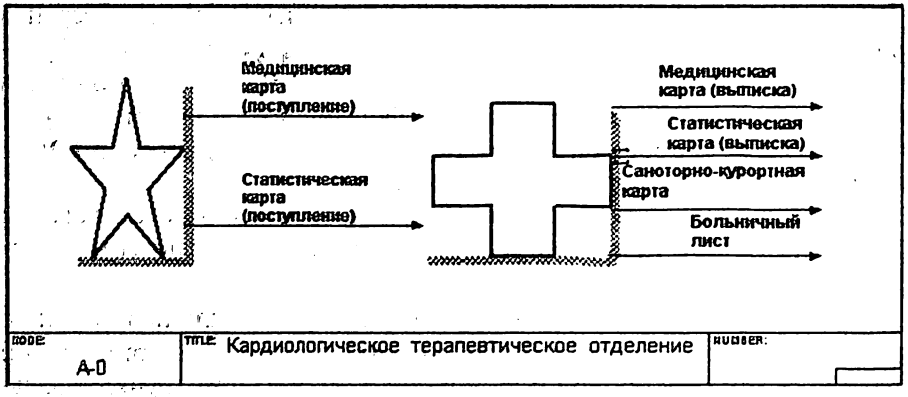


Рис. 4.16. Диаграмма потоков данных в нетрадиционном синтаксисе

Для использования нетрадиционного синтаксиса необходимо щелкнуть по работе и выбрать в контекстном меню пункт **Box Style**. Во вкладке Box Style следует выбрать опцию **Custom** и указать геометрическую фигуру (**Shape**) для работы и изображение (**Bitmap**). После щелчка по кнопке OK на диаграмме работа отображается в нетрадиционном синтаксисе.

Использование нетрадиционного синтаксиса удобно не только для визуального эффекта, но и при преобразовании диаграммы IDEF3 в имитационную модель для пакета Arena.

4.5. Стоимостный анализ (Activity Based Costing, ABC)

Пакет BPwin предоставляет аналитику инструмент для оценки модели — стоимостный анализ, основанный на работах (**Activity Based Costing, ABC**). Исходными данными для функционального оценивания являются затраты на ресурсы, необходимые при эксплуатации данной информационной системы (зарплата персонала, стоимость арендуемых помещений и компьютерной техники, расходных материалов, коммунальных услуг и т. д.).

Стоимостный анализ основан на модели работ, потому что количественная оценка невозможна без детального понимания функциональности предприятия. Обычно ABC применяется для того, чтобы

понять происхождение выходных затрат и облегчить выбор нужной модели работ при реорганизации деятельности предприятия (Business Process Reengineering, BPR).

С помощью стоимостного анализа можно решить такие задачи, как определение действительной стоимости эксплуатации системы, определение действительной стоимости поддержки клиента, идентификация работ, которые стоят больше всего (те, которые должны быть улучшены в первую очередь), обеспечение менеджеров финансовой мерой предлагаемых изменений и др.

При проведении стоимостного анализа в BPwin сначала задаются единицы измерения времени и денег. Для задания единиц измерения следует вызвать диалог **Model Properties** (меню **Model/Model Properties**), вкладка **ABC Units** (рис. 4.17).

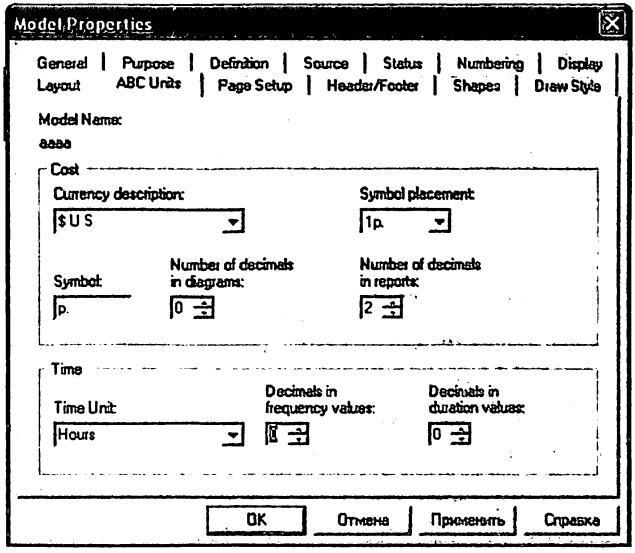


Рис. 4.17. Настройка единиц измерения валюты и времени

Затем описываются **центры затрат (cost centers)**. Для внесения центров затрат необходимо вызвать диалог **Cost Center Dictionary** из меню **Dictionary/Cost Center** (рис. 4.18). Каждому центру затрат следует дать подробное описание в окне **Definition.** Для отдельной модели задается один набор функциональных центров.

Для задания стоимости работы (для каждой работы на диаграмме декомпозиции нижнего уровня) следует щелкнуть ПКМ по работе и на всплывающем меню выбрать **Costs** (рис. 4.19).

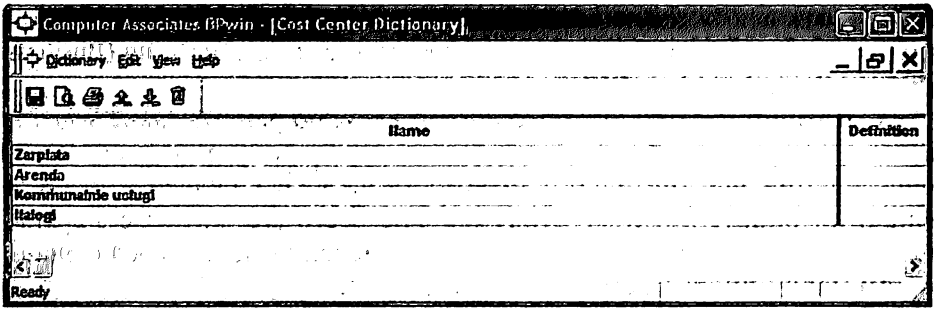


Рис. 4.18. Диалог Cost Center Dictionary

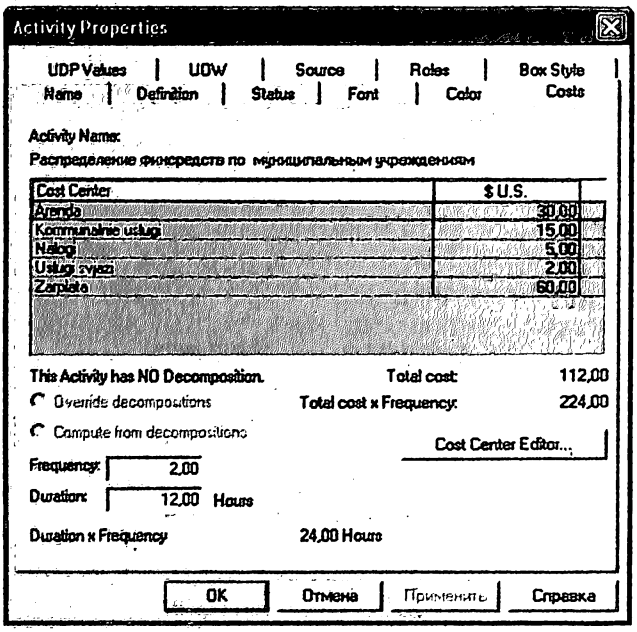


Рис. 4.19. Задание стоимости работ в диалоге Activity Cost

Во вкладке **Costs** диалога **Activity Properties** указывается частота проведения данной работы в рамках общего процесса (окно **Frequency**) и продолжительность (**Duration**). Затем следует выбрать в списке один из центров затрат и в окне **Cost** задать его стоимость. Аналогично назначаются суммы по каждому центру затрат, т. е. задается стоимость каждой работы по каждой статье расхода.

Общие затраты по работе рассчитываются как сумма по всем центрам затрат. При вычислении затрат вышестоящей (родительской)

работы сначала вычисляется произведение затрат дочерней работы на частоту работы (число раз, которое работа выполняется в рамках проведения родительской работы), затем результаты складываются. Если во всех работах модели включен режим **Compute from Decompositions** (в окне **Activity Properties**), подобные вычисления автоматически проводятся по всей иерархии работ снизу вверх.

Результат расчета стоимости размещается внутри каждого функционального блока.

4.6. Создание баз данных логического и физического уровней

4.6.1. Общие сведения о методологии IDEF1X

Методология IDEF1X представляет собой семантическое моделирование данных и применяется для построения информационной модели в виде ER-диаграммы (рис. 4.20), которая представляет структуру информации, необходимой для поддержания функции производственной системы или среды.

Основными конструкциями **ER-диаграммы** являются:

- предметы (сущности), к которым относятся данные;

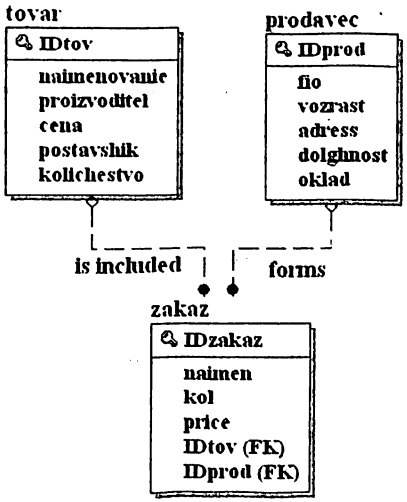


Рис. 4.20. ER-диаграмма

• отношения между этими предметами, которые изображаются с помощью линий, соединяющих эти блоки;

• характеристики этих предметов, изображаемые именами атрибутов внутри блоков.

Таким образом, построение ER-диаграммы предполагает определение **сущностей, атрибутов и первичных ключей**.

Сущность представляет собой множество реальных или абстрактных предметов, обладающих общими атрибутами или характеристиками. Отдельные элементы этого множества называются **экземпляром** сущности. Объект (предмет) может быть представлен в нескольких сущностях. Кроме этого, экземпляр сущности может представлять собой комбинацию существующих объектов.

Каждой сущности присваивается уникальное имя, которое помещается над блоком. Имя является грамматическим оборотом существительного в единственном числе (у существительного могут быть прилагательные и предлоги). Сущность может иметь список синонимов и псевдонимов, и все они должны быть приведены в глоссарии модели.

Одна и та же сущность может быть изображена на любом числе диаграмм, но на каждой конкретной диаграмме она должна быть представлена только один раз.

Правила, связанные с сущностями:

1. Каждая сущность должна иметь уникальное имя.

2. Сущность обладает одним или несколькими атрибутами, которые либо принадлежат сущности, либо указываются через отношения (внешние ключи).

3. Сущность обладает одним или несколькими атрибутами, которые однозначно идентифицируют каждый экземпляр сущности (первичные и альтернативные ключи).

4. Каждая сущность может обладать любым количеством отношений с другими сущностями модели.

5. Если внешний ключ целиком используется в качестве первичного ключа или его части, то сущность является зависимой (от идентификатора), и наоборот, если используется только часть внешнего ключа или вообще не используются внешние ключи, то сущность является независимой от идентификатора.

Проектировщик должен таким образом определить хранящуюся информацию в конкретной сущности и в конкретном атрибуте, чтобы обеспечить полную информационную поддержку для выполнения всех функций, заложенных в информационную систему. Поэтому

в основе построения ER-модели должна находиться функциональная иерархическая модель (в общем случае интегрированная IDEF0-модель), стрелки в которой обозначают информацию, используемую в моделируемой системе.

Обычно исходят из того, что каждой стрелке должна соответствовать либо сущность, либо атрибут сущности. При этом следует учитывать, что информация, которая моделируется в виде одной стрелки в модели процессов, может содержаться в нескольких сущностях и атрибутах модели данных. Например, стрелке «новый товар» будут соответствовать атрибуты «название товара», «цена товара», «производитель товара».

Кроме того, на диаграмме модели процессов могут присутствовать различные стрелки, изображающие одни и те же данные, но на разных этапах обработки (например, оптовая цена товара, розничная цена, уцененная стоимость и т. п.).

Проектировщику необходимо учитывать эту неоднозначность информации, определяемую названием стрелок, чтобы обеспечить компактность и непротиворечивость хранения данных.

Атрибут представляет собой характеристику объектов. Сущность должна обладать **первичным ключом (Primary Key, PK)** — атрибутом или комбинацией атрибутов, чьи значения однозначно определяют каждый экземпляр сущности. В дополнение к собственным атрибутам сущность-потомок может наследовать атрибуты через отношения или отношения категоризации от сущности-родителя. Через отношения могут передаваться только атрибуты первичных ключей, которые в сущности-потомке становятся **внешним ключом (Foreign Key, FK)**.

Каждый атрибут идентифицируется уникальным именем и выражается грамматическим оборотом существительного. Каждый атрибут внутри блока сущности занимает одну строку, атрибуты, определяющие первичный ключ, размещаются наверху списка и отделяются горизонтальной чертой (см. рис. 4.20).

Выбор ключевых атрибутов (возможны первичные, альтернативные и внешние ключи) необходимо начинать с тех сущностей, которые не являются ни сущностью-потомком, ни сущностью-категорией. При определении первичных ключей следует руководствоваться следующими принципами: миграция ключей от родительских сущностей к сущности-потомку является обязательной; нельзя в качестве первичных ключей использовать атрибуты, которые обращаются в ноль; сущности с составными ключами не могут быть разбиты на несколько сущностей с более простыми ключами.

Если первичный ключ сущности-потомка содержит все атрибуты внешнего ключа, то сущность-потомок называется **зависимой** от идентификатора относительно родительской сущности.

Отношение в этом случае называется **идентифицирующим**, рисуется сплошной линией, а сущность-потомок — с закругленными углами.

Если какие-либо атрибуты внешнего ключа не принадлежат первичному ключу сущности-потомка, то сущность-потомок является **независимой** от идентификатора относительно родительской сущности. Отношение называется **неидентифицирующим**, рисуется пунктирной линией, а сущность-потомок рисуется с прямыми углами.

Отношение родитель—потомок — это связь между сущностями, при которой одна сущность, называемая родительской, и может быть связана с произвольным (в том числе и нулевым) количеством экземпляров второй сущности, называемой сущностью-потомком. Каждый экземпляр сущности-потомка связан в точности с одним экземпляром сущности-родителя. Таким образом, экземпляр сущности потомка может существовать только при наличии сущности-родителя.

При установлении зависимости отношений между двумя сущностями отношение должно быть проверено в обоих направлениях. Это делается посредством определения мощности на каждом конце отношений. Для правильного определения мощности необходимо предположить существование экземпляра одной сущности, а потом уже определять, сколько экземпляров второй сущности может быть связано с первой. Затем следует повторить анализ, поменяв сущности ролями.

Мощность отношения определяет, какое количество экземпляров сущности-потомков может существовать для каждого экземпляра сущности-родителя.

Существует четыре типа мощности отношения:

- каждый экземпляр сущности-родителя может иметь **ноль, одну и более** связанных с ним экземпляров сущностей-потомков;
- каждый экземпляр сущности-родителя может иметь **не менее одного** связанного с ним экземпляра сущности-потомка (**P**);
- каждый экземпляр сущности-родителя может иметь **не более одного** связанного с ним экземпляра сущности-потомка (**Z**);
- каждый экземпляр сущности-родителя связан с **некоторым фиксированным числом экземпляров** сущности-потомка (**N**).

Отношения связи отображаются линией с точкой на конце у сущности-потомка. Рядом с этой линией указывается мощность отношения (P, Z или N).

Отношению дается имя, выраженное грамматическим оборотом глагола, и помещается рядом с отношением. Имя каждого отношения между двумя данными сущностями должно быть уникально, но имена отношений в модели не обязаны быть уникальными. Имя отношения формируется всегда **с точки зрения (со стороны) сущности-родителя.** Следует обратить внимание на то, что отношение должно оставаться по-прежнему верным при формулировке от сущности-потомка к сущности-родителю, хотя на диаграмме оно не именуется.

Правила отношений

1. Экземпляр сущности-потомка всегда должен быть связан в точности с одним экземпляром сущности-родителя.

2. Экземпляр сущности-родителя может быть связан с любым числом (0 и более) экземпляров сущности-потомка, в зависимости от указанной мощности отношения.

3. В идентифицирующем отношении сущность-потомок всегда является зависимой от идентификатора сущностью.

4. Сущность может быть связана с любым количеством других сущностей, как в качестве потомка, так и в качестве родителя.

В методологии IDEF1X применяется также **отношение категоризации**, когда некоторые объекты являются категориями других существующих объектов, а следовательно, и соответствующие сущности в некотором смысле являются категориями других сущностей (рис. 4.21).

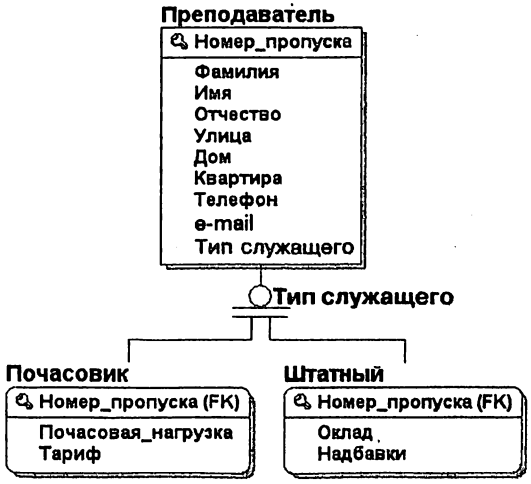


Рис. 4.21. Фрагмент ER-диаграммы с отношением полной категоризации

Существует понятие **отношения полной и неполной категоризации. Отношение полной категоризации** — это отношение между двумя или более сущностями, в котором каждый экземпляр одной сущности, называющейся общей сущностью, связан в точности с одним экземпляром одной и только одной из других сущностей, называемых **сущностями-категориями**. Каждый экземпляр общей сущности и связанный с ним экземпляр одной из сущностей-категорий изображают один и тот же предмет реального мира и поэтому обладают одним и тем же уникальным идентификатором.

Допускается существование неполного множества категорий. Например, если существует экземпляр общей сущности, не связанный ни с одним экземпляром сущности-категории, то такое отношение называется **отношением неполной категоризации**.

Сущности-категории, связанные с одной общей сущностью, всегда являются взаимоисключающими. То есть экземпляр общей сущности может соответствовать экземпляру только одной сущности-категории: служащий не может быть одновременно и штатным, и почасовым. При этом общая сущность и сущности-категории должны иметь **одинаковые ключевые атрибуты**.

В экземпляре общей сущности значение некоторого атрибута определяет, с какой из возможных сущностей-категорий он связан. Этот атрибут называется **дискриминатором отношения категоризации**. В нашем примере дискриминатором может быть атрибут **ТИП_СЛУЖАЩЕГО**. Имя атрибута общей сущности, которая используется в качестве дискриминатора, отображается рядом с кружком.

Дискриминатор отношения полной категоризации изображается в виде круга, подчеркнутого двумя линиями. Однократное подчеркивание означает неполноту множества категорий.

Отношение категоризации обозначается линией, ведущей от общей сущности к подчеркнутому кругу. Отдельные линии ведут из подчеркнутого круга к каждой из сущности-категории. Сущности-категории всегда зависимы от идентификаторов, а общая сущность всегда независима от них.

Отношение категоризации не именуется, но может звучать как «может быть».

Правила отношений категоризации:

1. Сущность-категория может иметь только одну общую сущность.
2. Сущность-категория может быть общей сущностью.
3. Атрибуты первичного ключа общей сущности и сущности-категории должны совпадать.

4. Все экземпляры сущности-категории имеют одно и то же значение дискриминатора.

При первоначальной разработке модели иногда бывает удобно устанавливать **неспецифические отношения** (многие-ко-многим) между двумя сущностями (рис. 4.22). Имя неспецифическому отношению дается в обоих направлениях.

При дальнейшей разработке неспецифические отношения заменяются на специфические путем введения третьей сущности, называемой сущностью-пересечением или ассоциативной сущностью (рис. 4.23).

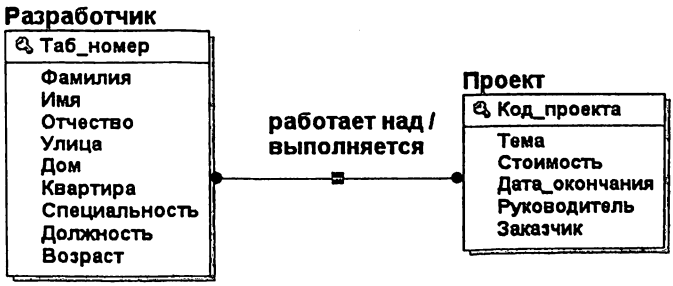


Рис. 4.22. Пример неспецифического отношения

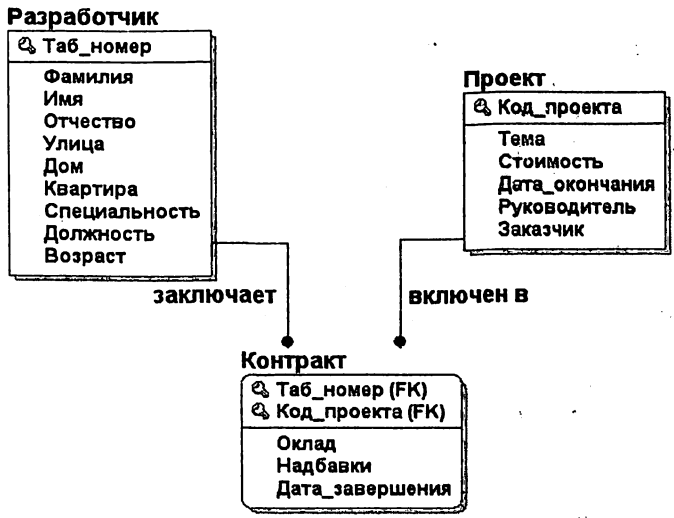


Рис. 4.23. Пример введения сущности-пересечения

4.6.2. Реализация методологии IDEF1X в пакете All Fusion ERwin Data Modeler (ERwin)

Модель данных обычно начинают создавать с логического уровня, который представляется сущностями и атрибутами. Логический уровень модели данных является универсальным, он не связан с какой-либо конкретной СУБД, и данные представляются так, как они выглядят в реальном мире, например «Товар», «Фамилия», «Адрес».

Физический уровень модели содержит информацию обо всех объектах базы данных (таблицах, столбцах, индексах, процедурах и т. п.) и зависит от выбранной СУБД. Пакет ERwin поддерживает практически все современные СУБД. Выбор конкретного СУБД выполняется командой **Database/Choose Database**. В диалоговом окне выбирается СУБД, тип данных и правила ссылочной целостности.

Для одного логического уровня можно построить несколько разных физических уровней для различных СУБД (Oracle, Informix, Sybase, Ingress, Access и т. д.). На основе физической модели ERwin может сгенерировать системный каталог СУБД или соответствующий SQL-скрипт, запустив который в среде соответствующей СУБД получают базу данных физического уровня, готовую для заполнения данными.

Пакет ERwin позволяет создавать модели трех типов: модель, имеющую только логический уровень (**Logical**); модель, имеющую только физический уровень (**Phisical**); модель, имеющую как логический, так и физический уровень (**Logical/Phisical**).

В ERwin различают три подуровня логического уровня модели данных, отличающихся по глубине представления информации о данных: диаграмма сущность—связь; модель данных, основанная на ключах; полная атрибутивная модель [25].

Диаграмма сущность—связь (Entity Relationship Diagram (ERD)) включает сущности и взаимосвязи, она не слишком детализирована, и в нее включаются основные сущности и связи между ними. ERD-диаграмма может включать связи «многие-ко-многим» и не включать описание ключей. Обычно этот тип диаграммы используется для презентаций и обсуждения структуры данных с заказчиком.

Модель данных, основанная на ключах (Key Based model (KB)), включает описание всех сущностей и первичных ключей, обеспечивая более полное представление структуры данных.

Полная атрибутивная модель (Fully Attributed model (FA)) представляет данные в третьей нормальной форме, включает все сущно-

сти, атрибуты и связи. Эта модель обеспечивает наиболее детальное представление структуры данных.

Построение логической модели данных предполагает определение сущностей, атрибутов и первичных ключей. Проектировщик должен таким образом определить хранящуюся информацию в конкретной сущности и в конкретном атрибуте, чтобы обеспечить полную информационную поддержку для выполнения всех функций, заложенных в информационную систему. Поэтому в основе построения ER-модели должна находиться функциональная иерархическая модель (IDEF0-модель), дуги в которой обозначают информацию, используемую в моделируемой системе.

Для построения ER-модели необходимо запустить пакет ERwin. В режиме создания новой модели в диалоге **Create Model** следует выбрать тип новой модели: логический, физический или логический/физический. Рекомендуется выбирать тип модели (**Logical/ Phisical**), так как для генерации базы данных потребуется физический тип модели.

Затем необходимо для создания логических моделей выбрать одну из двух международно признанных систем обозначений (нотации): **Integration Definition for Information Modeling (IDEF1X)** и **Information Engineering (IE)**. Переключение между нотациями можно сделать после выполнения команды **Model/Model Properties** в диалоге **Model Properties** на вкладке **Notations.**

Поскольку логическая модель имеет несколько уровней отображения диаграммы, то также надо выбрать требуемый уровень (уровень сущностей, уровень атрибутов и т. д.) с помощью кнопок на панели инструментов или с помощью контекстного меню, выбрав пункт **Display Label** и затем необходимый уровень отображения.

Для изучения панелей инструментов ERwin и принципов моделирования данных можно воспользоваться обучающей программой, которая содержит восемнадцать уроков и запускается командой **Help/Tutorial**.

Для внесения сущности в модель необходимо (убедившись предварительно, что вы находитесь на уровне логической модели) щелкнуть на иконке «сущность» на панели инструментов, затем щелкнуть на том месте диаграммы, где необходимо расположить новую сущность. Щелкнув правой клавишей мыши по сущности, вызывают контекстное меню, в котором выбирают пункт **Entity Properties**, и в появившемся диалоговом окне **Entities** определяется имя, описание и комментарии сущности (рис. 4.24).

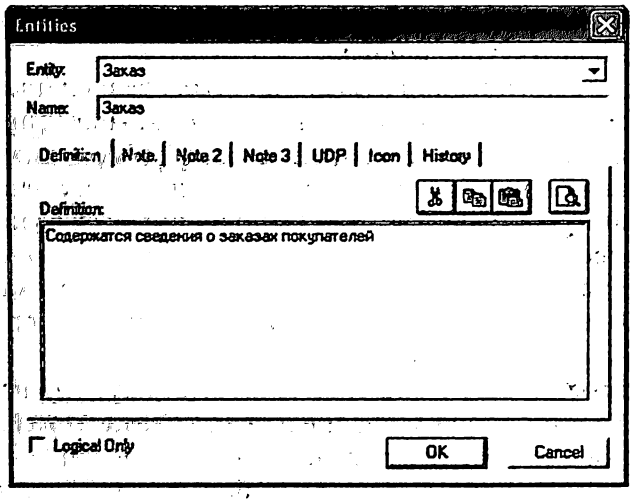


Рис. 4.24. Диалоговое окно Entities

Для описания атрибутов следует, щелкнув правой кнопкой по сущности, выбрать в появившемся меню пункт **Attributes**. Появляется диалоговое окно **Attributes** (рис. 4.25). Если щелкнуть по кнопке **New**, то в появившемся диалоге **New Attribute** можно указать домен, имя атрибута и имя соответствующей ему в физической модели колонки.

Для атрибутов первичного ключа во вкладке **General** диалога **Attributes** необходимо сделать пометку в окне выбора **Primary Key**.

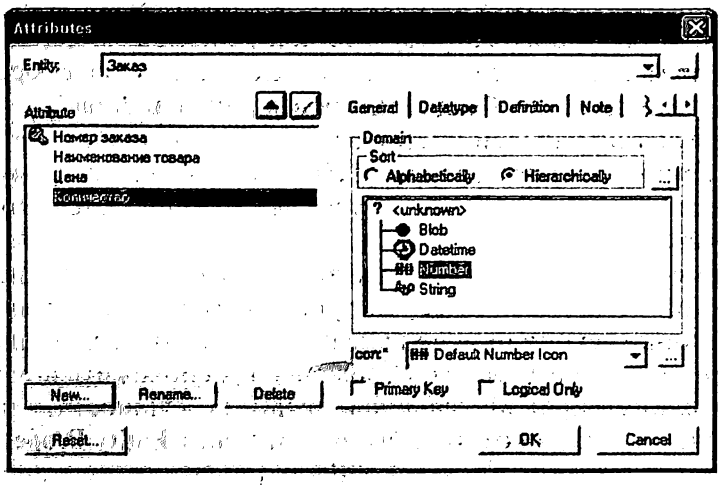


Рис. 4.25. Диалоговое окно Attributes

Вкладка **Definition** позволяет записывать определения отдельных атрибутов.

Для большей наглядности диаграммы каждый атрибут можно связать с иконкой. При помощи списка выбора **Icon** во вкладке **General** можно связать иконку с атрибутом.

Домен атрибута будет использоваться при определении типа колонки на уровне физической модели.

В пакете ERwin домен может быть определен только один раз и использоваться как в логической, так и в физической модели.

Домен может быть создан на основе другого домена и наследовать все свойства домена-прародителя. По умолчанию ERwin имеет четыре предопределенных домена: **String**, **Number**, **Blob**, **Datetime**. Создать домен можно во вкладке Domains окна **Model Explorer**. Домены позволяют облегчить работу с данными как разработчикам на этапе проектирования, так и администраторам на этапе эксплуатации системы.

Для создания новой связи следует (рис. 4.26):

• установить курсор на нужной кнопке (идентифицирующая или неидентифицирующая связь) в палитре инструментов и нажать левую кнопку мыши;

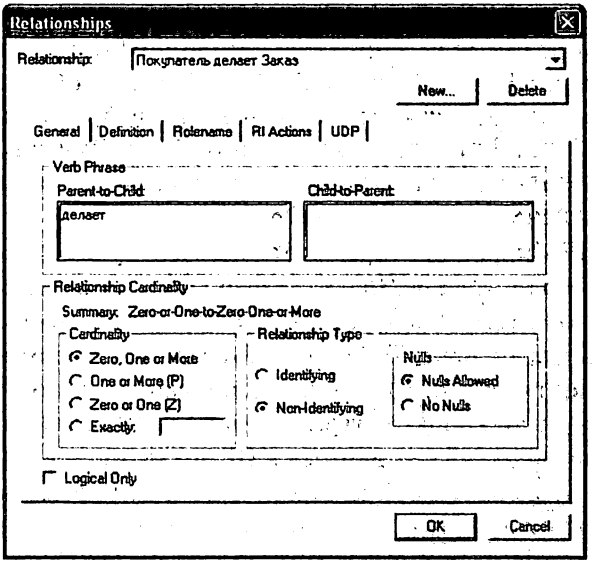


Рис. 4.26. Диалоговое окно для построения связей

• щелкнуть сначала по родительской, а затем по дочерней сущности.

В палитре инструментов кнопка соответствует идентифицирующей связи, кнопка — связи «многие-ко-многим» и кнопка соответствует неидентифицирующей связи.

Для редактирования свойств связи следует щелкнуть правой кнопкой мыши по связи и выбрать в контекстном меню пункт **Relationship Properties**. Во вкладке **General** появившегося диалога можно задать мощность, имя и тип связи.

4.7. Интеграция IDEF0- и IDEF1X-моделей и связывание объектов модели данных со стрелками и работами

После завершения разработки базы данных логического уровня ее необходимо средствами пакета BPwin связать с функциональной моделью проектируемой системы. Для успешного связывания моделей необходимо соответствие версий пакетов ERwin и BPwin. Связывание моделей гарантирует завершенность анализа и полную информационную поддержку для всех функциональных модулей системы и позволяет создавать спецификации на права доступа к данным для каждого процесса [25].

Для экспорта модели данных из ERwin в BPwin (способ экспорта-импорта через файлы формата **.EAX — .BPX**.) необходимо в ERwin открыть модель данных (рис. 4.27) и выбрать пункт меню **File/Export/BPwin**. В появившемся диалоге **Select BPwin Export File** необходимо выбрать каталог, вставить имя создаваемого файла экспорта с расширением ***.eax** и нажать **«OK»**.

Затем в BPwin нужно открыть **функциональную модель** (рис. 4.28), выбрать в меню пункт **File/Import/Erwin (EAX)**, а в диалоговом окне выбрать нужное имя файла (***.eax**) и нажать **«Открыть»** (рис. 4.29).

Появится диалог **Import Differences Preview**, в котором показывается протокол импорта (рис. 4.30). Для внесения данных в модель процессов следует щелкнуть по кнопке **Accept**. Кнопка **Cancel** отменяет импорт.

После внесения модели данных в BPwin можно связать сущности и атрибуты с дугами. Правой кнопкой мыши нужно щелкнуть по любой дуге (например, по дуге ***ценники***) функциональной модели (рис. 4.28) и выбрать в контекстном меню **Arrow Data**. Появляется вкладка **Arrow Data** диалога **Arrow Properties** (рис. 4.31).

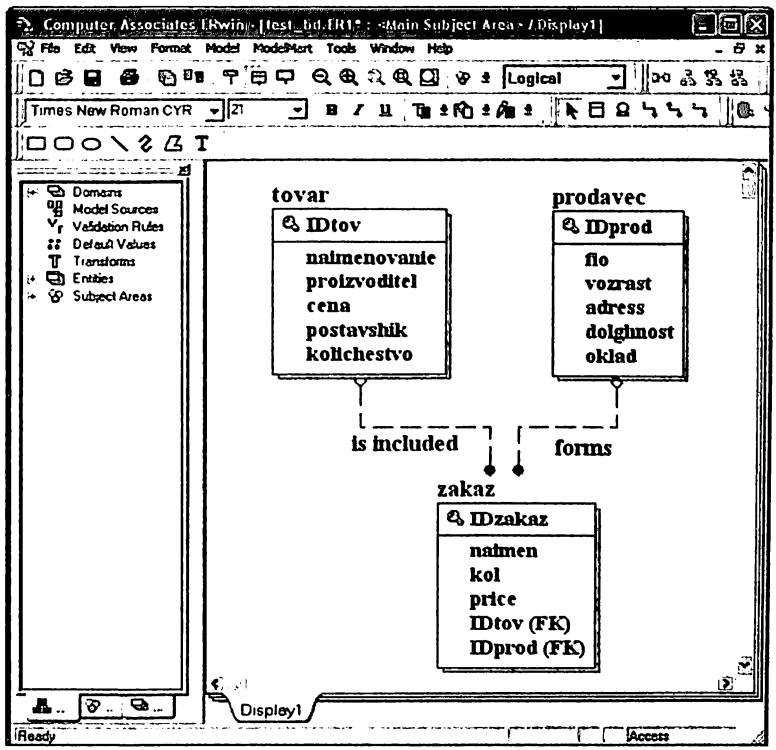


Рис. 4.27. Модель данных, открытая в ERwin

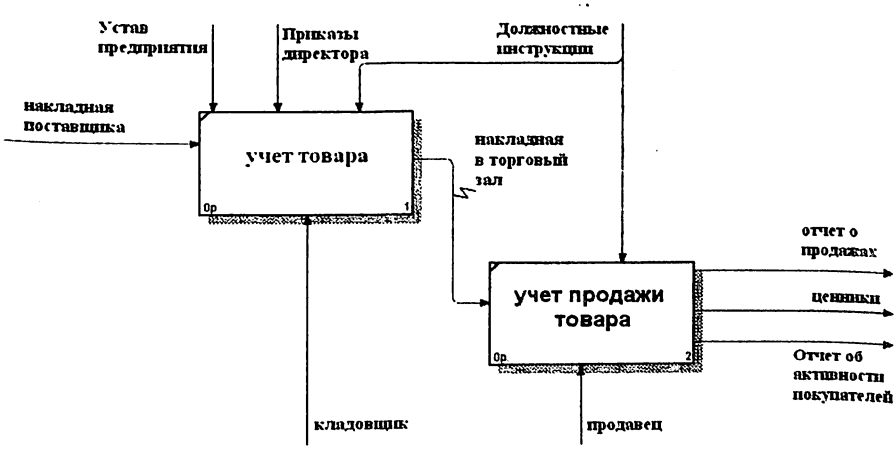


Рис. 4.28. Функциональная IDEF0-модель

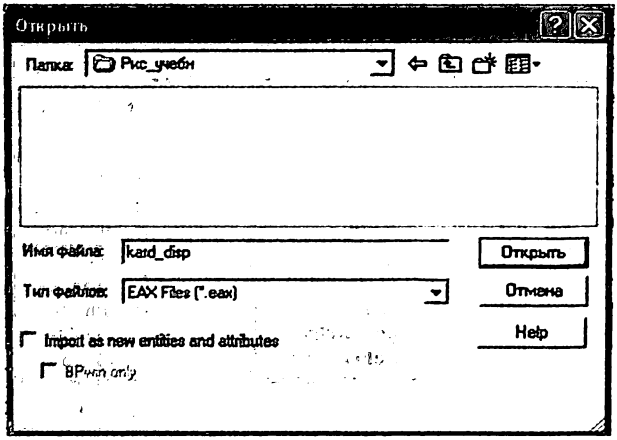


Рис. 4.29. Окно для импорта файла из Erwin

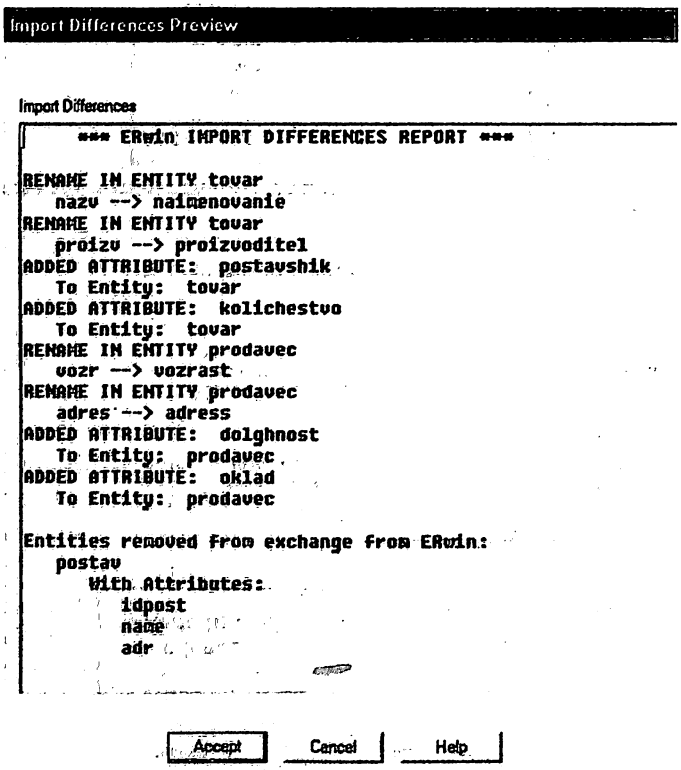


Рис. 4.30. Диалог Import Differences Preview

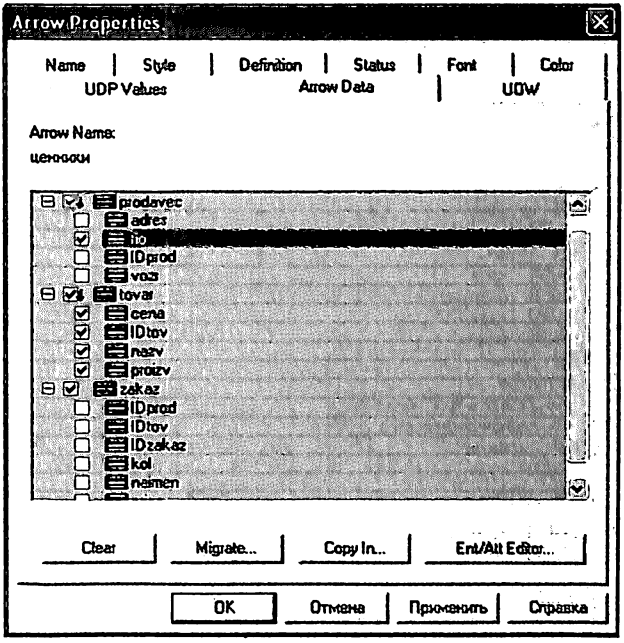
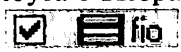


Рис. 4.31. Вкладка Arrow Data диалога Arrow Property для дуги ценники

Для связывания атрибута с выбранной дугой достаточно щелкнуть по иконкам выбора  в иерархическом списке соответствующих атрибутов. На рис. 4.31 они отмечены галочкой. При этом сущность автоматически связывается с дугой. Каждая дуга в функциональной модели может быть связана с несколькими атрибутами различных сущностей (см. рис. 4.35).

Следует также учесть воздействие функциональных блоков на данные, т. е. они определяют, какие данные для них являются входящими и исходящими. Для документирования такого воздействия необходимо щелкнуть правой клавишей мыши по блоку и в появившемся контекстном меню выбрать пункт меню **Data Usage Editor**.

В появившемся диалоге **Data Usage Editor** (рис. 4.32) в виде иерархического списка показываются все функциональные блоки модели (учет продаж, учет товара), дуги (ценники, накладная в торговый зал, накладная поставщика и др.), которые касаются блоков, сущности (**prodavec**, **tovar**, **zakaz**) и атрибуты (**fio**, **cena**, **nazv** и др.), которые были связаны со дугами. Для задания ассоциации достаточно щелкнуть по окну  в иерархическом списке.

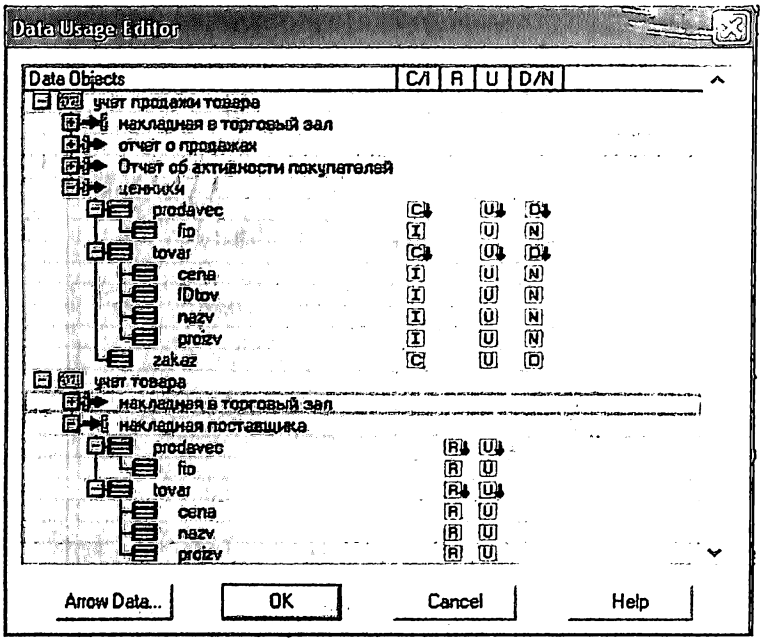


Рис. 4.32. Диалог BPwin Data Usage Editor

Для сущностей задается ассоциация **CRUD** (Create, Read, Update, Delete), для атрибутов — **IRUN** (Insert, Read, Update, Nullify). Ассоциации CRUD и IRUN — это правила использования сущностей и атрибутов работами, т. е. то, что могут делать работы с входящими или исходящими данными. Данные не могут использоваться работами произвольно. Стрелки входа представляют данные, которые работа преобразует в выход или потребляет. Такие данные могут быть обновлены (Update) или прочитаны (Read), но не могут быть созданы (Create, Insert) или удалены (Delete, Nullify). Данные, связанные со стрелками выхода, могут быть обновлены (если им соответствуют данные стрелок входа), удалены (Delete, Nullify) или созданы (Create, Insert). Для стрелок управления и механизма ассоциации не устанавливаются.

Результат связывания объектов модели процессов можно отобразить в отчете **Data Usage Report** (**Tools/Reports/Data Usage Report**) (рис. 4.33).

В окне **Standarts Reports** можно установить пять видов отчетов, указать их формат (в группе **Report Format**) и задать состав полей и их порядок следования в отчете.

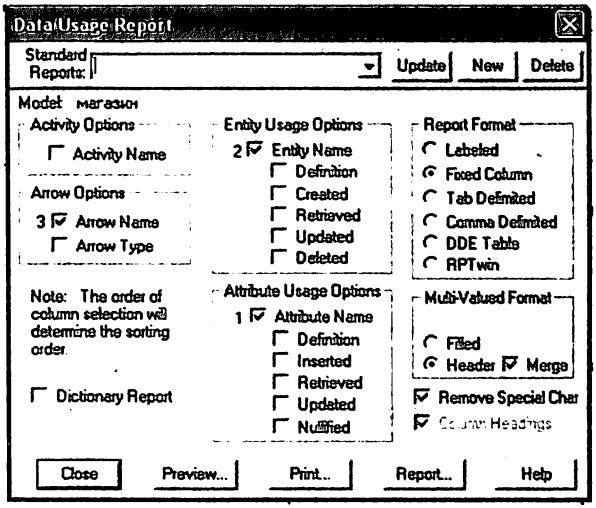


Рис. 4.33. Диалоговое окно для формирования отчетов

На рис. 4.33 установлены опции отчета, показанного на рис. 4.34 (вид отчета — **Activity Entity Attribute Association**). Этот вариант отчета позволяет определить, какие атрибуты сущностей задействованы в дугах. На рис. 4.35 приведены другие установки опций в окне **Data Usage Report**.

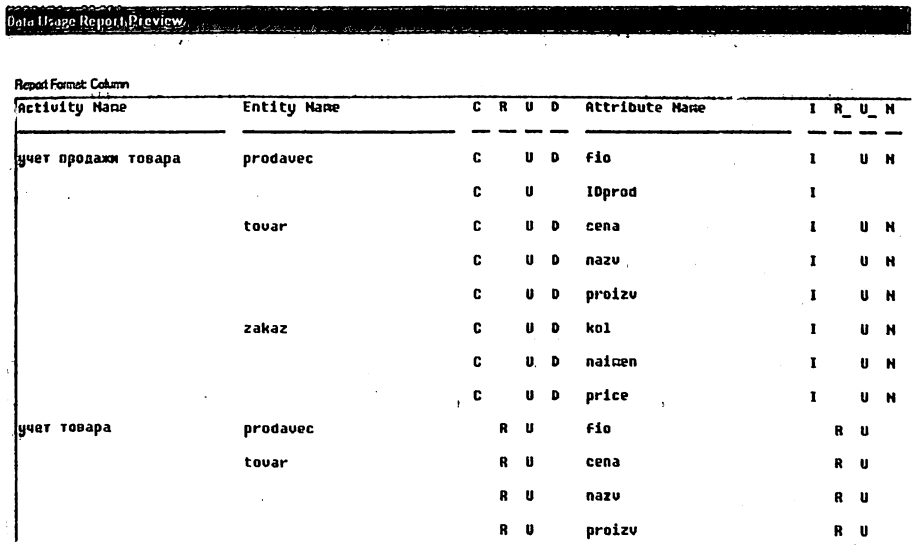


Рис. 4.34. Отчет о связях функциональных блоков с сущностями и атрибутами

Report Format: Column

Attribute Name	Entity Name	Arrow Name
adres	prodavec	prodavec
		кладовщик
		накладная продавца
cena	tovar	накладная поставщика
		накладная продавца
		отчет об активности покупателей
		ценники
fio	prodavec	prodavec
		кладовщик
		накладная поставщика
		накладная продавца
		отчет о продажах
		отчет об активности покупателей
		ценники
kol	zakaz	Отчет об активности покупателей

Рис. 4.35. Отчет о связях стрелок с сущностями и атрибутами

Анализ отчетов (см. рис. 4.35) и диалоговых окон (см. рис. 4.23) позволяет определить те атрибуты, которые **не используются** во входных и выходных документах, а следовательно, являются лишними, если в них не планируется хранение каких-либо вычисляемых при эксплуатации данных.

Если в процессе связывания дуг с объектами модели данных окажется, что каких-либо сущностей или атрибутов не хватает, их можно добавить прямо в BPwin, а затем экспортировать в ERwin.

Для того чтобы отредактировать сущности в BPwin, необходимо выполнить команду **Dictionary/Entity**. В появившемся диалоговом окне Entity создается новая сущность. Редактирование атрибутов созданных сущностей выполняется в диалоговом окне **Attribute Dictionary**, которое вызывается командой **Dictionary/Entity/Attribute** (рис. 4.36).

Новые сущности и атрибуты могут быть использованы для связывания со стрелками сразу же после их создания, т. е. до их экспорта в

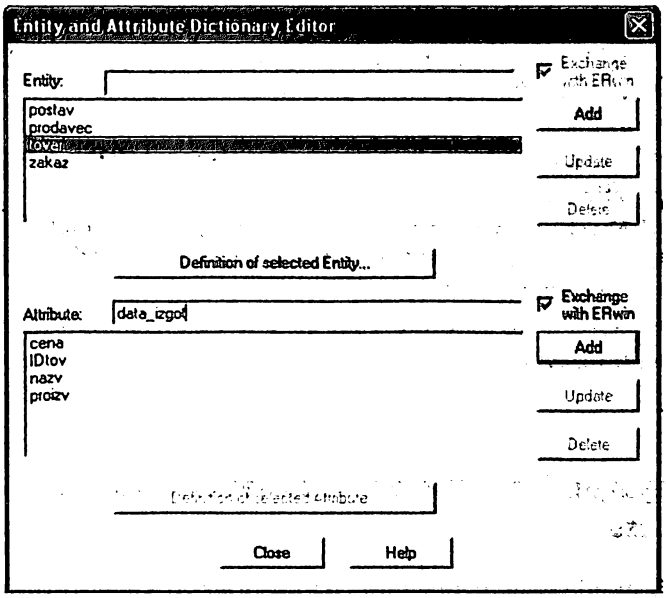


Рис. 4.36. Пример добавления атрибута data_izgot в сущность tovar

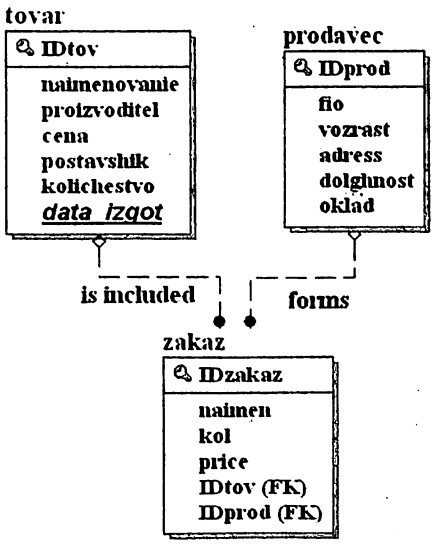


Рис. 4.37. Модифицированная в BPwin ER-диаграмма

ERwin. После создания сущностей или атрибутов необходимо сохранить данные и выйти из словаря.

Если в модель данных были внесены изменения, то для ее экспорта из BPwin следует выбрать команду **File/Export/ERwin(BPX)** и указать имя нового файла, в который будет «выгружена» информация об измененной информационной модели.

В ERwin следует выбрать меню **File/Import/BPwin** и в диалоге **ERwin Open File** указать файл **BPX**, в который была «выгружена» информация о модели. Возникает диалог **ERwin/BPwin Import**, в котором отображаются сущности и атрибуты, имеющиеся в **BPX**-файле, но отсутствующие в модели ERwin.

После щелчка по кнопке **Import** запускается процесс импорта BPX-файла и получаем сущность **tovar** с новым атрибутом **data_izgot** (рис. 4.37).

4.8. Генерация базы данных физического уровня в среде СУБД Access

Физический уровень представления модели зависит от выбранного сервера. Пакет ERwin поддерживает более 20 реляционных и нереляционных БД. При генерации схемы физической базы данных ERwin автоматически создает индекс на основе первичного ключа каждой таблицы, а также на основе всех альтернативных ключей и внешних ключей, поскольку эти колонки наиболее часто используются для поиска данных. Имена таблиц и колонок будут сгенерированы по умолчанию на основе имен сущностей и атрибутов логической модели.

Подготовка к генерации базы данных физического уровня начинается с создания пустой БД в среде той СУБД, куда планируется генерировать ER-диаграмму. Для этого надо запустить СУБД Access, выполнить команду на создание новой БД, присвоить ей имя и сохранить (рис. 4.38).

Затем открываем ER-диаграмму в среде ERwin и с помощью списка выбора в стандартной панели инструментов производим переключение между логической и физической моделью (рис. 4.39). При переключении, если физической модели еще не существует, она будет создана автоматически.

Теперь необходимо выбрать СУБД, в которой будем производить генерацию БД физического уровня. Для этого следует выполнить команду **DATABASE/Choose database**, в появившемся диалоговом окне (рис. 4.40) выбрать интересующую СУБД Access и щелкнуть по кнопке <OK>.

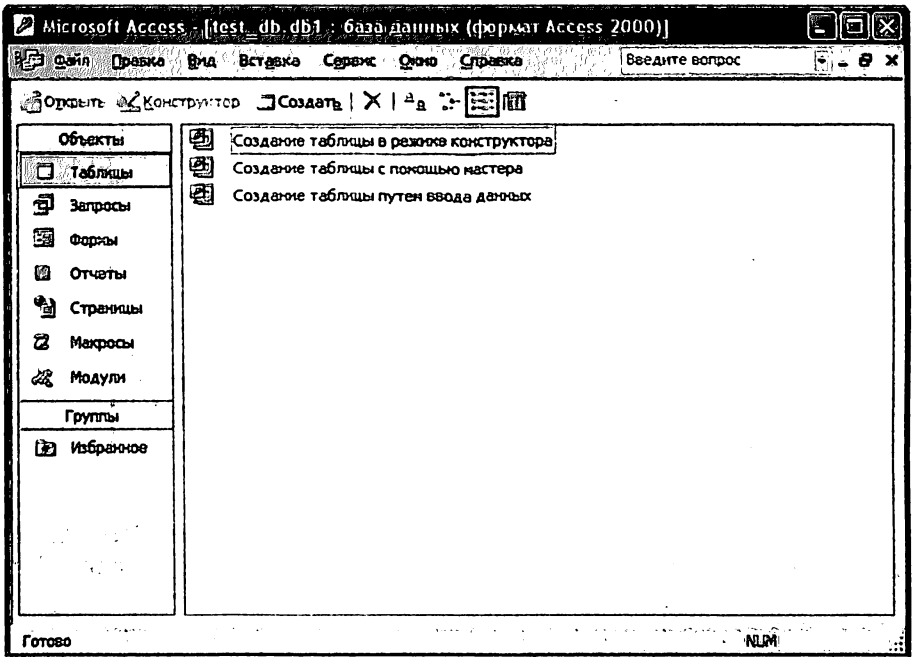


Рис. 4.38. Пустая БД с именем test_db в СУБД Access

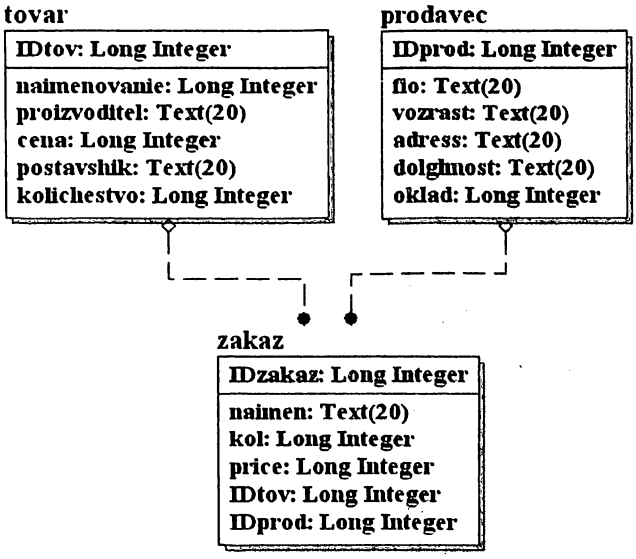


Рис. 4.39. Физическая модель БД в ERwin

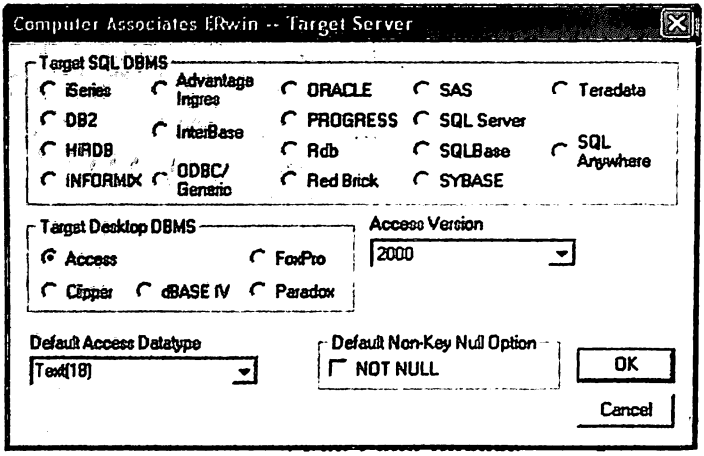


Рис. 4.40. Диалог выбора СУБД (сервера)

Для установления соединения БД из ERwin с целевой СУБД Access необходимо выполнить команду **DATABASE/Database connection.** В появившемся диалоговом окне (рис. 4.41) необходимо указать путь к БД в СУБД Access, вписать имя **admin** и нажать кнопку **Connect**.

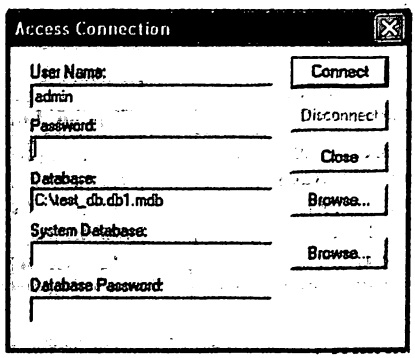


Рис. 4.41. Диалог присоединения к СУБД Access

Для генерации БД физического уровня в среде СУБД Access необходимо выполнить команду **TOOLS/Forward Engineering/Schema Generation.** В итоге получают диалог генерации схемы БД (рис. 4.42), который имеет три закладки.

Options. Служит для задания опций генерации объектов базы данных — триггеров, таблиц, представлений, колонок, индексов и т. д. Для задания опций генерации какого-либо объекта следует выбрать

объект в левом списке закладки, после чего включить соответствующую опцию в правом списке.

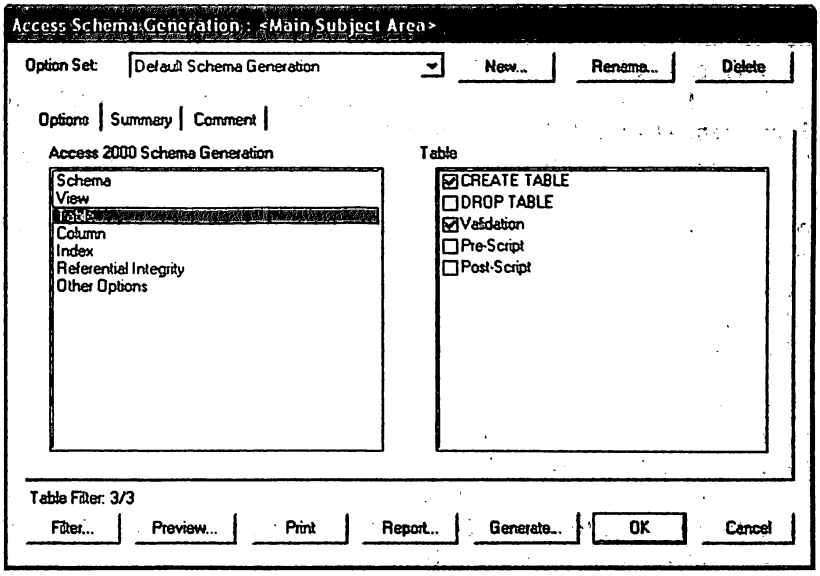


Рис. 4.42. Диалог генерации схемы БД

Во вкладке **Summary** отображаются все опции, заданные во вкладке **Options**. Список опций в **Summary** можно редактировать так же, как и в **Options**.

Comment. Позволяет внести комментарий для каждого набора опций.

Каждый набор опций может быть именован (окно Option Set, кнопки New, Rename и Delete) и использован многократно.

Кнопка **Preview** вызывает диалог **Schema Generation Preview**, в котором отображается SQL-скрипт, создаваемый ERwin для генерации системного каталога СУБД (рис. 4.43).

Кнопка **Print** диалога предназначена для вывода на печать создаваемого ERwin SQL-скрипта.

Кнопка **Report** сохраняет тот же скрипт в ERS- или SQL-текстовом файле. Эти команды можно в дальнейшем редактировать любым текстовым редактором и выполнять при помощи соответствующей утилиты сервера.

Нажатие на кнопку **Generate** приведет к запуску процесса генерации схемы. Возникает диалог связи с базой данных, устанавливается

```
CREATE TABLE "prodavec"
Set ERwinTableDef = ERwinDatabase.CreateTableDef("prodavec")
Set ERwinField = ERwinTableDef.CreateField("IDprod", DB_LONG)
ERwinTableDef.Fields.Append ERwinField
Set ERwinField = ERwinTableDef.CreateField("fio", DB_TEXT, 20)
ERwinTableDef.Fields.Append ERwinField
Set ERwinField = ERwinTableDef.CreateField("vozrast", DB_TEXT, 20)
ERwinTableDef.Fields.Append ERwinField
Set ERwinField = ERwinTableDef.CreateField("adress", DB_TEXT, 20)
ERwinTableDef.Fields.Append ERwinField
Set ERwinField = ERwinTableDef.CreateField("dolghnost", DB_TEXT, 20)
ERwinTableDef.Fields.Append ERwinField
Set ERwinField = ERwinTableDef.CreateField("oklad", DB_LONG)
ERwinTableDef.Fields.Append ERwinField
ERwinDatabase.TableDefs.Append ERwinTableDef

' CREATE TABLE "tovar"
Set ERwinTableDef = ERwinDatabase.CreateTableDef("tovar")
Set ERwinField = ERwinTableDef.CreateField("IDtov", DB_LONG)
ERwinTableDef.Fields.Append ERwinField
Set ERwinField = ERwinTableDef.CreateField("naimenovanie", DB_LONG)
ERwinTableDef.Fields.Append ERwinField
Set ERwinField = ERwinTableDef.CreateField("cena", DB_LONG)
ERwinTableDef.Fields.Append ERwinField
Set ERwinField = ERwinTableDef.CreateField("proizvoditel", DB_TEXT, 20)
ERwinTableDef.Fields.Append ERwinField
Set ERwinField = ERwinTableDef.CreateField("postavshik", DB_TEXT, 20)
ERwinTableDef.Fields.Append ERwinField
Set ERwinField = ERwinTableDef.CreateField("kolichestvo", DB_LONG)
ERwinTableDef.Fields.Append ERwinField
ERwinDatabase.TableDefs.Append ERwinTableDef
```

Рис. 4.43. Программа генерации таблиц БД (SQL-скрипты)

сеанс связи с сервером базы данных (СУБД Access), и начинает выполняться SQL-скрипт. При этом возникает диалог **Generate Database Schema** (рис. 4.44).

По умолчанию в диалоге **Generate Database Schema** включена опция **Stop If Failure**. Это означает, что при первой же ошибке выполнение SQL-скрипта прекращается. Щелкнув по кнопке **Continue**, можно продолжить выполнение. Кнопка **Abort** прерывает выполнение. При выключенной опции **Stop If Failure** SQL-скрипт будет выполняться, несмотря на встречающиеся ошибки.

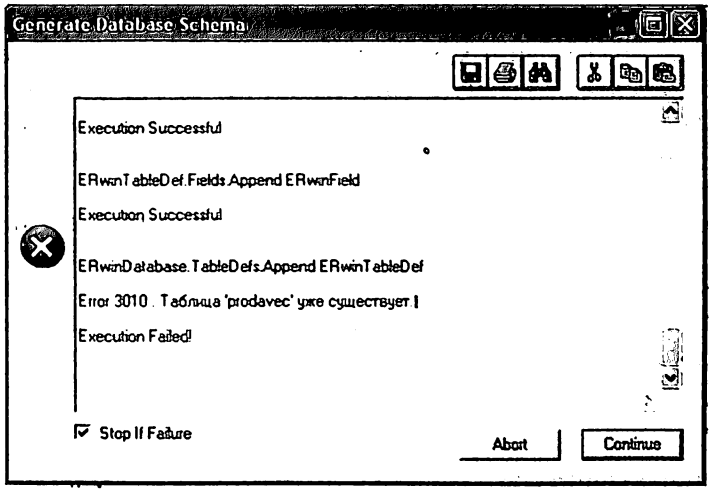


Рис. 4.44. Диалог Generate Database Schema

Для выполнения обратного проектирования следует выбрать пункт меню **Tools/Reverse Engineer**. После выполнения SQL-скрипта (см. рис. 4.43) в среде СУБД Access создается БД физического уровня (рис. 4.45).

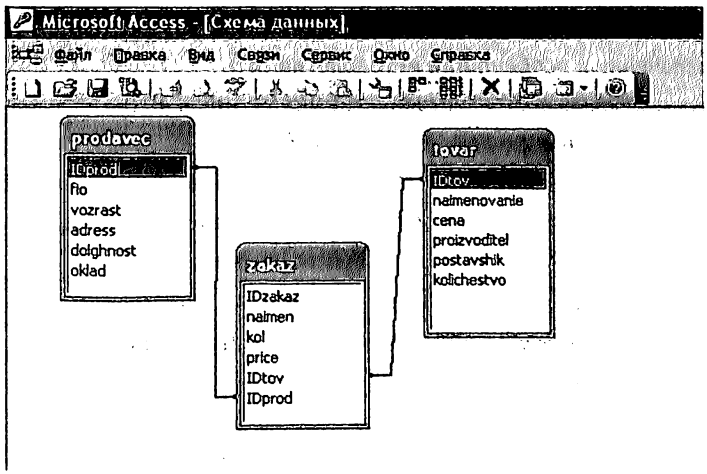


Рис. 4.45. Структура БД физического уровня в СУБД Access

Таким образом, на основе физической модели ERwin можно сгенерировать системный каталог СУБД или соответствующий SQL-скрипт. Этот процесс называется прямым проектированием (**Forward**

Engineering). Тем самым достигается масштабируемость — создав одну логическую модель данных, можно сгенерировать физические модели под любую поддерживаемую пакетом ERwin СУБД. С другой стороны, ERwin способен по содержимому системного каталога или SQL-скрипту воссоздать физическую и логическую модель данных (**Reverse Engineering**).

На основе полученной логической модели данных можно сгенерировать физическую модель для другой СУБД и затем сгенерировать ее системный каталог. Следовательно, ERwin позволяет решить задачу по переносу структуры данных с одного сервера на другой. Например, можно перенести структуру данных с **Oracle** на **Informix** (или наоборот).

Контрольные вопросы

1. Назначение и содержание диаграммы Swim Lane.
2. Какие словари необходимо создать для построения организационной диаграммы и Swim Lane diagram?
3. Назначение и основные достоинства методологии IDEF0.
4. Каким образом реализован структурный анализ в методологии IDEF0?
5. Назначение и особенности методологии IDEF3.
6. Назначение и особенности методологии DFD.
7. Сравнительный анализ методологий IDEF0, IDEF3, DFD.
8. Нетрадиционный синтаксис методологии DFD.
9. Назначение и содержание стоимостного анализа ABC в пакете BPwin.
10. В каких ситуациях используется отношение категоризации?
11. Как выбираются первичные и внешние ключи?
12. Типы связей в ER-диаграмме и их характеристики.
13. Для чего выполняется интеграция IDEF0- и IDEF1X-моделей в среде пакета BPwin?

Глава 5
СРАВНИТЕЛЬНЫЙ АНАЛИЗ СТАНДАРТОВ НА ОРГАНИЗАЦИЮ ЖИЗНЕННОГО ЦИКЛА СОЗДАНИЯ И ИСПОЛЬЗОВАНИЯ ИС. ПРОФИЛИ СТАНДАРТОВ

5.1. Общие сведения о стандартах на организацию ЖЦ ИС и профилях стандартов

Все существующие стандарты в области информационных технологий можно сгруппировать по трем следующим критериям: по предмету стандартизации, по утверждающей организации и по методическому источнику [18]. В свою очередь, стандарты, сгруппированные по предмету стандартизации, можно разделить на две группы: функциональные стандарты (стандарты на языки программирования, интерфейсы и протоколы) и стандарты на организацию жизненного цикла создания и использования ИС и программного обеспечения.

В стандартах жизненного цикла обобщен опыт и результаты исследований нескольких поколений специалистов и рекомендуются современные методы и процессы создания и развития прикладных систем. Стандарты ЖЦ могут использоваться как непосредственные директивные, руководящие или рекомендательные документы, а также как организационная база при автоматизации технологических этапов или процессов. Стандартизация процессов ЖЦ ИС отражается не только на их технико-экономических показателях, но и в первую очередь на качестве и надежности создаваемых ИС [22].

ЖЦ в стандартах отражается набором этапов, частных работ и операций, выполняемых в определенной последовательности и взаимосвязи на всех стадиях от подготовки технического задания до завершения испытаний версий и окончания эксплуатации ИС.

Стандарты включают описания исходной информации, способов и методов выполнения операций и работ, устанавливают требования

к результатам и правилам их контроля, а также регламентируют содержание технологических и эксплуатационных документов на ИС. Кроме этого, они определяют организационную структуру, обеспечивают распределение и планирование работ, а также контроль над ходом разработки.

Например, в первом стандарте жизненного цикла **DOD-STD-2167A**, который был разработан в 1985 г. для проектирования программных средств систем военного назначения по заказам Министерства обороны США, регламентировано восемь этапов ЖЦ и около 250 типов обязательных требований к процессам и объектам проектирования на всех этапах разработки.

В общих требованиях представлены планирование и управление разработкой ИС, правила взаимодействия с субподрядчиками и испытателями, а также документирование результатов. Изложены общие требования к технологии и средствам автоматизации разработки программ, к структуре и организации комплекса программ и поддерживающей его базы данных.

Специальный раздел посвящен требованиям к квалификационным испытаниям, к средствам и организации тестирования программ на всех этапах разработки. Изложены требования к организации, выполнению и документированию оценок качества программной продукции.

В России создание и испытание ИС регламентированы **стандартами серии 34**: ГОСТ 34.601—90 (Стадии создания АС), ГОСТ 34.602—89 (ТЗ на создание АС), ГОСТ 34.603—92 (Виды испытаний АС), РД 50-34.698—90 (Требования к содержанию документов). Однако создание, сопровождение и развитие ИС в этих стандартах отражено слабо, отдельные их положения устарели с точки зрения построения и обработки данных.

Проектирование и информационная поддержка организации разработки ИС, включая программное обеспечение и базы данных, традиционно поддерживается многими стандартами и фирменными методиками. Поэтому для каждого серьезного проекта требуется тщательный подбор и правильное применение базовых стандартов и нормативных документов.

Правильно выбрать базовые стандарты из нескольких сотен стандартов, действующих в сфере информационных технологий, достаточно сложно и ответственно. Дело в том, что степень адаптации стандартов к создаваемым ИС обычно недостаточна и не покрывает полностью потребности в стандартизации объектов и процессов соз-

даваемых систем. Кроме этого, надо иметь в виду предпочтения в этом вопросе заказчика, самих разработчиков и реальное состояние проекта (новая разработка, адаптация типового проекта или модификация действующей информационной системы).

Для сокращения стоимости и обеспечения качества создаваемой ИС выбранный стандарт ЖЦ следует адаптировать для конкретного проекта системы. Должны быть определены характеристики окружения проекта, которые могут воздействовать на адаптацию. Этими характеристиками могут быть: функции ЖЦ ИС; требования к системе и программному обеспечению; организационные основы коллективов специалистов, процедуры и стратегии их работы; размер, критичность и типы систем; число задействованного персонала и сторон-участников [22].

В процессе адаптации стандарта ЖЦ должны быть включены особенности пользователей, поддерживающего персонала, потенциальных покупателей. Все решения по адаптации стандарта должны быть задокументированы вместе с обоснованием их целесообразности и поддержаны комплексом наиболее эффективных инструментальных средств.

Одним из наиболее распространенных способов адаптации стандартов является разработка и применение **профилей стандартов ЖЦ**, под которыми понимают совокупность нескольких (или подмножество одного) базовых стандартов и других нормативных документов с четко определенными и гармонизированными подмножествами обязательных и факультативных возможностей, предназначенная для реализации заданной функции или группы функций [23].

При этом профиль не может противоречить использованным в нем базовым стандартам и нормативным документам. На базе одной и той же совокупности базовых стандартов могут формироваться различные профили для различных проектов ИС.

Профиль формируется исходя из функциональных характеристик объекта стандартизации. В профиле выделяются и устанавливаются допустимые возможности и значения параметров каждого базового стандарта и/или нормативного документа, входящего в профиль.

Поэтому представляется полезным провести сравнение по основным характеристикам **трех наиболее популярных стандартов**, которые при этом максимально отличаются друг от друга назначением применения, содержанием и степенью адаптивности к создаваемым проектам. Первый — это стандарт **ISO/IEC 12207:1995-08-01** на организацию ЖЦ продуктов программного обеспечения, который имеет мак-

симальную степень адаптивности и характеризуется как инструкция участникам разработки, содержащая всю информацию только о том, «что надо сделать». Второй стандарт, выбранный для сравнения, — **отечественный комплекс российских стандартов ГОСТ 34**, который считается достаточно устаревшим, но полезным с тех позиций, что в нем детально прописано, «как надо делать», и он имеет высокую степень адаптивности. И наконец, в качестве третьего стандарта выбрана **методология Oracle CDM** по разработке прикладных информационных систем под заказ. Она подробнейшим образом описывает действия участников разработки на уровне этапов, процессов и задач, но степень адаптивности минимальна: используется только для программного инструментария фирмы Oracle и не допускает никаких изменений в содержании и составе этапов, процессов и задач. Методология Oracle CDM представляет собой конкретный материал, детализированный до уровня заготовок проектных документов, которые рассчитаны на прямое использование в проектах ИС.

Стандарты ГОСТ 34 и **CDM** в первую очередь ориентированы на действия по созданию и поддержке систем, а **ISO 12207** — на приобретение и эксплуатацию систем, при этом разработка является процессом, логически вытекающим из приобретения.

5.2. Методология Oracle CDM

Методология Oracle CDM возникла как развитие давно разработанной версии Oracle CASE-Method (ныне Oracle Designer 10g). Она самым тесным образом опирается на использование инструментария Oracle, хотя имеются утверждения о возможности приспособления CDM к проектам, в которых используется другой инструментальный комплекс.

Общая структура ЖЦ формируется из определенных этапов проекта и процессов, каждый из которых выполняется в течение нескольких этапов, т. е. модель ЖЦ имеет двумерную структуру.

Этапы модели ЖЦ CDM:

1) «определение требования» (или стратегия);

2) «анализ» (формулирование детальных требований к системе);

3) «проектирование» (преобразование требований в детальные спецификации системы);

4) «реализация» (написание и тестирование приложений);

5) «внедрение» (установка новой ИС, подготовка к началу эксплуатации);

6) «эксплуатация» (поддержка и слежение за приложением, планирование будущих функциональных решений).

Процессы модели ЖЦ CDM: определение производственных требований, исследование существующих систем, определение технической архитектуры, проектирование и построение БД, проектирование и реализация модулей, конвертирование данных, документирование, тестирование, обучение, переход к новой системе, поддержка и сопровождение.

Каждый процесс декомпозируется на последовательность задач, которые размещаются на диаграммах в хронологическом порядке их выполнения. При этом задачи различных процессов взаимосвязаны с помощью явно указанных ссылок.

Методология CDM наиболее сильно связана с методикой Oracle PJM по организации управления проектом.

Степень адаптивности CDM ограничивается тремя моделями ЖЦ: классическая (Classic) — предусмотрены все работы/задачи и этапы; быстрая разработка (Fast Track) — присутствует три этапа и все процессы, еще в большей степени ориентирована на использование инструментов моделирования и программирования Oracle; облегченный подход (Lite) — содержит только два этапа, отсутствуют два процесса (исследование существующих систем и конвертирование данных), рекомендуется в случае малых проектов и возможности быстрого прототипирования приложения.

Классическая модель ЖЦ относится к типу каскадной, а Fast Track и Lite — к моделям ЖЦ быстрого прототипирования.

Методика CDM не предусматривает удаление задач или включение дополнительных задач, а также изменение последовательности выполнения задач, особенно по ходу процесса проектирования.

5.3. Международный стандарт ISO/IEC 12207:1995-08-01. Процессы ЖЦ программного обеспечения

Международный стандарт **ISO/IEC 12207 (ГОСТ Р ИСО/МЭК 12207: 1995)** является базовым стандартом процессов ЖЦ ПО, ориентированным на различные виды ПО и типы проектов ИС. Стандарт определяет стратегию и общий порядок в создании и эксплуатации

ИС, он охватывает все этапы от концептуализации идей до снятия ИС с эксплуатации.

В отличие от Oracle CDM стандарт ISO 12207 равносильно ориентирован на организацию действий как поставщика (разработчика), так и покупателя (пользователя). ***Общая структура*** в стандарте базируется на трех крупных компонентах, которые, в свою очередь, состоят из процессов (табл. 5.1): основные процессы ЖЦ ПС (5 процессов); поддержка ЖЦ ПС (8 процессов) и организация ЖЦ ПС (4 процесса). По сравнению с CDM процессы в стандарте ISO 12207 являются более крупными и обобщенными, можно сказать, что один такой процесс сравним со всеми процессами CDM, вместе взятыми.

Таблица 5.1. **Структура стандарта ГОСТ Р ИСО/МЭК 12207: 1995**

5. Основные процессы ЖЦ ПС	6. Поддержка ЖЦ ПС	7. Организация ЖЦ ПС
5.1. Приобретение	6.1. Документирование	7.1. Управление
5.2. Процесс поставки	6.2. Конфигурационное управление	7.2. Инфраструктура
5.3. Разработка	6.3. Обеспечение качества	7.3. Совершенствование
5.4. Эксплуатация	6.4. Верификация	7.4. Обучение
5.5. Сопровождение	6.5. Валидация	
	6.6. Управление проектом	
	6.7. Ревизия отчетов	
	6.8. Решение задач (устранение дефектов)	

Каждый **процесс** разделен на набор **действий** (работ), каждое действие — на **набор задач**. Например, **процесс поставки** включает работы: инициация, подготовка предложений, заключение контракта, планирование, исполнение и контроль, проверка и оценка, постановка и завершение процесса. А работа **«заключение контракта»** содержит две задачи: поставщик обязан согласовать и внести в контракт пункт о передаче покупателю программного продукта или сервиса ПО; поставщик может потребовать изменить контракт, действуя в рамках механизма управления изменениями.

Важным отличием для ISO 12207 является то, что каждый процесс, действие или задача инициируется и выполняется другим процессом по мере необходимости. Причем заранее не устанавливается

жестко определенных последовательностей, естественно должна сохраняться логика связей по исходным данным задач.

Ядром стандарта являются пять основных процессов ЖЦ ПС.

5.1. Процесс приобретения определяет действия предприятия-покупателя, которое приобретает ИС, программный продукт или сервис ПО.

5.2. Процесс поставки определяет действия предприятия-поставщика, которое снабжает покупателя системой, программным продуктом или сервисом ПО.

5.3. Процесс разработки определяет действия предприятия-разработчика, которое разрабатывает принцип построения ИС.

5.4. Процесс функционирования (или эксплуатации) определяет действия предприятия-оператора, которое обеспечивает обслуживание системы в процессе эксплуатации в интересах пользователей.

5.5. Процесс сопровождения определяет действия персонала сопровождения — управление модификациями программного продукта, поддержка его текущего состояния и функциональной пригодности, а также инсталляция и удаление программного изделия.

Вспомогательные процессы поддержки ЖЦ ПС обеспечивают должное качество проекта и его документирование.

Степень адаптивности максимальная, множество процессов, работ и задач сконструированы так, что возможна их адаптация для различных проектов ИС путем исключения процессов, работ и задач, не применимых в конкретном проекте. Возможно добавление уникальных или специфических процессов, работ и задач, но это требует юридического оформления в контракте.

Стандарт ISO 12207 принципиально **не содержит конкретные заготовки решений или документации**, как это реализовано в CDM. Он описывает архитектуру процессов ЖЦ ИС, но не конкретизирует в деталях, как реализовать или выполнить услуги и задачи, включенные в процессы.

Стандарт ISO 12207 не предписывает конкретную модель ЖЦ, но определяет, что стороны — участницы использования стандарта ответственны за выбор модели ЖЦ для проекта ИС, за адаптацию процессов и задач стандарта к этой модели, за выбор и применение методов разработки ПО, за выполнение работ и задач, подходящих для конкретного проекта ИС.

Недостатком стандарта можно считать предельно малое описание задач, направленных на проектирование БД. Конкретная польза стандарта в том, что он содержит наборы задач, характеристик ка-

честв и критериев оценки, которые дают всесторонний охват проектных ситуаций.

Дальнейшее развитие стандарта ISO 12207 реализовано в двух более поздних стандартах **ISO/IEC 15288:2002** и **ГОСТ Р ИСО/МЭК 14764—2002.**

Стандарт ISO/IEC 15288:2002 «Системная инженерия — Процессы жизненного цикла систем» описывает общую структуру процессов, составляющих жизненный цикл любого рода систем, из которых разработчик может ***конструировать модели жизненного цикла*** систем, соответствующие его продуктам и услугам. Стандарт констатирует обязательность оценки качества процессов сбора, обработки и распространения данных, качества используемой информации, ее своевременности, полноты, достоверности, конфиденциальности и полезности.

Стандарт ГОСТ Р ИСО/МЭК 14764—2002 «Информационная технология. Сопровождение программных средств» уточняет требования к сопровождению программных средств.

5.4. Комплекс российских стандартов ГОСТ 34

Стандарты **комплекса ГОСТ 34** рассчитаны на взаимодействие заказчика и разработчика, однако они в меньшей степени ориентированы на столь явное симметричное отражение действий обеих сторон, как в ISO 12207. Стандарты ГОСТ 34 в основном уделяют внимание содержанию проектных документов и помогают создавать их качественно и на современном уровне, что позволяет оставаться этому стандарту в числе самых востребованных.

Общая структура представлена комплексом взаимоувязанных стандартов, наиболее популярными из них являются **ГОСТ 34.601—90** («Стадии создания АС»), **ГОСТ 34.602—89** («ТЗ на создание АС»), **ГОСТ 34.603—92** («Виды испытаний АС») и методические указания **РД 50-34.698—90** («Требования к содержанию документов»).

Стандарты предусматривают стадии и этапы выполнения работ по созданию ИС (см. разд. 2.3), но не предусматривают сквозных процессов в явном виде по этапам, как это реализовано, например, в CDM. В стандартах детально описано содержание документов, разрабатываемых на каждом этапе, что делает их особенно полезными для построения профилей стандартов.

Стадии и этапы, выполняемые организациями — участниками работ по созданию ИС, фиксируются в договорах, техническом задании и техническом проекте, что близко к подходу в ISO 12207.

Ключевым документом взаимодействия заказчика и разработчика является **техническое задание** на создание ИС. ТЗ является основным исходным документом для создания ИС и ее приемки, оно определяет важнейшие точки взаимодействия заказчика и разработчика. В качестве приложения к ТЗ разрабатывается календарный план выполнения работ для всего проекта. На этапе приемки ИС полезным является и технический проект, в котором более точно и детально на уровне спецификаций прописаны все требования к системе.

Техническое задание разрабатывает обычно организация-разработчик (по ГОСТ 34.602—89), но формально выдает ТЗ разработчику заказчик (по РД 50-680—88). В ТЗ необходимо указать, кто будет разрабатывать программу и методики испытаний, чтобы избежать конфликтных ситуаций при наступлении этапа ввода системы в эксплуатацию.

Полная прежняя обязательность применения только стандартов ГОСТ 34 в настоящее время отменена. Материалы ГОСТ 34 по сути стали методической поддержкой и для разработчика, и для заказчика благодаря имеющемуся в стандарте набору требований к содержанию технико-экономического обоснования, ТЗ, технического проекта и проведению испытаний. В последнее время роль ГОСТ 34 многократно возросла из-за их широкого применения при формировании профилей стандартов ЖЦ ИС.

Стадии и этапы стандарта на практике ориентируют разработчиков на каскадную модель ЖЦ или близкую к ней спиралевидную.

Степень адаптивности стандартов ГОСТ 34 формально определяется следующими возможностями:

- исключать стадию эскизного проектирования в несложных проектах;
- объединять стадии «Технический проект» и «Рабочая документация» в одну — «Технорабочий проект»;
- исключать или объединять документы, разделы документов и работы;
- в ходе выполнения проекта создавать частные технические задания (ЧТЗ) для придания большей гибкости каскадной модели ЖЦ ИС.

Приведенное выше сравнение трех популярных стандартов показывает, что ни один из них не является полным, не описывает все виды

действий и задач, реально требующихся в конкретных проектах ИС. Вероятно, такая ситуация является объективно неизбежной для любых достаточно конкретных стандартов и фирменных методик, которые ориентированы на реализацию вполне определенных конкретных действий [18].

Так, CDM явно вводит принципиально важный в реальных проектах процесс смены поколений — процесс «Преобразование данных», который не выделен в ISO 12207:1995 и косвенно отражен в ГОСТ 34.601—90. Кроме этого, CDM вводит процесс проектирования баз данных в трактовке, близкой к классической.

В свою очередь, ISO 12207:1995 имеет набор процессов, работ и задач, которые охватывают наиболее широкий спектр возможных ситуаций при максимальной адаптации стандарта к разрабатываемому проекту ИС.

Этот стандарт является примером того, как должен строиться хорошо организованный стандарт, содержащий минимум ограничений, предполагая, что все детальные определения процессов и форм документов целесообразно выносить в различные функциональные стандарты, ведомственные нормативные документы или фирменные методики, которые могут быть использованы или не использованы в конкретном проекте.

По этой причине стандарт ISO 12207:1995 выбирается в качестве центрального в процессе построения профиля стандартов для конкретного проекта. Другие стандарты или материалы полезно включать в профиль теми положениями, которые определяют конкретные способы анализа или программирования, формы проектных документов и инструменты проектирования, применяемые в каждом конкретном процессе.

5.5. Профили стандартов на разработку программных средств

5.5.1. Общие сведения о профилях стандартов

Большинство используемых моделей ЖЦ ИС содержит ряд базовых этапов, но несколько отличается графическим представлением, количеством этапов и содержанием работ на этапах. На рис. 5.1 представлена обобщенная каскадная модель ЖЦ ИС, в которой этапы

располагаются последовательно, символизируя передачу выполненных работ с предыдущего этапа на последующий. Между этапами имеются обратные связи, указывающие на итерационный процесс совершенствования ПС.

Рис. 5.1. Обобщенная модель ЖЦ ИС

При практическом использовании моделей ЖЦ для реального планирования и управления проектами требуется более подробная информация о содержании процессов. Должны быть представлены исходные данные, содержание частных работ, ожидаемые результаты их выполнения, а также структура и содержание документов, сопутствующих их реализации. Подобная детализация на десятки и даже сотни частных работ производится при формировании технологии

и средств поддержки разработки, сопровождения и эксплуатации фирмами-производителями программных инструментариев.

При проектировании, разработке и сопровождении сложных ИС требуется тщательный подбор и правильное применение базовых стандартов и различных нормативных документов. Однако несколько сотен международных и национальных стандартов не покрывает полностью потребности в стандартизации объектов и процессов создания ИС. При этом степень адаптации большинства стандартов к создаваемым ИС недостаточна, поэтому в настоящее время активно применяются так называемые **профили стандартов**.

Формирование и применение профилей конкретных информационных систем выполняются на основе использования международных и национальных стандартов, ведомственных нормативных документов, а также стандартов де-факто при условии доступности соответствующих им спецификаций.

Разработка и применение профилей являются органической частью процессов проектирования, разработки и сопровождения информационных систем. Профили характеризуют каждую конкретную информационную систему на всех стадиях ее жизненного цикла, задавая согласованный набор базовых стандартов, которым должна соответствовать система и ее компоненты.

Применение профилей при создании и применении ИС обеспечивает снижение трудоемкости, длительности и стоимости проекта, повышает качество разрабатываемых или применяемых покупных компонентов ИС, обеспечивает переносимость прикладных программ и данных между разными аппаратно-программными платформами.

Выбор стандартов и документов для формирования профилей ИС зависит от того, какие из этих целей являются приоритетными. При практическом формировании и применении профилей ИС допускается использование региональных, национальных стандартов и ведомственных нормативных документов.

Подготовка профилей к применению должна также учитывать реальное состояние проекта ИС: планируется создание новой системы, имеется типовой проект ИС и предстоит его адаптация, существует и эксплуатируется реальная ИС, которую необходимо модифицировать. Жизненный цикл конкретной ИС должен быть поддержан профилем ИС в соответствии с основными процессами создания, сопровождения и развития ИС.

ЖЦ в профилях ИС должен представляться набором этапов, работ и операций, обеспечивающих разработку ИС, начиная от подготовки

ТЗ до завершения испытаний, а также сопровождение и развитие ИС вплоть до окончания эксплуатации. Профили должны включать описание исходной информации, способов и методов выполнения операций и работ, устанавливать требования к результатам и правилам их контроля, определять состав технологических и программных документов.

Профиль ЖЦ ИС обычно определяют как подмножество процессов, работ и задач стандарта ISO 12207, выбирая их с учетом характеристик проекта конкретной системы, заданных в ТЗ. Первичный профиль обычно создается на основе базовых стандартов, которые важны с точки зрения заказчика и задаются в ТЗ. То, что не задано в ТЗ, первоначально остается на усмотрение разработчика системы, который, руководствуясь требованиями ТЗ, может дополнять и развивать профили системы и впоследствии согласовывать их с заказчиком.

Таким образом, профиль конкретной системы не является статичным, он развивается и конкретизируется в процессе проектирования информационной системы и оформляется в составе документации проекта системы. После завершения проектирования и испытаний системы, в ходе которых проверяется ее соответствие профилю, он применяется как основной инструмент сопровождения системы при эксплуатации, модернизации и развитии [23].

5.5.2. Предложения по профилю стандартов на разработку программных средств

Приведенная на рис. 5.1 обобщенная модель ЖЦ ИС в наибольшей степени удовлетворяет большинству наиболее популярных международных стандартов, хотя и отличается от стандарта ГОСТ 34.601—90 отсутствием таких этапов, как техническое задание, эскизный и технический проекты. При формировании этой модели были определены основные крупные этапы с достаточно завершенными и контролируемыми результатами, а также коррелированными с наиболее популярными моделями ЖЦ ИС [http://www.devcomplexsoft.ru].

Большинство работ завершается созданием определенных документов. Формирование всей совокупности документов позволяет фиксировать завершение каждого этапа и возможность перехода к следующему этапу разработки. Тем самым обеспечивается формализованный контроль реализации процессов разработки и качества результатов на последовательных этапах проекта.

Целью разработки данного профиля стандартов является объединение структуры модели ЖЦ, используемой в международных стандартах, с работами и документами, используемыми в российских стандартах ГОСТ 34 и являющимися более привычными и понятными для российских разработчиков и заказчиков.

Профиль стандартов будет реализован в виде табл. 5.2, в которой помещены названия этапов, работ и наименований разделов соответствующих базовых стандартов, также номера создаваемых документов в поле **«Отчетные документы»**, содержание которых приведено в разд. 5.5.3.

Таблица 5.2. **Профиль стандартов**

Этапы и работы	Нормативные документы	Отчетные документы
Этап 1. Системный анализ предметной области		
1.1. Обследование объекта информатизации, обоснование необходимости создания ИС. Построение модели «AS IS»	РД 50-34.698—90, Приложение 1; ISO 12207—95, п. 5.3.1; РД IDEF0—2000	Документ 1.1
1.2. Первичная формулировка исходных данных и потенциальных решений ИС с учетом потребностей пользователя, ресурсов на разработку	РД 50-34.698—90, Приложение 1; ISO 12207—95, п. 5.3.1	Документ 1.1
1.3.Предварительная оценка технико-экономических показателей проекта, сроков, бюджета и стратегии риска	ISO/IEC15288-02	Документ 1.1
1.4. Разработка функциональной модели «ТО ВЕ». Формализация функций ИС, требований к качеству решаемых задач	РД IDEF0-2000; ISO 12207—95, п. 5.3.2, 5.3.3, Приложение А и В	Документ 1.1
1.5. Предварительное описание архитектуры ИС, его БД, первичных требований к интерфейсам пользователей	Стандарт IDEF1X РД 50-34.698—90	Документ 1.1
1.6. Разработка предварительного ТЗ на создание базовой версии ИС, заключение контракта на разработку базовой версии ИС	ГОСТ 34.602—89 ISO 12207—95, п. 5.1.3, 5.2.3, 5.3.4	Документ 1.2

Продолжение табл. 5.2

Этапы и работы	Нормативные документы	Отчетные документы
Этап 2. Предварительное (эскизное) проектирование		
2.1. Анализ функциональной и информационной моделей, архитектуры комплекса программ, описание алгоритмов, структур данных ИС	ISO 12207—95, п. 5.3.4	Документ 1.1
2.2. Выбор СУБД, проектирование структуры БД	РД 50-34.698—90, п. 5.3, 5.5, 5.7	Документ 1.1
2.3. Разработка предварительного руководства для пользователей и обслуживания базовой версии ИС	ISO 12207—95, п. 7.4; РД 50-34.698—90, п. 3.4	Документ 2.1; Документ 2.2; Документ 2.3
2.4. Уточнение и утверждение заказчиком ТЗ на разработку базовой версии ИС	ГОСТ 34.602—89	Документ 1.2
Этап 3. Детальное (техническое) проектирование		
3.1. Уточнение и документирование архитектуры ИС, спецификаций требований и методов решения задач	ISO 12207—95, п. 5.3.6	Документ 3.1
3.2. Разработка спецификаций требований к функциональным группам программ и модулям версии ПС	РД 50-34.698—90	Документ 2.4
3.3. Разработка технического проекта базовой версии ИС в соответствии с ТЗ	РД 50-34.698—90, п. 2.2	Документ 2.4
3.4. Документирование техпроекта, уточнение спецификаций требований и условий контракта	ISO 12207—95, п. 5.3.6, 6.1; РД 50-34.698, п. 2.2	Документ 2.4
Этап 4. Кодирование (программирование), отладка и документация компонентов базовой версии ИС		
4.1. Разработка исходных текстов программных модулей	ISO 12207, п. 5.3.7	Документ 2.3
4.2. Трансляция исходных текстов и устранение ошибок в программах	ISO 12207, п. 5.3.8	Документ 2.3
4.3. Тестирование и отладка модулей, устранение дефектов, корректировка текстов программ	ISO 12207—95, п. 5.3.7, 6.3, 6.4, 6.5	Документ 4.1

Продолжение табл. 5.2

Этапы и работы	Нормативные документы	Отчетные документы
4.4. Документирование исходных и объектных текстов компонентов, результатов их тестирования, качества и технических характеристик	РД 50-34.698—90; ISO 12207—95, п. 6.1; 5.3.7; ЕСПД	Документ 4.1
Этап 5. Интеграция и комплексная отладка		
5.1. Интеграция компонентов, тестирование и определение характеристик качества программ при решении основных функциональных задач ИС	ISO12207—95, п. 5.3.8, 5.3.9, 5.3.10, 5. 3.11, 6.6	Документ 4.1
5.2. Разработка проекта плана, программы и методик приемосдаточных испытаний базовой версии ИС	ISO 12207—95, п. 5.3.13, 5.4.1; РД 50-34.698—90, п. 2.14	Документ 5.1; Документ 5.2
5.3. Документирование результатов предварительных испытаний и характеристик версий для предъявления на приемосдаточные испытания заказчику	ISO 12207—95 п.п. 5.3.12; 6.1	Документ 5.3; Документ 5.4
5.4. Разработка комплекта эксплуатационной документации для пользователей, оформление текстов программ и информации БД для приемосдаточных испытаний заказчиком	ГОСТ 34.201—90; ISO 12207—95, п. 5.3.13; 6.1; ЕСПД	Документ 2.1; Документ 2.2; Документ 2.3
Этап 6. Испытания и документирование		
6.1. Разработка программы, методик и средств обеспечения приемосдаточных испытаний ИС (совместно с заказчиком)	ГОСТ 34.603—92; РД 50-34.698—90, п. 2.14	Документ 5.1; Документ 5.2
6.2. Проведение тестирования ИС по программе приемосдаточных испытаний на соответствие функциональным и техническим характеристикам, заданным в контракте	РД 50-34.698—90, п. 2.14	Документ 5.3; Документ 5.4
6.3. Опытная эксплуатация ИС представителями заказчика и оформление отчета о результатах	ГОСТ 34.603—92; РД 50-34.698—90; ISO 12207—95, п. 5.4.1; 5.4.2	Документ 6.1

Продолжение табл. 5.2

Этапы и работы	Нормативные документы	Отчетные документы
6.4. Проведение приемосдаточных испытаний ИС в составе информационной системы совместно с заказчиком	РД 50-34.698—90, п. 2.14	Документ 6.1
6.5. Документирование результатов приемосдаточных испытаний	ГОСТ 34.603—92; ЕСПД	Документ 6.1
6.6. Оформление полного комплекта документов на базовую версию ИС	ГОСТ 34.201—90 РД 50-34.698—90, п. 3.4; ЕСПД	Документ 2.1; Документ 2.2; Документ 2.3
6.7. Официальное завершение разработки и оформление акта приемки базовой версии ИС	РД 50-34.698—90, Приложение 2	Документ 6.2
Этап 7. Поддержка эксплуатации		
7.1. Обучение и консультации пользователей в процессе эксплуатации базовой версии ИС	ISO 12207—95, п. 5.3.12, 5. 3.13, 5.4.1	Документ 7.1
7.2. Накопление и обработка отчетов пользователей о результатах эксплуатации	ISO 12207—95, п. 5.4.2, 5.4.3, 5.5.1	Документ 7.2
7.3. Информирование пользователя о частных изменениях в эксплуатируемой базовой версии ИС	ISO 12207—95, п. 5.5.5	Документ 7.3
7.4. Планирование перехода к новой версии	ISO 12207—95, п. 5.5.3; 5.5.4.	Документ 7.4
7.5. Прекращение поддержки эксплуатации базовой версии	ISO 12207—95, п. 5.5.6	Документ 7.4
Этап 8. Сопровождение		
8.1. Анализ предложений на модификацию версий ИС, предлагаемых изменений программ и данных, необходимых затрат, риска и возможных альтернатив	ISO 12207—95, п. 5.5.1; 6.8	Документ 7.1

Окончание табл. 5.2

Этапы и работы	Нормативные документы	Отчетные документы
8.2. Создание новой версии (полное или частичное повторение этапов 1—6)	См. этапы 1—6	Документ 7.4 и документы на этапах 1—6
8.3. Испытания и утверждение новой версии ИС разработчиком и заказчиком	РД 50-34.698—90, п. 2. 14; Приложение 2	Документы 5.1—5.4
8.4. Приемка заказчиком и передача в эксплуатацию новой версии ИС	ГОСТ 34.603—92	Документ 6.1

5.5.3. Структура и содержание технологической и эксплуатационной документации

Предложения по профилю стандартов на разработку программных средств содержат структуру технологических и эксплуатационных документов, которые необходимы на этапах создания, применения и сопровождения ИС. Выполнение процесса ЖЦ ИС считается законченным, если выполнены требуемые задачи и оформлены все документы в соответствии с заранее установленными требованиями.

Под технологической документацией понимают подробные технические описания для специалистов, ведущих проектирование, разработку и сопровождение ИС. **Эксплуатационная документация** создается для конечных пользователей ИС, чтобы они могли освоить и квалифицированно применять эти средства при эксплуатации и сопровождении системы.

Каждый разрабатываемый документ должен позволять контролировать результаты и качество выполненных работ. В общем случае каждый документ должен иметь:

- наименование;
- назначение;
- категории пользователей, для которых он разрабатывается;
- авторов разработки;
- этапы работ, на которых его надо применять;
- функциональную, содержательную часть в соответствии с его назначением.

Далее предлагается набор рекомендуемой стандартами технологической и эксплуатационной документации, а также ее содержание

(номера документов соответствуют номерам работ в табл. 5.2 профиля стандартов на разработку ИС).

Документ 1.1. Технико-экономическое обоснование (ТЭО):

- описание результатов изучения объекта информатизации;
- цели и ограничения создания ИС;
- функции и задачи создаваемой ИС;
- характеристики комплекса задач;
- рекомендации по созданию ИС, БД, интерфейсов пользователей, выбору СУБД;
- условия испытания и приемки ИС;
- входная информация (формы представления, сроки и частота поступления, источники информации);
- выходная информация (описание выходных сообщений, периодичность выдачи, допустимое время задержки, получатели и назначение выходной информации);
- ожидаемые технико-экономические результаты создания системы (перечень основных источников экономической эффективности, получаемых в результате создания системы, и оценка ожидаемых изменений основных технико-экономических и социальных показателей деятельности объекта; оценка ожидаемых затрат на создание и эксплуатацию системы с распределением их по очередям создания системы и по годам; ожидаемые обобщающие показатели экономической эффективности системы);
- выводы и предложения о производственно-хозяйственной необходимости и технико-экономической целесообразности создания системы, о совершенствовании организации и технологии процесса деятельности;
- рекомендации по созданию системы.

Документ 1.2. Техническое задание на проектирование ИС

Этот документ предполагает поэтапное уточнение и детализацию предлагаемых ниже разделов, подробно описанных в ГОСТ 43.602—89:

- титульный лист с утверждающими и согласующими подписями;
- назначение и цель разработки ИС;
- общие сроки выполнения проекта;
- общие технические требования и базовые нормативные документы для выполнения ИС;
- общие требования к ИС;
- требования к структуре и функционированию системы;
- требования к оформлению и содержанию эксплуатационной и технологической документации;

• требования к составу и содержанию работ по внедрению ИС в эксплуатацию;

• этапы и сроки выполнения основных работ;

• ожидаемые результаты и формы их представления;

• порядок контроля и приемки результатов работы.

Документ 2.1. Руководство администратора:

• функции администрирования при эксплуатации данной системы;

• процедуры по инсталляции и подготовке информационной системы к эксплуатации;

• способы и формы контроля исполнения заданий;

• настройка ИС.

Документ 2.2. Руководство пользователя:

• введение (область применения, требования к уровню подготовки пользователей, перечень эксплуатационной документации, с которым необходимо ознакомиться пользователю);

• назначение и условия применения;

• подготовка к работе;

• описание операций обработки данных;

• аварийные ситуации и действия оператора при их наступлении;

• контрольный пример, правила его запуска и выполнения.

Документ 2.3. Документация на разработанный модуль ИС (руководство программиста):

• ТЗ и/или спецификация требований на разработку программы;

• описание программы (краткое описание задачи, алгоритма, структуры и функций модуля, пользовательского интерфейса, входных и выходных компонент, межмодульных интерфейсов, описание способов проверки работоспособности и контрольные примеры);

• исходный текст программы на бумажном носителе;

• исходный текст и объектный код программы на магнитных или иных носителях.

Документ 2.4. Пояснительная записка к эскизному и техническому проектам ПС:

• общие положения;

• описание процесса деятельности;

• основные технические решения (по структуре и функциям ИС, режимам функционирования, составу комплексов задач и интерфейсов, назначение и характеристики программных модулей и таблиц БД, функциям СУБД, по структуре технических средств, по организа-

ционной структуре, по языкам программирования, по организации и ведению базы данных и т. п.);

- мероприятия по подготовке объекта информатизации к вводу ИС в действие;
- перечень всех технологических и эксплуатационных документов.

Документ 3.1. Утвержденные спецификации требований и алгоритмы на функциональные группы программ, программные и информационные компоненты:

- назначение и характеристики алгоритма (постановка задачи, краткие сведения о процессе (объекте), ограничения на возможность и условия применения, характеристики качества решения, общие требования к входным и выходным данным);
- используемая (входная) информация программного модуля;
- результаты решения (выходная информация);
- математическое описание;
- требования к разработке программ.

Документ 4.1. Отчеты о результатах тестирования:

- журнал тестирования;
- отчет об аномальных событиях;
- итоговый отчет.

Документ 5.1. Программа испытаний:

- объект испытаний;
- цель испытаний;
- собственно программа испытаний;
- методики испытаний.

Документ 5.2. Методики испытаний:

- объект испытаний;
- цель испытаний;
- общие положения методов испытаний;
- объем испытаний по разделам программы;
- условия и порядок проведения испытаний в соответствии с ТЗ;
- материально-техническое обеспечение;
- отчетность и документирование результатов испытаний;
- акт и итоговый отчет о результатах испытаний.

Документ 5.3. Протокол испытаний:

- наименование объекта испытаний;
- список должностных лиц, проводивших испытания;
- цель испытания;

• назначение тестирования и раздел требований технического задания, по которому проводились испытания;

• перечень пунктов программы испытаний и указание соответствующих методик, обработка и оценка результатов;

• условия и сценарии проведения тестирования и характеристики исходных данных;

• сведения об отказах, аварийных ситуациях;

• сведения о корректировках параметров объекта испытаний и технической документации;

• обобщенные результаты испытаний с оценкой их на соответствие техническому заданию и технической документации, а в некоторых случаях и техническому проекту;

• выводы о результатах испытаний и соответствии созданной системы требованиям ТЗ.

Документ 5.4. Акт завершения работ:

• наименование завершенной работы;

• список представителей заказчика и разработчика, составивших акт;

• дата завершения работ;

• наименование документа, на основании которого проводилась работа;

• основные результаты завершенной работы;

• заключение о результатах завершенной работы.

Документ 6.1. Акт приемки ИС в промышленную эксплуатацию:

• наименование ИС, принимаемой в эксплуатацию;

• сведения о статусе приемочной комиссии;

• период времени работы комиссии;

• состав функций ИС, принимаемых в эксплуатацию;

• перечень документов, предъявляемых комиссии;

• заключение о результатах опытной эксплуатации ИС;

• оценка соответствия принимаемой системы техническому заданию на ее создание;

• краткая характеристика работ по созданию ИС и основные результаты;

• оценка экономической эффективности от внедрения ИС;

• решение комиссии;

• рекомендации комиссии по дальнейшему развитию ИС.

К акту прилагаются:

• программа и протокол испытаний;

• протоколы заседания комиссий;

• перечень технических средств, которые использовала комиссия при приемке системы.

Документ 6.2. Акт о завершении приемосдаточных испытаний и результаты выполнения контракта на разработку ИС:

• наименование завершенной работы;

• список представителей заказчика и разработчика, составивших акт;

• дата завершения работ;

• наименование документа, на основании которого проводилась работа;

• основные результаты завершенной работы;

• заключение о выполнении контракта.

Документ 7.1. Отчеты пользователей о выявленных дефектах и предложениях по корректировке версий ИС:

• идентификатор пользователя;

• номер пользовательской деятельности;

• подробное описание сценария и исходных данных, при которых выявлен дефект;

• предположение о причине появления дефекта;

• предложения по модификации ИС или совершенствование функционирования программы.

Документ 7.2. Журнал выявленных дефектов и предложений по совершенствованию версий ИС:

• идентификатор разработчика, которому отдан отчет пользователя для анализа;

• тесты, исходные данные и сценарий, при которых проявляется дефект;

• результаты анализа причины и источника выявленного дефекта;

• предложения по устранению дефекта;

• оценка сложности, трудоемкости и срочности модификации программ и БД;

• оценки влияния предлагаемых изменений на эксплуатацию версий ПС, имеющихся у пользователей.

Документ 7.3. Журнал подготовленных и утвержденных корректировок:

• идентификатор специалиста, разработавшего модификацию программы и базы данных;

• дата разработки модификации;

• причина изменения программ и базы данных;

• содержание изменений программ, базы данных и документации;

- результаты тестирования;
- результаты испытаний;
- решение по распространению версии среди пользователей;
- адрес хранения корректировок, документов и тестов новой версии ИС.

Документ 7.4. Извещение пользователей о выпуске новой версии ИС или о прекращении сопровождения определенной версии ИС:

- краткое обоснование причин модификации или прекращения сопровождения версии ИС;
- описание содержания основных изменений в новой версии;
- рекомендации по приобретению или замене пользовательской версии ИС.

Контрольные вопросы

1. Назначение и применение стандартов жизненного цикла ИС.
2. Сравнительный анализ стандартов ГОСТ 34 и методологии Oracle CDM.
3. Общая структура международного стандарта ISO/IEC 12207 (ГОСТ Р ИСО/МЭК 12207: 1995).
4. Сравнительный анализ стандартов ГОСТ 34 и международного стандарта ISO/IEC 12207:1995.
5. Сравнительный анализ трех стандартов по степени их адаптируемости к разрабатываемому проекту ИС.
6. Назначение и состав профиля стандартов.
7. Состав технологической и эксплуатационной документации в профиле стандартов.

Глава 6
РЕАЛИЗАЦИЯ НАЧАЛЬНЫХ ЭТАПОВ ПРОЕКТИРОВАНИЯ ИС

6.1. Основные компоненты в проектировании ИС

В предыдущих главах описано все необходимое для того, чтобы приступить к проектированию ИС.

Во-первых, был определен состав информационных систем, необходимый для реализации CALS-технологий: класс экономических информационных систем MRP/MRP II/ERP/CSRP и класс вспомогательных систем, выводящих стандарт ERP «за ворота предприятия», — CRM, SCM, OLAP, B2B, B2C и ИЭТР.

Во-вторых, познакомились с содержанием понятия «технология проектирования ИС» и с самими технологиями проектирования ведущих производителей программных инструментариев. Кроме этого, детальным образом изучили реализацию CASE-технологии в инструментальной среде Oracle Designer 10g.

В-третьих, познакомились с современными программными инструментариями и их правильным выбором для проектирования конкретной ИС. Особенно тщательно были изучены методологии инструментариев BPwin и ERwin, входящих в пакет All Fusion Modeling Suite, которые чрезвычайно популярны в учебном процессе и в проектировании простых ИС.

И, в-четвертых, изучили наиболее востребованные в настоящее время стандарты (ГОСТ 34, ISO/IEC 12207:1995, ISO/IEC 15288:2002, Oracle CDM и др.), на которых базируются технологии проектирования ИС. Познакомились также с принципами создания профилей стандартов для разрабатываемых систем.

Прежде чем перейти непосредственно к проектированию системы, следует отметить, что проектирование охватывает четыре основные области:

- проектирование процессов;

- проектирование объектов данных;
- проектирование программ, экранных форм, отчетов;
- разработка архитектуры ИС.

Главная цель **проектирования процессов** заключается в определении функциональности ИС в результате построения функциональной иерархической модели. Для этого с помощью графических моделей переходят от текстового описания деятельности (содержащегося в нормативных документах организации, таких как положения о подразделениях, должностные инструкции, технологические карты производственных процессов) к полному формализованному графическому описанию.

Описание деятельности организации исключительно трудоемкая работа, которая обычно осуществляется постепенно — с описания наиболее значимых процессов верхнего уровня с их последующей детализацией. При этом корректным считается такой подход к моделированию, когда диаграмма любого уровня (кроме верхнего) является детализацией объекта какой-либо диаграммы предыдущего уровня:

- процессы верхнего уровня;
- подпроцессы;
- сценарии процессов;
- процедуры;
- функции.

Такой подход, в частности, поддерживается инструментом AllFusion Process Modeler (BPwin) компании Computer Associates. Детализация (декомпозиция) — это условный прием, позволяющий представить систему в виде, удобном для восприятия и анализа как для разработчиков, так и для представителей заказчика.

Функции, полученные на нижнем уровне иерархии функциональной модели, представляют собой модули информационной системы, которые определяют интерфейсы программ: разметку меню, вид окон, горячие клавиши и связанные с ними вызовы. Конечным продуктом проектирования процессов является **набор спецификаций модулей** системы.

Проектирование объектов данных заключается в формировании базы данных логического и физического уровней. Проектировщики в качестве исходной информации получают результаты анализа документооборота в построенной функциональной модели. Полученная в процессе анализа информационная модель сначала преобразуется в логическую, а затем в физическую модель данных. Конечным продуктом проектирования объектов данных является **схема базы данных**.

При этом проектирование процессов продолжается параллельно с проектированием схемы базы данных, поскольку часть бизнес-логики (программных модулей) обычно реализуется в базе данных (ограничения, триггеры, хранимые процедуры).

Проектирование программ, экранных форм, отчетов производится на основе спецификаций (описания) всех модулей ИС, таблиц базы данных и элементов пользовательского интерфейса.

Разработка архитектуры ИС включает в себя выбор конкретной среды или технологии, а именно: топологии сети, конфигурации аппаратных средств, используемой архитектуры (файл-сервер или клиент-сервер), параллельной обработки, распределенной обработки данных и т. п.

В качестве примера разработаем проект экономической ИС для управления **муниципальными предприятиями города** в среде инструментариев BPwin и ERwin на базе российских стандартов ГОСТ 34 для первых двух этапов каскадной модели: формирование требований к ИС и разработка концепций ИС.

6.2. Этап формирования требований к ИС

6.2.1. Реализация этапа на основе методологий пакета BPwin

Целью этапа является более детальное изучение объекта информатизации с помощью функциональных моделей «AS IS» («Как есть») или моделей процессов в среде Oracle Process Modeler. Этап начинается с обследования и системного анализа объекта информатизации. Это самый сложный и ответственный этап, поскольку он определяет не только масштаб и сложность проекта, но и состав будущих пользователей системы.

В процессе изучения объекта автоматизации обязательно выявляются классификации **MuSCoW**. Данная аббревиатура содержит информацию о четырех видах функций предметной области.

1. **Must have** — функции, необходимые для включения в состав проектируемой ИС.

2. **Should have** — желательные функции в составе системы, их включение в состав функциональности ограничивается временными и финансовыми рамками.

3. **Could have** — желательные функции также ограничиваются временными и финансовыми рамками.

4. **Won't have** — отсутствующие функции отражают границы проекта.

При определении функций, которые должны быть автоматизированы, следует не забывать о **принципе 80/20**, согласно которому 20 % усилий (затрат) дают 80 % результата, а остальные 80 % усилий обеспечивают только 20 % результата.

Работа на этом этапе выполняется системными аналитиками совместно с представителями заказчика. Сложность работы заключается в том, что никто на автоматизируемом предприятии в деталях не знает, как выполняются все бизнес-процессы: руководитель знает, как работает руководство высшего звена, как должны выполняться работы в соответствии с руководствами и должностными инструкциями, но не знает в деталях, как на самом деле выполняются рутинные работы подчиненными. Подчиненные работники, в свою очередь, хорошо знают выполнение всех бизнес-процессов на нижнем уровне, но ничего не знают о бизнес-процессах в других подразделениях.

Поэтому первое, что должны сделать системные аналитики, это совместно с представителями заказчика построить организационную структуру предприятия (**Organization chart diagram**) в пакете BPwin, представив с ее помощью будущих пользователей системы (рис. 6.1).

Затем с каждой из категорий их пользователей необходимо провести интервьюирование по тем бизнес-процессам, в которых каждый из них принимает участие.

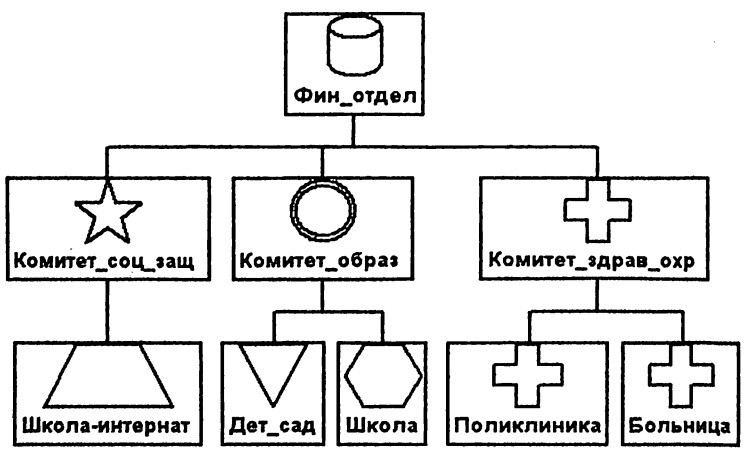


Рис. 6.1. Организационная структура предприятия

Анализ недостатков и «узких мест» в управлении предприятием начинается с построения модели «AS IS», которая отражает существующие в организации бизнес-процессы. На первой диаграмме модели рекомендуется отобразить взаимодействие всех внешних сущностей по отношению к моделируемому процессу в виде схемы «звезда». Для этих целей удобнее всего применить диаграмму потоков данных (рис. 6.2) или диаграмму прецедентов из пакета Rational Rose.

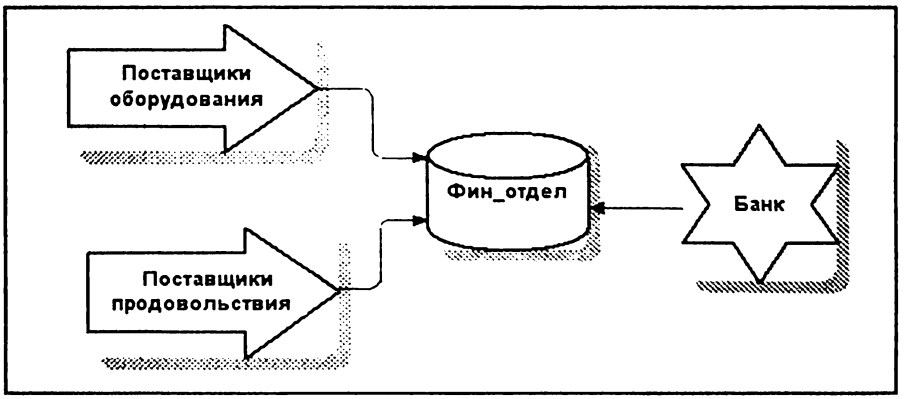


Рис. 6.2. Диаграмма потоков данных с внешними объектами

Модель «AS IS» строится в нотациях диаграмм Swim Lane (рис. 6.3) или модели процессов в среде Oracle Designer 10g (см. рис. 6.4). При этом продолжается дополнительное изучение документации (должностных инструкций, положений о предприятии, приказов, отчетов внутренних и для вышестоящих органов и т. п.), анкетирование и опрос служащих предприятия, создание фотографий рабочего дня будущих пользователей системы и других источников.

Количество диаграмм плавательных дорожек в модели «AS IS» определяется количеством сценариев или бизнес-процессов, которые системные аналитики обнаружат при изучении предметной области. При разработке модели необходимо исходить из тех позиций, что каждый бизнес-процесс является объектом управления, выход которого представляет собой определенную ценность как для внутреннего клиента, так и для внешнего (потребители продукции или услуг, акционеры, банки, налоговые органы и т. п.). Обычно процессы подразделяют на основные и вспомогательные [32].

К **основным процессам организации** относят процессы производства, сбыта и снабжения (процессы маркетинга, закупок, производства,

Рис. 6.3. Диаграмма плавательных дорожек «Управление поставками в городе (модель «AS IS»)»

хранения, поставки и сервисного обслуживания), т. е. процессы, добавляющие ценность.

Вспомогательные процессы не добавляют ценность, но увеличивают стоимость изделия или услуги. К таким процессам относят управление персоналом, управление документацией, техническое обслуживание оборудования, бюджетное управление и т. п.

При моделировании бизнес-процессов очень важно определить их содержание, т. е. из каких функций должен состоять конкретный бизнес-процесс. Особенно это важно на этапе построения моделей «ТО BE», но и на этапе построения модели «AS IS» следует также ориентироваться на процессный подход к управлению [32]. Признаком бизнес-процесса можно считать наличие в его составе **пяти основных элементов**: планирование и осуществление деятельности, регистрация фактической информации, контроль и анализ, принятие решений.

При правильном построении модели все функции, выполняемые в подразделениях, будут распределены по бизнес-процессам. Полученная модель «AS IS» содержит все имеющиеся недостатки в организации деятельности предприятия: выявляются неуправляемые работы, работы, не обеспеченные ресурсами, ненужные и неэффективные работы, дублирующие работы и т. д.

Если построить информационную систему, базируясь на функциональной модели «AS IS», то будет реализована автоматизация

предприятия по принципу «все оставить как есть, только чтобы компьютеры стояли». Такой подход недопустим, поскольку подобная система автоматизирует несовершенные бизнес-процессы и дублирует, а не заменяет существующий документооборот.

База данных логического уровня обычно не строится для модели «AS IS».

Результатом этапа формирования требований к ИС является документ — **технико-экономическое обоснование** (документ 1.1 в разделе 5.5.3). В ТЭО кратко описывается существующая система управления предприятием, указываются существующие недостатки в управлении, определяются пути их устранения и объясняются ожидаемые технико-экономические результаты.

6.2.2. Реализация этапа формирования требований к ИС на основе методологий пакета Oracle Designer 10g

Для построения модели процессов, которая может исполнять роль модели «AS IS», в Oracle Designer 10g используется более мощный инструмент **Process Modeler** по сравнению с диаграммами Swim Lane. Его уникальным свойством является возможность создания моделей процессов, представляющих отделы, подразделения или отдельных работников компании со свойственными им бизнес-процессами, потоками данных и операциями по сохранению данных [20].

Кроме этого, можно перейти от процессов верхнего уровня к процессам нижнего уровня и отражающим их моделям подпроцессов. То есть имеется возможность декомпозировать шаги процессов по аналогии с декомпозицией функциональных блоков в методологии IDEF0. Количество уровней декомпозиции не ограничивается.

В результате модель представляет последовательность шагов процессов, создающих услугу или изделие в рамках предприятия, а также информационный поток между ними. Данные, полученные из диаграммы процессов, хранятся в репозитарии (Repository) и доступны для использования другими инструментальными средствами Designer 10g. В частности, шаги процессов хранятся в репозитарии как функции, к которым можно получить доступ при использовании инструмента **Function Hierarchy Diagrammer** в целях построения функциональной диаграммы в автоматическом режиме на основании построенной модели процессов.

Запуск утилиты Process Modeler производится из окна Designer 10g щелчком по соответствующему ярлыку. Для включения режима создания новой диаграммы необходимо выполнить команду **File/ New Diagram**. В появившемся диалоговом окне необходимо ввести имя модели процессов и, щелкнув по клавише <Enter>, получают режим построения новой диаграммы (модели) процессов.

Вначале необходимо построить организационные единицы (**Organization Unit**), которые аналогичны плавательным дорожкам диаграммы Swim Lane. Имя организационной единицы размещается в левой части полосы, а справа от имени размещается трасса потока: шаги процесса (аналог работ IDEF3), хранилища и пункты принятия решений (логический элемент да/нет) (рис. 6.4).

Для построения организационной единицы необходимо щелкнуть по кнопке **Organization Unit** на панели инструментов и в появившемся окне ввести параметры новой организационной единицы: название (полное и краткое), текстовое описание и щелкнуть по клавише <OK>. На экране появляется новая организационная единица. В качестве названия организационных единиц используют названия отделов, подразделений, должностей и т. п., которые участвуют в выполнении моделируемого бизнес-процесса.

Для организационных единиц можно указать **отношение родитель—сын** (например, в подчинении у заместителя директора по кадрам находятся отдел кадров, отдел охраны, канцелярия и т. п.). Это отношение будет графически отображено на диаграмме. Для построения шага процесса, хранилища или пункта принятия решения необходимо установить курсор на соответствующую полосу (организационную единицу) и щелкнуть клавишей мыши, что приведет к появлению окна **Create Process Step**. В это окно вводится имя процесса и выбирается из списка тип процесса (шаг процесса, хранилище или пункт принятия решения).

Можно ввести также время и стоимость для шага процесса, текстовое описание, средства мультимедиа для отображения его в пиктографическом режиме. Щелкнув по кнопке **<OK>**, получают отображение объекта на соответствующей полосе. Для редактирования этих объектов используется редактор **Edit Process Step**, который вызывается двойным щелчком мыши по соответствующему шагу процесса.

Все объекты на модели процессов соединяются потоками (дугами), которые отображают движения информации или материалов. Они создаются с помощью кнопки **Create Flow** на панели инструментов.

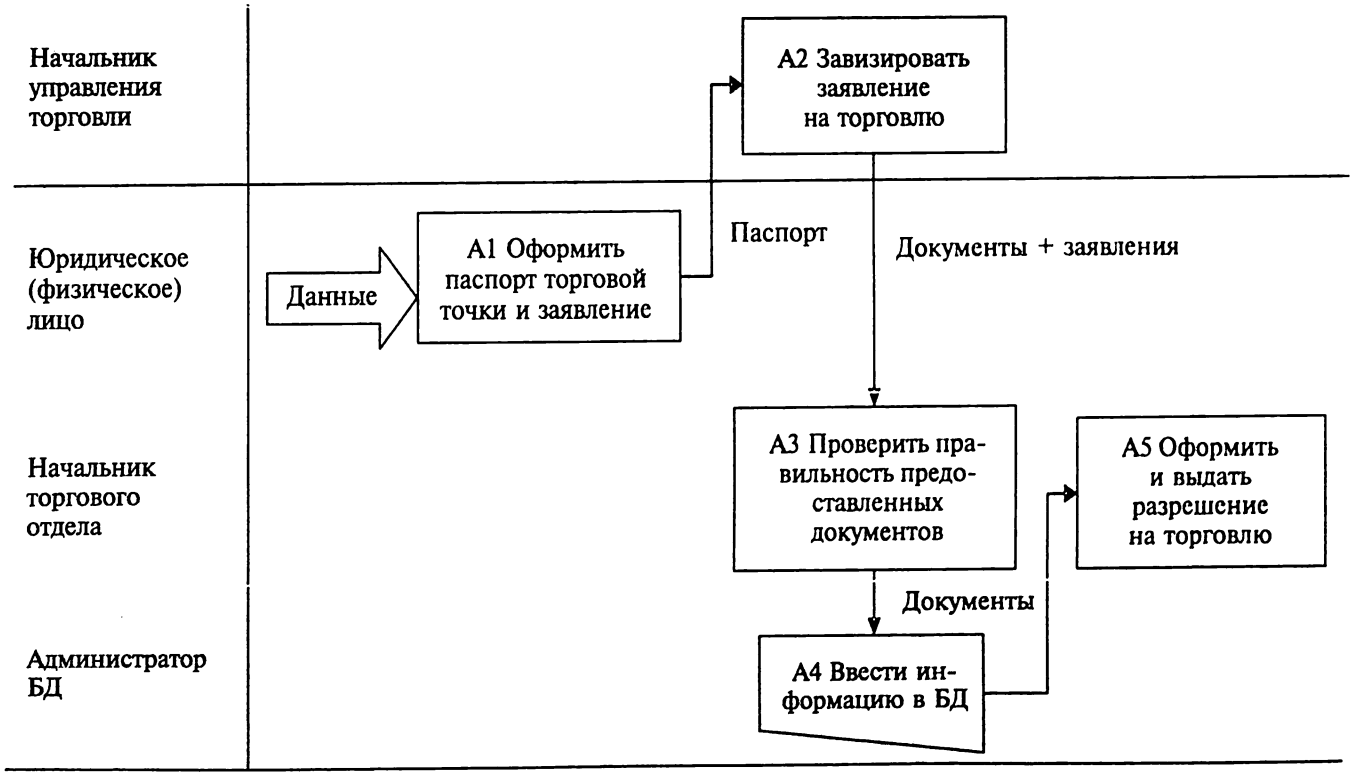


Рис. 6.4. Диаграмма из модели процессов «Выдача разрешения на торговлю»

После активизации этой кнопки следует щелкнуть поочередно на предыдущем и последующем шагах процесса для получения диалогового окна построения потока между этими объектами. В окне выбирают **тип потока** (данные, материалы, временной или просто поток), вводят имя потока, продолжительность и стоимость передачи. Для добавления потока в диаграмму следует щелкнуть по клавише **<OK>**.

В модели процессов используются еще **входной триггер** и **выходной триггер**. Входной триггер представляет собой событие, с которого начинается бизнес-процесс, например поступление товара на склад. Выходной триггер выражает результат выполнения бизнес-процесса, например сформированная накладная на выдачу товара.

Для создания входного триггера необходимо щелкнуть по кнопке **Create Trigger** на инструментальной панели, щелкнуть затем по соответствующему шагу процесса и в появившемся диалоговом окне ввести имя триггера («Поступление товара на склад»). Затем щелкнуть по кнопке **<OK>**.

Для создания выходного триггера следует щелкнуть по кнопке **Create Outcome** и выполнить те же действия.

Для декомпозиции шагов процессов (функций) модели следует выделить родительский шаг процесса и выполнить команду **File/Open Down**. Эта процедура открывает новую дочернюю диаграмму, на которую автоматически передаются организационные единицы родительской диаграммы. Из них оставляют только те, которые необходимы для нового уровня декомпозиции, и при необходимости добавляют новые. Процесс построения дочерней модели аналогичен: на диаграмму помещаются функции (шаги процесса), потоки, места хранения и пункты принятия решений. На рис. 6.5 представлена дочерняя диаграмма для блока A5 (см. рис. 6.4).

Количество уровней иерархий в модели не ограничивается — декомпозиция шагов процесса осуществляется до тех пор, пока каждый шаг процесса можно будет реализовать одним программным модулем. Переход из диаграммы нижнего уровня на более высокий обеспечивается командой **File/Open Up**.

Для отображения на мониторе полученной модели процессов Process Modeler имеет три режима:

- **Symbol** (по умолчанию) — используется два вида графических объектов;
- **Enhanced Symbol** — расширенная версия, для каждого вида объектов используется свое графическое изображение;

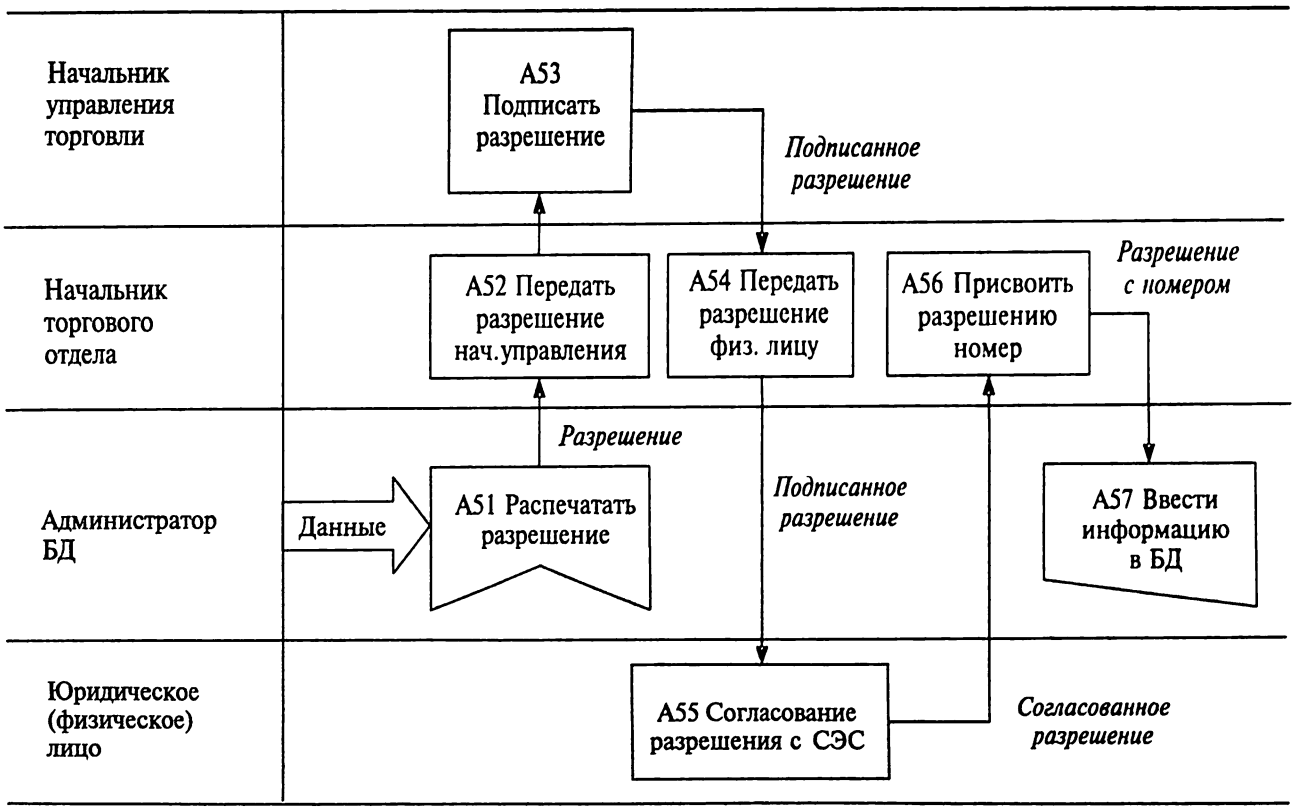


Рис. 6.5. *Диаграмма декомпозиции блока A5 «Оформить и выдать разрешение на торговлю»*

• **Iconic** — все объекты отображаются пиктограммами, которые в этом режиме оживляются, обеспечивая режим анимации.

Диаграммы в Process Modeler могут иметь мультимедийное расширение, поэтому можно показать, например, движущееся представление потоков от одного шага процесса к другому. Можно встроить видео- и аудиоклипы для подчеркивания или пояснения шагов процессов.

Допустимо создать ссылку на внешнюю программу, например для запуска презентационных слайдов, либо вывести электронную таблицу или график, отображающие операции процесса. Можно промоделировать временную последовательность после присваивания шагам процессов характеристик времени. Затем можно запустить диаграмму на выполнение, наблюдая последовательное выполнение шагов процессов, отображенных в виде пиктограмм, за установленное время.

При этом можно убедиться в одновременности наступления событий или выяснить узкие места в работе системы. Все отмеченные возможности позволяют в наглядной форме донести до заказчика и будущих пользователей цели и намерения разработчиков.

Для сохранения модели процессов следует выполнить команду **File/Save Diagram**, для закрытия Process Modeler — команду **File/Exit**.

На основе этой модели процесса можно в автоматическом режиме построить функциональную иерархическую модель системы «AS IS», в которой каждому шагу процесса будет соответствовать функциональный блок.

Утилита **Function Hierarchy Diagrammer** позволяет отображать функции системы в иерархическом или многоуровневом представлении. В отличие от IDEF0-модели эта диаграмма размещается на одном листе и более похожа на типичную диаграмму «дерево узлов» для IDEF0-модели.

На этой функциональной диаграмме представляется деятельность организации с точки зрения задач, но при этом не выделяются группы пользователей, отделов, потоков данных или мест хранения данных. Линии между функциями означают не потоки информации, а представляют взаимосвязи «родитель—потомок» (рис. 6.6).

В отличие от диаграммы «дерева узлов», в функциональной модели можно закрыть дочерние уровни, щелкнув дважды по красному кружку со знаком минус у родительской функции, либо показать дочерние уровни, щелкнув по красному кружку со знаком плюс. Для отображения всех дочерних и всех родительских функций необходимо выполнить команду **Layout/Expand Tree**.

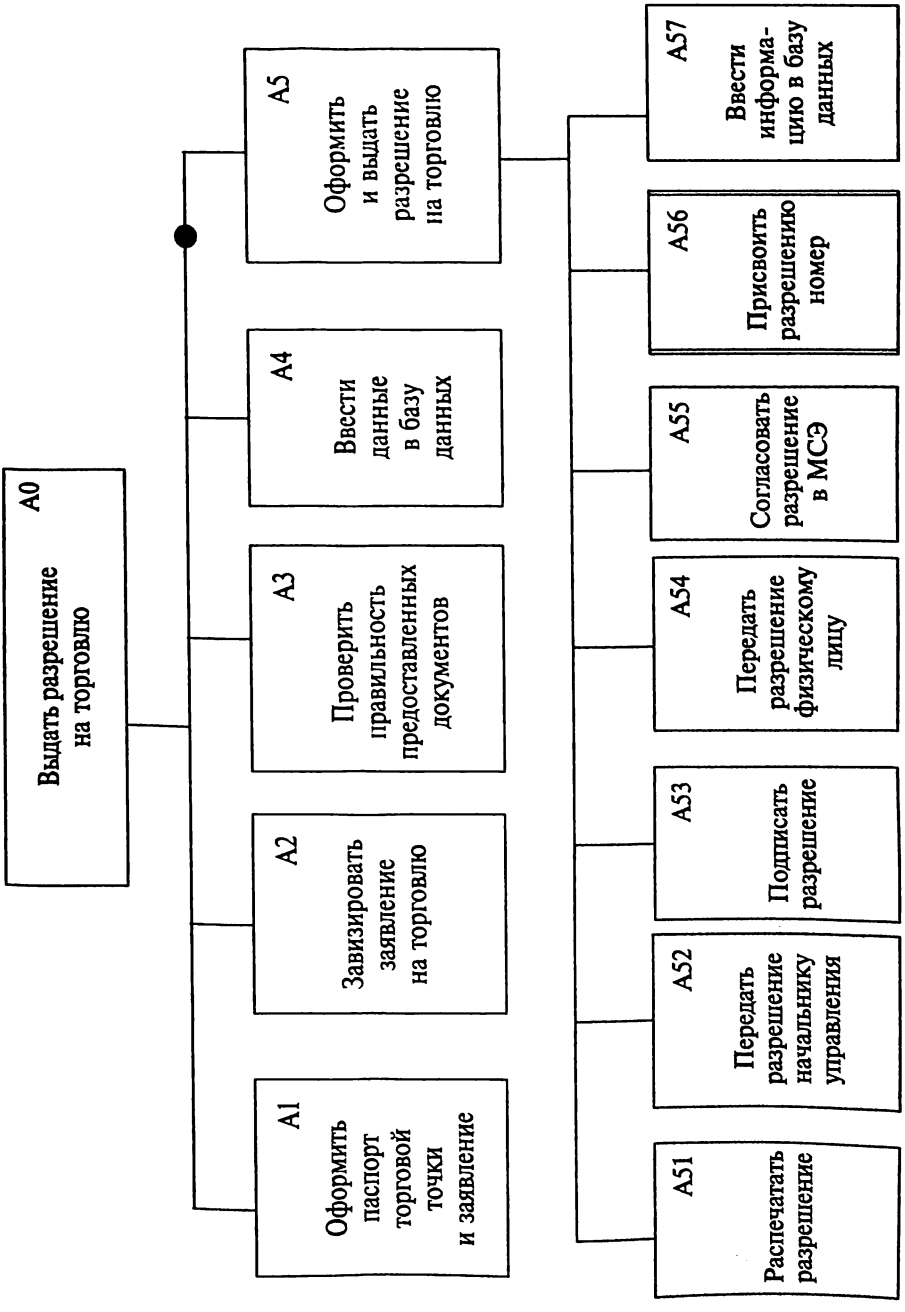


Рис. 6.6. Функциональная иерархическая диаграмма

На диаграмме можно вводить новые функции, перетаскивать дочерние функции к другим родителям, изменять функции родителей и т. д. Однако лучше делать эти изменения на модели процессов, которая является основным источником информации для проектировщиков, а функциональная диаграмма более полезна в качестве единого представления всех функций системы в их иерархической упорядоченности.

Для запуска режима построения функциональной диаграммы необходимо в диалоговом окне Designer 10g щелкнуть **по ярлыку Function Hierarchy**. На экран будет выдано диалоговое окно **Function Hierarchy Diagrammer**, в котором надо выполнить команду **File/New**, и в открывшемся диалоговом окне ввести имя функциональной диаграммы и имя модели процессов, на основе которой эта диаграмма будет строиться.

Щелкнув по кнопке **<OK>**, получают функциональную иерархическую диаграмму системы, которую можно легко редактировать: обеспечивать вертикальную, горизонтальную или гибридную компоновку; перемещать функции по диаграмме, изменять родительские (корневые) функции и т. п.

Для сохранения диаграммы выполняется команда **File/Save**. Результатом выполнения этапа является оформление отчета, который должен включать разделы, определяющие цели и ограничения при создании ИС, ее функции и задачи, входную и выходную информацию и т. д.

6.3. Этап разработки концепций ИС

6.3.1. Основные принципы реинжиниринга бизнес-процессов

Результатом этапа разработки концепций ИС является выбор оптимального варианта функциональной IDEF-модели «TO BE» из нескольких разработанных моделей, а также построение базы данных логического уровня и предложений по составу пользовательского интерфейса. Целью построения функциональных моделей обычно является выявление наиболее слабых и уязвимых мест деятельности организации, анализ преимуществ новых бизнес-процессов и степени изменения существующей структуры организации бизнеса [25].

Основным инструментом при построении модели «TO BE» является **Business process reengineering (BPR)**, который предполагает ради-

кальное переосмысление и перепроектирование бизнес-процессов для достижения резких, скачкообразных улучшений главных показателей деятельности компании, таких как стоимость, качество, сервис и темпы.

Понятие «реинжиниринг бизнес-процессов» лежит на стыке менеджмента и информационных технологий, поэтому требуется создание специальных инструментальных средств поддержки. Как правило, реинжиниринг сопровождается **внедрением новой информационной системы** как средства закрепления обновленных бизнес-процессов на предприятии. Кроме этого, для закрепления функций и областей ответственности необходима разработка должностных инструкций для сотрудников компании. При этом в новых условиях может потребоваться обучение и переподготовка специалистов.

Одной из особенностей реинжиниринга бизнес-процессов является то, что в его результате сотрудники (исполнители бизнес-процессов) наделяются возможностью самостоятельно принимать решения. Это повышает эффективность их работы, а также работы предприятия в целом, поскольку при этом уменьшается количество проверок, согласований с руководством и т. п. Кроме того, устраняется излишняя централизация, т. е. задачи выполняются в том подразделении, где это наиболее целесообразно.

Основным предметом реинжиниринга являются бизнес-процессы компании. При этом под **бизнес-процессом** понимают множество «внутренних шагов» предприятия, заканчивающихся созданием продукции, необходимой потребителю. Назначение каждого бизнес-процесса состоит в том, чтобы предложить потребителю продукцию (услугу), удовлетворяющую его по стоимости, сервису и качеству. При этом оптимизируется результативность бизнес-процесса путем его организации на основе упорядочения горизонтальных связей в структуре управления компанией.

Как следует из определения BPR, предполагается процессный подход к управлению при его реорганизации. Бизнес-процессы существовали всегда, но лишь относительно недавно некоторые компании стали использовать их как объект управления. Изначально более логичным решением считалось руководство структурными подразделениями, которые выполняют строго определенные функции и имеют свое руководство. Майкл Хаммер (Michael Hammer), крупнейший американский специалист в области управления, изобретатель понятия «реинжиниринг бизнес-процессов», рекомендовал называть процессы, используя границы «от» и «до», как, например, «от размещения

рекламы и до продажи товара». Именно данными мелочами и характеризуются бизнес-процессы, ведь они создаются с использованием множества связей между подразделениями, которые передают друг другу некоторое ключевое задание.

Каждый бизнес-процесс характеризуется:

- четко заданным во времени началом и концом;
- внешними интерфейсами, которые либо связывают его с другими бизнес-процессами внутри организации, либо описывают выход во внешнюю среду;
- последовательностью выполнения функций и правилами их выполнения (бизнес-правилами);
- для каждой функции, входящей в бизнес-процесс, определены ее место в общей последовательности работ, исполнитель, условия инициации, время.

Бизнес-процессы определяют прохождение потоков работ **независимо от иерархии и границ подразделений**, которые их выполняют. В силу этих обстоятельств реинжиниринг (реорганизация) бизнес-процессов направлен на решение следующих задач:

- в первую очередь, на выявление объективной структуры бизнес-процесса;
- во вторую очередь, на оптимальное распределение выявленных функций по структурным подразделениям и исполнителям и их автоматизацию с учетом ограничений на ресурсы.

Основная идея реинжиниринга — радикальное переосмысление с последующим перепроектированием предприятия — полностью основана на инженерном подходе. Описание бизнес-процессов для реинжиниринга представляет собой моделирование организации для последующего изменения модели согласно текущим и перспективным задачам организации. То есть сначала строится модель «Как есть», а затем, в результате реинжиниринга бизнес-процессов — «Как должно быть».

Бизнес-процессы весьма разнообразны, но существуют определенные требования, которым каждый из них должен соответствовать. Эти требования обеспечивают принципы организации бизнес-процессов, которые необходимо выполнять в ходе проведения реинжиниринга.

1. Горизонтальное сжатие бизнес-процесса. Несколько рабочих процедур объединяются в одну. Для перепроектированных процессов характерно отсутствие технологии «конвейера», в рамках которой на каждом рабочем месте выполняются простые задания или рабочие

процедуры. Теперь они интегрируются в одну — происходит **горизонтальное сжатие бизнес-процесса**. Если не удается привести все шаги процесса к одной работе, то создается команда, отвечающая за данный процесс. Горизонтальное сжатие ускоряет выполнение процесса примерно в десять раз.

2. Децентрализация ответственности (вертикальное сжатие процессов). Исполнители принимают самостоятельные решения в тех случаях, когда при традиционной организации работ он должен был обращаться к управленческой иерархии. Наделение сотрудников большими полномочиями и увеличение роли каждого из них в работе приводит к значительному повышению их отдачи.

3. Логика реализации бизнес-процессов. Линейное выполнение работ заменяется логическим порядком, т. е. реинжиниринг процессов освобождает от линейного упорядочения рабочих процедур, позволяя **распараллеливать процессы** там, где это возможно.

4. Диверсификация бизнес-процессов. Новые процессы имеют различные варианты исполнения, каждый вариант ориентирован только на одну соответствующую ему ситуацию. Поэтому в отличие от традиционных процессов они ясны и просты. В то время как традиционный процесс доложен выполняться одинаково для всех входов, приводя к согласованным выходам. Поэтому они оказываются очень сложными из-за излишней детализации и учета всевозможных исключений и частных случаев.

5. Рационализация горизонтальных связей. Работа выполняется в том месте, где это целесообразно. В традиционных компаниях она организуется по функциональным подразделениям: отдел заказов, транспортный отдел, отдел снабжения и т. п. Реинжиниринг распределяет работу между границами подразделений, **устраняя излишнюю интеграцию**, что приводит к повышению эффективности процесса в целом.

6. Рационализация управленческого воздействия. Уменьшается количество проверок и снижается степень управленческих воздействий, которые не приводят непосредственно к получению материальных ценностей. Задача реинжиниринга — сократить их до экономически целесообразного уровня. Вместо проверки каждого из выполняемых заданий перепроектированный процесс часто агрегирует эти задания и осуществляет проверки и управляющие воздействия в **отложенном режиме**, что заметно сокращает время и стоимость процессов.

7. Культура решения задачи. Минимизируется количество согласований, так как они тоже не имеют материальной ценности. Задача реинжиниринга состоит в **минимизации согласований** путем сокраще-

ния внешних контактов. Речь идет о стирании граней между функциональными подразделениями.

8. «Уполномоченный» менеджер обеспечивает **единую точку контакта.** Механизм «уполномоченного» менеджера применяется в тех случаях, когда шаги процесса либо сложны, либо распределены таким образом, что их не удается объединить силами небольшой команды. «Уполномоченный» менеджер играет роль буфера между сложным процессом и заказчиком, выступая ответственным за весь процесс. Менеджер должен быть способен отвечать на любые вопросы заказчика и решать его проблемы

9. Централизация информационной поддержки. При реинжиниринге необходимо совершенствовать информационную поддержку процессов. Современные информационные технологии позволяют действовать **полностью автономно** на уровне подразделений, сохраняя при этом возможность, пользоваться централизованными данными.

Задачи, которые приходится решать в ходе реинжиниринга, обычно характеризуются высокой степенью сложности и большой ответственностью. Рассмотрим построение модели «ТО ВЕ» на примере управления городом, используя для этого стилизованные диаграммы потоков данных «объектного» типа.

На рис. 6.2 и 6.7 представлена модель управления муниципальными предприятиями города «AS IS», при которой весь груз проблем по обеспечению своей жизнедеятельности (поиск поставщиков, закупка оборудования и материалов, оплата коммунальных услуг, выплата зарплаты и пр.) несут на себе муниципальные учреждения: школы, больницы, детские сады и т. п. При этом все эти действия выполняются неквалифицированными специалистами, которые не всегда правильно ориентируются в выборе поставщиков, качественного оборудования и товаров, а также возможны различные финансовые нарушения и махинации.

Модель «ТО ВЕ» управления финансовыми и материальными потоками муниципального образования построена по принципу корпорации, в которой централизован ряд финансово-хозяйственных процедур (рис. 6.8). При ее построении использовался принцип горизонтального сжатия бизнес-процессов: муниципальные предприятия были освобождены от функций поиска товаров и выполнения закупок, которые были переданы вновь созданному структурному подразделению — службе муниципальных закупок.

В этой модели служба закупок, собрав заявки от муниципальных учреждений города, например на приобретение компьютеров, осуще-

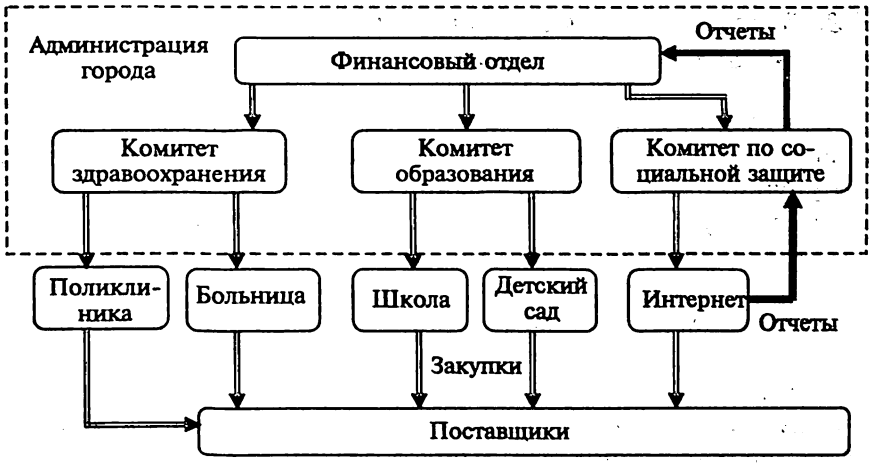


Рис. 6.7. Модель управления городом «AS IS»

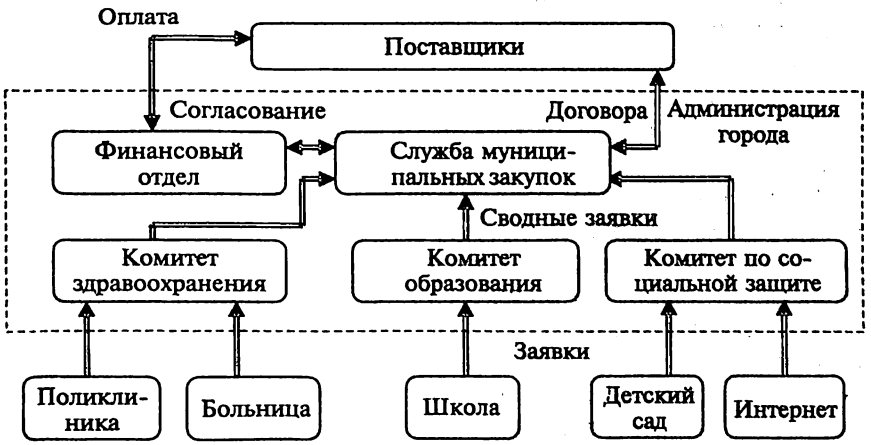


Рис. 6.8. Модель управления городом «TO BE»

ствляет крупные централизованные закупки, которые могут оказаться значительно выгоднее за счет объема и конкурсного отбора поставщиков. Закупками занимается узкий круг подготовленных специалистов, возможность различного рода финансовых нарушений существенно уменьшается. При этом учреждения освобождаются от составления всевозможных отчетов о закупках и затраченных средствах.

Переход на единую схему муниципальных закупок дает экономию в 15—20 %, автоматизация учета расходов местного бюджета может принести до 20 % экономии.

6.3.2. Построение функциональной модели «TO BE» на этапе разработки концепций ИС

На этом этапе системный аналитик должен выполнить тщательное обследование предметной области с позиций системного подхода:

- уточнить характер информации, циркулирующей внутри организации между подразделениями и отдельными сотрудниками;
- уточнить перечень выходных документов организации;
- уточнить перечень входных документов организации;
- уточнить характер информации, циркулирующей между данной организацией и различными организациями, относящимися к предметной области;
- разработать (унифицировать) формы документов, циркулирующих внутри организации между подразделениями и отдельными сотрудниками;
- разработать (унифицировать) формы документов, циркулирующих между организацией и различными организациями, относящимися к предметной области;
- разработать систему внутрифирменных нормативных документов, описывающих порядок работы отдельных подразделений и всей организации.

Модель «TO BE» отражает необходимые изменения бизнес-процессов с учетом внедрения ИС. Поэтому при построении модели следует максимально удовлетворять требованиям процессного подхода к управлению: правильно определять состав функций каждого бизнес-процесса на нижних уровнях иерархии. Выходом каждого бизнес-процесса должен воспользоваться хотя бы один клиент, необходимо максимально сократить количество подразделений, участвующих в бизнес-процессе, поскольку большинство проблем возникает на границах между подразделениями организации. Для этого необходимо либо изменить структуру подразделений, либо изменить поток работ.

При построении функциональной IDEF0-модели принципиально важно правильно определить состав функций на различных уровнях декомпозиции и сформировать содержание бизнес-процессов. Субъективность распределения функций по процессам может привести к противоречиям при дальнейшей работе с моделями: кто отвечает за процесс в целом, кому целесообразно делегировать те или иные функции, как распределена ответственность и т. д. При отображении бизнес-процессов верхнего уровня необходимо использовать функции одного уровня.

Для представления результатов исследования бизнес-процессов в виде **графической модели** необходимо руководствоваться следующими правилами:

- ограниченное количество функций на схеме (не более шести—восьми);
- все функции должны быть одного уровня;
- на схеме должна быть отображена информация по управлению процессом;
- все функции должны быть привязаны к подразделениям.

Первый шаг построения IDEF-моделей — построение контекстной диаграммы, для чего требуется определение внешних входов и выходов организации, нормативной документации, в соответствии с которой выполняются бизнес-процессы, и производственных (людских) ресурсов, которые являются владельцами этих бизнес-функций (рис. 6.9). В первую очередь рабочая группа выделяет входы и выходы, связанные с основной деятельностью организации. Обычно это набор входных и выходных документов, формируемых для внутренних нужд и отчетов перед вышестоящими организациями или партнерами.

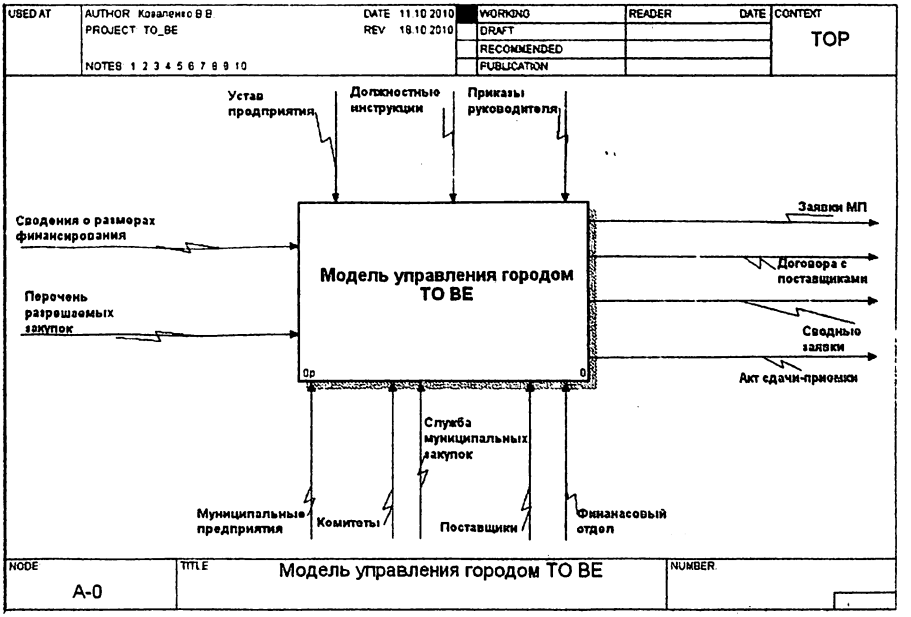


Рис. 6.9. Контекстная диаграмма модели «ТО ВЕ»

После определения входов и выходов составляется **предварительный перечень бизнес-процессов верхнего уровня** (например, бизнес-процесс сбыта готовой продукции, бизнес-процесс снабжения, бизнес-процесс производства готовой продукции и т. п.), после чего осуществляется распределение функций по процессам. Эта ответственная работа выполняется на основе результатов изучения предметной области и носит субъективный характер.

Первый уровень декомпозиции бизнес-процессов обычно соответствует процессам, которыми управляют заместители руководителя организации. Примером верхнего уровня может быть **процесс закупки сырья и материалов** для производства, который включает такие работы, как планирование закупок, заключение договоров, оформление заказов, получение товарно-материальных ценностей (ТМЦ), оплата ТМЦ, отпуск ТМЦ в производство [32]. Для нашего примера это уровень отделов администрации и муниципальных предприятий (рис. 6.10).

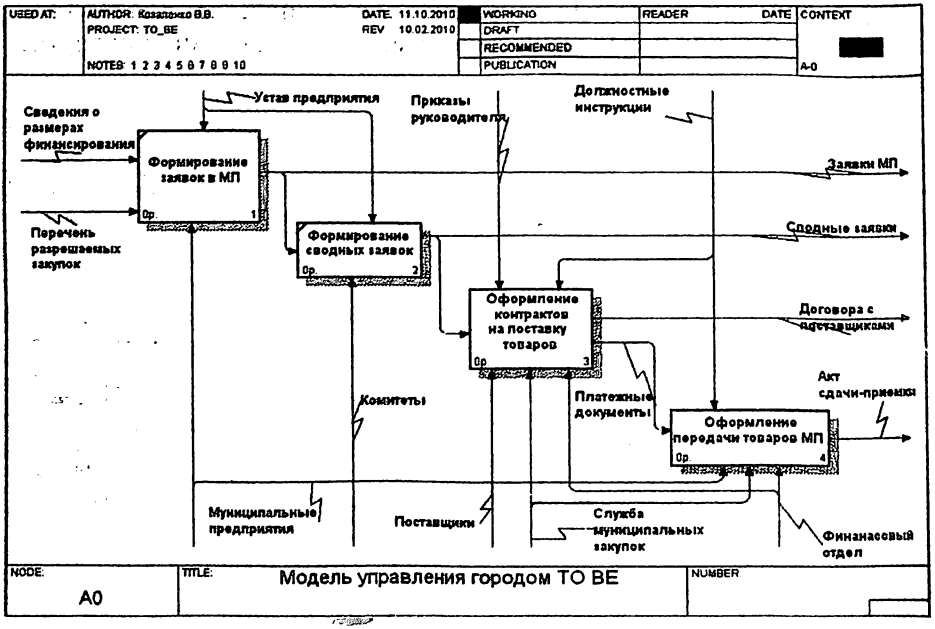


Рис. 6.10. Первый уровень декомпозиции бизнес-процессов в модели «TO BE»

Второй уровень декомпозиции рекомендуется рассматривать на уровне процессов крупных подразделений предприятия.

Третий уровень — это уровень функций подразделений и отделов, **четвертый уровень** — операции, выполняемые на рабочих местах, и т. д. Обычно для этих целей используется методология **IDEF3** (при необходимости моделирования большого количества логических операций) или методология **DFD** (при моделировании процессов документооборота).

Декомпозиция функций выполняется до тех пор, пока каждая функция нижнего уровня декомпозиции может быть реализована **одним программным модулем**.

Итоговый результат моделирования удобно представить в виде **диаграммы «дерева узлов»**, которая отображает все уровни иерархии модели «ТО BE» (рис. 6.11).

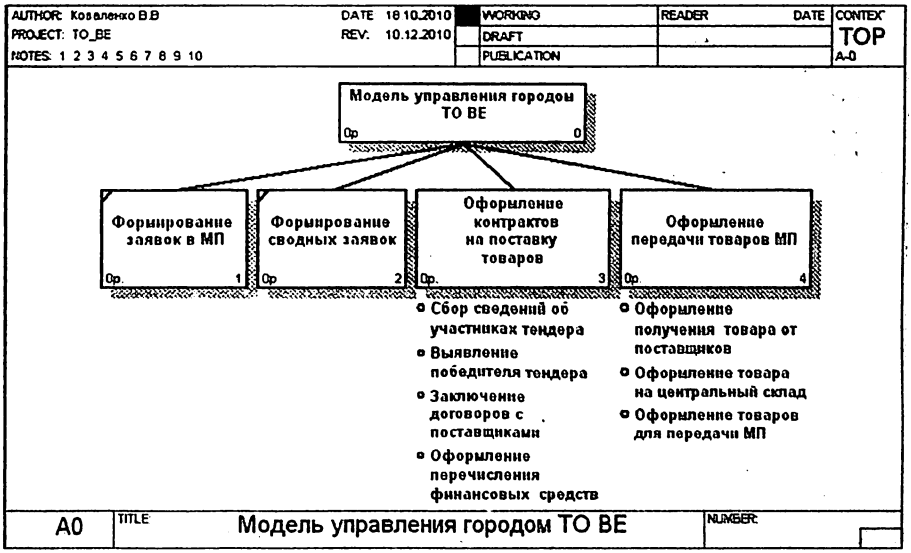


Рис. 6.11. «Дерево узлов» модели «ТО BE»

Как правило, создают несколько моделей «ТО BE», а затем на основе расчетов, например с помощью функционально-стоимостного анализа ABC, выбирают оптимальную модель, которая является основой для создания будущей ИС. Нижний уровень иерархии «дерева узлов» определяет состав **программных модулей системы.** Документооборот, представленный в модели, совместно с входными и выходными документами определяет основу для построения **базы данных логического уровня** (рис. 6.12).

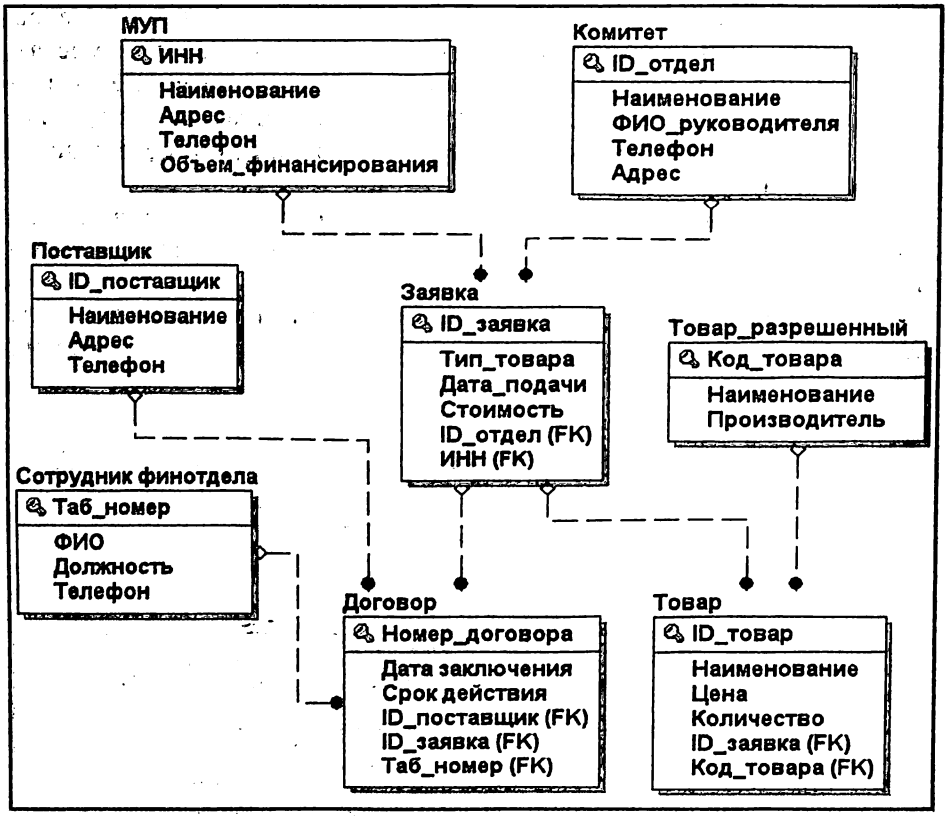


Рис. 6.12. База данных логического уровня

В свою очередь, набор программных модулей, полученный в результате построения функциональной модели, совместно с базой данных логического уровня определяет структуру **пользовательского интерфейса**, который на данном этапе представляется **деревом меню** (рис. 6.13).

В отличие от рассмотренного выше подхода, когда функциональная модель строится «сверху вниз», рядом авторов, проповедующих процессный подход, предлагается выстраивать модель **«снизу вверх»**, чтобы максимально эффективно реализовать основные процессы [21]. В этом случае происходит не распределение функций (дробить можно бесконечно), а объединение (заведомо конечное).

Тогда функциональная модель выстраивается самым оптимальным образом для конкретного бизнеса. Группировка функций у кон-

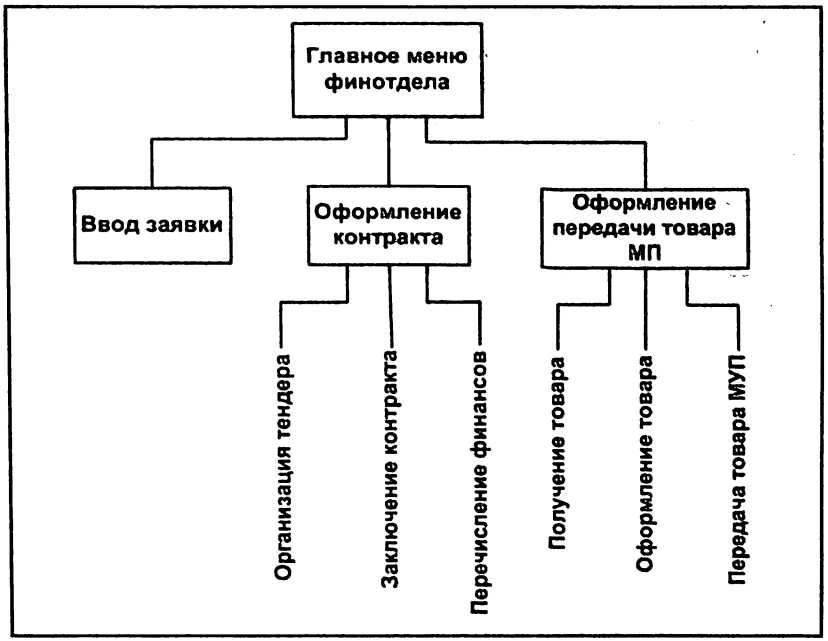


Рис. 6.13. Дерево меню информационной системы

кретных работников получается наиболее естественной, рациональной и, следовательно, эффективной, а их должностные инструкции становятся работоспособными, практичными.

Однако этот подход может иметь место для небольших предприятий, когда количество функций нижнего уровня иерархии незначительно и с ними со всеми можно работать одновременно. Для крупных предприятий этот подход можно использовать на этапе построения модели «ТО BE» для модификации и перегруппировки функций нижнего уровня иерархии в составе родительских функциональных блоков.

В данной главе достаточно подробно рассмотрена реализация двух этапов каскадной модели ЖЦ системы. Остальные этапы этой модели с меньшими подробностями представлены в **приложении А**.

Контрольные вопросы

1. Какова главная цель проектирования процессов?
2. Объяснить содержание классификации MuSCow.

3. Для каких целей на этапе формирования требований строится организационная структура предприятия и диаграмма потоков данных объектного уровня с внешними объектами?
4. Почему модель «AS IS» рекомендуется строить в виде набора диаграмм Swim Lane?
5. Каковы особенности построения модели «AS IS» в среде Process Modeler?
6. Содержание требований реинжиниринга бизнес-процессов к процессам: горизонтальное сжатие, децентрализация ответственности и т. п.
7. Для каких целей применяется реинжиниринг бизнес-процессов при построении модели «TO BE»?

Глава 7
КОМПЛЕКС СИСТЕМ УПРАВЛЕНИЯ ДЛЯ РЕАЛИЗАЦИИ УПРАВЛЕНЧЕСКОГО СТАНДАРТА CSRP

7.1. Интегрированные системы управления предприятием

В последнее время на предприятиях и фирмах различного уровня повысился интерес к системам управления предприятиями. На российском рынке представлен целый ряд программных продуктов, претендующих на роль таких систем и имеющих кардинальные различия между собой. Одна категория ИС создана с учетом стандартов ERP (ERP-системы), другая категория не отвечает этим стандартам (не ERP-системы).

С точки зрения базовых принципов, заложенных в идеологию представленных на российском рынке программных продуктов, можно выделить два направления их разработки и развития.

Первое направление (не ERP-системы) берет начало от автоматизации учетных бухгалтерских функций, а затем путем постоянной разработки и подключения новых модулей к системе автоматизации бухгалтерии создавалась информационная система. Продуктов такого типа достаточно много, к ним относятся практически все российские системы управления, так называемые корпоративные информационные системы (КИС) и некоторые заказные. С точки зрения производства такие системы разрабатывались «неестественным» путем, исходя из задачи первоначальной автоматизации финансовых функций. В этих системах управления производство оказывалось на втором плане после управления финансовыми ресурсами и было «непрозрачно» для управленческих кадров. Объединение различных модулей в составе такой не ERP-системы не позволяло обеспечить их интеграцию в соответствии с концепцией ERP.

Примером не ERP-системы может служить система комплексной автоматизации торговли (рис. 7.1), предназначенная для фирм, которым необходим полный оперативный учет и анализ внутрихозяйственной деятельности для принятия правильных и эффективных решений с ведением документации и автоматизацией торгово-финансовых операций.

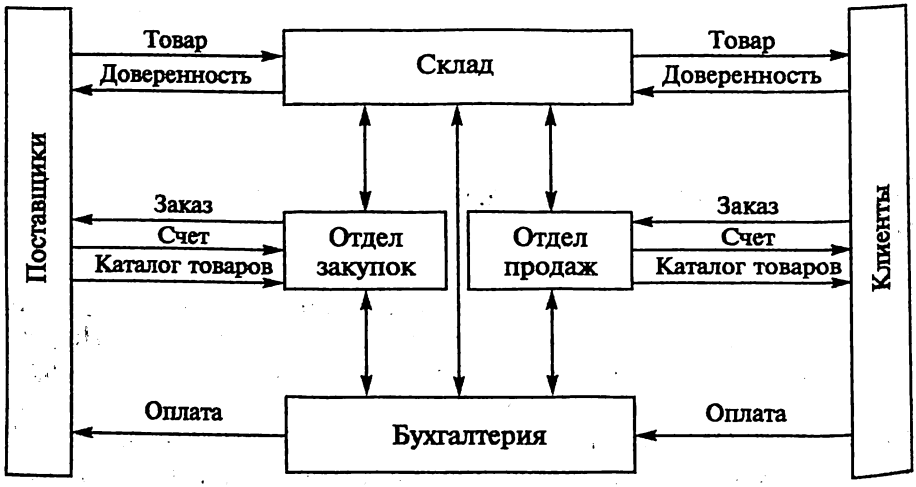


Рис. 7.1. Система комплексной автоматизации торговли

Не ERP-системы разрабатываются для выполнения лишь одной из общих функций, например финансовой.

Второе направление **(ERP-системы)** основано на автоматизации управления всеми ресурсами предприятия. Новые модули системы интегрировались с производственным ядром естественным путем, т. е. исходя из необходимости обеспечения производства материалами, компонентами, оборудованием, финансами и т. д.

Другим признаком ERP-системы является условие, при котором ядро системы создается на основе выполнения **пяти основных функций**, состав которых не зависит от последовательности выполнения технологических цепочек и структуры предприятия: планирование производства, подготовка производства, обеспечение производства, производство продукции и сбыт.

Результаты выполнения этих функций используются общими функциями, которые присутствуют на предприятии любого типа: руководство предприятием; функции поддержки (работа с кадрами, де-

лопроизводство, информационное и техническое обеспечение, юридическая деятельность и др.); финансовая деятельность; взаимодействие с дочерними предприятиями и фирмами.

Продуктов этого направления на российском рынке достаточно много — системы Oracle Application, Baan IV, R/3, MAX, Renaissance CS, Concorde, Syte Line, Exact, IFS Applications и другие. Следует отметить российские интегрированные ИС, которые по своей функциональности приближаются к ERP-системам: БООС (компания АйТи), «Парус» (корпорация «Парус»), «Галактика» (корпорация «Галактика») и др.

На рис. 7.2 представлен программный комплекс управления предприятием IFS Applications, который разработан шведской компанией Industrial & Financial Systems (IFS). Эта система адаптирована для российских условий и поставляется с исходными текстами программ. Комплекс является полномасштабным интегрированным продуктом, который входит в десятку лидеров систем класса ERP и представляет собой набор модулей, которые заказчик выбирает по своему усмотрению для своего проекта.

Состав модулей IFS-системы дает полное представление о функциональности ERP-системы.

IFS Поставки (Управление закупками, продажами, движением товарных потоков). Модуль реализует вопросы резервирования товаров, формирования планов/заказов на закупку и в производство, а также учет клиентов, поставщиков, долгосрочные и краткосрочные планы работы с ними и вопросы формирования цен, ведение прайс-листов. Модуль обеспечивает документирование процессов закупки, продажи, определение сроков выполнения заказов с учетом времени закупки/изготовления, транспортировки и учет календаря (рабочие, нерабочие дни) при ведении заказов. Ведется полный складской учет, управление движением материальных ценностей внутри предприятия содержит богатый набор аналитических функций и запросов.

Основные пользователи — работники подразделений снабжения и сбыта, складов, плановики.

IFS Производство (Управление производственными процессами). Модуль обеспечивает объемно-календарное планирование, планирование мощностей, ресурсов, потребностей в материалах (MRP). В полном объеме ведется учет спецификаций на изделия (связь с системой IFS Технолог), планирование и контроль операций, расчет себестоимости изделий и формирование производственных планов по

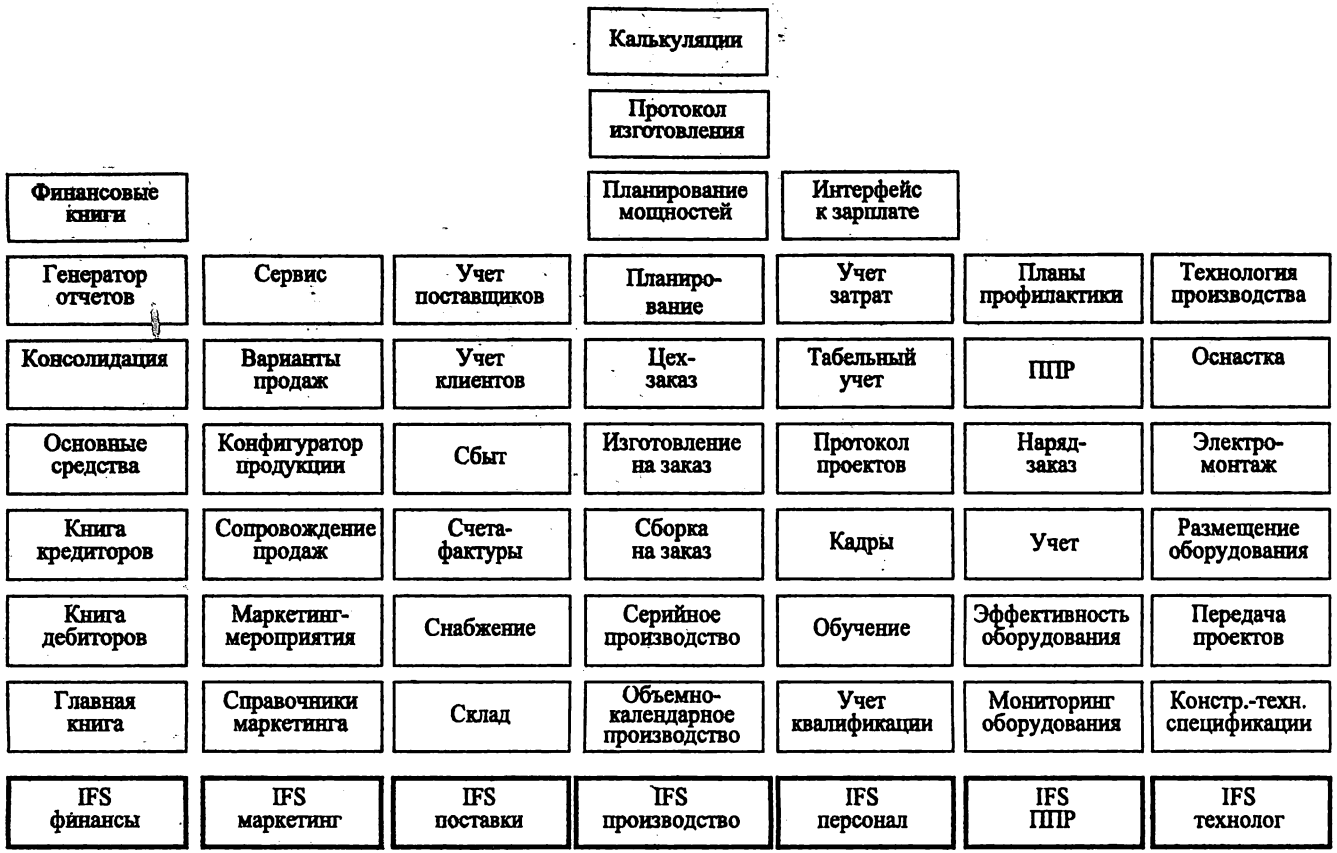


Рис. 7.2. Система комплексного управления предприятием IFS Applications

участкам, по изделиям. В модуле реализуется контроль сборочных процессов и сроков по всему технологическому процессу производства, контроль качества продукции. Модуль осуществляет ведение заявок и заказов на производство, координацию работ с модулем IFS Поставки.

Основные пользователи — управленческий, инженерный, технический персонал производственных участков.

IFS Технолог (Регистрация, учет и контроль конструкторско-технологической спецификации на продукцию). Модуль обеспечивает поддержку структуры изделий с учетом вариантов изменений и модификаций, контроль по срокам действия версий/модификаций и ведение извещений об изменениях. Модуль реализует ведение технологических карт с поддержкой замен и вариантов изготовления, поддержку технологической подготовки производства в части оснастки, размещения оборудования, электромонтажа, а также тесную координацию с модулем IFS Производство.

Основные пользователи — персонал технологических служб.

IFS ППР (Планово-предупредительные и текущие ремонты). В модуле реализована мощная система планирования, поддержки, учета работ по поддержанию работоспособности оборудования, применяемая в различных отраслях (энергетика, коммуникации, транспорт, перерабатывающая, пищевая промышленность, коммунальное хозяйство и др.). Ее составная часть — системы учета оборудования. Выполняется поддержка разнообразной (административной, технической и т. п.) документации, учет ресурсов, материалов, времени при планировании и проведении деятельности ППР, обеспечивается тесная связь практически со всеми IFS-модулями.

IFS Маркетинг. Модуль представляет собой мощный инструмент совершенствования деятельности маркетинговых служб. Обеспечивает подготовку предложений по продажам, представление продукции с нужной степенью детализации как для клиентов, так и для служб своего предприятия. Модуль позволяет формировать планы, контракты, заказы по конкретным клиентам, вести отчеты и истории контрактов, использовать имеющуюся информацию для тренировки и обучения сотрудников.

Основные пользователи — персонал маркетинговых служб.

IFS Персонал (Функции планирования и учета деятельности персонала фирмы). Модуль обеспечивает максимальную автоматизацию табельного учета (рабочее время, больничные, неполные рабочие дни,

нерабочие дни, сменность, возможные графики замен), почасовой учет рабочего времени в цехах.

Таким образом, ИС предприятия на базе стандартов ERP обеспечивают весь необходимый набор информационных услуг от планирования до управления складами и производством, от закупок сырья, материалов и комплектующих до сбыта готовой продукции, включая бухгалтерию и финансы.

В Главной книге ERP-системы естественным образом (обычно автоматически) собираются результаты из всех основных модулей.

Комплекс **IFS Application** построен в архитектуре трехуровневого клиент-сервера. Уровень хранения данных выполнен на современной промышленной СУБД Oracle. Уровень бизнес-логики реализован процедурными средствами СУБД Oracle на Oracle-хранимых процедурах.

При разработке или выборе ИС для предприятия необходимо вначале определить, какого класса система соответствует его размеру, характеру деятельности и возможностям. Экономически невыгодно применять ERP-системы для небольших предприятий, которые имеют простой производственный процесс, несложную организационную структуру, консерватизм к изменениям (это предприятия, не занимающиеся производством: научные, сервисные, социальные и т. п.; небольшие торговые компании; предприятия с простым производственным процессом). Для автоматизации управления такими предприятиями успешно используются локальные или малые системы не ERP-класса.

Крупные и средние предприятия, для которых первоочередное значение имеет управление производством, фактически не имеют альтернативы ERP-системам (крупные холдинговые промышленные предприятия, финансово-промышленные группы, управляющие компании, организации со сложной распределенной структурой, развивающиеся предприятия и организации).

Однако в настоящее время уровень ERP-системы уже не обеспечивает предприятию конкурентные преимущества. Появляется необходимость в расширении стандарта управления до CSRP, т. е. к функциям ERP-системы необходимо добавить функции систем управления взаимоотношениями с клиентами, OLAP-систем, систем электронной торговли и других. Поэтому в последующих разделах этой главы рассмотрим функциональность этих систем, «выводящих ERP-систему за ворота предприятия», а также особенности их проектирования и внедрения для конкретных предприятий.

7.2. Системы управления взаимоотношениями с клиентами (CRM-системы)

7.2.1. Общие сведения об управлении взаимоотношениями с клиентами

Термин «управление взаимоотношениями с клиентами» появился в России несколько лет назад, хотя на Западе над этой проблемой работают около 20 лет. CRM-системы обеспечивают интеграцию данных о клиентах и повышают качество их обслуживания, что в последнее время вызывает к ним интерес. Ведь сложно говорить об управлении взаимоотношениями с клиентами, если неизвестно, кто они, как с ними работали раньше, каковы их потребности и какую ценность представляет каждый из них для компании.

Основная задача **CRM-систем (Customer Relationship Management)** заключается в управлении циклом продаж, контроле над информацией о клиентах, увеличении эффективности процессов взаимодействия с клиентами. Наиболее эффективно применение CRM-систем на высококонкурентных рынках, где обеспечение клиентов качественным сервисом является обязательным условием.

Решения класса CRM представляют собой приложения для автоматизации, оптимизации и повышения эффективности бизнес-процессов, направленных на взаимодействие с клиентами за счет персонализации взаимоотношений в сферах продажи, маркетинга и обслуживания. Это позволяет компании обращаться к клиентам с интересными предложениями в наиболее удобный момент времени и наиболее удобным качеством связи.

В итоге эта стратегия приводит к существенному увеличению эффективности взаимодействия с клиентами за счет создания единого источника информации по клиентам и полной истории взаимоотношений с ними, персональных отношений с клиентами, привлечения и удержания клиентов, увеличения количества повторных продаж, повышения вероятности заключения потенциальных сделок.

В общем виде бизнес-задачи CRM для туристических фирм можно представить в следующем виде (рис. 7.3) [www.ls-crm.ru].

На технологическом уровне CRM-система представляет набор приложений, связанных единой бизнес-логикой и интегрированных в корпоративную информационную среду на основе единой базы данных, в которой ежедневно фиксируются:

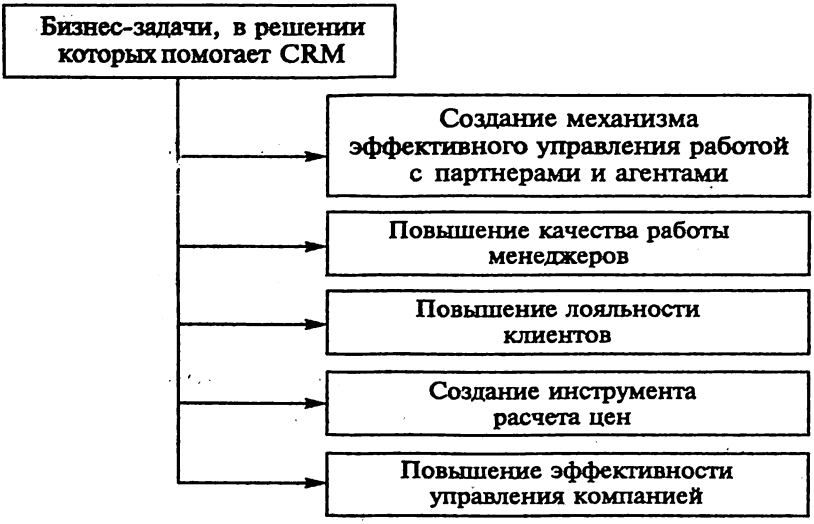


Рис. 7.3. Бизнес-задачи CRM-системы

- все контакты с настоящими и потенциальными клиентами;
- любая новая информация о покупателях;
- причины лояльности и источники информации покупателей;
- результаты маркетинговых воздействий и количество новых покупателей;
- заявленные претензии и рекламации;
- заказы и контроль их исполнения;
- счета на оплату и контроль оплаты;
- информация о конкурентах;
- неудовлетворенный спрос.

Системы категории CRM чрезвычайно различаются по функциональности, и их можно классифицировать по нескольким признакам. Наиболее распространенный способ классификации **по целевому использованию**: оперативные и аналитические [1]. Однако это деление довольно условное, поскольку одна CRM-система может включать и оперативные, и аналитические модули.

Оперативные CRM содержат функции маркетинга, продажи и сервиса, т. е. соответствуют стадиям привлечения клиента, самого акта совершения сделки и послепродажного обслуживания. Системы этого класса осуществляют взаимодействие предприятие—клиент, обеспечивая непосредственный доступ к информации при работе с клиентом в ходе продаж и обслуживания.

Основным компонентом такой системы является приложение, которое позволяет сотрудникам вносить накопленную информацию по отдельным клиентам в базу данных и затем эффективно ее использовать.

Примером подобного рода российских систем является **Sales Expert** (компания «Про-Инвест») для малых предприятий и **«Управление деловыми процессами. Парус-Клиент»** (компания «Парус») для крупных предприятий. В любом случае системы этого типа используются на предприятиях, которые имеют небольшой штат сотрудников (сервисный персонал, менеджеры по продажам), работающих с клиентами, или специфика выпускаемой продукции такова, что количество клиентов невелико.

Аналитические CRM начинают использовать в тех случаях, когда компания имеет большую базу данных о клиентах. Такие системы позволяют реализовать совместный анализ данных, характеризующих деятельность как клиента, так и фирмы, получить новые знания, выводы и рекомендации.

Они представляют собой мощный инструментарий для полномасштабного анализа клиентской базы и моделирования. Здесь часто используются сложные математические модели, трехмерная визуализация, осуществляется поиск статистических закономерностей для выработки наиболее эффективной стратегии маркетинга, продаж и обслуживания клиента.

Из российских систем хорошие аналитические возможности имеет система **Marketing Analitic** компании «Курс», **Marketing Expert** компании «Про-Инвест». Ряд фирм выпускают целые линейки программных продуктов с различной функциональностью. Например, фирма «Эксперт системс» предлагает четыре вида CRM-систем:

Quick Sales Free — бесплатная версия для одного пользователя с интуитивно понятным интерфейсом обеспечивает ведение клиентской базы, простейшую аналитическую обработку информации, построение отчетов и т. д.

Quick Sales — особенно удобна для неподготовленных пользователей своей простотой и удобством пользовательского интерфейса, является идеальным вариантом для тех, кто понимает, что держать все в голове или вести клиентов в Excel крайне неудобно. Позволяет легко искать клиентов, строить отчеты, раздавать права доступа и т. п.

Sales Expert представляет собой гибкое настраиваемое решение для продвинутых пользователей, ее иногда классифицируют в качестве маркетинговой информационной системы. В конфигураторе есть возможность редактировать поля, изменять вид карточки клиентов,

формировать простые и расширенные запросы, обладает мощной аналитикой и многим другим.

Монитор CRM предназначается для среднего бизнеса, легко интегрируется с большим количеством распространенных бухгалтерских систем, в системе реализована настройка интерфейса и конфигурации под вкусы пользователя.

Из главного окна CRM-системы **Quick Sales** (рис. 7.4) есть быстрый доступ к наиболее часто используемым модулям (панель слева): Клиенты (управление клиентами), Календарь (распорядок дня), Работы (анализ продаж), Пакеты (отправка информации по почте), Отчеты (отчеты по продажам), База знаний (база знаний для службы поддержки и сервиса) и Воронка (воронка продаж).

Крупным фирмам можно предложить комплексное использование оперативных и аналитических CRM-систем. Рекомендуется использовать CRM-системы и компоненты финансового контура одного производителя, поскольку очень важна четкая стыковка всех модулей, особенно когда данные из CRM-системы активно используются системой управления цепочками поставок (**SCM**), а оттуда информация передается в складские и торговые модули. Такое решение обладает множеством достоинств, но дорого и требует серьезной постановки задачи.

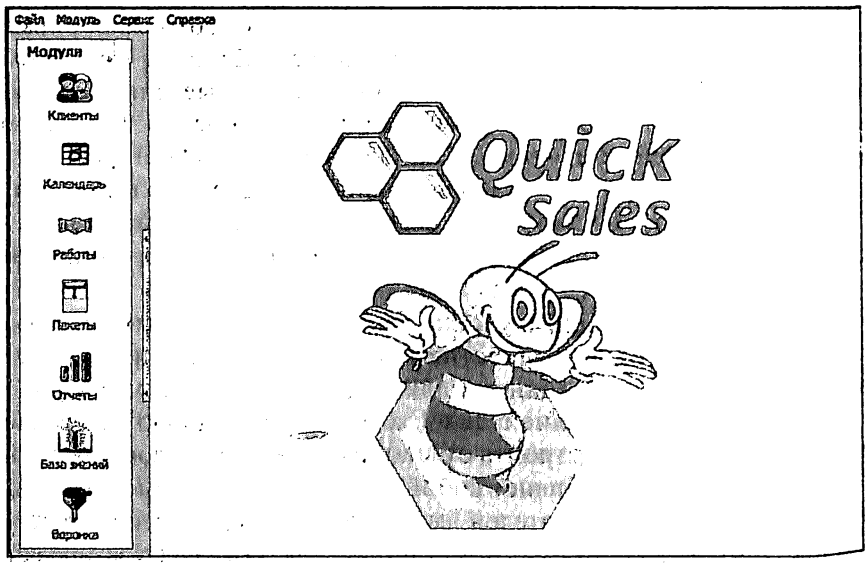


Рис. 7.4. Главное окно CRM-системы Quick Sales

Существует еще одна достаточно **популярная классификация CRM-систем**, основанная на размере предприятия, которая содержит три группы.

1. Системы для офисной работы с клиентами и организации документооборота.
2. Системы для дистанционной работы с клиентами и приема заказов на услуги и товары через Internet.
3. Call-центры, позволяющие автоматизировать и хранить информацию о телефонных переговорах с клиентами.

Первый тип систем нужен для небольших фирм, чтобы нормализовать работу с клиентами, повысить ее продуктивность. В них обычно используются стандартные решения, имеющие небольшой диапазон настроек, которые могут регулироваться самим пользователем. Такие системы можно запускать быстро и эффективно, они почти всегда приносят хотя бы небольшой эффект. Цены на такие системы от 150 до 300 долл. на одно рабочее место.

Рекомендуется использовать такой модуль во взаимодействии с установленной учетной или торговой компьютерной системой. Тогда, например, торговое приложение позволит отпустить клиенту только те товары и по тем ценам, которые прописаны в договоре, заполненном в модуле CRM.

Системы для дистанционной работы иногда называют интернет-заказом, они ориентированы на средние и крупные организации и рассчитаны на заказ товаров постоянными клиентами непосредственно через Интернет. В этом случае предварительно происходят необходимые проверки уровня доступа к базе данных, во время диалога с клиентом автоматически проверяется наличие товара, доступность его данному клиенту, назначаются соответствующие скидки. Клиент может также просмотреть историю своих предыдущих заказов, сведения о запрошенных им ранее товарах и т. д.

Такие системы требуют учета специфики каждой организации или каждого вида ее деятельности: формирование скидок, подготовки справочников, отчетности и т. д. Соответственно цены на эти системы гораздо выше.

Call-центры также предназначены для средних и крупных предприятий. Они обеспечивают в автоматическом и полуавтоматическом режимах сбор сведений о телефонных контактах. Системы для дистанционной работы имеют свою базу знаний, позволяющую давать ответы на типичные вопросы. Она может быть настроена на автоматическую генерацию ответов. Обеспечивается прослушивание всей

истории переговоров с клиентом, документальное обоснование тех или иных действий. Call-центры недешевы и требуют специальной настройки при установке.

Довольно часто предлагается комплексное решение, когда CRM идет как модуль, который без проблем объединяется с ERP-компонентами в рамках единой информационной базы. Однако чаще CRM-системы интегрированы только с финансовой подсистемой, поэтому даже однократный ввод документов в торговой системе сразу же отражается в разделе расчетов с клиентом.

Если CRM-система используется как составная часть ERP- компонент компании, то она становится центральным звеном IT-инфраструктуры, где сходятся основные бизнес-процессы компании (http://www.mscrm.ru). Примером такого решения может служить система **Microsoft CRM** (рис. 7.5), которая обеспечивает:

- объединение существующих информационных систем компании в единое решение;
- поддержку интернет-технологий со всех трех точек зрения;

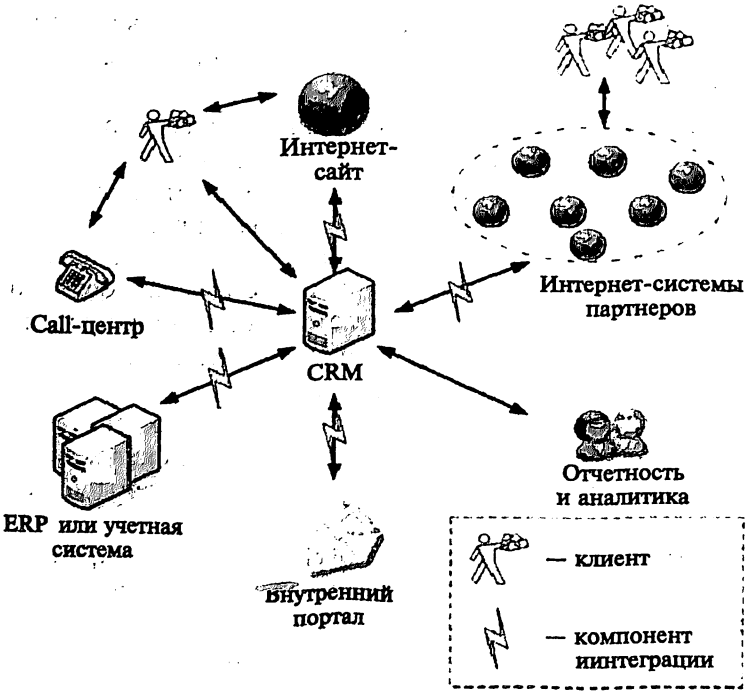


Рис. 7.5. CRM-система как центральное звено ИТ-инфраструктуры

• инфраструктуру, готовую к практической реализации CRM-стратегии.

Результаты от внедрения CRM можно оценить с помощью среднерыночных показателей:

• административные издержки снижаются на 10—20 %;

• объем продаж увеличивается на 10—30 % за счет более качественного обслуживания, ускорения процессов контроля над всеми этапами сделки и возможности реагировать на сбои;

• объем заключенных сделок увеличивается на 5—15 %;

• степень удовлетворенности клиентов увеличивается на 3—10 %.

Кроме этого, появляется возможность анализировать и оценивать работу каждого сотрудника, сбытового подразделения в целом, отдельно по регионам или другим переменным. Предоставляется также возможность анализировать сбытовой оборот в целом и по каждому продукту.

К новейшему поколению решений для управления продажами, мероприятиями и взаимоотношениями с клиентами относятся системы класса **CRM OnDemand** («CRM по требованию»), построенные по модели **SaaS** («Software as a Service» или «Программное обеспечение как услуга») (рис. 7.6).

CRM OnDemand предоставляется пользователям через Интернет по подписке за фиксированную ежемесячную арендную плату за каждого пользователя. Все программное обеспечение и база данных располагаются на сервере провайдера и доступны через интернет-браузер. Клиенты могут воспользоваться решениями CRM OnDemand легко, быстро и по доступной цене, без предварительных затрат на закупку лицензий, оборудование или контрактов на сопровождение.

Система «**CRM Libra OnDemand**» разработана специально для гостиничной индустрии и предоставляет полноценную функциональ-

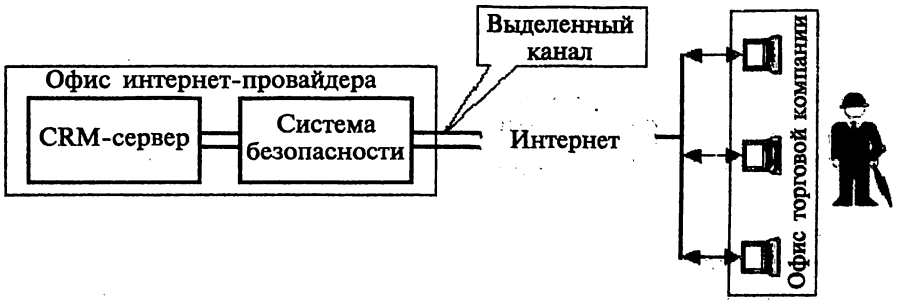


Рис. 7.6. Блок-схема системы класса «CRM по требованию»

ность по управлению взаимоотношениями с клиентами, программами лояльности, клубными программами, продажами и мероприятиями в гостиничной отрасли. Она автоматизирует бизнес-процессы отдела продаж и маркетинга, бронирования и банкетной службы в гостинице, что позволяет полностью контролировать работу менеджеров по продажам и маркетологов, организовывать успешные мероприятия и собирать максимально подробную статистику по каждому гостю [www.libraondemand.com].

Данный продукт уже полностью интегрирован с системами управления гостиницей «**Epitome PMS**», рестораном «**Epitome POS**» и существенно расширяет функциональные возможности этих систем.

Другая подобная система «**Siebel CRM OnDemand**» фирмы «Siebel Systems, Inc.» обладает полноценной функциональностью в сфере продаж, маркетинга и обслуживания; встроенной системой клиентского анализа; технологией виртуального информационного центра.

7.2.2. Особенности проектирования и внедрения CRM-систем

Миф о том, что разработать и внедрить CRM-систему легче, чем, например, ERP-систему, неоднократно опровергнут на практике. CRM-система действительно в технологическом плане гораздо проще, но при проектировании требует более мощной консалтинговой поддержки. Рабочим материалом в данном случае выступают уже **не ресурсы предприятия**, а клиенты, заказчики, т. е. люди. Поэтому малейшая ошибка в настройке, например, продаж может вызвать серьезные последствия.

Для каждого заказчика разрабатывается индивидуальный CRM-проект, учитывающий особенности бизнес-процессов и организационной структуры предприятия, а также и его клиентов. Проектирование CRM-систем осуществляется на основе **типовых моделей ЖЦ**, однако присутствует ряд особенностей, особенно на начальных этапах (рис. 7.7).

На первом этапе, как обычно, проводится системный анализ предметной области предприятия, но функциональная **модель «AS IS»** строится для **основных бизнес-процессов** работы с клиентами.

После построения модели «AS IS» необходимо **определить CRM-стратегию**, которая предусматривает поэтапную фокусировку бизнеса на потребностях клиентов. Извне этот процесс видится как поступательное движение по пути совершенствования управления

Рис. 7.7. Модель ЖЦ CRM-системы

взаимоотношениями с клиентами и неизбежно сопровождается внутренней трансформацией предприятия. При этом в изменения вовлечены и руководители высшего звена, и менеджеры, и все сотрудники (http://www.mscrm.ru).

Стратегия CRM не является набором универсальных правил, которые готовы к немедленному применению: ее нельзя купить или внедрить локально на отдельных участках бизнес-процессов. Она представляет собой эффективную организацию работы с клиентами, стратегию действий компании, которая основывается на регулярном анализе взаимоотношений с клиентами.

Формирование CRM-стратегии обычно начинают с разработки **маркетинговой стратегии**, которую можно представить в виде пяти этапов:

- сегментация клиентов (объединение в группы);
- определить ключевые факторы успеха для каждой из этих групп;

• оценить каждую группу клиентов с точки зрения ее привлекательности для бизнеса фирмы;

• выбрать целевые группы клиентов, наиболее привлекательные для фирмы;

• определить маркетинговую политику в отношении каждой из целевых групп клиентов [41].

Сегментация клиентов (объединение в группы) производится на основе схожести потребностей и поведенческих характеристиках клиентов: они должны быть одинаковы в каждой из групп, но значительно отличаться между группами. Например, можно разделить клиентов на небольшие предприятия, занимающиеся мелкооптовыми и розничными продажами, а также крупные торговые сети и строительные компании, если фирма занимается производством строительных материалов.

Группировка или сегментация клиентов, с одной стороны, огрубляет картину рынка, поскольку невозможно точно классифицировать клиентов и учесть при этом все их многообразие. С другой стороны, отсутствие групп с большой вероятностью сведет на нет все дальнейшие усилия системных аналитиков — рынок слишком сложен, чтобы описать все его многообразие в единой модели.

При определении ключевых факторов успеха для каждой из групп клиентов необходимо ответить на вопрос: что наиболее важно для достижения успеха при работе с данной группой. Например, при работе с мелкими и средними клиентами важны типовые отлаженные бизнес-процессы, а при работе с крупными клиентами — индивидуальный подход.

Исходя из выделенных ключевых факторов успеха, необходимо определить конкурентоспособность фирмы, т. е. насколько фирма заказчика способна добиться успеха при работе с исследуемой группой клиентов, другими словами, насколько клиенты заинтересованы в фирме, ее продукте или услуге.

При оценке каждой группы клиентов с точки зрения ее привлекательности для бизнеса фирмы необходимо ответить на вопрос, какой она имеет потенциал продаж, уровень конкуренции, насколько динамично развиваются входящие в нее компании.

Затем надо **выбрать целевые группы**, исходя из того, насколько они, с одной стороны, интересны бизнесу фирмы, а с другой — фирма сама интересна этой группе.

На последнем этапе **определяется маркетинговая политика** в отношении каждой из выбранных целевых групп: определяются мero-

приятия в части улучшения продукции, ценообразования, формирования политики продвижения товара и организации сбыта.

И только после выполнения всех этих действий можно переходить **к формулировке CRM-стратегии**. Например, если системные аналитики совместно с заказчиком решили, что главное для работы с этой группой клиентов — это качество продукта или услуги, то CRM-стратегия должна быть направлена на сбор и анализ требований клиентов, реализацию эффективной сервисной поддержки, возможно вовлечение потребителя в процесс создания продукта или услуги. Если же основная проблема — найти и убедить клиента в покупке продукта, то CRM должна помочь организовать бизнес-процессы поиска клиентов и работы с ними, осуществить анализ и контроль их функционирования.

После формулировки CRM-стратегии на основе анализа клиентов ее следует **уточнить**, проанализировав организационную структуру самой фирмы и состав ее продукции с точки зрения возможного состава клиентов. Этот вид анализа следует проводить вместо анализа клиентов, если фирма только начинает свою деятельность и не имеет постоянного контингента клиентов.

Проведем подобный анализ на примере **фирмы по продаже дорогостоящей уникальной техники**. Продажа на таких фирмах обычно производится под заказ, повторные продажи практически отсутствуют. Работа с клиентом состоит из большого количества этапов, в которых задействованы многие сотрудники из различных подразделений. Со стороны клиента в принятии решения о выборе поставщика также участвует большое количество людей. В некоторых случаях для выбора поставщика может проводиться тендер.

Для такого типа компаний можно выделить **три основных бизнес-процесса** работы с клиентом:

- продажа вплоть до заключения контракта;
- исполнение контрактных обязательств, включая поставку техники и монтаж;
- послепродажное и гарантийное обслуживание клиента.

Опыт работы подобных компаний показывает, что ключевыми факторами для них в борьбе за клиента являются:

- гибкость в предложении товара и способность предоставлять комплексный набор услуг;
- эффективность прямых продаж, включающих поиск клиента, формирование предложения и работу с лицами, влияющими на принятие решений;

• репутация продавца, которая зависит от качества и точности исполнения контрактных обязательств перед клиентами.

Именно эти факторы определяют, кого из продавцов выберет клиент. Для достижения преимущества по всей совокупности перечисленных факторов в CRM-системе необходимо обеспечить качественную организацию, планирование и контроль работы с клиентами. То есть CRM-стратегия должна фокусироваться на ***отладке всех бизнес-процессов работы с клиентами, а также планировании и контроле работ в рамках этих бизнес-процессов***.

В конце третьего этапа на основе сформированной CRM-стратегии разрабатываются регламенты и правила, процессы по работе с клиентами, должностные инструкции сотрудников.

Четвертым этапом проектирования является **выбор функций** будущей CRM-системы на основе результатов CRM-стратегии с одновременным построением функциональных моделей «TO BE». С учетом трех **основных бизнес-процессов** работы с клиентом, определенных на этапе анализа CRM-стратегии, производится выбор функций на примере оперативной CRM-системы [6], которая обычно реализует автоматизацию продаж и автоматизацию обслуживания клиентов.

Автоматизация продаж является ядром CRM-системы и должна содержать следующие функции:

• ведение календаря событий и планирование работы;

• управление контактами (ни одно обращение клиента не будет пропущено);

• работа с клиентами (каждый клиент будет обслужен на высочайшем уровне благодаря зафиксированной истории взаимодействия с ним);

• повышение точности прогноза продаж;

• предоставление информации о ценах;

• автоматическая подготовка коммерческих предложений;

• автоматическое формирование отчетов по результатам деятельности;

• организация продаж по телефону (создание списка потенциальных клиентов, автоматический набор номеров, регистрация звонков, прием заказов) и ряд других вспомогательных функций.

Автоматизация обслуживания клиентов в последнее время приобретает первостепенное значение, так как в условиях жесткой конкуренции удержать прибыльного клиента можно прежде всего высоким качеством обслуживания. Для быстрого, точного и эффективного

удовлетворения индивидуальных потребностей клиентов необходимо обеспечить выполнение в CRM-системе следующих функций:

- мониторинг потребностей клиента (сотрудник отдела всегда в курсе проблем и предпочтений каждого клиента);
- мониторинг прохождения заявок (процесс отслеживается автоматически);
- мониторинг продаж (в любой момент времени можно получить информацию о качестве выполненной услуги, ее стоимости, удовлетворенности клиентов, сроках выполнения заявки и др.);
- ведение базы знаний;
- контроль выполнения сервисных соглашений (сроки и условия отслеживаются автоматически);
- управление запросами клиентов с помощью присвоения приоритетов.

Кроме указанных можно указать общие функции для этих двух направлений:

- составление отчетов для высшего руководства;
- интеграция с ERP-системой;
- электронная торговля (управление закупками с помощью систем электронной коммерции типа B2B и B2C).

После завершения формирования функциональности CRM-системы в виде функциональной модели «ТО BE» проектирование системы продолжается в соответствии с этапами каскадной (см. рис. 7.7) или иной модели ЖЦ: разрабатывается база данных логического уровня, дерево меню, техническое задание, технический проект и т. д.

Следует иметь в виду, что не существует единых методик и технологий проектирования CRM-систем, применимых ко всем компаниям. Система управления взаимоотношениями с клиентами должна строиться **на основе маркетинговых целей и стратегий**. Только в этом случае CRM будет работать на повышение прибыли компании в краткосрочной и долгосрочной перспективе.

С учетом выбранной функциональности CRM-системы можно произвести реорганизацию бизнес-процессов в сфере продаж, маркетинга и обслуживания клиентов, построив функциональную модель «ТО BE» **для самой фирмы**.

Однако в последнее время ряд специалистов утверждают, что такого рода формальная реорганизация с целью подогнать бизнес-процессы под развертываемую в будущем CRM-систему может сделать работу компании нестабильной. Поэтому в настоящий момент, как правило, рекомендуется сначала внедрить в компании CRM-систему

и только затем перестраивать и улучшать бизнес-процессы самой фирмы, фиксируя изменения в настройках CRM-системы.

Внедрению CRM-системы должна предшествовать подготовка структуры предприятия. Подразумевается наличие уже сформированной клиентской базы данных, которая в дальнейшем составит ядро системы, налаженной системы сбора информации для этой базы, а также опыта анализа полученных данных. В принципе возможны три варианта внедрения покупной CRM-системы:

- внедрение полностью выполняется собственными силами IT-подразделения компании;
- внедрение осуществляется совместными силами заказчика и поставщика CRM-системы;
- поставка CRM-системы «под ключ», когда заказчик в результате получает функционирующую и полностью настроенную под свои бизнес-процессы систему.

Первый вариант оправдан в том случае, если компания располагает достаточно мощным IT-подразделением и не слишком ограничена сроками внедрения. В этом случае в рамках компании создается рабочая группа, которая обучается непосредственно в процессе внедрения и в дальнейшем занимается сопровождением. При этом варианте сокращаются прямые затраты на внедрение CRM, но появляется опасность надолго «застрять» на этапе внедрения.

Во втором случае в компании также создается рабочая группа, тесно взаимодействующая с представителями компании-поставщика, которая производит экспресс-диагностику бизнес-процессов для выявления уязвимых и проблемных мест. По ее результатам уточняются этапы модели ЖЦ для данной компании. При таком подходе время внедрения сокращается, однако сохраняется опасность «застрять» на последующих этапах, так как основную работу выполняют специалисты самой компании-заказчика.

При поставке CRM-системы «под ключ» компания-поставщик выполняет проектирование системы, запуск ее в эксплуатацию и оказание технологической поддержки. Это самый дорогостоящий для заказчика вариант, при котором существует опасность, что после ухода специалистов компании-поставщика компания-заказчик опять окажется «лицом к лицу» со старыми проблемами. Нередки случаи, когда спустя два-три месяца оборот компании заказчика снижается почти до прежнего уровня.

Срок окупаемости CRM-системы в среднем составляет около одного года.

7.3. Системы электронной коммерции типа B2B

7.3.1. Общие сведения о системах B2B

В последнее время предприятия активно используют Интернет не только в качестве мощного инструмента маркетинга, но также и в качестве инструмента прямых продаж с помощью систем электронной коммерции. Под **электронной коммерцией** (**e-commerce**) понимают совокупность технических и организационных форм, предназначенных для ведения коммерческой деятельности и совершения сделок с использованием электронных систем и сети Интернет как средства взаимодействия с партнерами, банком, поставщиками, потребителями товаров и услуг. В системах электронной коммерции обычно присутствуют все этапы совершения сделки: поиск требуемой продукции или услуг, уточнение деталей, оплата, получение (доставка) заказа.

Электронный бизнес (**e-business**) является весьма близким к электронной коммерции понятием. Под ним понимается вся совокупность производственных и организационных отношений между работниками одного предприятия, между различными предприятиями, государством, учреждениями и общественными организациями, причем взаимодействие этих субъектов происходит непосредственно в электронной форме, через сеть Интернет. Системы электронного бизнеса, в отличие от систем электронной коммерции, могут иметь или не иметь коммерческой составляющей.

Например, системы электронной коммерции типа **B2C** (**Business-to-Consumer**) обеспечивают коммерческие взаимоотношения между организацией (Business), обычно в лице **интернет-магазина**, и частным, так называемым конечным потребителем (Consumer).

Основные обороты электронной коммерции приходятся на системы типа «**business-to-business**» (**B2B**), которые являются не просто средством заключения сделок между фирмами, а реализуют технологию поддержки межкорпоративных отношений. Их основная задача состоит в повышении эффективности работы компаний за счет снижения затрат на подготовку торговых процедур и расширения географии бизнеса до масштаба всего мира.

Межфирменная торговля в Интернете осуществляется через системы электронной торговли, посредством организации **электронных торговых площадок** (**ЭТП**), на которых компании могут выступать

и в качестве заказчиков (покупателей), и в качестве поставщиков (продавцов) (рис. 7.8) [11].

Типичный бизнес-процесс материально-технического снабжения ***потребность → закупка → поставка*** через ЭТП может быть описан следующим образом [14]. Процесс закупки начинается в тот момент, когда снабженец производит поиск и подбор поставщиков по номенклатуре согласно утвержденной потребности. Обычно поиск ведется по внутренней корпоративной справочной базе, а также по различным каталогам, справочникам, рекламным материалам и через Интернет.

После осуществления предварительного поиска проводится коммерческая переписка с поставщиками, уточнение характеристик и

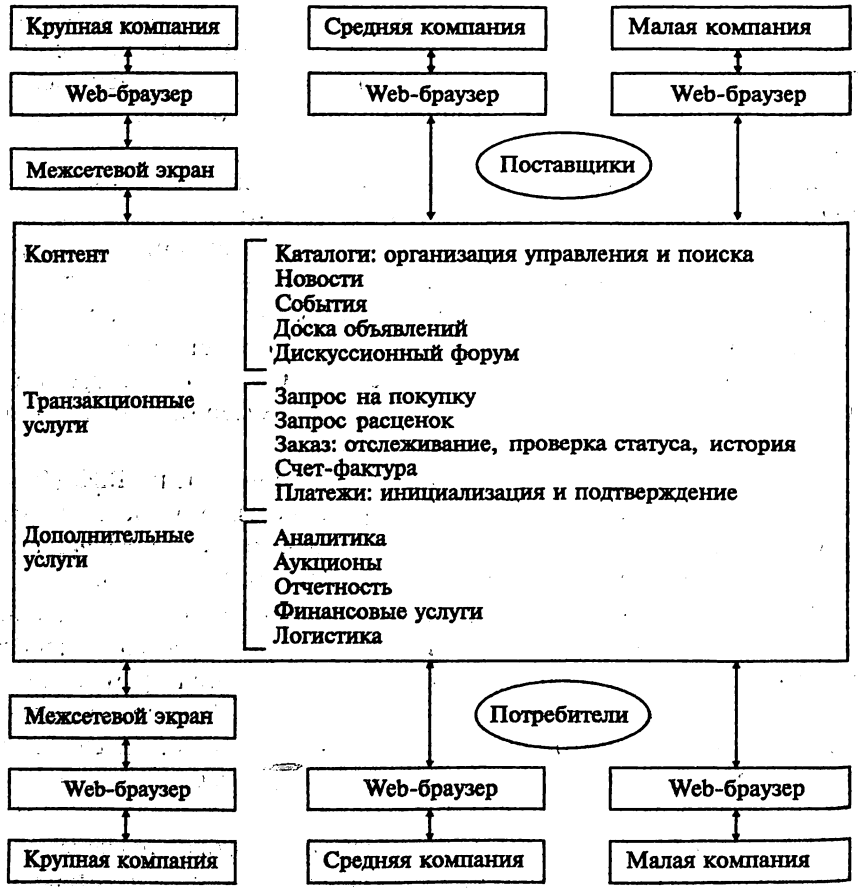


Рис. 7.8. Схема электронной торговой площадки

возможных условий поставки. Затем с выбранным поставщиком заключается договор поставки. Если с данным поставщиком уже имеется заключенный договор, то достаточно сформировать заказ.

В электронной системе снабжения все указанные операции осуществляются в электронной форме, для подписи используется электронная цифровая подпись или документы распечатываются на бумажный носитель.

После документального закрепления сделки, на фазе выполнения поставки, поставщик производит отгрузку продукции. При этом система электронного снабжения позволяет отслеживать фактические отгрузки в режиме реального времени. Заказчик также может производить оплаты поставщику через свою систему электронного снабжения, используя доступ в систему электронных расчетов своего операционного банка.

Система электронной коммерции B2B — это аппаратно-программный комплекс, позволяющий поддерживать бизнес-отношения между предприятиями. Условно эти комплексы делятся на следующие типы:

- **корпоративный сайт компании** предназначен для общения данной фирмы с партнерами и контрагентами — поставщиками и потребителями, действующими и потенциальными инвесторами. Сайт, как правило, содержит информацию о компании, ее персонале, руководстве, а также каталоги продукции и описание услуг;
- **интернет-магазин** предназначен для сбыта продукции, может быть встроен в корпоративный сайт. Он позволяет размещать заказы, проводить электронные платежи, обеспечивать доставку;
- **служба закупок** ищет поставщиков, получает коммерческие предложения, осуществляет электронные платежи, контролирует выполнение заказов;
- **информационные сайты и вертикальные порталы** предназначены для размещения информации об отрасли, входящих в нее компаний, параметров состояния рынка, отраслевых стандартов;
- **брокерские сайты** играют роль посредников между покупателем и продавцом. Их задача — получить через интернет-сайт заказ от одного предприятия, а затем поместить его выполнение на другом;
- **интегрированные комплексы** обеспечивают прямое взаимодействие внутрикорпоративных систем управления (ERP-систем) с внешней системой электронной коммерции B2B (электронной торговой площадкой) и являются наиболее законченным и привлекательным решением в области электронной коммерции. Интегрированные комплексы позволяют полностью автоматизировать все функции ма-

териально-технического снабжения и поэтапно увязывать в единую цепочку все звенья внутрикорпоративных бизнес-процессов: анализ, планирование, бухгалтерию и финансы, учет материальных ценностей (склады), сбыт, снабжение, логистику;

• **электронные торговые площадки** (ЭТП) существуют как отдельные сайты и предназначены для непосредственной организации онлайновой деятельности специалистов служб сбыта и снабжения различных предприятий. На ЭТП создаются «рабочие места» для предоставления пользователям следующих услуг:

— создание и поддержка фирменных каталогов;
— поиск поставщиков и потребителей, проведение тендеров, аукционов и других видов конкурсов в режиме on-line;
— подбор комплекса средств интерактивного on-line взаимодействия контрагентов;
— выполнение маркетингового и конъюнктурного анализа;
— обеспечение предконтрактной и контрактной подготовки;
— проведение платежных операций;
— осуществление контроля поставок.

Электронная торговая площадка позволяет объединить в одном информационном и торговом пространстве поставщиков и потребителей различных товаров и услуг и предоставляет участникам ЭТП ряд сервисов, повышающих эффективность их бизнеса. **Заказчики** получают возможность проводить электронные торги — тендеры, аукционы, запросы цен и предложений, — оптимизируя затраты, а **поставщики** — участвовать в проводимых закупках, размещать информацию о предлагаемой продукции и услугах (рис. 7.9).

На торговой площадке, как правило, существует процедура авторизации участников. Желающие стать пользователями данной ЭТП проходят регистрацию, позволяющую установить и ввести в систему необходимую информацию о предприятии пользователя. После этого пользователю предоставляется ключ (логин и пароль) для входа в систему, которая по этому ключу «опознает» данного пользователя и открывает ему доступ к ресурсам системы.

ЭТП напоминает хорошо организованный торговый центр, где разные продавцы арендуют площади, а администрация обеспечивает рекламу, привлечение покупателей, а также условия и сервис для заключения сделок (купли-продажи). При этом администрация заинтересована в том, чтобы покупательский спрос был удовлетворен и по ассортименту, и по качеству и чтобы продавцы не остались без покупателей.

Рис. 7.9. Схема электронной торговой площадки «Инмарсис»

За право работы на ЭТП обычно приходится платить. Причем на некоторых из них взимаются комиссионные в размере нескольких процентов или долей процента от стоимости проводимых операций (транзакций). На других доступ оплачивается фиксированной суммой, не зависящей от проводимых операций. При этом плата за работу на торговой площадке несопоставимо ниже стоимости создания своего интернет-магазина или своей электронной службы снабжения. Точно так же соотносятся плата за аренду площади в торговом центре и стоимость строительства и поддержания собственного магазина.

Торговая площадка представляет собой сложно организованную систему со своей инфраструктурой. Ее функционирование обеспечивают группа специалистов в данной предметной области, служба технической поддержки и другие сервисные подразделения.

7.3.2. Интеграция ERP- и B2B-систем

Интегрированные комплексы являются наиболее законченным и привлекательным решением в области электронной коммерции B2B, обеспечивая прямое взаимодействие внутрикорпоративных систем управления (ERP) с внешней системой электронной коммерции B2B (торговой площадкой).

Процесс интеграции ERP-систем с электронным бизнесом на текущий момент находится в стадии развития. Достаточно широко компаниями используются отраслевые электронные торговые площадки: Chimforum.ru (для участников рынка химической отрасли), Lesprom.ru (рынок лесной отрасли), Conditer.ru (кондитерский рынок), Stroyteh.ru (дорожно-строительная техника), Medprom.ru (медицинское оборудование и комплектующие), Russiatele.com (телекоммуникационное оборудование), MtsZerno.ru (зерновой рынок), Platts.ru (площадка по нефтепродуктам, нефти и газу), Cislink.ru (продукты питания и другие потребительские товары), eMatrix.ru (для участников ИТ-рынка) и множество других площадок.

Подтверждением процесса интеграции отраслевых ЭТП является распоряжение Правительства Российской Федерации от 01.06.2009 г. № 755р, которое для проведения открытых торгов в электронной форме в интересах федеральных заказчиков определяет оператором электронной торговой площадки **www.etp.zakazrf.ru** отраслевые B2B-системы: «Индексное агентство РТС», «Единая электронная

торговая площадка» (ЕЭТП), «Сбербанк-АСТ», площадка Республики Татарстан и «ММВБ-Информационные технологии».

Использование компаниями ERP-систем для оптимизации деятельности в Интернете не ограничивается B2B-площадками. Необходимо отметить, что на данном этапе чаще всего наблюдается слияние этих систем внутри программных продуктов, разработанных одной компанией-производителем. Все большее количество организаций использует свои интернет-ресурсы для взаимодействия с поставщиками или дистрибьюторами.

Компании-производители ERP-систем активно продвигают дополнительные модули или даже полноценные совокупные продукты, позволяющие покупателю системы создавать интернет-представительства и торговые площадки на базе систем поддержки внутренней инфраструктуры компании. Примерами таких компаний могут служить Oracle, IBM, PeopleSoft, SyBase, SAP, из российских компаний можно отметить 1С, которая на протяжении долгого времени вела разработку таких систем, в результате чего был создан совместный продукт с компанией «Аркадия».

Российская компания **«Интертех»** разработала и внедрила два типа решений в области электронной коммерции между предприятиями: открытую электронную торговую площадку «Инмарсис» (рис. 7.9) и корпоративную систему электронного снабжения АС-МТС (рис. 7.10) [14].

В функции ЭТП «Инмарсис» входят:

- поиск продукции и поставщиков товаров по наименованию, техническим характеристикам и моделям;
- запрос и получение коммерческих предложений;
- проведение маркетингового и конъюнктурного анализа, тендеров и аукционов on-line;
- размещение информации на доске объявлений, а также ряд специальных функций.

Ввод и поддержка прайс-листов и каталогов осуществляется как самими поставщиками, так и администрацией (службой технической поддержки) системы «Инмарсис». Для обеспечения функционирования службы технической поддержки специально создана автоматизированная система обслуживания клиентов (CRM-система), которая выполняет функции поддержки контента (баз данных, каталогов и т. п.), регистрации пользователей, мониторинга работы операторов, фиксации отношений с ними, бухгалтерских операций и др.

ЭПТ «Инмарсис» создана также в виде «отторгаемого» продукта: оболочки для построения электронных торговых площадок на базе функционала площадки «Инмарсис».

В **корпоративной системе электронного снабжения** АС-МТС использование ERP-систем позволяет оптимизировать процессы как закупок, так и продаж. Информация о заказах, полученных через Интернет, интегрируется с данными склада, отделов доставки, продаж, сервисных центров, что позволяет создать единый профиль клиента, эффективно обрабатывать заказы и быстро отвечать на них, создавать и хранить данные обо всех его обращениях, анализировать их и прогнозировать новые обращения (рис. 7.10).

Клиент, в свою очередь, отправив заказ через сайт, получает возможность контролировать процесс обработки заказа: автоматическое уведомление о его принятии и начале работы, данные о подготовке заказа, когда его заявка получена складом, сведения об отгрузке, когда заказ сформирован и отправлен, а также пакет необходимых документов.

Аналогично используется система при закупках. Менеджер через систему получает доступ к каталогу товаров поставщика, выбирает необходимые товары и отправляет заказ. При расчете поставщик автоматически получает уведомление об оплате счета и данные о получении заказа, когда товары будут доставлены на склад клиента.

Электронная торговая площадка «Инмарсис» используется системой АС-МТС-ЮКОС в качестве внешнего каталога, представляющей собой корпоративную систему обеспечения закупок.

Таким образом, в интегрированной системе ERP+B2B рабочее место специалиста по снабжению или сбыту оказывается подключенным как к внутренней зоне, в которой решаются локальные задачи управления ресурсами, так и к внешней — торговой площадке, через которую он связывается и взаимодействует с партнерами, контрагентами — поставщиками и потребителями.

Преимущества работы на ЭТП **для заказчика**:

- значительная экономия рабочего времени;
- экономия денежных средств на организации и проведении закупок;
- прозрачность и открытость процесса закупок;
- честная конкуренция, исключающая работу недобросовестных сотрудников со «своими» фирмами-поставщиками;
- участие в торгах возможно из любой точки мира, не выходя из своего офиса;

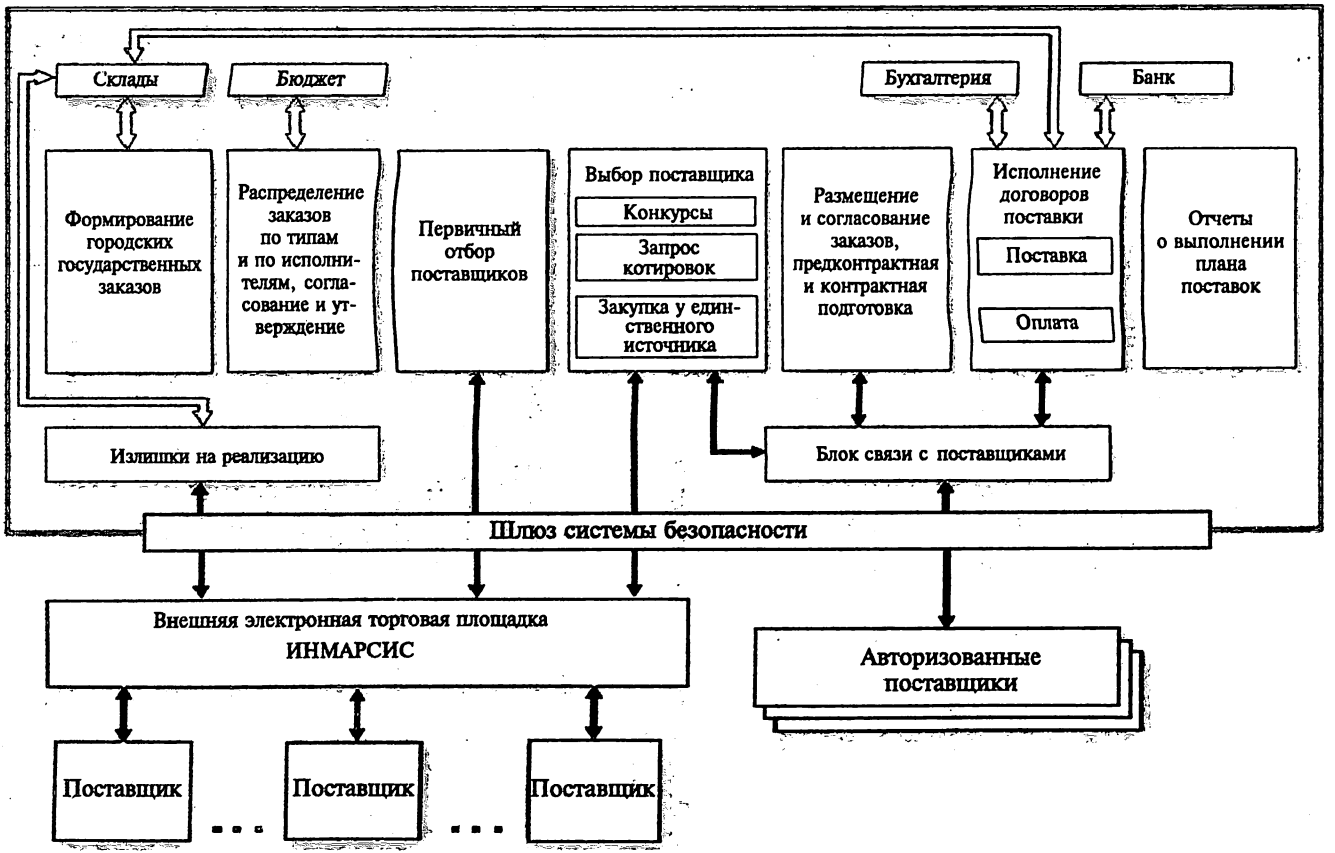


Рис. 7.10. Схема корпоративной системы электронного снабжения АС-МТС

• доступность для представителей любого бизнеса — цена и условия лота ничем не ограничены.

Преимущества работы на ЭТП **для поставщика**:

• быстрый поиск интересующих торгов;

• экономия средств на рекламной кампании;

• прозрачность и открытость процесса продаж;

• честная конкуренция, исключающая неценовые методы ведения борьбы;

• равные права всех поставщиков товаров, работ и услуг;

• участие в торгах возможно из любой точки мира, не выходя из своего офиса (http://ru.wikipedia.org).

Несмотря на то что разные категории площадок имеют свои преимущества и недостатки, их общая отличительная черта — снижение издержек предприятий. По оценке компании IBS, экономия от использования схем B2B может достигать 15 % со стороны закупок и 22 % со стороны сбыта.

7.3.3. Особенности проектирования систем электронной коммерции B2B

С точки зрения методологии и технологии процесс проектирования систем электронной коммерции состоит из тех же классических этапов, что и процесс разработки любой ИС (рис. 7.11). Команда разработчиков часто включает представителей нескольких специальностей: бизнес-аналитиков, экспертов-консультантов, постановщиков задачи, программистов, технических писателей, специалистов по отладке и тестированию программ, специалистов по внедрению и обучению персонала и других. Разработка начинается с обследования текущего состояния и методики работы служб снабжения и сбыта, а также системы управления ресурсами предприятия. Рекомендуется провести следующие организационные мероприятия: прописать этапы оформления заказа, прописать этапы движения товара до покупателя после оформления заказа, определить алгоритмы расчета стоимости доставки, выбрать приемлемые способы оплаты и платежные системы.

Модель разрабатываемой системы электронной коммерции строится с учетом результатов обследования и анализа бизнес-процессов, а также рекомендаций экспертов и решений руководства компании.

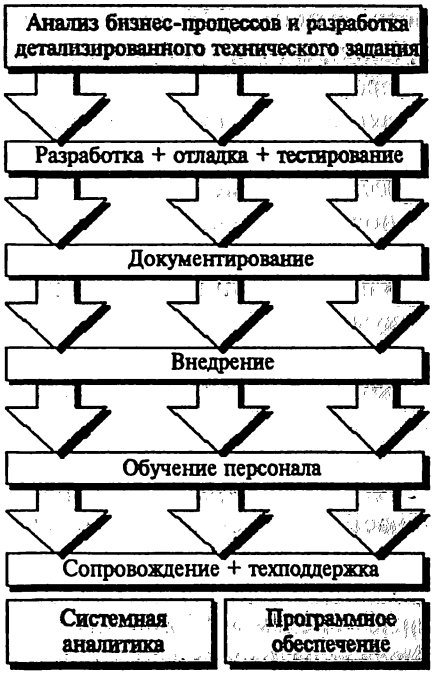


Рис. 7.11. Этапы разработки проекта системы типа B2B

На основе этой модели создается документ — **техническое задание на разработку**. Хотя в ряде случаев, если для построения системы электронной коммерции используется типовой программный пакет, вместо технического задания на разработку создается другой документ — **протокол обследования** с приложениями в виде таблиц вопросов-ответов, необходимых для настройки этого программного пакета.

Особое внимание следует уделить **подсистеме защиты информации от несанкционированного доступа**, чтобы конкуренты или иные посторонние лица не могли при желании добывать конфиденциальную информацию с корпоративных компьютеров. Работа подсистем защиты информации ЭТП должна основываться на процедурах строгой идентификации, аутентификации, разграничения прав доступа к данным и криптографической защиты транзакций.

Необходимо также предусмотреть возможность предоставления «гостевого» доступа к выделенной части информации.

На следующем этапе (если не используется готовый программный продукт) происходят техническое проектирование, разработка, отладка и тестирование созданного комплекса программ. Следует отметить,

что даже при разработке системы B2B на базе готового программного продукта все равно, как правило, требуется определенная совокупность настроечных операций, а иногда даже и программирование, при этом этапы отладки и тестирования являются неизбежными.

Обязательным требованием является представление разработчиком рабочей и технической документации: руководства для специалиста (инженера) по поддержке и сопровождению системы, руководства для пользователей, инструкции оператора и т. д. Сюда же могут входить рекомендации по организации работы персонала, документообороту и формам документов. При внедрении системы обычно возникает необходимость дополнительного оснащения рабочих мест компьютерной техникой, сетевых настроек, а также серверных настроек или установок.

Таким образом, можно выделить две основные составляющие разработки: выбор (создание) программного средства для реализации проекта и работы по настройке и внедрению системы (внедренческий консалтинг).

В качестве программного обеспечения могут быть использованы как готовые «пакетные» системы-оболочки, так и требующие программирования авторские разработки. В качестве готовых решений следует упомянуть интегрированные системы ERP+B2B компаний SAP или Oracle (Oracle Applications) и оболочки для электронной коммерции фирм Ariba, Commerce One, Intershop, JD Edwards и др.

Требующие программирования системы могут разрабатываться на платформах Microsoft (SQL, ASP, VBS, IIS, серверы семейства .NET Enterprise Servers 2000 и т. п.) и UNIX (Java, PHP и др.).

Следует отметить тот факт, что размещение заказов на разработку и программирование в местных IT-компаниях создает квалифицированные рабочие места в области высоких технологий. Это весьма важный социально-экономический фактор, так как спрос на IT-специалистов на местном рынке повышает престижность профессии, способствует росту уровня научных разработок, стимулирует вузы к подготовке соответствующих специалистов. Однако гораздо лучше размещать заказы на разработку у наиболее продвинутых и перспективных отечественных IT-компаний, для которых качество и результативность разработок — неотъемлемая часть борьбы за выживание в условиях жесткой рыночной конкуренции.

Какой бы путь ни выбрала для себя фирма, развертывающая систему типа B2B, ей придется воспользоваться консалтинговыми услугами. Эти услуги по внедрению связаны с изучением и структуризаци-

ей бизнес-процессов, заполнением системы конкретными пользовательскими данными и информацией, настройкой всего программного обеспечения на определенные схемы бизнес-процессов, доработкой под требования (спецификации) заказчика, внедрением системы на объектах заказчика и обучением персонала.

Независимо от выбора типа программного обеспечения функции внедренческого консалтинга, системного анализа и построения схем бизнес-процессов, настройки и наполнения системы являются уникальными для каждой крупной компании в той мере, в какой уникальны сами бизнес-процессы этой компании. Для осуществления такого рода деятельности необходимы высококвалифицированные специалисты в данной прикладной области и в смежных областях, обладающие помимо хорошего профессионального образования особыми навыками структурного мышления и системного анализа.

Одним из популярных подходов к внедрению и эксплуатации промышленных систем электронной коммерции по аналогии с CRM-системами (см. рис. 7.6) является **аутсорсинг (outsourcing)**. В этом случае компания-держатель системы B2B (**провайдер**) по заданию другой компании (заказчика) берет на себя функции хостинга по сопровождению и обеспечению функционирования комплекса задач сбыта или материально-технического снабжения полностью на своей площадке, используя свой компьютерный парк, программное обеспечение и труд своих специалистов. В результате провайдер обеспечивает «внешнее» решение «внутренних» задач «под ключ» [14].

7.4. Системы управления цепочками поставок SCM

Информационные системы класса ERP предназначены для управления всеми ресурсами предприятия. Все, что существует на предприятии: рабочее время, станки, материалы, финансы — рассматривается как ресурсы, управление которыми означает преобразование и распределение ресурсов в целях выпуска товаров и услуг. Однако большинство компаний, внедривших у себя ERP-системы, сходятся во мнении, что они не решают всех задач предприятия.

Для минимизации рисков, издержек, сохранения конкурентных преимуществ в условиях современной экономической и рыночной ситуации необходимо планировать и управлять не только внутренними ресурсами, но находиться в тесном взаимодействии со всеми участниками производственного процесса. Понятие оптимизации бизне-

са в принципе должно предполагать оптимизацию всего: закупка по минимальной цене только в необходимом количестве, доставка с минимальными расходами, продажа с максимальной скоростью.

На рис. 7.12 схематично отражены потребности в управлении ресурсами предприятия и какую часть этих потребностей покрывают информационные системы: большую их часть решает ERP-система, какие-то задачи решают системы CRM, SCM и так далее. Все эти системы интегрированы между собой и потому на рисунке пересекаются [2].

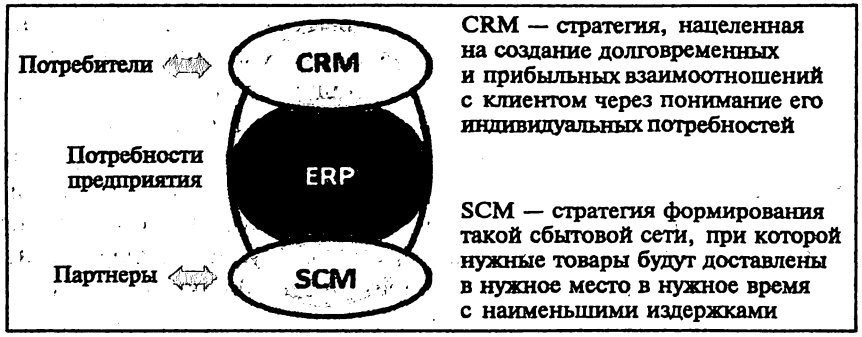


Рис. 7.12. Схема управления ресурсами предприятия

Ранее мы установили, что CRM — это концепция ведения бизнеса, когда во главу угла ставятся индивидуальные потребности клиента. Но если CRM — философия работы с клиентами, SCM (Supply Chain Management) — философия работы с партнерами. **Концепция SCM** подразумевает формирование такой сбытовой сети, при которой нужные товары будут доставлены в нужное место в нужное время с наименьшими издержками (рис. 7.13).

Фактически SCM-системы являются интеллектуальной надстройкой ERP-системы, оптимизируя поступление в компанию товаров и услуг от поставщиков. Дело в том, что деятельность любого предприятия состоит из трех этапов: закупка, производство, сбыт. Каждый из этих этапов может быть сложной цепочкой с промежуточными звеньями. Оптимизация закупок, производства, сбыта, логистики, доходов и прибыли — это и есть **назначение SCM-системы**. Достижение этой цели возможно только при эффективной интеграции поставщиков, производителей, дистрибьюторов и продавцов.

Имеется еще одна причина, по которой системы CRM и SCM изображены вместе на рис. 7.12. Существует тесная связь между

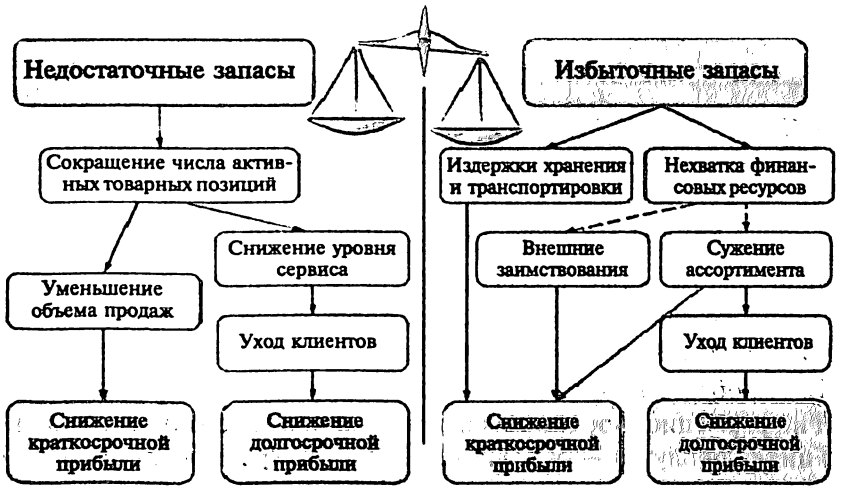


Рис. 7.13. Схема влияния запасов на прибыль предприятия

складскими запасами и CRM. Качественное управление ассортиментом и запасами возможно только при условии качественной работы с данными об истории взаимоотношений с клиентами. Оптимальный уровень запасов по отдельным товарным позициям рассчитывается исходя из прогноза продаж и страховых запасов, необходимых для покрытия рисков. Величина страховых запасов, в свою очередь, должна зависеть от точности прогноза продаж, экономической и маркетинговой значимости товарных позиций и их покупателей. Все эти параметры могут быть рассчитаны только на основе данных о клиентах и истории взаимоотношений с ними [5].

SCM-система состоит из множества звеньев, связанных между собой информационными, денежными и товарными потоками. Исследователи, как правило, выделяют **шесть основных областей**, на которых сосредоточено управление цепочками поставок: производство, поставки, месторасположение, запасы, транспортировка и информация. SCM охватывает весь цикл закупки сырья, производства и распространения товара (рис. 7.14).

Система управления цепочками поставок представляет собой процесс организации планирования, исполнения и контроля потоков сырья, материалов, незавершенного производства, готовой продукции, а также обеспечения эффективного и быстрого сервиса за счет получения оперативной информации о перемещениях товара. Важно, что одни звенья могут целиком принадлежать одной организации (от-

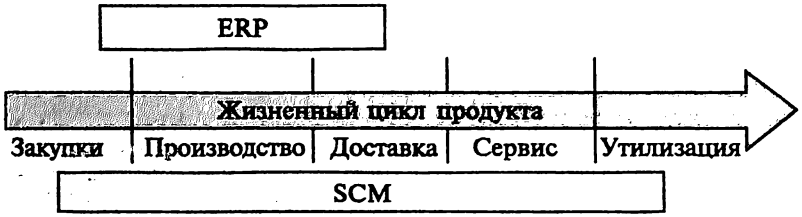


Рис. 7.14. Структура цепочки непрерывных поставок

дел производства, отдел продаж), другие — компаниям-контрагентам (клиентам, поставщикам и дистрибьюторам), т. е. в цепочку поставок обычно входят несколько организаций.

Таким образом, SCM-система обеспечивает планирование ресурсов и информационное сопровождение на протяжении всего жизненного цикла продукта — от заказа на разработку до послепродажного сервиса и утилизации, тогда как ERP-система обеспечивает лишь планирование ресурсов, необходимых для разработки продукта.

В результате система управления цепочками поставок создает **единое информационное пространство** для всех компаний, участвующих в производстве продукта, его транспортировке, продаже и постпродажном обслуживании. Благодаря этому повышается уровень обслуживания клиентов, у них появляются дополнительные возможности, например отслеживание состояния заказа в режиме реального времени.

Система управления цепочками поставок в первую очередь предназначена для организаций со следующими свойствами:

- работа большим количеством поставщиков;
- необходимость в автоматическом выборе в режиме реального времени наиболее выгодного для предприятия поставщика по заданным критериям (по цене, по срокам доставки, по надежности, по качеству и т. д.);
- сложные системы расчетов, учитывающие в том числе оплаченные товары в пути или продукцию, изготавливаемую поставщиком по сформированным первоначальным заказам;
- необходимость отслеживания истории оплат и отгрузок по первоначальным заказам и поставленной продукции;
- автоматизированный контроль месторасположения товара в пути, сроки доставки и стоимость транспортировки.

Структуру SCM удобно рассмотреть на примере программного продукта **«Фолио Заказ-Поставка (SCM)»**, который автоматизирует

работу отделов закупки и логистики предприятия и предназначен для отслеживания перемещения товара от заказа поставщику до прихода на склад [www.folio.ru]. Система состоит из ряда модулей, отвечающих за сбор заявок, их консолидацию, выбор поставщика, согласование заказов, контроль исполнения поставок, формирование отчетов (рис. 7.15).

Модуль «Сбор заявок» обеспечивает накопление заявок на товары от организаций-покупателей с указанием дат поставки. Все заявки внутри системы управления цепочкой поставок привязаны к единому каталогу товаров с внутренним артикулом. При наличии интеграции с программой складского учета «ФолиоWinСклад» и модулем CRM в последнем можно настроить передачу товаров по договору поставки с указанием количества и сроков. Тогда в модуле сбора заявок появится заявка со ссылкой на покупателя и номер договора. Возможно также формирование сводной заявки для поставщика, в которую будут включены артикулы товаров без их привязки к конкретному заказчику, что оправданно при больших оборотах, большом количестве клиентов и при необходимости поддержки минимального запаса товаров на собственных складах.

Рис. 7.15. Архитектура SCM-системы управления цепочками поставок «Фолио Заказ-Поставка (SCM)»

Модуль «Консолидация заявок» содержит механизм подключения разных алгоритмов консолидации заявок для реализации разных закупочных сценариев пользователей и в зависимости от способа регистрации заявок от покупателей. **SCM-система Фолио** допускает интерактивный ввод заказа поставщику, а также проведение конкурса на выбор последнего, если поставщиков несколько.

Модуль «Выбор поставщика» обеспечивает выбор оптимального варианта, если один и тот же товар предоставляют несколько поставщиков, в соответствии с тем или иным критерием, задаваемым пользователем, например по наименьшей закупочной цене. Возможен также отбор поставщиков по совокупности критериев. В алгоритм может быть заложена проверка на наличие достаточного количества запрашиваемого товара на складах поставщика, сроки изготовления и доставки, надежность и приоритетность поставщика по качеству продукции. Допускается прямой выбор поставщика из каталога.

Модуль «Согласование заказов» позволяет в случае отказа или изменения условий работы конкретного поставщика автоматически изменять цепочку поставки путем консолидации всех неподтвержденных заказов для формирования заказа другому поставщику и может предложить замену товара на аналогичный (возможность и ассортимент для такой замены заранее определяются с помощью настроек). Обеспечивает контроль исполнения поставок.

В оставшихся модулях выполняются следующие действия. Сообщение поставщика об отгрузке товара регистрируется в **SCM-системе**. Для каждой партии груза (посылка) цепочка поставки отслеживается по планируемым срокам прохождения этапов пути от грузоотправителя до собственных складов. Приход груза регистрируется признаком прихода посылки в системе закупок и приходной накладной в складской системе. Если товар пришел под конкретного заказчика, то одновременно с приходной накладной автоматически создаются учитываемые счета, резервирующие пришедший товар под указанного заказчика.

Более подробно работа SCM-системы «Фолио Заказ-Поставка» представлена на рис. 7.16, где отображены четыре основных этапа прохождения документов.

На первом этапе создаются заявки клиентов либо через склад, либо через CRM, либо интерактивно в самой программе. Из заявок клиентов формируются заказы поставщикам, кроме этого, существует возможность создавать заказы под свободную продажу, с учетом данных со склада.

Рис. 7.16. Типовая схема прохождения документов в программе «Фолио SCM»

Заказы формируются с учетом параметров принятых прайс-листов и согласовываются с поставщиком. На основании подтверждения вводятся счета поставщиков и оплата по этим счетам. После отгрузки товара поставщикам документы переходят в последнее четвертое состояние — «товары в пути».

Все заказы, отгруженные поставщиками, группируются в несколько реестров, объединенных группой Доставка. При прибытии грузов автоматически формируются приходные накладные и счет, резервирующий этот товар под клиента, который сделал этот заказ.

Расчет размера заказа может выполняться по нескольким алгоритмам, один из которых представлен на рис. 7.17.

Этот способ расчета размера заказа позволяет формировать заказ на заданный временной интервал по прогнозу спроса, соответствующего темпу продаж определенного периода.

Являясь модулем корпоративной ИС **«Фолио Купец»**, программный продукт **«Фолио Заказ-Поставка (SCM)»** может работать либо самостоятельно, либо в составе корпоративной системы, обмениваясь данными с программами-модулями: «Складской учет», «Складская логистика», «Бухгалтерский учет» и «Управление взаимоотношениями с клиентами» (CRM) (www.folio.ru).

В отличие от других модулей корпоративной системы, отвечающих за автоматизацию внутренних бизнес-процессов, **SCM-система** отвечает за взаимодействие с «внешним миром», обеспечивая полный цикл документооборота, связанного с закупками. При этом обеспечивается полный цикл финансового, складского и управленческого учета предприятия.

Как показывает практика деятельности ведущих зарубежных и отечественных компаний, в связи с диктуемой рынком необходимостью увеличения скорости движения материальных и информационных потоков в цепочке поставок все большим спросом пользуется **идея кросс-докинга** (cross-docking) (рис. 7.18).

Кросс-докинг представляет собой процесс внутри цепочки поставок, в ходе которого приемка и отправка продукции потребителю так скоординированы по времени и объему, что можно **избежать помещения товаров на склад**. В результате можно исключить все виды деятельности, характерные для склада хранения. При этом концепция кросс-докинга выходит за рамки физического процесса товарооборота. Тесное информационное взаимодействие всех партнеров обеспечивает эффективное планирование, организацию и контроль всей цепочки поставок.

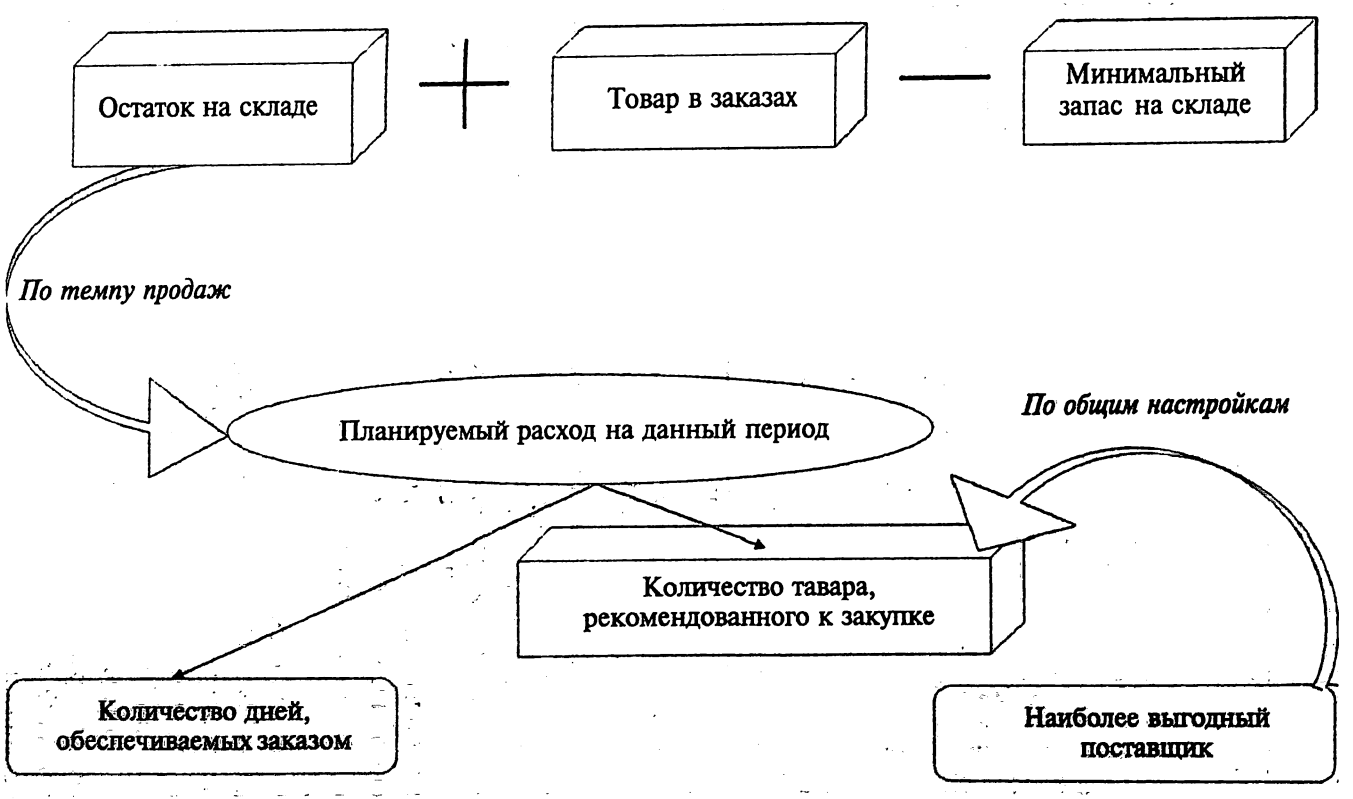


Рис. 7.17. Способ расчета размера заказа «Прогнозирование»

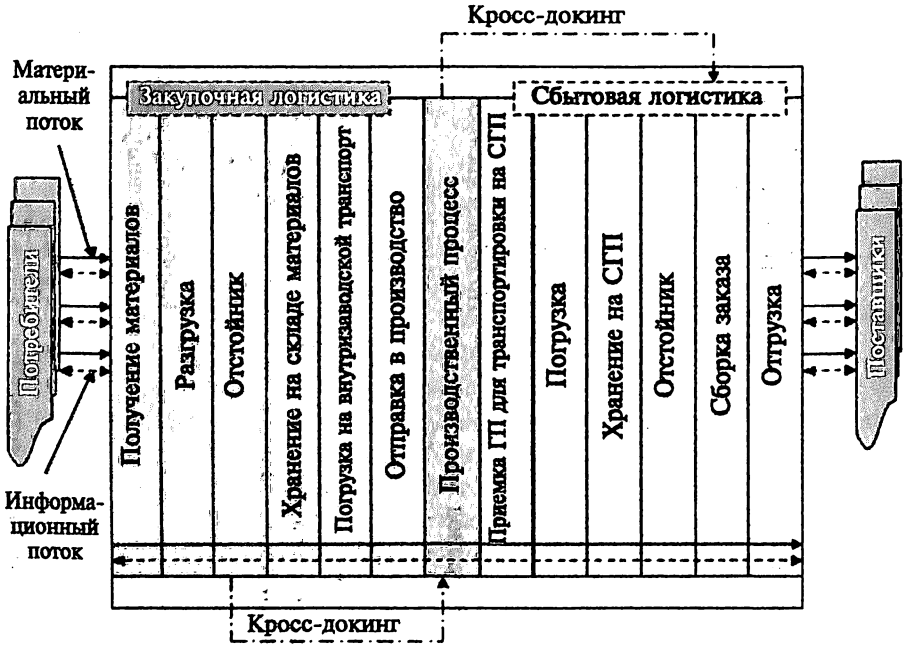


Рис. 7.18. Логистические процессы на предприятии в цепи поставок по типу кросс-докинга

Преимущества кросс-докинга состоят не только в сокращении запасов и издержек на складирование, но и в устранении всех операций, не добавляющих ценности продукту, которые появляются при загрузке материалов на хранение и последующей перегрузке.

Успешное внедрение систем управления цепочками поставок дает компаниям следующие преимущества:

- рост прибыли на 5—15 %;
- сокращение времени и стоимости обработки заказа от 20 до 40 %;
- сокращение закупочных издержек на 5—15 %, складских запасов на 20—40 % и производственных затрат на 5—15 %;
- сокращение времени выхода на рынок;
- контроль местонахождения, состояния и движения товара в реальном масштабе времени.

Поэтому крупнейшие разработчики ERP-систем уже не могут игнорировать потребность клиентов в SCM-системах и предлагают соответствующие модули к своим системам. Также появились компании, специализирующиеся на разработке и внедрении конкретно

SCM-решений. Например, компания **«ИнтегПрог»** использует для разработки **платформу interLogistics**, которая обеспечивает быструю разработку заказных информационных систем. Она предназначена для разработки решений по управлению информационными, логистическими процессами и событиями [www.integprog.ru].

Особенностью этой платформы является возможность построения систем, обеспечивающих работу в едином информационном пространстве, поддержку единой технологии обработки данных, централизованное и децентрализованное управление данными, возможность интеграции с приложениями сторонних разработчиков.

В принципе существует три альтернативных варианта внедрения SCM-системы: внедрение универсальной системы (такие системы, как Oracle E-Business Suite, SAP), приобретение коробочного решения и заказная разработка (рис. 7.19).

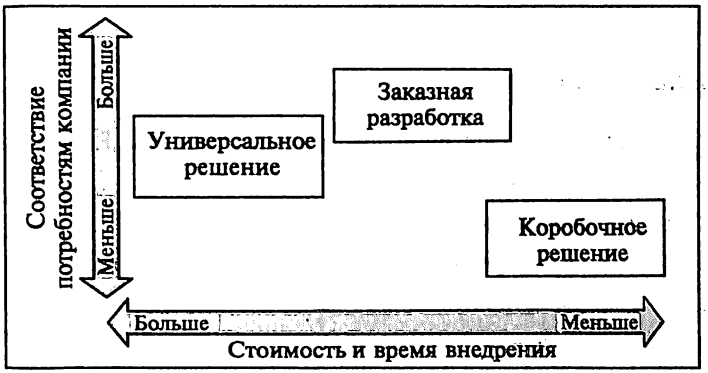


Рис. 7.19. Варианты внедрения SCM-системы

Внедрение универсальной системы достаточно дорогостоящий и длительный процесс. Компания платит не только за лицензии на рабочие места, но еще и за работу специалистов, настраивающих эту систему под особенности бизнеса компании. Причем чем сильнее отличаются бизнес-процессы от так называемых лучших практик (best practice), тем дольше и дороже получается проект.

В случае **с коробочным решением** компания в целом получает сразу готовый продукт. Конечно, некоторая настройка решения выполняется, но она не такая трудоемкая и дорогостоящая, как в первом случае.

Этот вариант подойдет для тех компаний, которые готовы мириться с функциональными ограничениями коробочной версии и ко-

торые готовы несколько видоизменить свою работу таким образом, чтобы соответствовать требованиям программы.

Разработка индивидуального решения по стоимости и сроку внедрения находится между первыми двумя вариантами, но по сравнению с ними имеет важные отличия:

- разработанное решение идеально соответствует бизнес-процессам компании, поскольку разрабатывается именно под нее;
- заказная разработка позволяет реализовать уникальные сервисы заказчика;
- заказная разработка позволяет сократить расходы на дорогостоящем консалтинге;
- возможность отразить в решении существующие бизнес-процессы, не изменяя их;
- возможность оперативной адаптации системы под изменившиеся бизнес-процессы заказчика.

Контрольные вопросы

1. Прокомментировать функциональный состав системы IFS Applications.
2. Назначение и функциональный состав CRM-систем.
3. Классификация CRM-систем.
4. Специфика работы класса систем «CRM по требованию».
5. Особенности проектирования CRM-систем.
6. Особенности внедрения CRM-систем.
7. Назначение и функциональный состав систем электронной коммерции B2B.
8. Назначение и функциональный состав электронной торговой площадки.
9. Назначение интеграции ERP- и B2B-систем.
10. Особенности проектирования систем электронной коммерции типа B2B.
11. Особенности внедрения систем электронной коммерции типа B2B.
12. Назначение и функциональный состав SCM-систем.
13. Особенности систем класса кросс-докинг как процесса внутри SCM-систем.
14. Особенности внедрения SCM-систем.

Глава 8
СИСТЕМЫ ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЙ — OLAP-СИСТЕМЫ

8.1. Введение в OLAP-технологию

В области информационных технологий всегда существовало два взаимодополняющих друг друга направления развития:

- системы, ориентированные на оперативную (транзакционную или операционную) обработку данных;
- системы, ориентированные на анализ данных, — системы поддержки принятия решений.

Все, что говорилось об информационных системах в предыдущих главах, относится исключительно к ИС для оперативной обработки данных. Они предназначаются для автоматизации оперативной деятельности предприятий и называются системами On-Line Transaction Processing — (**OLTP-системы**). Их основная задача — обеспечить регистрацию некоторых фактов, их непродолжительное хранение с последующим размещением в архивах.

Информационную основу таких систем обеспечивают реляционные базы данных. Именно для этого класса систем изначально создавались и на это были ориентированы реляционные СУБД, которые сегодня стали основным средством построения информационных систем самого различного масштаба и назначения. Но, являясь высокоэффективным средством реализации систем оперативной обработки данных, реляционные СУБД оказались менее эффективными в задачах аналитической обработки.

Дело в том, что не каждая система позволяет использовать в полном объеме имеющуюся в ней информацию для поддержки принятия решений, принимаемых руководством предприятия с целью достижения конкурентных преимуществ. Разработка руководителем решений по управлению попадает в разряд областей, наиболее сложно поддающихся автоматизации. Традиционным подходом являются попытки

использовать уже построенные оперативные системы для решения задач поддержки принятия решений. Обычно пытаются строить развитую систему запросов к OLTP-системе и использовать полученные после интерпретации отчеты непосредственно для принятия решений.

Однако осуществить полноценную поддержку принятия решений непосредственно на основе данных OLTP-систем, автоматизирующих сбор и первичную обработку данных о деятельности предприятия, невозможно. Для этого следует использовать так называемые квазиинтеллектуальные **OLAP-системы** (On-Line Analytical Processing), обеспечивающие превращение «руды» — информации из OLTP- систем — в готовое «изделие», которое руководство и аналитики могут использовать для выработки управленческих решений.

Полная структура информационно-аналитической системы, построенной на основе хранилища данных, показана на рис. 8.1.

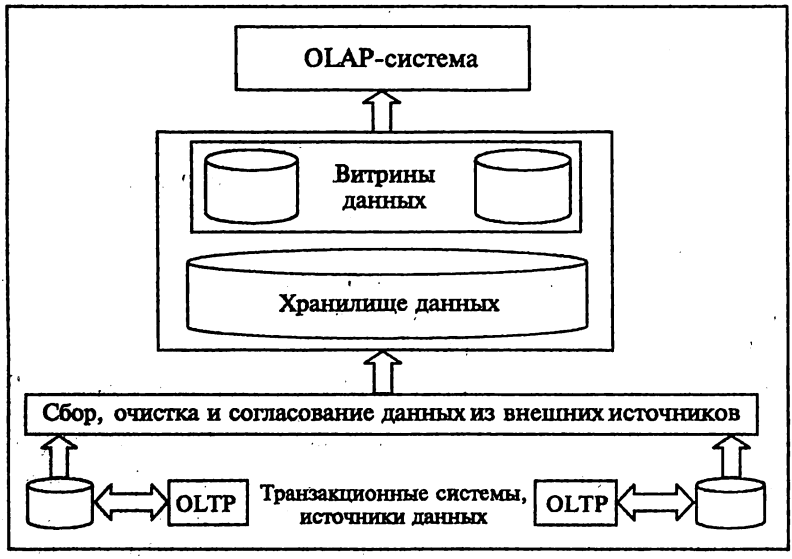


Рис. 8.1. Полная структура информационно-аналитической системы

Суть работы OLAP-систем состоит в предоставлении пользователю **многомерной таблицы**, автоматически суммирующей данные в различных разрезах и позволяющей интерактивно управлять вычислениями и формой отчета.

Эти требования к многомерности представления данных объясняются тем, что аналитики — это особые потребители корпоративной

информации. **Задача аналитика** — находить закономерности в больших массивах данных. Поэтому аналитик не обращает внимания на отдельно взятый факт, что в понедельник шестого числа контрагенту Иванову была продана партия цемента — ему нужна информация о сотнях и тысячах подобных событий. Аналитик отбрасывает ненужные ему подробности вроде ИНН покупателя, его точного адреса и номера телефона, индекса контракта и тому подобного. В то же время данные, которые требуются аналитику для работы, обязательно содержат числовые значения — это обусловлено самой сущностью его деятельности (http://www.interface.ru/home.asp).

Современные информационные системы масштаба предприятия, как правило, содержат в виде подсистем либо в виде отдельных систем OLAP-приложения, предназначенные для комплексного многомерного анализа данных, их динамики, тенденций и т. п. Такой анализ в конечном итоге призван содействовать принятию решений. Нередко эти системы так и называются — системы поддержки принятия решений.

Фактически OLAP являются естественным продолжением и развитием идей электронных таблиц типа Excel для многомерных БД. Визуальный интерфейс OLAP является той же электронной таблицей, но оснащенной мощной машиной вычислений и особым стандартом представления данных и управления ими. Поэтому для специалистов, владеющих электронными таблицами, не возникает проблем в овладении OLAP-инструментом.

Таким образом, аналитику нужно много данных, эти данные являются выборочными, а также носят характер **«набор атрибутов — число»**. Поэтому аналитики должны быть обеспечены информацией, образец которой приведен в табл. 8.1.

Здесь столбцы «Место дислокации», «Вид спорта» и «Время» являются атрибутами, а столбцы «Количество спортсменов» и «Объем финансирования» содержат числовые значения.

Анализируя табл. 8.1, можно заметить, что она легко преобразуется в три измерения: по одной из осей откладывается «Место дислокации», по другой — «Вид спорта», по третьей — «Время». А числами в этом трехмерном массиве у нас будут «Количество спортсменов» или «Объем финансирования». Именно такой трехмерный массив используется в качестве базы данных в OLAP-системах и называется кубом (рис. 8.2).

Куб совсем не обязательно должен быть трехмерным. Он может быть и двумерным, и многомерным — в зависимости от решаемой за-

Таблица 8.1. **Данные для анализа подготовки спортсменов**

Место дислокации	Время	Вид спорта	Объем финансирования, тыс. долл.	Количество спортсменов
Сочи	2007	Бобслей	700	55
Сочи	2008	Бобслей	750	60
Сочи	2007	Коньки	600	256
Сочи	2009	Коньки	650	300
Сочи	2007	Лыжи	450	314
Сочи	2008	Лыжи	470	340
Москва	2007	Бобслей	550	42
Москва	2007	Коньки	620	322
Москва	2009	Коньки	630	340
Москва	2007	Лыжи	320	230
Томск	2007	Бобслей	250	26
Томск	2008	Бобслей	430	40
Томск	2007	Коньки	210	120
Томск	2009	Коньки	220	135
Томск	2007	Лыжи	300	210
Томск	2008	Лыжи	310	215

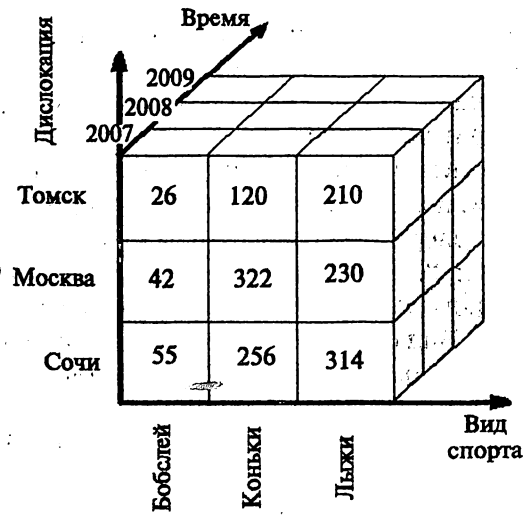


Рис. 8.2. Трехмерное представление табл. 8.1

дачи. Особо опытным аналитикам может понадобиться порядка 20 измерений, и мощные OLAP-системы именно на такое количество и рассчитаны. Более простые настольные приложения поддерживают не более шести измерений. При этом подсчет суммарных величин в ячейках куба уже не требует времени непосредственно на этапе анализа, поскольку все возможные суммы уже подсчитаны на этапе построения многомерной базы данных.

Таким образом, в основе OLAP лежит идея многомерной модели данных, обычно представляемая в виде гиперкуба (см. рис. 8.2). И это очень удобно для проведения анализа потому, что человеческое мышление многомерно по определению. Когда человек задает вопросы, он налагает ограничения, тем самым формулируя вопросы во многих измерениях, поэтому процесс анализа в многомерной модели приближен к реальности человеческого мышления.

В OLAP должны использоваться **агрегированные данные.** Дело в том, что пользователя, занимающегося анализом, редко интересуют детализированные данные. Чем выше уровень пользователя (руководителя, управляющего, аналитика), тем выше уровень агрегации данных, используемых им для принятия решения. Например, коммерческого директора фирмы по продаже автомобилей мало интересует вопрос: «Какого цвета "Жигули" успешнее всего продает один из менеджеров компании — Петров: белого или красного?» Для него важно, какие модели и какие цвета предпочитают в данном регионе. Его также мало интересует детализация на уровне контракта, часа или даже дня. Для правильного формирования запасов на складе ему важна и необходима информация на уровне декады, месяца или даже квартала.

Важнейшим свойством данных в аналитических задачах является их **Исторический характер.** После того как зафиксировано, что в Сочи в 2007 г. проходили подготовку 314 лыжников, данные об этом событии становятся историческим (свершившимся) фактом. Информация об этом факте может многократно считываться из БД, но уже не может и не должна быть изменена.

Другим неотъемлемым свойством Исторических данных является обязательная спецификация **Времени**, которому эти данные соответствуют. Причем Время является не только наиболее часто используемым критерием выборки, но и одним из основных критериев, по которому данные упорядочиваются в процессе обработки и представления пользователю. Кроме этого, время является стандартным параметром практически любой аналитической, статистической или

финансовой функции (прогноз, нарастающий итог, переходящий запас, скользящее среднее и т. д.). Наличие Исторических данных в системе обеспечивает выполнение такой важной функции, как **прогнозирование данных**, т. е. данных о событии, которое еще не происходило.

Если сравнить данные из табл. 8.1 и куба на рис. 8.2, то легко заметить, что OLAP не дает никаких новых данных, которых не было у пользователей при работе с реляционной БД. Аналитики могли заказать у IT-специалистов какой-то новый отчет и из него получить любую цифру. Весь вопрос в том, додумались ли бы они до формулировки задачи и за какое время этот отчет был бы сделан. То есть OLAP предоставляет больше возможностей по работе с данными, но сами данные для его работы нужны те же, с которыми работают и не OLAP-системы.

Комплексный взгляд на собранную в хранилище данных информацию, ее обобщение и агрегация, гиперкубическое представление и многомерный анализ являются задачами систем оперативной аналитической обработки данных (OLAP).

8.2. Многомерное представление при описании структур данных

Основными понятиями, с которыми оперируют пользователь и проектировщик в многомерной модели данных, являются:

- **измерение** (Dimension);
- **ячейка** (Cell).

Иногда вместо термина «Ячейка» используется термин «**Показатель**».

Измерение — это множество однотипных данных, образующих одну из граней гиперкуба. Например, Дни, Месяцы, Кварталы, Годы — это наиболее часто используемые в анализе временные Измерения. Примером географических измерений являются: Районы, Города, Регионы, Страны.

В многомерной модели данных Измерения играют роль индексов, используемых для идентификации конкретных значений (**Показателей**), находящихся в **Ячейках** гиперкуба (рис. 8.3).

В свою очередь, **Показатель** — это поле (обычно количественное), значения которого однозначно определяются фиксированным набо-

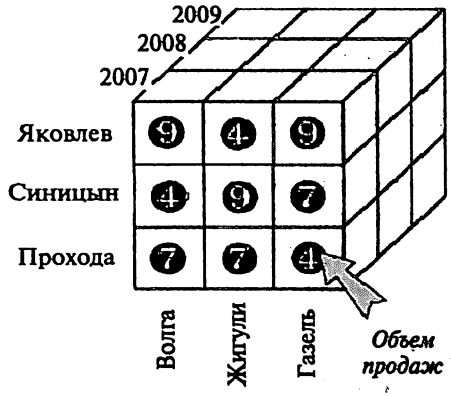


Рис. 8.3. Измерения и Показатели гиперкуба

ром Измерений. В Oracle Express Server, в зависимости от того, как формируются его значения, Показатель может быть определен как:

- **переменная** (Variable) — значения таких Показателей один раз вводятся из какого-либо внешнего источника или формируются программно, а затем в явном виде хранятся в многомерной базе данных (МБД);
- **формула** (Formula) — значения таких Показателей вычисляются по некоторой заранее специфицированной формуле, т. е. для Показателя, имеющего тип Формула. В этом случае в БД хранятся не его значения, а формула, по которой эти значения вычисляются при каждом обращении.

Это различие существует только на этапе проектирования и полностью скрыто от конечных пользователей. На рис. 8.3 каждое значение показателя Объем продаж однозначно определяется комбинацией полей (значениями Измерений):

- модель автомобиля;
- менеджер;
- время (например, Год).

Поэтому в терминах многомерной модели речь будет идти уже не о двумерной таблице, а о трехмерном гиперкубе:

- первое Измерение — Модель автомобиля;
- второе Измерение — Менеджер, продавший автомобиль;
- третье Измерение — Время (Год).

На пересечении граней куба находятся значения Показателя — «Объем продаж».

В отличие от Измерений, не все значения Показателей должны иметь и имеют реальные значения. Например, менеджер Синицын в 2009 г. мог уже не работать в фирме, и в этом случае все значения Показателя «Объем продаж» за этот год будут иметь для него неопределенные значения.

В различных многомерных СУБД используются два основных варианта организации данных:

- гиперкубическая модель;
- поликубическая модель.

Системы, поддерживающие **поликубическую модель** (примером является Oracle Express Server), предполагают, что в МБД может быть определено несколько гиперкубов с различной размерностью и с различными Измерениями в качестве их граней. Например, значение Показателя «Рабочее время менеджера», скорее всего, не зависит от Измерения «Модель Автомобиля» и однозначно определяется двумя Измерениями: «День» и «Менеджер». В поликубической модели в этом случае может быть объявлено два различных гиперкуба:

двумерный — для Показателя **«Рабочее время менеджера»**;

трехмерный — для Показателя **«Объем продаж»**.

В случае же **гиперкубической модели** предполагается, что все Показатели должны определяться одним и тем же набором Измерений. То есть только из-за того, что «Объем продаж» определяется тремя Измерениями, при описании Показателя «Рабочее время менеджера» пришлось бы также использовать три Измерения и вводить избыточное для этого Показателя Измерение «Модель автомобиля» [33].

8.3. Операции манипулирования Измерениями

Формирование среза. Пользователь практически никогда не работает одновременно сразу со всем гиперкубом данных. Даже трехмерный куб сложно отобразить на экране компьютера так, чтобы были видны значения интересующих показателей. Для визуализации данных, хранящихся в кубе, применяются, как правило, привычные двумерные, т. е. табличные, представления.

Пользователь, анализирующий информацию, может «разрезать» куб по разным Измерениям, получать агрегированные, например, по годам или, наоборот, детализированные по месяцам сведения. А затем уже осуществлять прочие манипуляции, которые необходимо выполнить в процессе анализа.

Двумерное представление куба (см. рис. 8.2) можно получить, «разрезав» его поперек одной или нескольких осей (Измерений). На рис. 8.4 зафиксировано («разрезано») значение Измерения «Время» (2007 г.). В результате получают двумерный срез куба для одного Показателя — «Количество спортсменов» и двух «неразрезанных» измерений — «Дислокация» и «Вид спорта».

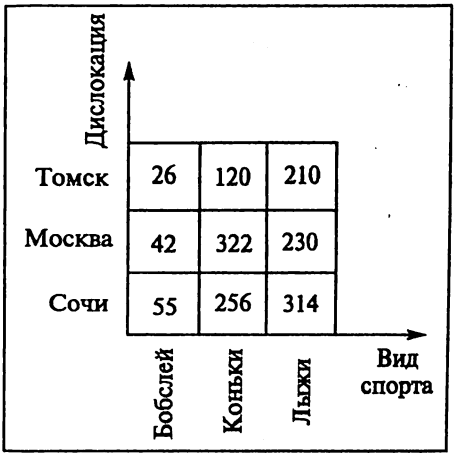


Рис. 8.4. Двумерный срез куба для одной меры

Двумерное представление куба возможно и тогда, когда «неразрезанными» остаются и более двух измерений. При этом на осях среза (строках и столбцах) будут размещены два или более Измерений «разрезаемого» куба.

Подмножество гиперкуба, получившееся в результате фиксации значения одного или более Измерений, называется **срезом**.

Операция вращения обеспечивает изменение порядка представления (визуализации) Измерений (обычно при двумерном представлении данных). Эта операция позволяет пользователю обеспечить визуализацию данных в комфортной для их восприятия форме в результате поворота осей координат на 90°.

Иерархические отношения. Значения Показателей в гиперкубах не являются первичными, а получены в результате суммирования (агрегирования) по более мелким элементам. Например, год делится на кварталы, кварталы на месяцы, месяцы на недели, недели на дни. Регионы состоят из населенных пунктов. Наконец, в самих городах можно выделить конкретные спортивные комплексы. Виды спорта

можно разделить на отдельные категории и так далее. В терминах OLAP такие многоуровневые объединения совершенно логично называются **Иерархиями**.

Средства OLAP дают возможность в любой момент перейти на нужный уровень иерархии. Причем для одних и тех же Измерений могут поддерживаться несколько видов Иерархий: например, для Времени: **день—неделя—месяц—квартал** или **день—декада—месяц—квартал** [33].

Исходные данные берутся из нижних уровней иерархий, а затем суммируются для получения значений более высоких уровней (рис. 8.5). Для того чтобы ускорить процесс перехода, просуммированные (агрегированные) значения для разных уровней рассчитываются на этапе заполнения куба данными и затем хранятся в нем постоянно. Таким образом, то, что со стороны пользователя выглядит одним кубом, на самом деле состоит из множества более примитивных кубов.

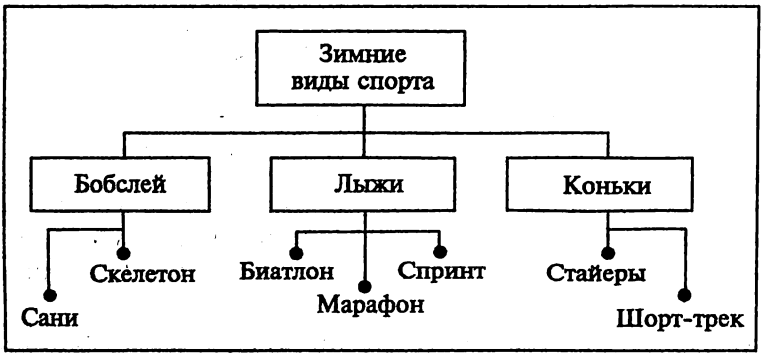


Рис. 8.5. Иерархия отношений по измерению «Виды спорта»

Измерения OLAP-кубов состоят из так называемых меток или членов. Например, измерение «Лыжи» состоит из меток: «Биатлон», «Марафон», «Спринт» и так далее. Совсем не обязательно должны быть заполнены все ячейки куба: если нет информации о конькобежцах-стайерах, тренирующихся в Сочи в 2008 г., то значение в соответствующей ячейке просто не будет определено. Единственным ограничением для создания Иерархии служит то, что она должна быть образована каскадом отношений ***«один ко многим»***. То есть в любой Иерархии каждая позиция на шкале должна иметь ровно одного родителя.

Например, каждый «Менеджер» может работать только в одном подразделении, а каждой «Модели автомобиля» однозначно соответствует фирма, которая ее выпускает (см. рис. 8.3):

Менеджер → Подразделение;

Модель Автомобиля → Фирма.

В свою очередь, множество Отношений может иметь иерархическую структуру — **Иерархические отношения**:

День → Месяц → Квартал → Год;

Менеджер → Подразделение → Регион → Страна;

Модель Автомобиля → Завод-производитель → Страна.

Иерархическая структура отношений позволяет не объявлять каждый раз новые Измерения и затем устанавливать между ними множество отношений, а использовать механизм Иерархических отношений [33]. Например, для рис. 8.3 можно добавить к множеству значений Измерения «Менеджер» (Яковлев, Синицын, Прохода, Смирнов) значения Измерения «Подразделение» (Филиал 1, Филиал 2, Филиал 3) и Измерения «Регион» (Центр, Юг) (рис. 8.6). Затем можно определить между этими значениями Отношения иерархий:

«Центр» → «РФ»	«Петров» → «Филиал 1»;
«Юг» → «РФ»	«Смирнов» → «Филиал 1»;
	«Иванов» → «Филиал 2»;
«Филиал 1» → «Центр»	«Кузнецов» → «Филиал 2»;
«Филиал 2» → «Центр»	«Ковалев» → «Филиал 2»;
	«Синицын» → «Филиал 3»;
«Филиал 3» → «Юг»	«Яковлев» → «Филиал 3»;
	«Прохода» → «Филиал 3».

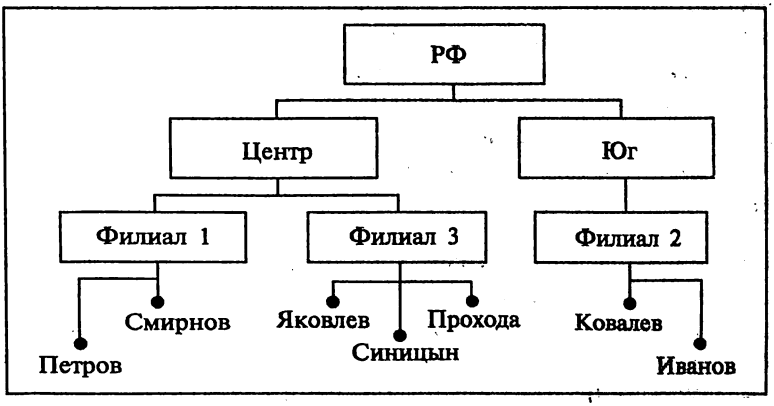


Рис. 8.6. Иерархия отношений по измерению «Менеджер»

Операция Агрегации. С точки зрения пользователя, Подразделение, Регион, Страна являются точно такими же Измерениями, как и Менеджер (см. рис. 8.6). Но каждое из них соответствует новому, более высокому уровню агрегации значений Показателя Объем продаж. В процессе анализа пользователь не только работает с различными Срезами данных и выполняет их Вращение, но и переходит от детализированных данных к агрегированным, т. е. производит операцию **Агрегации (Drill Up)**.

Например, посмотрев, насколько успешно в 2007 г. менеджер Синицын продавал модели «Жигули» и «Волга», управляющий может узнать, как выглядит соотношение продаж этих моделей на уровне Подразделения («Филиал 3»), где Синицын работает, а затем получить аналогичную справку по Региону («Юг»), в котором располагается это Подразделение, или в целом по Стране («РФ»).

Операция Детализации. Переход от агрегированных данных к более детализированным называется операцией **Детализации (Drill Down)**. Например, начав анализ на уровне Региона («Центр»), пользователь может получить более точную информацию о работе конкретного Подразделения («Филиал 1») или Менеджера (Петрова, Смирнова).

Расчет и вывод суммарного (по каждому дню) количества автомобилей всех моделей Волжского автозавода, проданных Менеджерами Петровым и Смирновым в первом квартале 2008 г., выполняется с помощью четырех команд:

```
Limit DAY to QUARTER Q1.2008 {только дни I квартала 2008 г.}
Limit MANAGER to "Петров" "Смирнов"{только Петров и Смирнов}
Limit MODEL_CAR to FIRMA_CAR "АВТОВАЗ"{модели АВТОВАЗА}
{В качестве аргументов функции Total указывается агрегируемый
Показатель и новые Измерения результата (DAY, MANAGER)}
Table Total (Amount Day Manager)
{Агрегация выполняется по «Моделям авто»}
END
```

Таким образом, гиперкубы OLAP представляют собой, по сути, метаотчеты. Разрезая метаотчеты (кубы) по Измерениям, аналитик получает, фактически, «обычные» двумерные отчеты на интересующих его уровнях иерархии Измерений. Это не обязательно отчеты в обычном понимании этого термина — речь идет о структурах данных с такими же функциями (см. рис. 8.4).

На основе такого представления можно получить следующие агрегатные данные в качестве ответов на вопросы типа:

- Какова суммарная стоимость продаж за 2007 г., совершенных продавцами региона «Юг»?
- Какова суммарная стоимость (или количество) продаж автомобилей фирмы «АВТОВАЗ» за 2008 г., совершенных продавцами магазина «Филиал 3»?
- Какова суммарная прибыль от продаж автомобилей фирмы «АВТОВАЗ» за 2009 год, совершенных продавцами региона «Центр»?

Преимущества кубов очевидны — данные необходимо запросить из реляционной СУБД всего один раз — при построении куба. Поскольку аналитики, как правило, не работают с информацией, которая дополняется и меняется «на лету», сформированный куб является актуальным в течение достаточно продолжительного времени. Благодаря этому не только исключаются перебои в работе сервера реляционной СУБД (нет запросов с тысячами и миллионами строк ответов), но и резко повышается скорость доступа к данным для самого аналитика.

Кроме того, как уже отмечалось, производительность повышается и за счет подсчета промежуточных сумм Иерархий и других агрегированных значений в **момент построения куба**. То есть если изначально наши данные содержали информацию о дневной выручке по конкретному товару в отдельно взятом магазине, то при формировании куба OLAP-приложение считает итоговые суммы для разных уровней Иерархий (недель и месяцев, регионов и стран).

8.4. Принципы работы пользователей на OLAP-системах

OLAP-система предоставляет удобные быстродействующие средства доступа, просмотра и анализа деловой информации. Для примера используем наиболее простой продукт из числа OLAP-клиентов — **«Контур Стандарт»** компании Intersoft Lab [3].

Все OLAP-продукты характеризуются общими принципами построения:

- В качестве внешнего интерфейса они предоставляют управляемую динамическую таблицу (рис. 8.7). На вход динамической таблицы подается многомерный массив (см. рис. 8.2, 8.3), который содержит данные двух типов: **Измерения** (оси координат куба) и **Показатели**

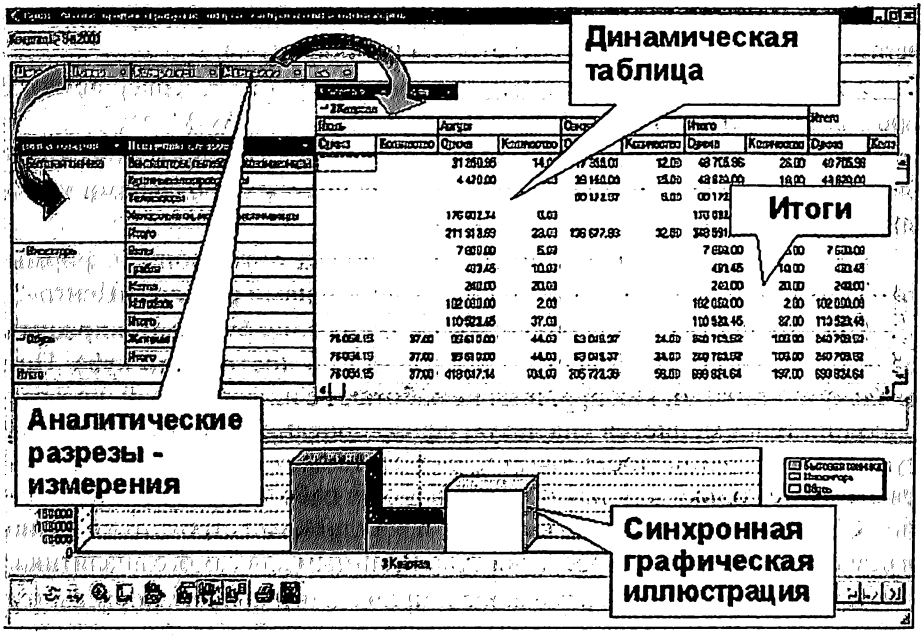


Рис. 8.7. Диалоговое окно пользователя в OLAP-системе

(значений в ячейках куба). Измерения становятся столбцами и строками динамической таблицы, в которых отображаются шкалы Измерений. На пересечении столбцов и строк размещены Показатели.

- Столбцы и строки являются основными инструментами управления таблицей. С их помощью пользователь может манипулировать исходными данными: удалять, менять местами строки и столбцы, устанавливать фильтры («разрезы») по Измерениям, детализировать информацию или, наоборот, агрегировать ее (рис. 8.7, 8.8). При этом промежуточные и окончательные итоги по Показателям автоматически пересчитываются. Выполнение этих операций обеспечивается OLAP-машиной (или машиной OLAP-вычислений). Сами манипуляции с данными носят название OLAP-операций.

- Еще одной важной стороной OLAP-анализа является **графическое отображение данных.** График синхронизирован с динамической таблицей. После выполнения любой OLAP-операции данные автоматически пересчитываются, а график перерисовывается (см. рис. 8.7, 8.8).

Результаты анализа, полученные в виде двумерных таблиц и соответствующих им графиков, называются **брифингами**. Их можно сохранять и использовать при последующих обращениях.

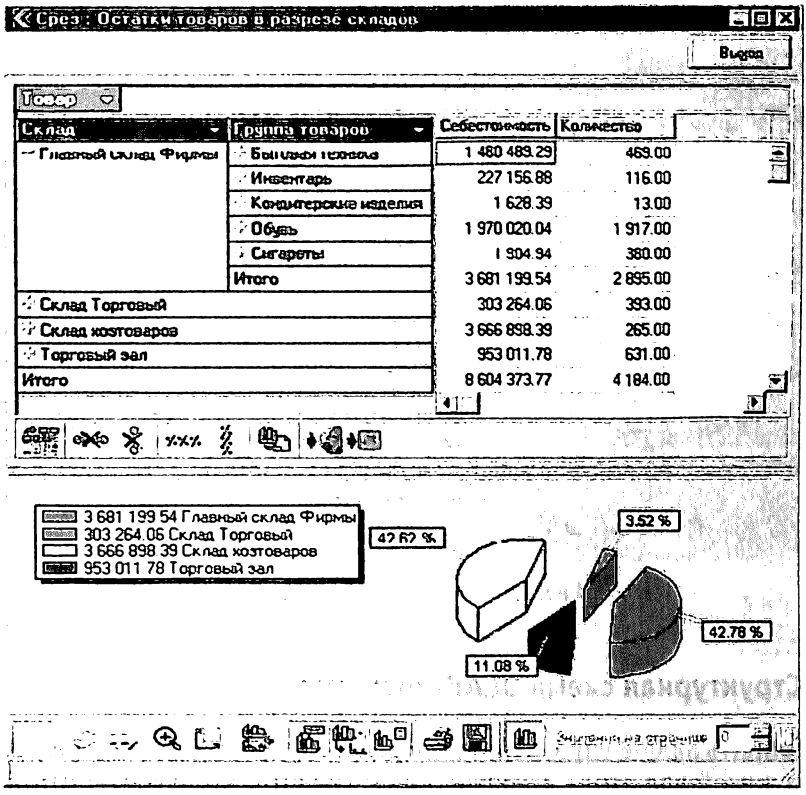


Рис. 8.8. Диалоговое окно для анализа остатков товара на складе

Для установки фильтров можно использовать и открытые Измерения. Чтобы сравнить динамику продаж конфет в Москве и Екатеринбурге, установим фильтры на измерения «Товар» и «Город» (рис. 8.9).

Закрыв ненужные Измерения и выбрав тип графика «Линия», получаем график, на котором можно проследить динамику продаж, оценить сезонные колебания и связь падений и роста сбыта товара в разных городах.

Таким образом, OLAP-технология позволяет пользователю из одного интерфейса выпустить десятки видов самых разных отчетов, управляя динамической OLAP-таблицей при помощи мыши. Задачей программиста, владеющего таким инструментом, становится не рутинное кодирование отчетных форм, а настройка OLAP-клиента на базы данных. При этом способы управления отчетом интуитивно понятны конечному пользователю.

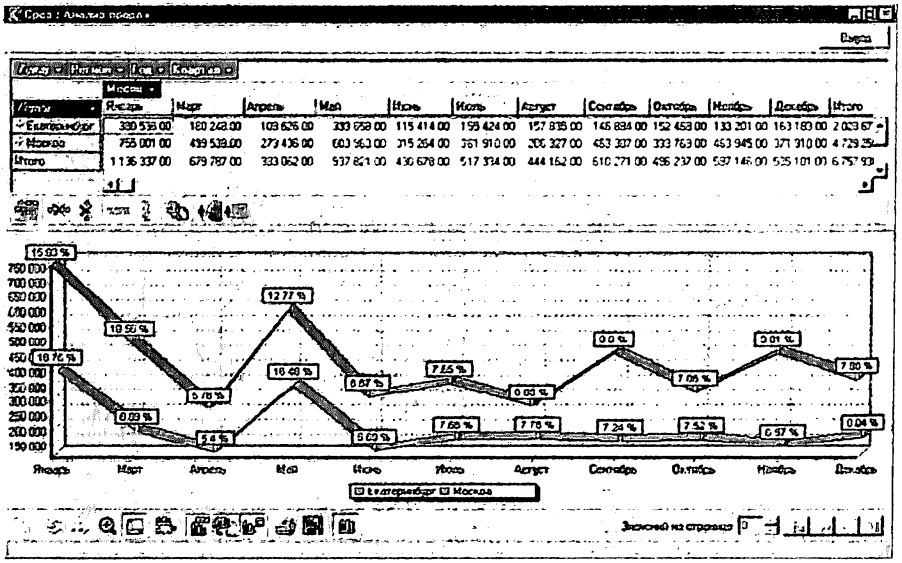


Рис. 8.9. Тип графика «Лента»

8.5. Структурная схема OLAP-системы

Принять любое управленческое решение невозможно, не обладая необходимой для этого информацией, обычно количественной. Для этого надо создавать **хранилища данных (Data warehouses)**, заполняя их информацией по итогам сбора, отсеивания и предварительной обработки данных с последующим предоставлением результирующей информации пользователям для статистического анализа, а нередко и для создания аналитических отчетов.

Ральф Кимбалл, один из авторов концепции хранилищ данных, описывал хранилище данных как «место, где люди могут получить доступ к своим данным» [16]. Он же сформулировал и основные требования к хранилищам данных:

- поддержка высокой скорости получения данных из хранилища;
- поддержка внутренней непротиворечивости данных;
- возможность получения и сравнения так называемых срезов данных;
- наличие удобных утилит просмотра данных в хранилище;
- полнота и достоверность хранимых данных;
- поддержка качественного процесса пополнения данных.

Удовлетворять всем перечисленным требованиям в рамках одного и того же продукта зачастую не удается. Поэтому для реализации хранилищ данных обычно используется несколько продуктов, одни из которых представляют собой собственно **средства хранения данных** (например, реляционные хранилища данных), другие — **средства их извлечения и просмотра** (OLAP-сервер), третьи — **средства их пополнения** и т. д. Все это в целом принято называть **OLAP-системой** (рис. 8.10).

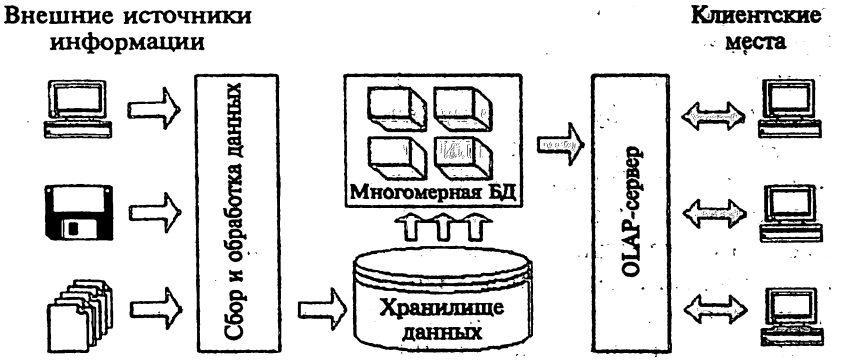


Рис. 8.10. Структурная схема OLAP-системы

Типичное **реляционное хранилище данных (ХД)**, как правило, отличается от обычной реляционной базы данных.

Во-первых, обычные БД предназначены для того, чтобы помочь пользователям выполнять повседневную работу, тогда как ХД предназначены для принятия решений. Например, продажа товара и выписка счета производятся с использованием БД, предназначенной для обработки транзакций, а анализ динамики продаж за несколько лет, позволяющий спланировать работу с поставщиками, — с помощью ХД.

Во-вторых, обычные БД подвержены постоянным изменениям в процессе работы пользователей, а ХД относительно стабильно: данные в нем обычно обновляются согласно расписанию (например, еженедельно, ежемесячно или ежеквартально — в зависимости от потребностей). В идеале процесс пополнения представляет собой просто добавление новых данных за определенный период времени без изменения прежней информации, уже находящейся в хранилище.

И, **в-третьих**, обычные БД чаще всего являются **источником данных**, попадающих в хранилище. Кроме того, ХД может пополняться

за счет внешних источников, например статистических отчетов, магнитных и бумажных носителей информации.

Не менее важно и то, что многие критически необходимые для оперативных систем функциональные возможности, реализуемые в реляционных СУБД, являются избыточными для аналитических задач.

Различие же на уровне OLTP- и OLAP-систем заключается, прежде всего, в том, что первые ориентированы на обработку транзакций в режиме on-line, т. е. они могут выполнять очень большое число небольших транзакций. Напротив, OLAP-системы предназначены для обработки в режиме off-line сравнительно простого запроса аналитика «Как изменилась доходность филиалов нашей организации по сравнению с прошедшим годом?». Но для ответа на этот вопрос в OLTP-системе потребуется чтение сотен тысяч записей! В OLAP-системе ответ на данный вопрос выдается в результате всего двух запросов через «дружелюбный» пользовательский интерфейс в табличном и графическом виде (см. рис. 8.7—8.9).

То есть вместо традиционных методов анализа данных, основанных на различных системах реализации SQL-запросов к реляционной базе данных, используется многомерная модель представления данных, в которой на смену таким понятиям, как отношения и сущности, приходят понятия измерений и кубов данных. Именно многомерность представления данных позволяет сделать механизм получения данных более доступным и интуитивно понятным аналитику, а многие ограничения языка SQL перестают существовать в силу принципиально иного механизма создания отчетов средствами OLAP.

Ядром OLAP-системы является многомерная база данных (МБД), предоставляющая возможность реализации механизма составления отчетов на основе OLAP-технологий. Система построена по принципу трехзвенной архитектуры «клиент—сервер» и обеспечивает удаленный и многопользовательский доступ к серверу МБД (см. рис. 8.10). При этом функции сервера приложений выполняет OLAP-сервер, а функции сервера баз данных — многомерная БД (гиперкуб).

Источником данных для OLAP-системы являются **внешние источники данных** — таблицы оперативных БД, представления, хранимые процедуры и т. п. Затем следует операция **выборка**, в которой настраиваются алгоритм объединения таблиц, условия фильтрации и набор полей, передаваемых в соответствии с современными воззрениями в специальную базу данных — хранилище данных (ХД).

Хранилище данных аккумулирует информацию, которая поступает из внешних источников. В зависимости от задачи ХД может быть

реализовано как на основе многомерной базы, так и на основе реляционной. Если хранилище данных реализуется на основе реляционной базы, то между ней и OLAP-системой все равно должен располагаться специализированный сервер многомерных баз данных. В этом случае ХД является поставщиком информации для многомерной БД, гиперкубическая структура которой позволяет манипулировать данными в соответствии с желаниями пользователя.

ХД чаще всего представляет собой реляционную базу данных, организованную по схеме «звезда» (см. рис. 8.11). Основная таблица ХД (**таблица фактов**) в этом случае содержит числовые значения показателей, по которым делаются запросы. С таблицей фактов связаны другие таблицы (**таблицы измерений**) в соответствии с топологией «звезда».

В целом создание хранилища данных из независимых источников данных — многоэтапный процесс, который предусматривает **извлечение данных** из каждого источника, **преобразование** их в соответствии со схемой ХД, **очистку**, а затем **загрузку** в хранилище.

Цель **этапа извлечения данных** — перенести данные из разнородных источников в базу данных, где их можно модифицировать и добавить в хранилище.

Чтобы поддерживать регулярные обновления и проверки качества данных, необходимо знать источник для каждого столбца в хранилище данных. Для документирования информации об источниках данных используется редактор **Data Warehouse Source Editor** из пакета **ERwin**. Поскольку ERwin может осуществить обратное проектирование из самых разнородных источников, имена таблиц и столбцов источников данных могут быть импортированы как из баз данных, так и из других моделей ERwin.

Для каждого столбца ХД можно внести информацию не только об использованном источнике, но и также дополнительную информацию о способах, режимах и периодичности переноса данных из источника в хранилище данных.

Этап преобразования данных обеспечивает устранение несоответствия в схеме и соглашениях о значениях атрибутов. Набор правил и скриптов, как правило, выполняет преобразование данных из исходной схемы в итоговую. Например, дистрибьютор может разделить имя каждого клиента на три части: имя, отчество (или инициалы) и фамилия. Для этого аналитик сначала должен извлечь записи, а затем, для каждой записи, преобразовать все столбцы с соответствующими тремя частями имени, чтобы получить значение для атрибута «Имя менеджера».

На этапе очистки устраняются ошибки, полученные при вводе данных или из-за различия в схемах. Они могут привести к тому, что таблица измерений «Менеджер» в схеме «звезда» будет иметь несколько соответствующих кортежей для одного менеджера, что приводит к неточным ответам на запросы и некорректным моделям ХД.

После того как данные извлечены и преобразованы, возможно, что их еще необходимо дополнительно обработать перед тем, как **загрузить** в хранилище. Как правило, утилиты фоновой загрузки поддерживают такие функции, как **проверка ограничений целостности, сортировка, суммирование, агрегирование** и выполнение других вычислений для создания производных таблиц, размещаемых в хранилище. Помимо наполнения хранилища, утилита загрузки должна позволять системным администраторам проверять статус; отменять, приостанавливать и возобновлять загрузку; возобновлять работу после ошибки без потери целостности данных

При **обновлении** ХД должны быть рассмотрены два вопроса: ***когда обновлять и как обновлять***. Обычно ХД обновляются периодически в соответствии с заранее установленным расписанием, например ежемесячно или еженедельно. Выполнять каждое обновление необходимо только в том случае, если для выполнения OLAP-запросов требуются текущие данные. Администраторы хранилища данных определяют правила обновления в зависимости от требований пользователей и трафика. Расписание обновлений может быть различным для разных источников данных.

В результате этих действий реляционная база данных, содержащая всю статистическую информацию о предметной области, преобразуется в хранилище данных в терминах OLAP, а процесс создания структуры аналитической системы сводится к определению измерений и, в случае необходимости, организации **витрин данных (data mart)**.

Дело в том, что некоторые организации вместо ХД, процесс конструирования которых достаточно сложный, строят витрины данных, содержащие информацию только для **конкретных подразделений**. Например, маркетинговая витрина данных может содержать только информацию о клиентах, продуктах и продажах и не включать в себя планы поставок. Витрины данных строятся значительно быстрее, чем ХД, но впоследствии могут возникнуть серьезные проблемы с их интеграцией, если первоначальное планирование проводилось без учета полной бизнес-модели.

ХД обычно на порядок больше оперативных баз, зачастую имеют объем от сотен гигабайт до нескольких терабайт. Как правило, храни-

лище данных поддерживается независимо от оперативных БД, поскольку требования к функциональности и производительности аналитических приложений отличаются от требований к транзакционным системам.

Рабочая нагрузка на ХД состоит из нестандартных, сложных запросов, которые обращаются к миллионам строк таблицы фактов и выполняют огромное количество операций сканирования, соединения и агрегирования. Время ответа на запрос в данном случае важнее, чем пропускная способность.

Данные, размещенные в гиперкубе, доступны пользователю для анализа с клиентских мест путем настройки OLAP-отчетов в **OLAP-сервере**: формирование сложных запросов, генерация отчетов в числовом и графическом исполнении, получение произвольных подмножеств данных и т. д.

На основании заданных Измерений и вычисляемых Показателей исследователь может легко формировать необходимые отчеты, активизируя соответствующие Измерения. В результате создается интерфейс аналитического приложения, который можно запускать в требуемые моменты времени и видеть соответствующие таблицы и графики, в которых, например, цены просуммированы по товарам, покупателям и периодам. При этом способы управления анализом интуитивно понятны конечному пользователю (см. рис. 8.7—8.9).

8.6. Структура хранилища данных по схеме «звезда»

Хранилище данных является одним из фундаментальных структур в OLAP-технологии. Аналогично реляционной БД, для создания ХД требуется проект схемы. Проектирование объектов хранилищ данных сходно с проектированием связей и сущностей для реляционной модели, но **отличается целями**. Если реляционная модель акцентируется на целостности и эффективности ввода данных, то **размерная модель (Dimensional)** ориентирована в первую очередь на выполнение сложных запросов к БД. В размерном моделировании для ХД принят стандарт модели, называемый **схемой звезда** (star schema), которая обеспечивает высокую скорость выполнения запроса посредством денормализации и разделения данных.

Идея схемы «звезда» заключается в том, что имеются таблицы для каждого Измерения, а все Показатели помещаются в одну таблицу (таблицу фактов), индексируемую множественным ключом, со-

ставленным из ключей отдельных Измерений. В результате хранилище состоит из **таблицы фактов**, которая описывает все транзакции, и **таблиц измерений** (рис. 8.11). Эта схема соответствует гиперкубу на рис. 8.3.

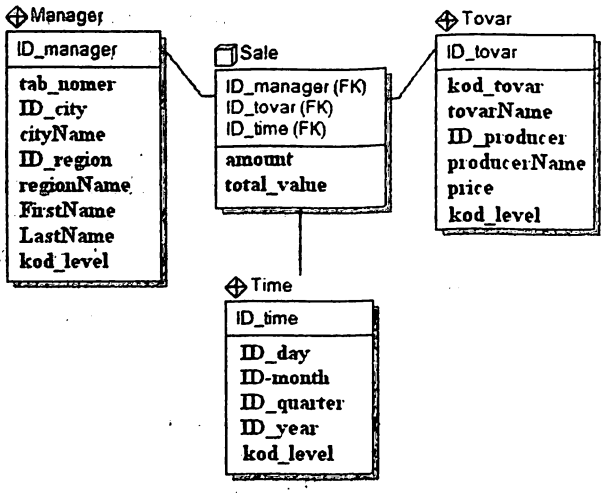


Рис. 8.11. Структура хранилища данных по схеме «звезда»

При этом таблицы измерений связаны с таблицей фактов отношениями «один ко многим»: таблицы измерений являются родительскими по отношению к таблице фактов.

Вычисление соединения между таблицей фактов и набором таблиц измерений — более эффективная операция, чем вычисление соединения произвольных реляционных таблиц. Но следует иметь в виду, что схема «звезда» обеспечивает наивысшую производительность при выполнении **только одного основного запроса** либо для группы похожих запросов.

Данные таблицы фактов — **это совокупность ключей и показателей.** Ключи связывают каждую строку таблицы фактов с ассоциированной строкой в таблице размерностей. Как и в схеме реляционной БД, первичный ключ **ID_tovar** таблицы размерностей **Tovar** становится внешним ключом в таблице фактов **Sale.**

Таблица фактов обычно содержит один или несколько столбцов (Показателей), дающих числовую характеристику какому-то аспекту предметной области (например, объем продаж для торговой компании или сумма платежей для банка), и несколько целочисленных колонок

(полей первичного ключа) — **ключей для доступа к таблицам измерений**. Таблицы измерений **расшифровывают ключи**, на которые ссылается таблица фактов; например, таблица измерений **«Tovar»** в схеме «звезда» может содержать сведения о названии товара, его производителе, типе товара.

В результате OLAP-система делает как бы мгновенный снимок реляционной БД и структурирует ее в пространственную модель для запросов. Заявленное время обработки запросов в OLAP составляет около 0,1 % от аналогичных запросов в реляционную БД.

В схеме «звезда» для каждого измерения куба содержится одна таблица измерений, в том числе и при наличии нескольких уровней иерархии (регион — город — табельный номер в таблице **«Manager»**, год — квартал — месяц — день в таблице **«Time»**). То есть за счет использования специальной структуры таблицы измерений **реализуется иерархия отношений Измерений**, в том числе ветвящаяся. По этой причине обычно данные в таблицах измерений денормализованы, за счет чего удается уменьшить число участвующих в операции соединения таблиц, что обычно приводит к значительному уменьшению времени выполнения запроса. Иногда тем не менее требуется произвести нормализацию таблиц измерений. Тогда получают схему, которая носит название **«снежинка»** (***snowflake schema***) [27].

Для разработки ХД по схеме «звезда» удобно использовать пакет **AllFusion ERwin Data Modeler (ERwin)**, который поддерживает методологию моделирования хранилищ благодаря применению специальной нотации для физической модели — **Dimensional**. Наиболее простой способ перейти к нотации Dimensional в ERwin — при создании новой модели (**меню File/New**) в диалоге ERwin **Template Selection** выбрать из списка предлагаемых шаблонов **Dimension**. В шаблоне Dimension присутствуют все необходимые для поддержки нотации размерного моделирования настройки.

Для каждой таблицы можно задать **шесть типов** правил манипулирования данными: обновление (Refresh), дополнение (Append), резервное копирование (Backup), восстановление (Recovery), архивирование (Archiving) и очистка (Purge). Для задания правила следует выбрать имя правила из соответствующего списка выбора. Каждое правило должно быть предварительно описано в диалоге **Data Warehouse Rule Editor** (меню **Edit/Data Warehouse Rule**). Для каждого правила нужно задать имя, тип, определение.

Например, определение правила дополнения данных может включать частоту и время дополнения (ежедневно в конце рабочего

дня), продолжительность операции и т. д. Связать правила с определенной таблицей можно с помощью диалога **Table Editor** [27].

Проект схемы «звезда», который показан на рис. 8.11, имеет несколько известных характеристик:

- каждая таблица в «звездной» схеме имеет индивидуальный первичный ключ, который устраняет противоречие между естественными первичными ключами и суррогатными первичными ключами. В хранилище базы данных назначение первичного ключа происходит как **суррогатное**, если есть натуральный ключ, который нужно сохранить для запросов. Тогда он определяется как **альтернативный ключ**. Например, в таблице **Tovar** атрибут **ID-tovar** введен в качестве суррогатного первичного ключа, а атрибут **kod_tovar** — в качестве альтернативного ключа;
- большинство полей **пустые** как в таблицах измерений, так и в таблицах с данными. Только первичные и внешние ключи обязательны для заполнения, и лишь первичный ключ уникален. Большинство полей должно иметь возможность быть nullable или оставаться незаполненными, потому что для этих полей в начале работы не будет данных;
- большинство полей в таблице фактов **числовые**. Таблица фактов есть цель исследований, и аналитикам требуются числа и факты.

Избыточность данных растет по всей схеме. Эта избыточность необходима для хранилища данных, чтобы добиться приемлемого уровня производительности. Количество данных в ХД обычно огромно по сравнению с количеством данных в реляционной БД. При написании запросов на языке SQL для работы с ХД по схеме «звезда» избыточность минимизирует число связей, необходимых для возврата данных, при этом производительность намного выше, чем при выполнении того же запроса в исходной реляционной БД [31].

Запрос к таблице фактов неподготовленный пользователь выполняет с помощью дружелюбного интерфейса, выбирая в его полях необходимые Измерения и их уровень иерархии. OLAP-система затем автоматически формирует **SQL-запрос** к схеме «звезда», который обычно содержит в себе:

- одно или несколько соединений таблицы фактов с таблицами измерений;
- несколько фильтров (SQL-оператор WHERE), применяемых к таблице фактов или таблицам измерений;
- группировку и агрегирование по требуемым элементам иерархии измерений.

Например, для получения среза за 2007 г., представленного на рис. 8.12, необходимо выполнить соединения всех таблиц схемы «звезда» (рис. 8.11), применить фильтр к таблицам «Time» и «Tovar», выполнить агрегирование по Измерениям «Tovar» , «Manager» для Показателя «Amount» для уровней иерархии «Производитель» и «Город».

SQL-запрос в этом случае будет выглядеть следующим образом [29].

```
SELECT
{Задаются Измерения среза на рис. 8.12}
Tovar.producer_id,
Manager.city_id,
{Рассчитывается величина Показателя «Amount»}
sum(Sale.amount) as summa
FROM
Sale, Manager, Time, Tovar
WHERE
{Выполняется соединение таблиц}
Sale.Manager_id = Manager.manager_id and
Sales.date_id = Time.date_id and
Sales.tovar_id = Tovar.tovar_id and
{Применяются фильтры}
Time.year_id = 2007 and
Tovar.producer _id = "АВТОВАЗ" Модель авто
GROUP BY
{Выполняется агрегирование по Измерениям «Tovar» , «Manager» для
Показателя «Amount»}
Tovar.producer _id, Manager.city_id.
END
```

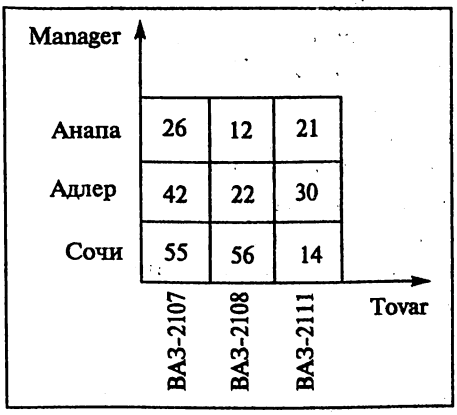


Рис. 8.12. Результат запроса к OLAP-системе (срез)

Результат запроса к OLAP-системе представляется в виде среза (см. рис. 8.12).

Особое свойство OLAP-систем — это **автоматический** расчет Показателя при иерархическом изменении размерностей Измерений. Например, «Сумма продаж» будет рассчитываться для всех сочетаний товаров, покупателей (или городов, или регионов) и периодов времени (дня, или недели, или месяца) при условии, что исходные данные (Показатели) занесены в таблицу фактов для самого младшего уровня.

OLAP-системы способны не только суммировать итоги, но и выполнять более сложные вычисления, включая статистический анализ.

8.7. Проектирование многомерной БД

Проектирование OLAP-систем существенно отличается от проектирования информационных систем класса OLTP тем, что для них основной задачей является разработка многомерной БД, а в качестве прикладного программного обеспечения и пользовательского интерфейса возможно использование штатных средств программного инструментария, например Oracle Express.

Поэтому обычно весь процесс проектирования OLAP-системы сводится к проектированию многомерной БД за несколько шагов.

Шаг 1. Анализ возможных запросов и определение основного вопроса. Первым шагом в проектировании МБД является определение запросов, с которыми конечные пользователи хотели бы обратиться к системе. Причем на этом этапе интерес представляют даже не сами тексты запросов, а понимание того, о каких личностях, местах, событиях и объектах в них спрашивается [15].

Например, пользователи хотели бы предсказывать продажи различных моделей автомобилей в разных регионах. Менеджеры по продажам хотели бы знать, какие продажи сделал каждый из сотрудников компании. В связи с тем что существует сезонная активность продаж автомобилей, менеджеры по продажам хотели бы оценить эффективность сезонных скидок. Руководство компании также хотело бы сравнивать объемы продаж различным потребителям по регионам с результатами за предыдущий квартал.

На основании анализа этих запросов формируют основной вопрос, на который должна давать ответы OLAP-система: «Какой объем продаж производится в выбранном временном интервале в заданной

точке продаж заданной модели автомобиля с учетом выбранного типа скидки?»

Шаг 2. Выделение Измерений. На основании сформированного основного вопроса производится выбор Показателей и Измерений. Измерениями, как правило, идентифицируются параметры, по которым пользователь запрашивает данные на самом нижнем уровне иерархии Измерений. Для нашего примера в качестве Измерений могут быть выбраны: «Месяц», «Регион», «Модель», «Типы скидок», а в качестве Показателя — «Объем продаж».

Шаг 3. Создание иерархий. После выделения Измерений необходимо создать Иерархии, которые их организуют и представляют собой отношение старшинства на множестве значений Измерения. Пользователь, перемещаясь по уровням Иерархии, выполняет операции детализации или агрегации. Поэтому очень важно правильно определить необходимое количество уровней Иерархии.

Если спросить заказчика, какой уровень детализации ему желателен, он не задумываясь ответит: максимально возможный. Однако стоит оценить, сколько такое решение может стоить, и попытаться определить возможный экономический эффект от наличия данных на каждом новом уровне детализации.

Например, выбрав в качестве уровня агрегации Год, можно проанализировать тенденции автомобильного рынка и спрогнозировать динамику его развития. Выбрав же в качестве уровня агрегации Месяц или Неделю, возможно прогнозирование спроса на конкретные модели в конкретные моменты времени.

Это позволит отследить возможные сезонные колебания, рациональнее формировать запасы на складе и более эффективно проводить политику формирования сезонных скидок и распродаж. А если в систему введена информация о затратах на маркетинг, появится возможность проследить эффект от каждого конкретного маркетингового мероприятия [33].

Однако переход на каждый следующий уровень детализации и добавление новых источников данных могут привести к увеличению, иногда более чем на порядок, размера целевой МБД и соответствующему удорожанию и усложнению аппаратного решения.

Поэтому в OLAP-системе важно найти баланс между тем, что хранится на диске, и тем, что можно подкачать в память. Какой объем памяти нужен на сервере — зависит от структуры куба и агрегатов, объема данных, запросов пользователя и их количества. При работе с Oracle Express используют «бытовой» индикатор: если во время агре-

гации или запроса пользователя процессор занят на 100 %, значит, системе хватает памяти. А если процессор не загружен, значит, происходят чтение и запись с дисков или свопирование. Какой потребуется объем памяти, если ее не хватает, устанавливают только экспериментально [9].

Рассмотрим в качестве примера Показатель «Объем продаж». Анализ предметной области показывает, что он однозначно определяется комбинацией четырех Измерений:

1) {Год \ Квартал \ Месяц \ Неделя \ День};
2) {Страна \ Регион \ Филиал \ Менеджер};
3) Фирма-производитель \ Завод-производитель \ Модель авто};
4) {Тип скидки}.

Если выбрать следующие уровни детализации:

- день (365 × 10 = 3650 различных значений);
- менеджер (300 различных значений);
- модель автомобиля (100 различных значений);
- тип скидки (4 различных значения),

то получим гиперкуб, состоящий из 438 000 000 ячеек. В результате многомерной БД будет сразу же зарезервировано место для всех 438 млн значений Показателя «Объем продаж». Причем цифры «300 менеджеров» и «100 моделей автомобилей» вовсе не означают того, что сегодняшняя номенклатура фирмы — 100 различных моделей, которые продают 300 человек. Цифра 300 говорит о том, что в фирме за 10 лет ее существования работало 300 различных менеджеров. Сегодня же их может быть, например, всего 30.

Теперь следует рассчитать, какой процент ячеек будет содержать реальные значения после 10 лет эксплуатации. Предположим, что в среднем в фирме постоянно работает около 30 менеджеров, менеджер продает в день 10 различных моделей и при продаже каждого автомобиля может быть использован один вариант скидки. Тогда (3650 × 30 × 10 × 1) = 1 095 000. То есть только 0,25 % ячеек куба будет содержать реальные значения данных.

В основе используемого в МСУБД способа хранения данных лежит предположение о том, что внутри гиперкуба нет пустот. Значения Показателей хранятся в виде множества логически упорядоченных блоков, имеющих фиксированную длину. При этом если блок не содержит ни одного значения Показателя, то он не подлежит хранению [33].

Шаг 4. Определение показателей. После выбора Измерений необходимо выбрать Показатели, исходя из содержания основного вопроса.

В нашем примере в роли Показателя выступает «Объем продаж», который однозначно определяется выбранными в предыдущем разделе Измерениями: «Время», «Точки продаж» и «Товар». В этом же пространстве Измерений могут быть выбраны и другие Показатели, например: «Себестоимость продаж авто», «Количество продаж» и «Доход».

При этом значения этих Показателей будут рассчитаны для всех сочетаний уровней иерархии при загрузке данных в куб и будут в нем храниться. Для выбранного уровня детализации опишем структуру данных: полный перечень Измерений и Показателей, используемых в примере (табл. 8.2).

Таблица 8.2. **Перечень Измерений и Показателей**

Имя	Дескриптор	Тип
DAY	День	Измерение
MANAGER	Менеджер	Измерение
MODELCAR	Модель автомобиля	Измерение
FIRMACAR	Фирма-производитель	Измерение
DEPARTMENT	Подразделение	Измерение
REGION	Регион	Измерение
MONTH	Месяц	Измерение
YEAR	Год	Измерение
TIPJDISCOUNT	Тип скидки	Измерение
TOTAL.COST	Объем продаж	Показатель
INTCOST	Себестоимость продаж	Показатель
QUANTITY	Количество продаж	Показатель
PROFIT	Доход	Показатель

Шаг 5. Объявление Измерений, Показателей, Отношений. В Oracle Express Server описание структур данных можно выполнить средствами языка описания данных Data Definition Language (DDL).

Объявление Измерений (табл. 8.2) выглядит следующим образом:

```
Define MANAGER Dimension Text
Define MODEL_CAR Dimension Text
Define DAY Dimension Date
Define MONTH Dimension Date
```

Define YEAR Dimension Date
Define FIRMA_CAR Dimension Text
Define TIP_DISCOUNT Dimension Text

Объявление Показателей реализуется несколько сложнее:

Define TOTAL_COST Variable Decimal <DAY MANAGER MODEL_CAR TIP_DISCOUNT>.

В этой строке определен Показатель «TOTAL_COST» как переменная Variable с указанием того, что данный Показатель определяется четырьмя Измерениями. При этом порядок, в котором перечислены Измерения, определяет последовательность, в которой будут отсортированы значения TOTAL_COST в БД: первым изменяется значение Измерения День, а затем — Менеджер, Модель автомобиля, Тип скидки.

Кроме этого, неопределенные значения TOTAL_COST, которые могут появиться из-за того, что Менеджер в данный момент времени не работал в фирме, будут идти подряд. Благодаря этому существенно снизятся непроизводительные затраты на хранение неопределенных значений в МБД.

Остальные показатели определяются аналогично:

Define INT_COST Variable Decimal <DAY MANAGER MODEL_CAR>
Define QUANTITY Variable Decimal <DAY MANAGER MODEL_CAR TIP_DISCOUNT>
Define PROFIT Formula TOTAL_COST - INT_COST Decimal

Показатель PROFIT имеет **тип Формула** и вычисляется как разность между TOTAL_COST и INT_COST.

Объявление Отношений.
Define FIRMA_CAR Relation FIRMA_CAR <MODEL_CAR>

Таким образом, объявлено, что каждой модели автомобиля (MODEL_CAR) однозначно соответствует выпускающая ее фирма (FIRMA_CAR). Для окончательного определения отношения требуется ввести соответствующие пары значений, соотносимых этим Отношением Измерений. Например, для фирмы-производителя АВТОВАЗ:

- АВТОВАЗ ВАЗ 2106;
- АВТОВАЗ ВАЗ 2108;
- АВТОВАЗ ВАЗ 2121.

И хотя при объявлении Показателя «Объем продаж» явно не указывается его взаимосвязь с «Фирмой-производителем», это не означает, что пользователь не сможет получать итоговые результаты не на уровне отдельных моделей, а на более высоком уровне агрегации — по каждой фирме.

В дальнейшем при любых операциях манипулирования данными (выборках, вычислениях и т. д.) пользователь имеет возможность ссылаться как на «Модель автомобиля», так и непосредственно на «Фирму-производителя». И для того чтобы получить Срез по все моделям, производимым на конкретной фирме, будет достаточно просто указать имя фирмы (например, АВТОВАЗ), а не перечислять все наименования моделей (ВА32106, ВА32108, ВА32108), выпускаемых на ней.

Аналогичным образом описываются Отношения для всех Измерений.

Шаг 6. Использование аналитических функций. Кроме функций агрегации, в Oracle Express включены обширные библиотеки встроенных функций для математической, финансовой, статистической обработки данных (табл. 8.3), а также обеспечивается возможность определения и подключения собственных, определяемых пользователем функций и написания пользовательских программ и моделей.

Таблица 8.3. **Список встроенных функций ORACLE Express (часть функций)**

Название функций	Описание
Функции, основанные на временных рядах	
CUMSUM LAG LAGABSPCT	Вычисляет нарастающий итог. Вычисляет значения за предшествующий временной период
Функции агрегации	
AVERAGE AVV	Возвращает среднее значение. Проверяет, есть ли значения, соответствующие заданному критерию
Финансовые функции	
DEPRDECL, DEPRSL DEPSOYD	Вычисляет амортизационные отчисления. Вычисляет расписание выплаты процентов по займам
Прочие функции	
REGRESS FORECAST TREND FORECAST	Вычисляет линейную регрессию. Вычисляет линейный прогноз. Вычисляет экспоненциальный прогноз Holt-Winters

В качестве примера рассмотрим вычисления на оси времени, используя исторические данные по продажам и встроенную функцию **FORECAST** (вычисляет линейный прогноз) из табл. 8.3, для прогнозирования своих действий на будущие периоды. Эта функция поддерживает несколько различных методов предсказания: линейный, экспоненциальный и другие.

Предположим, что менеджер по продажам хочет предсказать продажи в южном регионе в 2011 г. с помощью экспоненциального метода, который экстраполирует исторические данные в предположении о постоянстве скорости роста объема продаж от одного периода времени к другому. Для решения задачи менеджеру следует установить следующие параметры: «Регион» (Юг); «Товар» (автомобили АВТОВАЗа); временной период, на основе которого прогнозируется предсказание (два года); длина предсказания (один год).

На языке Oracle Express предсказание будет описано следующим кодом, который генерируется автоматически при установке пользователем в полях интерфейса указанных выше параметров:

```
define fsct_TOTAL_COST decimal <month>
limit product to АВТОВАЗ
limit region to Юг
limit month to JAN 2007 to DEC 2009
FORECAST length 24 exponential fcname- time month TOTAL_COST
```

Шаг 7. Программная реализация операций загрузки данных. В Oracle Express Server загрузка данных может производиться практически из внешнего источника данных, включая:

- различные реляционные СУБД (через соответствующий ODBC- драйвер);
- плоские файлы с фиксированной структурой записей;
- электронные таблицы (Lotus 1-2-3, Excel и т. д.);
- в интерактивном режиме через специально написанные пользовательские приложения или непосредственно Oracle Express Administrator.

В случае использования в качестве источника данных реляционной БД процедуры выгрузки и загрузки данных обычно оформляются в виде программ на языке манипулирования данными Oracle Express, в которые могут быть включены операторы языка SQL.

Это специальный язык типа 4GL, с помощью которого в базе можно выполнять следующие действия:

- целиком **определять многомерную базу** данных, включая шкалы, переменные, формулы и т. д.;

• **заносить в базу данные** — например, с помощью загрузки плоских файлов или путем непосредственного подключения к реляционной БД;

• **агрегировать данные по иерархиям**, или же задавать весьма сложные вычисления;

• **выбирать требуемые данные** для построения плоских файлов или сложных отчетов.

Язык манипулирования данными 4GL используют при взаимодействии с базой данных такие продукты, как Express Administrator или же Express Analyzer.

Например, следующая программа может быть использована для считывания данных из реляционной таблицы **TIME_TABLE (R_MANAGER, R_DAY, R_WORK_TIME)** и загрузки их в Показатель **WORK_TIME**:

```
PROGRAMM
{Определить курсор CURS_1 для чтения данных из таблицы TIME_TABLE
и определить имена колонок, из которых будут выбираться данные}
sql declare • CURS_1 cursor for
SELECT R_MANAGER, R_DAY, R_WORK_TIME
FROM TIMEJTABLE
{Открыть курсор}
sql open CURS_1
{В цикле считывать данные из реляционной таблицы в соответствующие
поля МБД}
: .Manager, :.Day, :.Work_Time }
while sqlcode eq 0 do
sql fetch CURS_1
into : .Manager, :.Day, :.Work_Time
doend
{По достижении конца таблицы или при возникновении ошибок закончить
сеанс}
sql close CURS_1
sql disconnect:
return.
```

Следует заметить, что в Oracle Express данные могут храниться как на постоянной основе, так и загружаться динамически в тот момент, когда к ним обратится пользователь. Таким образом, имеется возможность постоянно хранить в МБД только ту информацию, которая наиболее часто запрашивается пользователями. Для всех остальных данных хранятся только описания их структуры и программы их выгрузки

из центральной (обычно реляционной) БД. И хотя при первичном обращении к таким виртуальным данным время отклика может оказаться достаточно продолжительным, такое решение обеспечивает высокую гибкость и требует более дешевых аппаратных средств.

8.8. Классификация OLAP-систем

OLAP-продукты, реализующие MOLAP, весьма разнообразны: например, семейство Oracle Express, в состав которого входят сервер МБД Express Server, система анализа МБД Express Analyzer, средства разработки OLAP-приложений Express Objects, несколько утилит и готовых приложений (рис. 8.13). Эти продукты позволяют создавать сложные OLAP-приложения. В других продуктах OLAP-система реализована как опция, например в пакете Seagate Info.

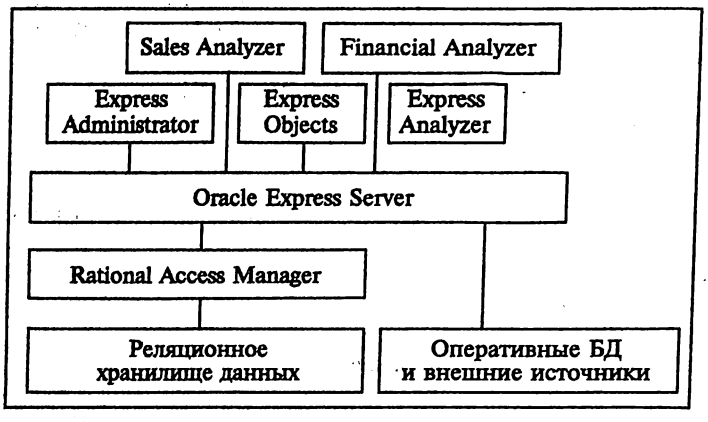


Рис. 8.13. Семейство программных продуктов Oracle Express

Аналитические модули появились в составе пакетов финансово-производственных приложений SAP R/3, Oracle Applications и других. Одновременно с лидерами мирового рынка подобные модули включили в свои системы российские фирмы «Парус» и «АйТи».

Многомерная модель в OLAP является логической и может быть реализована в двух вариантах: как базовая в СУБД — в этом случае речь идет о многомерной БД, т. е. MOLAP, и может быть смоделирована на основе реляционной БД (тогда говорят о ROLAP) благодаря применению схемы типа «звезда».

Поэтому наиболее популярной является классификация OLAP-продуктов **по способу хранения данных.** И исходные, и агрегатные данные для кубов могут храниться как в реляционных, так и многомерных базах данных. Поэтому в настоящее время применяются три способа хранения данных: **MOLAP (Multidimensional OLAP), ROLAP (Relational OLAP)** и **HOLAP (Hybrid OLAP).** Соответственно, OLAP-продукты по способу хранения данных делятся на три аналогичные категории.

Самые первые системы оперативной аналитической обработки (например, Oracle Express Server компании Oracle) относились к классу **MOLAP**, т. е. могли работать только со своими собственными многомерными БД. Они основываются на патентованных технологиях для многомерных СУБД и являются наиболее дорогими. Эти системы обеспечивают полный цикл OLAP-обработки.

Системы оперативной аналитической обработки реляционных данных (**ROLAP**) позволяют представлять данные, хранимые в реляционной базе, в многомерной форме, обеспечивая преобразование информации в многомерную модель через промежуточный слой метаданных.

Наконец, гибридные системы (**Hybrid OLAP, HOLAP**) разработаны с целью совмещения достоинств и минимизации недостатков, присущих предыдущим классам: они объединяют аналитическую гибкость и скорость ответа MOLAP с постоянным доступом к реальным данным, свойственным ROLAP.

В случае **MOLAP** исходные и многомерные данные хранятся в многомерной БД или в многомерном локальном кубе. Такой способ хранения обеспечивает высокую скорость выполнения OLAP-операций за счет превосходных свойств индексации. Но многомерная база в этом случае чаще всего будет избыточной, отсюда следует низкий коэффициент использования дискового пространства, особенно в случае разреженных данных. Объем куба, построенного на ее основе, будет сильно зависеть от числа измерений. При увеличении количества измерений объем куба будет экспоненциально расти, и иногда это может привести к «взрывному росту» объема данных.

Все рассмотренное выше основывалось на предположении, что данные для анализа загружаются в МБД и именно МБД является хранилищем всей информации.

В **ROLAP** все данные хранятся в реляционной БД, а для пользователя создается многомерное представление на логическом уровне в виде слоя метаданных (таблиц) в той же реляционной БД. Этот слой отображает реляционные термины (**таблицы, связи, ключи**) в много-

мерные (**измерения, иерархии, показатели**), скрывая, таким образом, от конечного пользователя сложную структуру реляционной БД (схема «звезда»).

Агрегатные данные могут помещаться в служебные таблицы в той же БД. Преобразование данных из реляционной БД в многомерные кубы происходит по запросу OLAP-средства. При этом скорость построения куба будет сильно зависеть от типа источника данных.

Главным недостатком ROLAP по сравнению с многомерными БД является меньшая производительность. Для обеспечения производительности, сравнимой с MOLAP, реляционные системы требуют тщательной проработки схемы базы данных и настройки индексов, т. е. больших усилий со стороны администраторов БД. Только **при использовании звездообразных схем** производительность хорошо настроенных реляционных систем может быть приближена к производительности систем на основе многомерных БД.

В случае использования **Гибридной архитектуры (HOLAP)** исходные данные остаются в реляционной базе, а агрегаты размещаются в многомерной. Построение OLAP-куба выполняется по запросу OLAP-средства на основе реляционных и многомерных данных. Такой подход позволяет избежать взрывного роста данных. При этом можно достичь оптимального времени исполнения клиентских запросов.

В заключение необходимо сказать, что было бы не совсем правильно противопоставлять или говорить о какой-либо серьезной взаимной конкуренции реляционного и многомерного подходов. Правильнее сказать, что эти два подхода взаимно дополняют друг друга. Реляционный подход никогда не предназначался для решения на его основе задач, требующих синтеза, анализа и консолидации данных. Изначально предполагалось, что такого рода функции должны реализовываться с помощью внешних по отношению к реляционной СУБД инструментальных средств. Именно на решение таких задач и ориентированы OLAP-системы. Область, где они наиболее эффективны, — это хранение и обработка высокоагрегированных и стабильных во времени данных.

Контрольные вопросы

1. Выполнить сравнительный анализ OLTP- и OLAP-систем.
2. Объяснить основные достоинства хранения информации в многомерном кубе.

3. Дать характеристику основным понятиям многомерной модели: Измерение, Показатель.
4. Объяснить назначение операций манипулирования с Измерениями: срез, вращение, детализация и агрегация.
5. Назначение и содержание иерархии отношений в многомерном кубе.
6. Особенности работы пользователей с OLAP-системами.
7. Объяснить структурную схему OLAP-системы.
8. Структура хранилища данных по схеме «звезда».
9. Дать характеристику содержания шагов проектирования МБД.
10. Использование аналитических функций для обработки данных в МБД.
11. Программная реализация операций загрузки данных в МБД.
12. Классификация OLAP-продуктов по способу хранения данных: MOLAP, ROLAP и HOLAP.

Глава 9
ВЫБОР И ВНЕДРЕНИЕ ИНФОРМАЦИОННЫХ СИСТЕМ

9.1. Общие вопросы выбора ИС

Рассмотренный ранее подход в проектировании ИС, называемый **каноническим или оригинальным**, предполагает разработку системы от постановки задачи до ввода в эксплуатацию. В качестве программного инструментария для его реализации используются CASE-средства: Oracle Designer 10g, SQL Server, Rational Rose и другие. Однако существует еще и другой подход, так называемое **типовое проектирование**, который реализуется на основе выбора информационной системы, наиболее подходящей для данной предметной области, с ее последующей адаптацией по определенным технологиям.

Выбор и внедрение интегрированной, целостной информационной системы, которая позволяет автоматизировать «сквозные» бизнес-процессы предприятия, охватывающие все аспекты его деятельности, — задача сложная, трудоемкая, сопровождающаяся высокими рисками и требующая значительных временных и финансовых затрат. Поэтому владельцы и руководители предприятий приходят к этому решению постепенно, критично оценивая возможности отдачи от инвестиций и свой кадровый потенциал, тщательно выбирая функционал ИС, поставщиков систем и состав оборудования.

Компания, которая решила внедрить у себя ERP-систему, должна заранее определить рамки проекта с позиций оптимального соотношения его стоимости, сроков выполнения и качества. Нередко принимается решение о сужении функциональных рамок и «фрагментарном» внедрении ERP, что снижает эффективность системы.

Поэтому поставщики программных решений и компании-консультанты по созданию информационных систем направляют усилия на то, чтобы оптимизировать стоимость, сроки и качество реализации IT-проектов и одновременно заложить основу для создания интегри-

рованной информационной среды предприятия. Для уменьшения стоимости и сроков внедрения производители программного обеспечения ERP предлагают сегодня предварительно настроенные, «типовые» отраслевые решения, в которых можно последовательно и постепенно наращивать функциональность до требуемого уровня [8, 12].

После выбора информационную систему необходимо профессионально установить и настроить, организовать обучение персонала, обеспечить надежную, бесперебойную работу техники и четкую организацию технологических процедур и, наконец, добиться эффективной эксплуатации установленных технологий. Причем это касается любого типа систем — от самых простейших систем управления взаимоотношениями с клиентами до самых сложных ERP-систем.

Если выбрана стандартная ИС, то существует два варианта внедрения: типовое и индивидуальное [9].

Типовое внедрение рекомендуется предприятиям, бизнес-процессы которых близки или идентичны схемам, заложенным в основу стандартных проектов. Использование общности черт и задач позволяет привязать готовые решения (модели и программы) к условиям конкретного пользователя и его задачам. Например, большинство организаций решает типовые задачи в бухгалтерском учете, финансах, организации управленческого труда, автоматизации документооборота, управлении кадрами и т. п. В рамках таких задач использование типовых решений будет оправданным и эффективным. Особенно это касается малого бизнеса. При типовом внедрении максимально используются штатные средства настройки ИС без изменения программного кода, и поэтому объем выполняемых работ ограничен:

- установка программного обеспечения;
- обучение пользователей;
- ввод в эксплуатацию программного обеспечения.

Индивидуальное внедрение применяется при необходимости внесения значительных изменений в программные продукты. Но только оно позволяет максимально точно решить задачи, стоящие перед предприятием, и учесть все особенности деятельности. При индивидуальном внедрении выполняется:

- технический проект на доработку программного обеспечения;
- программирование и тестирование изменений;
- установка программ, пусконаладочные работы, обучение пользователей, перенос данных из старой системы в новую ИС;
- опытная эксплуатация, передача созданной системы в эксплуатацию.

Многие заказчики и пользователи ИС придерживаются установившегося мнения, что индивидуальное внедрение на порядок дороже, чем приобретение готовой системы с гибким механизмом настроек, т. е. типовое внедрение. Но практика адаптации ИС показывает, что стоимость настройки системы с готовым начальным функционалом сопоставима со стоимостью доработок программного кода системы.

Затем возникает альтернатива — нужно ли выбрать **монолитную систему** известного производителя, которая решает все задачи корпоративного управления, либо формировать ИС на основе подсистем лучших в своих классах продуктов. Выбор зависит от структуры и размера предприятия, оказываемых услуг и предпочтений руководства.

Некоторые предприятия выбирают монолитную систему известного производителя, например фирмы «Ист Консепт», которая поставляет на рынок полный набор взаимосвязанных систем для управления отелем, рестораном, кафе, столовой, магазином, санаторием, клубом, развлекательным центром, туристической компанией, интернет-бронированием. Учитывая тот факт, что все эти системы могут работать автономно, их внедрение можно производить последовательно, оптимально распределяя по времени затраты на приобретение, внедрение и сопровождение. Например, фирма «Libra» предлагает следующий набор основных функциональных блоков для автоматизации гостиницы:

- бронирование;
- групповое бронирование и управление группами;
- регистрация и поселение гостей;
- регистрация паспортов;
- расчеты с гостями;
- управление клубом, расчеты по клубным картам;
- управление номерным фондом;
- инженерная служба, управление ремонтом;
- работа горничных;
- история гостя, программы частого гостя (программы поощрений);
- подарочные сертификаты;
- история компании;
- работа с туристическими агентствами;
- работа с контрагентами;
- консьерж, телефонный оператор, электронный консьерж;
- управление апартаментами;

- ночной аудит;
- управление тарифами.

Другие пытаются сформировать свою систему из подсистем ИС различных производителей, которые, по их мнению, лучше подходят для автоматизации бизнес-процессов предприятия. Например, для управления санаторием можно выбрать подсистему фирмы «Ист Консепт» для учета номерного фонда, для диспетчеризации медицинских услуг — подсистему фирмы «Дейманд», для управления питанием — подсистему фирмы «Libra» и т. п.

Выбранные подсистемы объединяются для работы в едином информационном пространстве с помощью штатных или специально разработанных интерфейсов (шлюзов). Эти подсистемы можно поэтапно внедрять, считая этот путь менее затратным и трудоемким, а риск того, что все указанные системы выбраны неудачно или все бизнес-процессы предприятия будут одновременно заблокированы, чрезвычайно мал.

Именно этот подход достаточно мощно поддерживается корпорациями по разработке прикладного программного обеспечения, которые отказываются от создания систем, охватывающих все подразделения предприятия. Во-первых, каждое предприятие подбирает системы выборочно и постепенно, исходя из своих потребностей и бюджета. Во-вторых, универсальная система слишком тяжеловесна технически, трудно и долго внедряется, требует большого количества специалистов и технологов различных узких специализаций и практически не поддается сбалансированному развитию и качественному сопровождению.

Поэтому современные решения автоматизации предприятий базируются на скоординированном взаимодействии нескольких специализированных систем, поставляемых профессиональными компаниями, которые могут не только качественно установить свои системы и обучить персонал, но также обеспечить хороший уровень круглосуточного технического и технологического сопровождения [35].

Ответственной задачей является правильный **выбор производителя ИС**, а также **поставщика** системы, который будет продавать и внедрять систему. От их выбора зависит не только качество системы, но и качество обслуживания, сопровождения и даже стоимость лицензии и внедрения системы, схема оплаты. Обычно придерживаются следующих основных критериев выбора производителя и поставщика:

- время работы и известность на рынке ИС;
- количество успешных внедрений;

• перечень услуг при внедрении ИС (обследование предприятия, консалтинговые услуги, обучение персонала, проведение деловых игр и т. п.);

• условия гарантийного и послегарантийного обслуживания;

• лицензионная чистота программного продукта и средств разработки;

• позиции фирмы в рейтингах.

Чрезвычайно полезно посетить предприятия, на которых внедрены ИС данного производителя, и проанализировать качество и объем выполняемых функций, оценить окупаемость затрат, качество и стоимость оказываемых услуг.

Еще более ответственной задачей является **правильный выбор ИС**, которые сегодня представлены на рынке сотнями наименований. Довольно часто предприятие приступает к выбору ИС, не имея четкого представления о том, что же оно в конечном итоге хочет получить, ориентируясь в основном на стоимость системы.

Абсолютно недопустима ситуация, когда выбор информационной системы поручается сотрудникам собственного отдела автоматизации предприятия, которые не могут принять в расчет все особенности автоматизируемой предметной области. Лучше всего придерживаться точки зрения, что собственный IT-отдел принимает участие в выборе, но не играет при этом ведущей роли. Точно так же недопустима ситуация, когда руководитель авторитарным решением выбирает информационную систему, руководствуясь тем, что лучше него никто не знает, что на самом деле нужно предприятию.

На практике часто останавливают свой выбор на ИС, которая работает на предприятиях такой же отрасли. Однако необходимо помнить, что двух весьма похожих друг на друга предприятий не существует, у каждого предприятия есть своя «изюминка», за счет которой оно побеждает в конкурентной борьбе, есть своя собственная стратегия развития бизнеса, на поддержку которой, прежде всего, ориентирована ИС. Поэтому попытки слепо перенять чужой опыт могут привести к тому, что систему не удастся внедрить по техническим причинам (технологии, оборудование, особенности инфраструктуры и т. п.), или стоимость такого решения будет высокой и процедура настройки ИС под предприятие будет весьма трудоемкой.

Для исключения подобных крайностей необходимо до начала процесса непосредственного внедрения системы разработать модель бизнес-процессов предприятия, учитывая мнения руководителей и всех сотрудников, имеющих отношение к этим процессам. Именно

при создании бизнес-модели формируется «язык общения» консультантов, разработчиков, пользователей и руководителей предприятия, позволяющий выработать единое представление о том, ЧТО и КАК должна делать внедряемая информационная система.

Бизнес-модель является отображением предприятия, его информационно-управляющей системы и представляет собой совокупность графических и текстовых описаний, позволяющих не только понимать процессы управления предприятием, но и оптимизировать их в соответствии с общепринятыми эффективными принципами менеджмента, например Business Processes Redesign (BPR). Без подобной модели бизнес-процессов автоматизируемого предприятия, ревизии и реинжиниринга системы управления предприятием внедрение ИС не даст ожидаемого эффекта.

Построение модели бизнес-процессов и ее реижиниринг не следует поручать представителям разработчиков или поставщиков ИС, так как в этом случае будет разработана не модель собственных процессов, отражающих реальную ситуацию, а модель бизнес-процессов системы, которая «подогнана» под систему фирмы-разработчика. В случае потенциально крупного внедрения лучше всего для построения модели бизнес-процессов привлечь независимую консалтинговую компанию.

Отсутствие работ по описанию и реинжинирингу бизнес-процессов предприятия является самой распространенной ошибкой при выборе ИС. На сегодняшний день лишь небольшая доля предприятий ведет систематическую работу по формализации и оптимизации существующих бизнес-процессов. Реинжиниринг деятельности преследует три цели: повышение эффективности деятельности предприятия в целом, оптимизация бизнес-процессов для последующего внедрения системы, подготовка детального проекта и плана развертывания ИС. Всегда следует помнить об известном постулате: «Нельзя автоматизировать хаос!» Нередко для того чтобы ИС смогла дать положительный эффект, руководителю приходится перед внедрением системы привести в порядок не только бизнес-процессы, но и структуру управления предприятием.

Обычно уже сразу после этапа реинжиниринга бизнес-процессов повышается эффективность работы предприятия только за счет устранения дублирования функций и более четкого распределения ответственности за результат труда. Существует даже такое категоричное мнение некоторых руководителей при оценке эффекта от внедре-

ния ИС, что отдача от наведения порядка существенно выше, чем собственно от самого внедрения.

Только выполнение всех этих этапов позволит значительно снизить риски при выборе ИС и существенно облегчить работы на последующих этапах ее внедрения.

9.2. Этапы индивидуального внедрения ИС

Внедрение информационной системы должно осуществляться в тесном сотрудничестве поставщика и собственных специалистов предприятия, внедряющих данную систему. Процесс внедрения следует начать с назначения менеджера проекта и создания команды высококвалифицированных специалистов, подготовку которых и сертификацию желательно предварительно провести у поставщиков системы. При этом приоритет во внедрении следует отдавать команде заказчика, а команде поставщика оставить роль консультационной и технической поддержки. В целом успех IT-проекта напрямую зависит от квалификации специалистов, внедряющих систему.

Какой бы ни был выбран вариант — покупная ИС или заказная разработка, этапы индивидуального внедрения примерно одинаковы и совпадают с каскадной моделью жизненного цикла системы в соответствии с ГОСТ 34.601—90 «Автоматизированные системы. Стадии создания».

На этапе **формирования требований** проводится обследование предприятия, на котором будет внедряться ИС. Результаты обследования удобнее представить в виде организационной диаграммы, определяющей структуру предприятия, и диаграмм плавательных дорожек (Swim Lane), представляющих в графической форме основные сценарии управленческой деятельности предприятия. Для построения подобных графических диаграмм удобно воспользоваться пакетом All Fusion Modeling Suite [25, 17]. На практике для описания процессов управления предприятием требуются десятки подобных диаграмм, а их разработка и согласование занимает не одну неделю и даже не один месяц.

Данный комплект диаграмм может использоваться в качестве **бизнес-модели «AS IS»** («Как есть»). Эта модель позволяет понять, где находятся наиболее слабые места, в чем будут состоять преимущества новых бизнес-процессов и насколько глубоким изменениям подвергнется существующая структура организации производства. Детализа-

ция процессов позволяет выявить недостатки организации даже там, где функциональность кажется очевидной.

Найденные в модели «AS IS» недостатки должны быть исправлены при создании модели «TO BE» («Как должно быть») — модели новой организации управления процессами предприятия. Результирующим документом этапа формирования требований является **технико-экономическое обоснование** (ТЭО) (ГОСТ 24.202—80, РД 50-34.698—90).

На этапе **разработки концепции ИС** выполняются работы по наведению элементарного порядка в управленческих процессах предприятия и реструктуризации (реинжиниринга) бизнес-процессов на основании модели «AS IS». На этом этапе специалисты по внедрению должны выполнить тщательное обследование предметной области с позиций системного подхода.

Результаты обследования используются для разработки функциональной модели «TO BE» в виде иерархической функциональной модели на основе методологий IDEF0, IDEF3, DFD [25, 26]. Эта модель будет содержать весь набор функций, которые планируется автоматизировать с помощью ИС. Оптимизация бизнес-процессов в модели «TO BE» выполняется на основе последовательного использования технологий **ISD** (Information Systems Design — планирование информационных систем) и реинжиниринга бизнес-процессов (**BPR**).

Каждый функциональный блок модели может быть декомпозирован и представлен соответствующей диаграммой декомпозиции. Декомпозиция блоков диаграмм ведется до такого уровня, чтобы функциональные блоки нижнего уровня иерархии соответствовали элементарным программным модулям существующих ИС.

Итоговую функциональную модель удобно представить в виде «дерева узлов», на котором функциональные блоки нижнего уровня иерархии должны соответствовать функционалу внедряемой ИС (рис. 9.1). На диаграмме им соответствует весь перечень наименований блоков, не заключенный в прямоугольники. Этот набор блоков и является необходимой функциональностью для ИС, которую и выбирают из существующих на рынке систем по наличию именно этих функций.

Следует отметить, что при построении модели «TO BE» необходимо максимально удовлетворять требованиям процессного подхода к управлению, т. е. правильно определять состав функций каждого бизнес-процесса на нижних уровнях иерархии. Выходом каждого бизнес-процесса должен воспользоваться хотя бы один клиент, необходимо максимально сократить количество подразделений, поскольку большинство проблем возникает на границах между подразделе-

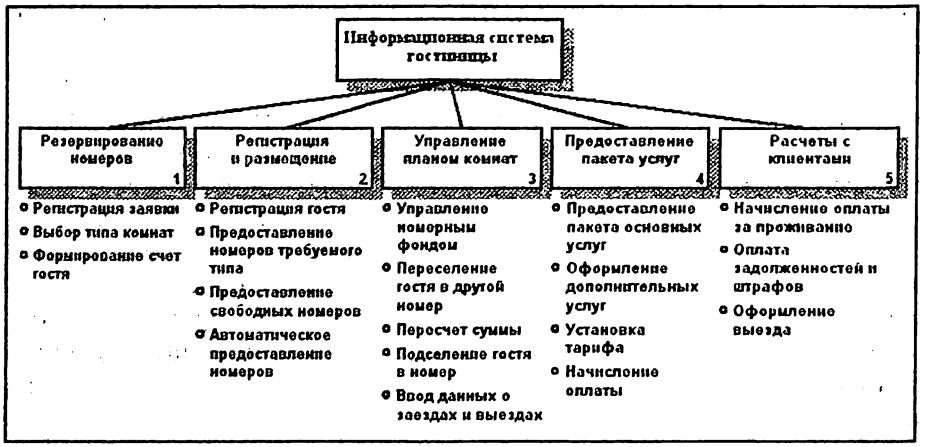


Рис. 9.1. «Дерево узлов» функциональной модели гостиницы

ниями организации. Для этого надо либо заранее описать действия на всех этапах бизнес-процесса, либо изменить структуру подразделений, либо изменить поток работ.

Полученная на предыдущем этапе функциональная модель позволяет определить функциональный состав, потоки информации, содержание входной и выходной информации, структуру пользовательского интерфейса требуемой системы. Этой информации достаточно для разработки **технического задания** (ТЗ) на ИС (ГОСТ 34.602—89), которое является основным юридическим документом во взаимоотношениях между поставщиком и заказчиком.

На этапе **технического проекта** необходимо разработать спецификации на программные модули, пользовательский интерфейс, описать требования к организации информационной базы на уровне входной и выходной информации.

Раздел для спецификаций программных модулей содержит перечень всех функциональных блоков нижнего уровня иерархии функциональной модели (см. рис. 9.1) и описание результата работы каждого блока: ввод документа, формирование документа или диалогового окна и т. д., со ссылкой на конкретную форму каждого документа.

Раздел для спецификаций пользовательского интерфейса включает сведения о составе пользователей будущей ИС, их функциональных обязанностях и правах доступа. Это необходимо для формирования такого состава диалоговых окон пользовательского интерфейса и обеспечения требуемых прав доступа, чтобы работа пользователей

ИС была эффективной и комфортной. Естественно, что пользовательский интерфейс должен обеспечивать управление всеми функциями, указанными в спецификации программных модулей.

Раздел организации информационной базы должен содержать перечень и описание входных и выходных сообщений.

На этапе **рабочая документация** производится выбор системы на рынке, которая в наибольшей мере соответствует требованиям технического проекта по функциональности, пользовательскому интерфейсу, составу входной и выходной документации. Дело в том, что чем шире функциональность системы, тем больше в ней содержится экранных форм и настроек и тем большее количество пользователей должно в такой системе работать. При этом каждый из этих пользователей должен получить на своем рабочем месте именно тот набор меню и опций, который ему необходим. Поэтому необходимо определить, как присутствие тех или иных функций повлияет на конечную стоимость системы, время внедрения и, что самое важное, соответствует ли предлагаемая функциональность целям компании.

На основе технического проекта производится отбор потенциальных кандидатов, отсеиваются системы, не подходящие по тем или иным критериям. Как правило, после завершения данного этапа специалистам предприятия остается подробно ознакомиться лишь с несколькими ИС. Затем уже организуется тендер среди прошедших предварительный отбор поставщиков. Обычно ориентируются на информационные системы, функциональность которых несколько шире, чем реальные бизнес-процессы компании.

В обязательном порядке следует договориться с поставщиками об организации посещения предприятий, на которых внедрены поставляемые ими системы, с целью получить отзывы о программном продукте, качестве установки и обучения, репутации компании-поставщика и взаимоотношениях с ней в ходе эксплуатации. Полезным мероприятием является проведение выездной презентации системы на фирме поставщика.

Возможно также осуществление на автоматизируемом предприятии пилотного внедрения ИС или деловой игры для специалистов предприятия. Для этого на предприятии развертывается конфигурация системы, которая функционально соответствует утвержденной версии модели бизнес-процессов «ТО BE», но с небольшим количеством рабочих мест только для ключевых пользователей. Организационная структура моделируемого предприятия может быть в разумных пределах также упрощена.

Группа ключевых пользователей по специально разработанному сценарию проводит **деловую игру**, имитируя управление работой своего предприятия. При этом желательно присутствие руководителей предприятия, которые могут разнообразить сценарий игры, создавая те или иные нештатные ситуации. При таком подходе руководители и ключевые пользователи получают возможность дополнительно проверить соответствие функциональности ИС потребностям предприятия и получить более глубокие знания о системе.

Этап **ввода в действие** начинается с проведения **опытной эксплуатации** системы после устранения поставщиком обнаруженных при деловой игре недостатков (ГОСТ 34.603—91). Это стадия реальной эксплуатации новой системы на ограниченном количестве рабочих мест. Все учетные операции в это время ведутся в «старой» системе, если она имеется. Для проведения опытной эксплуатации разрабатываются программа и методики испытаний, создается приказом руководителя предприятия комиссия по проведению испытаний (РД 50-34.698—90, ГОСТ 24.202—80). По результатам опытной эксплуатации создается акт, в котором отмечаются все обнаруженные ошибки или несоответствия техническому заданию и бизнес-модели.

Следует иметь в виду, что испытания могут проводить только представители заказчика, которые должны быть обучены поставщиком заранее.

После устранения поставщиком обнаруженных при опытной эксплуатации недостатков проводятся **приемочные испытания** ИС также по специально разработанным программе и методикам испытаний. Подписание членами приемочной комиссии акта о приеме ИС в промышленную эксплуатацию означает, что поставщиком выполнены все требования по контракту.

Этап **сопровождения ИС** заключается в выполнении гарантийных обязательств поставщиком обычно в течение года. После окончания гарантийного сопровождения большинство клиентов подписывают договор о дальнейшем сопровождении системы. Как правило, в стоимость такого сопровождения входит обновление системы и коррекция отчетных форм. Обучение вновь принятых на работу пользователей или разработка эксклюзивных отчетов оплачиваются отдельно.

Для эксплуатации ИС создается собственный коллектив либо система передается на обслуживание специализированному предприятию на условиях аутсорсинга. Их функции заключаются в анализе функционирования ИС, устранении обнаруженных ошибок и установке новых версий.

9.3. Типовое внедрение ИС

9.3.1. Типовые внедрения на базе пакетированных решений и модельно-ориентированного проектирования

Существует несколько современных подходов для решения задачи типового внедрения ИС. В области ERP-систем становится популярной концепция **«типовых решений»**, которая позволяет подобрать наиболее подходящую по функциональности готовую ИС с последующей ее адаптацией, если в этом появится необходимость. Поставщики ERP-решений предлагают ***предварительно настроенные системы*** в виде набора интегрированных функций, внедряемых в заданной последовательности, которые реализуют эталонные отраслевые модели бизнес-процессов. В этой концепции получили широкое распространение две стратегии внедрения: **пакетированные решения** и **модельно-ориентированное проектирование**.

Например, компания SAP разработала новую стратегию внедрения своих продуктов, основанную на применении пакетированного решения **SAP All-in-One ERP Package for IM&C** (http://www.topsbi.ru).

Пакетированное решение содержит предварительно настроенные, протестированные и детально документированные ***процедуры*** бизнес-процессов производственного предприятия, инструментарий автоматизации настроек системы и загрузки основных данных. Основной идеей таких решений является предоставление предприятию возможности максимально быстро начать использование ERP-системы в ключевых областях деятельности: в управлении производством, снабжением, сбытом, финансами.

В результате внедрения системы SAP будет реализована следующая система бизнес-процессов предприятия (рис. 9.2):

- управление производством;
- поддержка жизненного цикла продукта;
- логистика и финансовый учет;
- управление предприятием и службами поддержки;
- продажа и маркетинг;
- снабжение.

Пакетированное решение фактически обеспечивает **предварительно настроенную** интеграцию процессов. Оно содержит реализацию «сквозных» бизнес-процессов предприятия на основе передового опыта и дает возможность получить заранее известную цепочку свя-

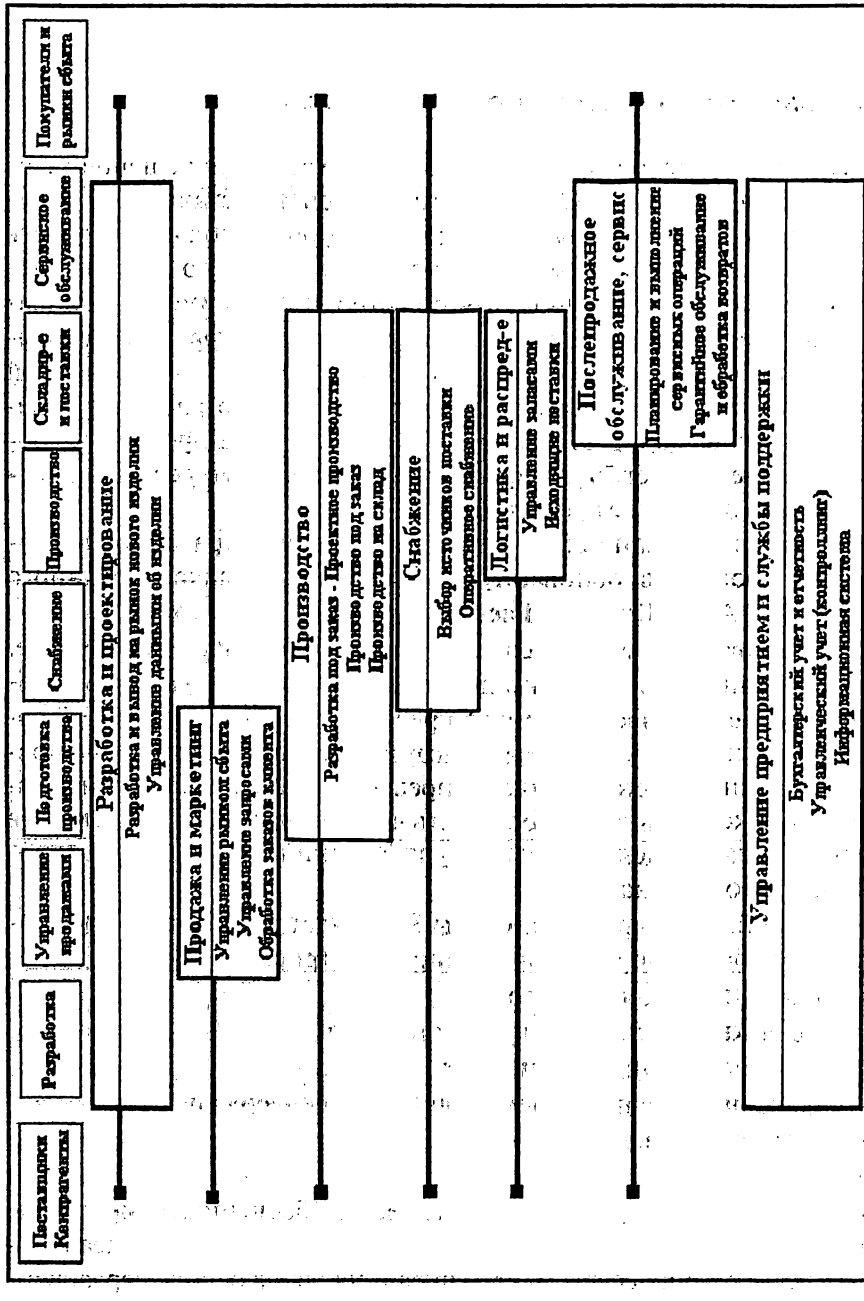


Рис. 9.2. Основные функции системы на основе пакетированного решения SAP для машиностроения

зей между различными функциями системы. Пакетированное решение предоставляет целый ряд преимуществ, связанных как со снижением рисков проекта, так и с сокращением целого спектра работ по внедрению, что сокращает сроки и снижает стоимость [8].

Подготовительный этап включает так называемую предконтрактную стадию, на которой обеспечивается четкое определение рамок проекта до его старта. Для этого заказчику предлагается проанализировать соответствие **преднастроенных моделей бизнес-процессов** специфике деятельности собственного предприятия. При выявлении отклонений уточняется объем необходимых доработок и принимается решение о готовности к использованию предлагаемых процессов.

Предпроектная стадия предполагает также подготовку менеджеров предприятия, определение стратегий управления проектом, техническую подготовку, организационную подготовку, разработку мотивации персонала, участвующего в проекте, создание собственной профессиональной проектной группы.

Опыт внедрения ERP-систем свидетельствует, что **стратегии внедрения** систем класса ERP можно свести к трем основным типам.

Стратегия «большого взрыва». При таком подходе предприятие осуществляет запуск одновременно ***большого количества функций системы*** и, соответственно, автоматизацию большого числа операций, ранее исполнявшихся вручную. С точки зрения сроков внедрения ERP-системы в широкой функциональности такой подход может принести определенный выигрыш, но обычно он сопровождается существенными рисками для бизнеса и требует от предприятия значительных трудовых и финансовых ресурсов.

«Внедрение по функциям» («внедрение по модулям»). Это довольно распространенная стратегия внедрения, предполагающая определение ***некоторой функциональной области бизнеса***, которая будет автоматизирована в первую очередь и где ожидается сразу получить ощутимый эффект. Такой областью может быть управление персоналом, управление финансами, управление материально-техническим снабжением и т. п. Обычно этот подход сопровождается рисками, связанными с тем, что при изолированном внедрении модулей недостаточно точно будут продуманы межфункциональные связи и при развитии системы возникнут проблемы интеграции модулей.

«Внедрение по процессам». Эта стратегия предполагает выбор функциональной области и всей цепочки связанных с ней бизнес-процессов, что обеспечивает автоматизированное выполнение ***определенной бизнес-задачи***. При выборе этой стратегии пакетирован-

ное решение также обладает преимуществами: оно содержит конкретные сценарии выполнения процессов, и эти сценарии, по сути, и являются готовыми стратегиями внедрения по процессам.

Для сокращения трудоемкости внедрения системы предлагается разбивать объект автоматизации ***на однородные по бизнес-процессам сегменты***. Например, при наличии у компании нескольких складов в качестве прототипа выполняется полное внедрение информационной системы на одном из складов. При этом охватываются все процессы объекта: учет поставок, движение материалов, материальный учет, отпуск в производство или продажа и т. д.

Полученное решение затем тиражируется на другие объекты (сегменты). Причем тиражирование на определенном этапе предприятие может выполнять уже силами собственных специалистов. Такой подход дает выигрыш по затратам и срокам внедрения и гарантированно приводит к успеху проекта.

Более гибким подходом для внедрения ИС является **модельно-ориентированное проектирование**, которое заключается в адаптации компонентов типовой ИС к модели предметной области конкретного предприятия. Для этого **технология проектирования** должна поддерживать как модель типовой ИС, так и модель конкретного предприятия, а также средства поддержания соответствия между ними (http://l2allpatch.narod.ru).

Таким образом, в этой стратегии в обязательном порядке предполагается построение модели объекта автоматизации с использованием специального программного инструментария (например, SAP Business Engineering Workbench (BEW), BAAN Enterprise Modeler) и ее периодической корректировкой. Для построенной модели предприятия находят в репозитарии наиболее подходящую **базовую модель**, на основе которой осуществляется конфигурация программного обеспечения проектируемой ИС (рис. 9.3).

Репозитарий в общем случае содержит метаинформацию базовой модели функциональности типовой системы, типовых моделей определенных классов ИС (для отраслей). Кроме этого, в него включаются модели предприятий, которые получаются на основе базовой или типовых моделей. Репозитарий поставляется вместе с программным продуктом и расширяется по мере накопления опыта проектирования информационных систем для различных отраслей и типов производства.

Базовая модель ИС в репозитории содержит описание бизнес-функций, бизнес-процессов, бизнес-объектов, бизнес-правил,

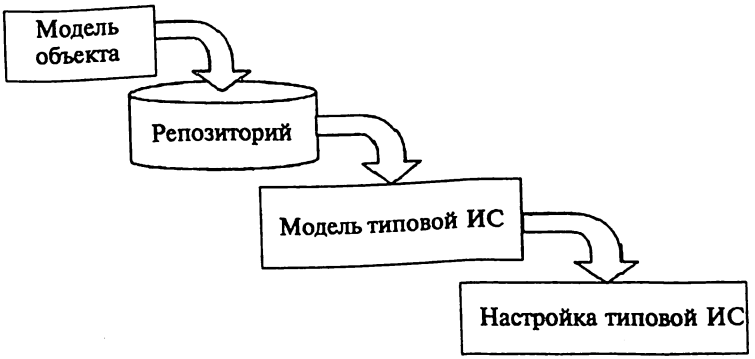


Рис. 9.3. Адаптация состава и характеристик типовой ИС в соответствии с моделью объекта автоматизации

организационной структуры, которые используются в программных модулях типовой ИС.

Типовые модели описывают конфигурации информационной системы для определенных отраслей (автомобильной, электронной, нефтегазовой и др.) или типов производства (единичного, серийного, массового, непрерывного и др.).

Модель конкретного предприятия строится либо путем выбора фрагментов основной или типовой модели в соответствии со специфическими особенностями предприятия (BAAN Enterprise Modeler), либо путем автоматизированной адаптации этих моделей в результате экспертного опроса (SAP Business Engineering Workbench).

Построенная модель объекта автоматизации хранится в репозитарии (специальной базе метаинформации) и при необходимости может быть откорректирована. На основе этой модели **автоматически осуществляется конфигурирование** и настройка информационной системы.

В обобщенном виде конфигурация корпоративных информационных систем на основе модельно-ориентированной технологии представлена на рис. 9.4. Рассмотрим компоненты модели предприятия более детально.

Модель функций представляет собой иерархическую декомпозицию функциональной деятельности предприятия. На первом уровне иерархии обычно указываются основные виды функциональных подсистем: сбыт, производство, логистика, сервис, финансы, персонал и т. д. На следующем уровне иерархии для каждой функциональной подсистемы показываются функциональные модули, например, подсистема «Логистика» включает в себя функциональные модули: пла-

Рис. 9.4. Конфигурация ИС на основе модельно-ориентированной технологии

нирование потребности в материалах, закупки, управление запасами, управление складами, проверка платежей и т. д.

Для функциональных модулей задаются **наборы бизнес-функций**, для каждой из которых в дальнейшем определяются **бизнес-процессы**. Например, для функционального модуля «Закупки» определяются бизнес-функции: оформление договоров, оформление заказов, выписка счетов и т. д.

Модель бизнес-процесса отражает последовательность выполнения работ (операций) для функций самого нижнего уровня модели бизнес-функций. Модель бизнес-процесса позволяет провести конфигурацию программных модулей информационной системы в соответствии с характерными особенностями конкретной проблемной области.

Модели объектов (данных) — это объекты-сущности в нотации языка UML, например заказы, счета, материалы, поставщики и т. д. С другой стороны, в отличие от обычных объектов-сущностей биз-

нес-объекты имеют стандартный интерфейс, написанный на языке описания интерфейсов IDL (Interface Definition Language), с помощью которого бизнес-объекты могут взаимодействовать друг с другом через объектную шину. На основе бизнес-объектов осуществляется интегрирование различных бизнес-процессов (приложений).

Модель организационной структуры предприятия представляет собой традиционную иерархическую структуру подчинения подразделений и персонала (организационных единиц). Назначение моделирования организационной структуры применительно к информационной системе заключается в распределении автоматизируемых функций по работникам подразделений и определении полномочий доступа к информационной системе.

Модели бизнес-правил — это специальные сведения, которые хранятся в репозитории и используются для контроля корректности построенной модели предприятия и процессов конфигурации и эксплуатации ИС.

Реализация модельно-ориентированного проектирования ИС выполняется по технологии, которая содержит **четыре основные стадии**:

- выбор типового проекта;
- разработка проектной модели предприятия;
- реализация проекта;
- ввод в эксплуатацию и поддержка функционирования.

На всех стадиях используется инструментарий моделирования предприятия.

Стадия выбора типового проекта начинается с анализа требований к ИС с последующим построением ***предварительной модели предприятия***, которая содержит требования к функциональности информационной системы (множеству автоматизируемых функций) и на основе которой будет осуществляться выбор программного комплекса.

Функциональная модель может быть оптимизирована в рамках проведения предварительного реинжиниринга бизнес-процессов. Но возможен вариант использования существующей организацией бизнес-процессов. В этом случае реально существующая модель предприятия будет адаптироваться на этапе эксплуатации информационной системы с целью оптимизации функционирования организационно-экономической системы. Полагают, что второй подход обеспечивает более быстрое внедрение корпоративной ИС, сокращение капитальных затрат и повышение эффективности эксплуатации ИС.

На стадии выбора типового проекта также строятся модели **бизнес-функций, бизнес-процессов, бизнес-объектов и организационной структуры.**

На основании построенных моделей руководство предприятия принимает решение ***о выборе типовой информационной системы***, модель предприятия которой в наибольшей степени соответствует целям автоматизации.

После закупки типовой информационной системы, формирования проектной группы внедрения, выделения всех необходимых ресурсов и формирования календарного плана-графика работ формируются ***технико-экономическое обоснование*** и ***техническое задание*** на внедрение типовой ИС.

На стадии «Разработка проектной модели предприятия» производится привязка модели предприятия к функциональности типовой ИС, на основе которой в последующем автоматически выполняется конфигурация информационной системы. На этой стадии выполняются следующие работы:

- инсталляция программного продукта, реализующего типовую ИС;
- проведение обучения проектной команды;
- привязка модели предприятия к компонентам типовой информационной системы;
- определение требований к доработке программного обеспечения;
- проектирование внешних интерфейсов системы.

Стадия «Реализация проекта ИС» сводится к конфигурированию ИС и генерации интерфейсов пользователей, а также к определению ***структуры базы данных***. Настройка программного комплекса типовой ИС и генерация интерфейса пользователей осуществляются автоматически на основе бизнес-правил и проектной модели предприятия. В исключительных случаях требуется доработка или создание новых программных модулей, которые производятся с помощью инструментальных средств программного комплекса.

На этой стадии выполняются следующие работы:

- конфигурирование программных модулей;
- настройка базы данных;
- генерация пользовательских интерфейсов;
- доработка программных модулей или разработка новых программных модулей и интерфейсов;
- комплексное тестирование всех компонентов созданной ИС.

При этом следует особо отметить, что доработка программных модулей или разработка новых программных модулей и интерфейсов

осуществляется на основе определенных ранее спецификаций на доработку программных модулей и интерфейсов с использованием языковых средств типовой системы.

Стадия «Ввод в эксплуатацию» осуществляется поэтапно в соответствии с определенным планом. Перед началом эксплуатации должны быть выполнены следующие работы:

- создание документации конечных пользователей и их обучение;
- установка программно-технической среды эксплуатации ИС;
- наполнение информацией новых баз данных или подключение и конвертация существующих баз данных.

9.3.2. Типовое внедрение готовых программных продуктов

Типовое внедрение систем, когда готовые решения (модели и программы) адаптируются к условиям конкретного пользователя и его задачам только ***за счет настройки штатных опций*** без изменения программного обеспечения, особенно распространено для малого бизнеса и рекомендуется для предприятий, ***функциональность которых близка к стандартным проектам***. Штатными средствами выполняется настройка под конкретный объект автоматизации прикладных программ, базы данных, пользовательского интерфейса, технической и эксплуатационной документации.

Подобными средствами обладают системы, предлагаемые фирмами «Ист Консепт», «Libra», «Информационные системы в медицине», «Hotel and Restaurant Systems» и другие. В этих системах предусмотрен максимальный уровень локализации и адаптации функциональных возможностей требованиям заказчика, без необходимости изменения основного программного кода разработчиками системы.

Достаточно подробно разработана технология типового внедрения фирмой «Libra» информационной системы для гостиницы «epitome POS». Если результатом выбора по технологии, описанной в предыдущей главе, оказалась эта система, то ее внедрение происходит следующим образом [www.Libra-Russia.com].

Для успешного осуществления проекта гостиница должна назначить менеджера проекта, который будет отвечать за координацию проекта со стороны гостиницы и будет основным ответственным лицом, представляющим интересы гостиницы в ходе реализации проекта.

Менеджером проекта может быть один из высококвалифицированных менеджеров гостиницы, который владеет вопросами технологического процесса гостиницы, имеет опыт работы в службе приема и размещения, в состоянии хорошо изучить систему и действовать в роли «эксперта на месте». Он координирует все работы по внедрению системы: согласовывает даты проведения работ, сроки поставки оборудования и программного обеспечения, расписание занятий с персоналом гостиницы, а также координирует работу всех специалистов и консультантов, участвующих в проекте.

Первым шагом реализации проекта внедрения системы управления является **установка необходимого программного обеспечения** с участием системного администратора гостиницы с тем, чтобы он в дальнейшем мог самостоятельно сопровождать установленную систему.

Одновременно проводится **семинар по управлению** для определения политики, процедур и стандартов гостиницы, которые будут использованы в настройках системы. Во время семинара обсуждаются вопросы технологии работы гостиницы, структура и взаимодействие отделов и служб, вопросы обслуживания клиентов, маркетинговой политики и финансового учета. Полезно обсудить также состав входной, выходной и справочной информации, используемой при управлении гостиницей, определиться с формой представления входной и выходной информации.

При подготовке к проведению данного семинара руководство гостиницы заполняет специальный вопросник по политике и процедурам (рис. 9.5), который позволит заранее подготовиться к обсуждению вопросов. Конечная цель семинара по управлению — определить цели и задачи, стоящие при внедрении системы, и согласовать основные параметры конфигурации.

На этапе **конфигурации системы** производится настройка параметров системы в соответствии с принципами, стандартами и правилами работы, согласованными в ходе семинара по управлению. В процессе конфигурации системы участвуют представители гостиницы, которые в дальнейшем смогут самостоятельно вносить необходимые изменения.

На начальном этапе реализации проекта с руководством гостиницы согласуется расписание **обучения персонала**, состав участников и круг вопросов, которые будут рассмотрены на обучающих занятиях. В ходе обучения работники гостиницы знакомятся с функциями системы, технологией эксплуатации, участвуют в практических занятиях по использованию системы и выполнению технологических процедур.

epitome Enterprise Solutions

ТАБЛИЦА ДЛЯ РАСЧЕТА КОНФИГУРАЦИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Название гостиницы: Гостиница

Количество номеров:

Система управления гостиницей epitome PMS

	Требуется ли модель (да/нет)	Кол-во номеров в гостинице
Базовый модуль		
История гостей и фирм		
Счета к получению		
Управление туристическими агентствами		
Групповые продажи		
Групповые контракты (оптовые продажи)		
Модуль управления мероприятиями		
Модуль регистрации и учета паспортов в LibraPass		
Подарочные сертификаты		

Система управления ресторанами epitome POS

	Требуется ли модель (да/нет)	Кол-во точек продаж
Система управления ресторанами		
Клиентский модуль		

Система управления производством (складская система) Libra F&B

	Требуется ли модель (да/нет)	Физическое кол-во складов
Система управления производством Libra F&B		

Интерфейсы

	Наименование внешней системы	Требуется ли интерфейс
Система тарификации электронных звонков		
Система бухгалтерского учета - стандартная выгрузка		
Система электронных замков		
Система тарификации доступа в Интернет		
Система точек продаж (ресторанная система)		
Система платного/интерактивного телевидения		
Система авторизации кредитных карт		
Система гостевого сервиса телефонной станции		
Система бронирования Академсервис		
Иные системы (укажите)		

Рис. 9.5. Специальный вопросник по политике и процедурам

После завершения обучения специалисты Libra Hospitality совместно с представителями гостиницы осуществляют процесс **ввода системы в эксплуатацию**. Эта работа проводится по четкому графику с соблюдением необходимых процедур, обеспечивающих плавный переход на использование новой системы и корректный перенос начальных данных. Одновременно производится настройка интерфейсов с внешними системами, функционирующими в гостинице: элек-

тронных замков, телефонного сервиса, бухгалтерского учета, управления ресторанами и барами и другие. Эти работы должны проводиться совместно с представителями компаний — поставщиков систем, с которыми производится стыковка.

В течение первых дней после ввода системы в эксплуатацию специалисты поставщика в круглосуточном режиме находятся в гостинице для оказания «живой» поддержки. Это позволяет персоналу гостиницы с уверенностью начать работу на системе и на практике отработать все технологические процедуры.

К началу реализации проекта необходимо решить вопросы установки сети, подготовки необходимого технического оборудования, а также выделения удобного помещения для проведения обучения.

Особое внимание при внедрении систем следует акцентировать на качестве работы **интерфейсных программ** (шлюзов), этих «перешейков» общей автоматизации предприятия. В подавляющем большинстве случаев гостиница является основным объектом комплекса, а рестораны, бары, центры развлечений и т. п. являются как бы сопутствующими. Такое же соотношение существует и в системах управления этими объектами.

ИС гостиницы обычно первой внедряется и является базовой, под нее уже подбираются сопутствующие специализированные гостиничные системы, которые достаточно функционально обособленны. Они разработаны для разных целей, имеют разную специфику, эксплуатируются разным персоналом в разных службах и требуют различных знаний, например системы управления мероприятиями, мини-баров, электронных ключей, финансово-бухгалтерского учета. Следовательно, структура базы данных, дизайн экранных форм и подход к отчетности в каждой системе могут быть специфическими [35].

Интерфейс как раз и предназначается для передачи из одной партнерской системы в другую всего нескольких полей (от 1 до 20), являющихся ключевыми для данного взаимодействия. Например, интерфейс обеспечивает передачу данных из системы управления рестораном в ИС гостиницы, необходимых для включения стоимости обеда в общий счет клиента. Чем лучше продуманы процедуры работы взаимосвязанных подразделений гостиницы и отлажены информационные потоки, тем изящнее и надежнее интерфейс, т. е. тем меньше информационных полей он передает. Тем самым минимизируется дублирование информации в партнерских системах гостиницы.

Большинство гостиничных интерфейсов работает в реальном режиме времени (on-line), некоторые функционируют в режиме перио-

дического экспорта-импорта информации. Согласование режима передачи, формата передаваемых между системами данных являются ключевыми вопросами при внедрении систем. В качестве примера можно привести фрагмент требований к интерфейсам ИС отеля из технического задания фирмы Libra [www.libra-russia.com].

Система должна обладать интерфейсами с российскими бухгалтерскими системами и обеспечивать ведение учетной политики предприятия с учетом реализации по оплате, по выписке гостя и по факту оказания услуг.

Пользователи системы на любом рабочем месте должны иметь доступ к функциям системы в соответствии со своими правами без ограничений.

При выборе и внедрении ИС следует учитывать размер автоматизируемого объекта: гостиницы, санатория, ресторана и т. п. Обычно полагают, что размер объекта мало влияет на требования к функциональным возможностям системы автоматизации. В большей степени здесь играют роль уровень и комплекс предоставляемых услуг.

Так, для небольшого бутик-отеля бесспорно важным будет наличие в ИС следующих функций: программа частого гостя, on-line бронирование, клубные программы, интеграция с системами платного телевидения, телефонии, доступ в Интернет из номера и другие возможности, которые предлагаются полнофункциональными системами управления. С другой стороны, для небольшой гостиницы экономического класса будет вполне достаточно функций управления бронированием, поселением и расчетами.

Вместе с тем размер объекта предъявляет определенные дополнительные требования к информационным системам. Например, в небольших отелях, как правило, все функции управления сосредоточены в руках генерального управляющего. Персонал стойки портье ограничивается одним человеком, который осуществляет и бронирование, и поселение с выпиской, и назначение номеров на уборку единственной горничной. Небольшие гостиницы не имеют собственных инженерных и компьютерных служб.

Поэтому для эффективной автоматизации объектов небольшого размера необходимо обращать внимание на следующие основные требования к системам управления:

- возможность доступа ко всем функциональным блокам системы с одного рабочего места и с максимальным удобством для пользователя;

• простота и интуитивность, минимальные требования к обучению пользователей, схожесть технологии работы во всех модулях системы;

• работа в среде ОС Windows и совместимость с офисными программами;

• минимальные требования по администрированию, отсутствие необходимости технического сопровождения собственными силами предприятия;

• низкая стоимость системы при наличии всех функциональных возможностей «больших» систем.

Данные требования определяют особенности предлагаемых поставщиками решений для автоматизации малых объектов. Так, стоимость системы управления гостиницей «epitome PMS», системы управления санаторием «Здравница» и других прямо пропорциональна размерам гостиницы или санатория, при этом функциональные возможности систем сохраняются в полном объеме.

В заключение следует отметить, что, несмотря на значительные финансовые затраты, внедрение ИС дает очевидные плюсы в изменении деятельности предприятия: повышение эффективности и оперативности принятия решений, более удачное позиционирование организации, направленность рекламы на целевые группы клиентов и, самое главное, повышение степени управляемости предприятия. Происходят также положительные информационные изменения в принятии решений: улучшение качества информации, уменьшение количества бумажной работы и непроизводительных затрат, рост правдоподобия результатов анализа, увеличение количественной информации по сравнению с качественной.

Однако прибыль, приносимую в результате внедрения информационной системы, очень трудно оценить количественно, и главные причины этого в следующем:

• выгоды реализуются в течение продолжительного интервала времени;

• природа выгод неосязаема;

• стратегические и конкурентные выгоды трудны для количественного выражения;

• результаты от введения информационных технологий непрямые и поэтому неразличимы от результатов других введенных факторов.

Некоторые авторы предлагают разделить все выгоды от внедрения ИС на прямые и косвенные. Прямые выгоды — это прямые результаты внедрения, которые легко поддаются оценке. Они обычно

связаны с уменьшением издержек, например, уменьшение работы по вводу данных вследствие электронной системы заказов, уменьшение количества бумажной работы, сокращение времени поиска информации и формирования выходных документов и т. д.

Косвенные результаты внедрения информационных технологий, в конечном счете, включают неосязаемые, непрямые и стратегические выгоды. Именно с этими тремя косвенными выгодами главным образом и связана проблема оценки пользы информационных систем.

Контрольные вопросы

1. Принципы выбора ИС.
2. Этапы индивидуального внедрения ИС.
3. Содержание пакетированного решения при внедрении ИС.
4. Содержание модельно-ориентированного проектирования.
5. Стадии реализации модельно-ориентированного проектирования.
6. Определение экономического эффекта от внедрения ИС.
7. Принципы типового внедрения за счет настройки штатных опций ИС.
8. Назначение интерфейсных программ (шлюзов) для партнерских ИС.

Заключение

В учебном пособии рассмотрены вопросы проектирования, разработки и внедрения конкретного класса информационных систем, обеспечивающих реализацию CALS-технологий на различных этапах жизненного цикла изделий: производство, поставка и эксплуатация. Наибольшее внимание уделено экономическим информационным системам, обеспечивающим управление ресурсами предприятия: определена их функциональность с позиций стандартов управления, рассмотрены особенности технологий проектирования для этого класса систем, применяемые программные средства для реализации этих технологий, в приложении приведен пример проектирования ИС для каскадной модели жизненного цикла системы.

Затем на уровне назначения, состава функциональности и особенностей проектирования рассмотрены системы управления цепочками поставок, электронной коммерции и управления взаимоотношениями с клиентами, которые расширяют функциональность ERP-системы до систем класса CSRP.

Выбор этого набора систем удобен также с методической точки зрения, поскольку они все реализуются в соответствии с архитектурой «клиент—сервер» и имеют почти одинаковые технологии проектирования, разработки и сопровождения, что существенно улучшает восприятие материала студентами и повышает эффективность обучения.

Особо рассмотрены системы поддержки принятия решения на примере OLAP-систем как класса систем, реализованных на основе многомерных баз данных для оперативного получения аналитической информации. Достаточно подробно показано происхождение преимуществ OLAP-систем по сравнению с OLTP-системами, построенными на основе реляционных баз данных, которые, в свою очередь, являются поставщиками информации для многомерных кубов.

В учебном пособии выполнена классификация существующих информационных систем с позиций их сложности и масштабности, даются рекомендации по применению современных программных инструментариев для их проектирования и разработки.

В работе подробно рассмотрены популярные современные программные инструментарии All Fusion Modeling Suite и Oracle Designer

10g, а также реализованные в них методологии Swim Lane Diagram, IDEF0, IDEF3, DFD, IDEF1X и Process Modeler. Автором предложена собственная методика применения этих методологий на всех этапах модели жизненного цикла информационных систем, что позволяет студентам в процессе выполнения курсовых и дипломных проектов разрабатывать реальные информационные системы.

Большое внимание уделено вопросу применения существующих стандартов в сфере информационных технологий. Выполнен сравнительный анализ российских и международных стандартов, дано представление о разработке собственного стандарта для каждой разрабатываемой системы в виде профиля стандартов. Приведен пример разработанного автором профиля с набором проектных документов для всех этапов жизненного цикла системы, начиная от процесса обследования предметной области и заканчивая приемкой в промышленную эксплуатацию и сопровождением системы.

Детально рассмотрен наиболее злободневный в настоящее время вопрос о выборе и внедрении готовых (покупных) ИС. С позиций индивидуального и типового внедрения предложены современные подходы на базе пакетированных решений и модельно-ориентированного проектирования.

В результате студент в пределах одной книги имеет возможность понять место проектируемой информационной системы среди существующих систем, выбрать модель ее жизненного цикла, технологию проектирования, программный инструментарий для ее реализации, а также определить необходимый состав стандартов и сформировать весь пакет проектной и эксплуатационной документации.

Кроме этого, читатель получает необходимые сведения о методике выбора покупных ИС и современных методах их внедрения с наименьшими затратами и максимальным эффектом для заказчика.

Литература

1. *Альбитов А.* Классификация гостеприимства // PC WEEK/RE. 2001. № 45.

2. *Арутюнян М.* Информационные системы управления цепочками поставок / Электронный ресурс. Режим доступа: http://www.integprog.ru.

3. *Архипенков С.* От переработки данных — к анализу // Банковские технологии. 1998. № 3.

4. *Ахтырченко К.В., Леонтьев В.В.* Распределенные объектные технологии в информационных системах // СУБД. 1997. № 5—6.

5. *Белоусова Ю.Г.* Система организации материальных и информационных потоков на предприятиях в рамках цепочки поставок / Электронный ресурс. Режим доступа: http://www.ekportal.ru/page-id-393.html

6. *Бирюков В., Дрожжинов В.* Введение в CRM // PC WEEK/RE. 2001. № 25.

7. *Бобровский С.* Программная инженерия развивается экстремальными методами // PC WEEK/RE. 2001. № 35.

8. *Бронникова Т., Романов А.* ERP: «Разделить нельзя интегрировать» или «Как внедрять быстро, качественно и недорого?» // CIO. 2006. № 10. Окт.

9. *Ветитнев А.М., Коваленко Вл.В., Коваленко В.В.* Информационные технологии в социально-культурном сервисе и туризме. Оргтехника. М.: ФОРУМ, 2010.

10. *Горчинская О.Е.* Designer/2000 — новое поколение CASE-продуктов фирмы Oracle // СУБД. 1995. № 3.

11. *Граничин О.Н., Кияев В.И.* Информационные технологии в управлении. ИНТУИТ.ру, 2008.

12. *Грекул В.И., Денищенко Г.Н., Коровкина Н.Л.* Проектирование информационных систем // ИНТУИТ.ру, 2008.

13. *Гудков Д.* Информационная поддержка изделия на всех этапах жизненного цикла / Электронный ресурс. Режим доступа: http://www.espotec.ru/art_info.htm

14. *Гулько Д.* B2B: промышленность движется к Hi-Tech // PC WEEK/RE. 2001. № 10.

15. *Дайв М.* Проектирование многомерной базы данных для OLAP // Oracle Magazine. 2000. Vol. 10. № 2.

16. *Елманова Н., Федоров А.* Введение в OLAP: часть 1. Основы OLAP // КомпьютерПресс. 2001. № 4.

17. *Емельянова Н.З., Партыка Т.Л., Попов И.И.* Проектирование информационных систем. М.: ФОРУМ, 2009.

18. *Зиндер Е.З.* Соотнесение и использование организации жизненных циклов систем // СУБД. 1997. № 3. С. 41—53.

19. Интеграция данных об изделии на основе CALS-технологий / Государственный межведомственный научно-исследовательский и образовательный центр CALS-технологий. М., 2002.

20. *Колетски П., Дорси П.* Oracle Designer. Настольная книга пользователя. М.: Лори, 1999.

21. *Королев В.А.* Основа повышения эффективности бизнеса // Методы менеджмента качества. 2004. № 10. С. 24—28.

22. *Липаев В.В.* Стандарты, регламентирующие жизненный цикл сложных комплексов программ информационных систем // Инженерное образование. 2005. № 8.

23. *Липаев В.В.* Формирование и применение профилей открытых систем // Открытые системы. 1997. № 5.

24. *Липаев В.В.* Технологические процессы и стандарты обеспечения функциональной безопасности в жизненном цикле программных средств // Информационный бюллетень «Jet Info 03(130)/2004».

25. *Маклаков В.С.* Создание информационных систем с All Fusion Modeling Suite. М.: ДИАЛОГ-МИФИ, 2003.

26. *Маклаков С.В.* BPwin и ERwin. CASE-средства разработки ИС. М.: ДИАЛОГ-МИФИ, 2001.

27. *Маклаков С.В.* Хранилища данных и их проектирование с помощью CA ERwin / Электронный ресурс. Режим доступа: http://www.interface.ru/

28. *Милаев В.* и др. Автоматизация управления в условиях многономенклатурного мелкосерийного производства // PC WEEK/RE. 2001. № 10.

29. *Некрасов В.В.* Введение в OLAP на практическом примере // PC WEEK. 2001. № 6.

30. *Никулина Н.О., Хусаинова Л.Р.* Сравнительный анализ стандартов жизненного цикла информационных систем ГОСТ 34.601—90 и ISO/IEC 12207. Управление в сложных системах. Уфа, 2009.

31. *Пуле М.* Исследование схемы «звезда» // КомпьютерПресс. 2001. № 1.

32. *Репин В.В., Елиферов В.Г.* Процессный подход к управлению. Моделирование бизнес-процессов. М.: Стандарты и качество, 2005.

33. *Сахаров А.А.* Принципы проектирования и использования многомерных баз данных // СУБД. 1996. № 3.

34. *Сидоров С.* «Подземелья» Oracle Express. / Электронный ресурс. Режим доступа: http://www.olap.ru/desc/oracle/

35. *Слепцова Н.* Автоматизация гостиниц. Системы, интерфейсы, персонал, Libra International // Гостиничное дело. 2007. № 12. С. 42—45.

36. Стандарт РД IDEF0—2000.

37. *Судов Е.В.* Интегрированная поддержка жизненного цикла машиностроительной продукции. М.: MBM, 2003.

38. *Урман С.* Oracle 8: Программирование на языке PL/SQL. М.: Лори, 1999.

39. *Чеботарев В.* Моделирование бизнеса: средства и методы // PC WEEK/RE. 2000. № 9.

40. *Шарашов К.В.* Автоматизация крупных предприятий // PC WEEK/RE. 1997. № 6.

41. *Шубанов С.* Как заставить CRM приносить прибыль? // Логинфо. 2005. № 3.

42. *Якумчук С.* MSF — философия создания IT-решений или голые амбиции лидера // IT-менеджер. 2004. № 4. С. 33—38.

43. Standard Intergration Definition for Information Modeling IDEF1X. FIPS, 1993.

Порядок выполнения работ при проектировании информационных систем по каскадной модели ЖЦ

Порядок выполнения работ для первых двух этапов рассмотрен в гл. 6. В данном приложении представлен процесс проектирования в соответствии с каскадной моделью на оставшихся пяти этапах: техническое задание, разработка технического проекта, рабочая документация, ввод в действие и сопровождение. Содержание всех документов, которые приводятся в данном приложении А, подробно изложено в разд. 5.5.3.

А.1. Формирование требований к ИС

На данном этапе, который подробно описан в разд. 6.2, проводится обследование предметной области, для которой будет разрабатываться ИС, с последующим обоснованием необходимости ее создания. Построение организационной диаграммы и модели «AS IS» как в виде IDEF0-модели, так и в виде диаграмм Swim Lane позволяет в удобной форме представить существующие в автоматизируемом предприятии бизнес-процессы.

Результирующим документом этапа формирования требований является технико-экономическое обоснование (ТЭО) (ГОСТ 24.202—80, РД 50-34.698—90).

А.2. Разработка концепций ИС

На этапе разработки концепций ИС, который подробно описан в разделе 6.3, создаются несколько моделей «TO BE», которые отражают необходимые изменения бизнес-процессов в разрабатываемой ИС. Результатом этого этапа является выбор оптимальной модели «TO BE», определение функциональности ИС на уровне набора программных модулей, построение базы данных логического уровня и дерева меню.

А.3. Техническое задание

Полученные на предыдущем этапе функциональная модель, база данных логического уровня и пользовательский интерфейс разрабатываемой

системы позволяют разработчику и заказчику определить сложность и трудоемкость разработки ИС, ее функциональный состав, структуру информации, условия эксплуатации. Этой информации достаточно для разработки технического задания (ТЗ) на ИС (ГОСТ 34.602—89). ТЗ является основным юридическим документом во взаимоотношениях между разработчиком и заказчиком и определяет цели создания ИС, требования к системе и основные исходные данные, необходимые для ее разработки, а также план-график создания ИС. Обычно ТЗ разрабатывает разработчик, а выдает его разработчику заказчик.

ТЗ содержит **титульный лист**, на котором помещают подписи заказчика, разработчика и согласующих организаций, которые скрепляют гербовой печатью. Подписи разработчиков ТЗ на ИС и должностных лиц, участвующих в согласовании и рассмотрении проекта ТЗ на ИС, помещают на последнем листе.

При разработке технического задания необходимо решить следующие задачи:

- установить общую цель создания ИС, определить состав подсистем и функциональных задач;
- разработать и обосновать требования, предъявляемые к подсистемам;
- разработать и обосновать требования, предъявляемые к информационной базе, математическому и программному обеспечению, комплексу технических средств (включая средства связи и передачи данных);
- установить общие требования к проектируемой системе;
- определить перечень задач создания системы и исполнителей;
- определить этапы создания системы и сроки их выполнения;
- провести предварительный расчет затрат на создание системы и определить уровень экономической эффективности ее внедрения.

ТЗ на ИС содержит следующие разделы, которые могут быть разделены на подразделы:

1) общие сведения;
2) назначение и цели создания (развития) системы;
3) характеристика объектов автоматизации;
4) требования к системе;
5) состав и содержание работ по созданию системы;
6) порядок контроля и приемки системы;
7) требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие;
8) требования к документированию;
9) источники разработки.

Эти разделы подробно описаны в ГОСТ 34.602—89, но особенно следует обратить внимание на следующие моменты.

В разделе **«Требования к системе»** в подразделе **требования к функциям (задачам)**, выполняемым системой, следует перечислить все блоки нижнего уровня иерархии «дерева узлов» модели «ТО ВЕ».

В подразделе **«Требования к системе в целом»** следует описать только те требования, которые имеют смысл для разрабатываемой системы. В обязательном порядке составляются требования **к защите информации от несанкционированного доступа**, требования **по сохранности информации** при авариях, отказах технических средств (в том числе — потери питания) и т. п., при которых должна быть обеспечена сохранность информации в системе.

В подразделе **«Требования к видам обеспечения»** в зависимости от вида системы приводят требования к математическому, информационному, лингвистическому, программному, техническому, метрологическому, организационному, методическому и другим видам обеспечения системы.

Раздел **«Состав и содержание работ по созданию (развитию) системы»** должен содержать перечень стадий и этапов работ по созданию системы (например, в соответствии с ГОСТ 34.601—90), сроки их выполнения. В данном разделе также приводят:

1) перечень документов по ГОСТ 34.201—89, предъявляемых по окончании соответствующих стадий и этапов работ;

2) вид и порядок проведения экспертизы технической документации (стадия, этап, объем проверяемой документации, организация-эксперт).

В разделе **«Порядок контроля и приемки системы»** указывают виды (приемочные испытания, опытная эксплуатация и приемочные испытания), состав, объем и методы испытаний системы на основе стандарта ГОСТ 34.603—92. Рекомендуется указать, кто (разработчик или заказчик) разрабатывает **программу и методики испытаний**.

Предварительные испытания проводят для определения работоспособности системы и решения вопроса о возможности ее приемки в опытную эксплуатацию.

Опытную эксплуатацию системы проводят обычно в течение одного месяца на ограниченном количестве рабочих мест, но с полной функциональностью с целью определения фактических значений количественных и качественных характеристик системы и готовности персонала к работе в условиях ее функционирования, а также определения фактической эффективности и корректировки, при необходимости, документации.

Приемочные испытания проводят для определения соответствия системы техническому заданию, оценки качества опытной эксплуатации и решения вопроса о возможности приемки системы в постоянную эксплуатацию.

Следует иметь в виду, что опытная эксплуатация и приемочные испытания проводятся только представителями заказчика.

В разделе **«Требования к документированию»** приводят согласованный разработчиком и заказчиком системы перечень подлежащих разработке комплектов и видов документов, соответствующих требованиям ГОСТ 34.201—89 и ЕСПД.

В последнее время рекомендуется разрабатывать **профиль стандартов** на разрабатываемую систему и утверждать его в составе ТЗ (раздел 5.5.2).

После утверждения ТЗ разработчик и заказчик заключают **контракт** на разработку ИС, с указанием этапов выполнения работ, сроков их выполнения и стоимости.

А.4. Технический проект

Этап технического проекта является самым ответственным и решающим в успешном завершении разработки ИС. Этап выполняется на основе стандарта РД 50-34.698—90.

Технический проект ИС содержит основные проектные решения по системе в целом, ее функциям и всем видам обеспечения ИС, которых должно быть достаточно для разработки программных кодов и рабочей документации. В стандарте ГОСТ 34.201—89 приведены **более 20 документов**, которые следует разработать на этом этапе (пояснительная записка к техническому проекту, ведомость технического проекта, перечень входных сигналов и данных, перечень выходных сигналов (документов), описание автоматизируемых функций, описание информационного обеспечения системы, описание организации информационной базы, описание комплекса технических средств и др.). Содержание этих документов приведено в стандарте РД 50-34.698—90.

Для упрощения оформления документации для этапа **технический проект** предлагаются так называемые **«Утвержденные спецификации требований и алгоритмы на функциональные группы программ, программные и информационные компоненты»** [2]. Эти спецификации требований являются основой для детального планирования процесса разработки программных средств и их компонентов. **Под спецификациями требований** понимается формальное описание свойств объектов будущего программного продукта: программных модулей, таблиц БД и элементов пользовательского интерфейса.

Поэтому в упрощенном варианте рекомендуется следующий комплект документов для этапа технического проекта:

- пояснительная записка к техническому проекту;
- спецификации требований и алгоритмы на функциональные группы программ, программные и информационные компоненты;

- описание организации информационной базы;
- структурная схема комплекса технических средств.

Содержание **пояснительной записки к техническому проекту** подробно описано в стандарте РД 50-34.698—90 и не требует дополнительных объяснений. Она оформляется в виде отдельного документа с титульным листом, содержащим **утверждающие и согласующие подписи**.

Спецификации требований и алгоритмы на функциональные группы программ, программные и информационные компоненты содержат описание свойств основных компонент ИС: программных модулей, таблиц БД и пользовательского интерфейса.

Спецификации для программного модуля содержат назначение и характеристику каждого программного модуля (постановка задачи, общие требования к входным и выходным данным, описание алгоритма функционирования) и результаты выполнения модуля (выходной документ, экранная форма и т. п.). В качестве программных модулей описываются **все функциональные блоки нижнего уровня иерархии** модели «ТО ВЕ».

Спецификации для каждого программного модуля выглядят следующим образом:

- **заголовок модуля** — имя модуля, имена и типы формальных параметров, краткое описание назначения модуля и выполняемые им функции;
- **паспорт модуля** содержит: описание всех входных данных (используемых полей таблиц БД), вызываемых модулем; функциональную схему модуля (блок-схема алгоритма или ссылка на реализуемую выходную или экранную формы).

Спецификации для таблиц БД содержат описание каждого поля таблиц (тип и размер поля, его смысловое значение).

Спецификации для пользовательского интерфейса содержат описание всех элементов пользовательского интерфейса (имя элемента, назначение, какой программный модуль запускается, какой результат получают).

Документ **«Описание организации информационной базы»** содержит разделы:

- входная информация;
- выходная информация;
- логическая структура базы данных;
- физическая структура базы данных.

Раздел **«Входная информация»** должен содержать перечень и описание входных сообщений: наименование, форму представления, сроки и частоту поступления, а также источник информации (документ, видеокадр, устройство, информационная база на машинных носителях и т. д.).

Раздел **«Выходная информация»** содержит перечень и описание выходных сообщений: наименование, форму представления сообщения (до-

кумент, видеокадр, сигнал управления), периодичность выдачи, сроки выдачи и допустимое время задержки решения; получателей и назначение выходной информации.

В разделе **«Логическая структура»** приводят описание состава данных, их форматов и взаимосвязей между данными (ER-диаграмма).

В разделе **«Физическая структура»** приводят описание избранного варианта расположения данных в среде конкретного СУБД.

Документ **Структурная схема комплекса технических средств** содержит схему размещения технических средств (рис. А1) с краткой аннотацией. Например, в состав комплекса технических средств входят следующие технические средства:

- серверы БД;
- серверы приложений;
- сервер системы формирования отчетности;
- веб-сервер;
- ПК пользователей;
- ПК администраторов.

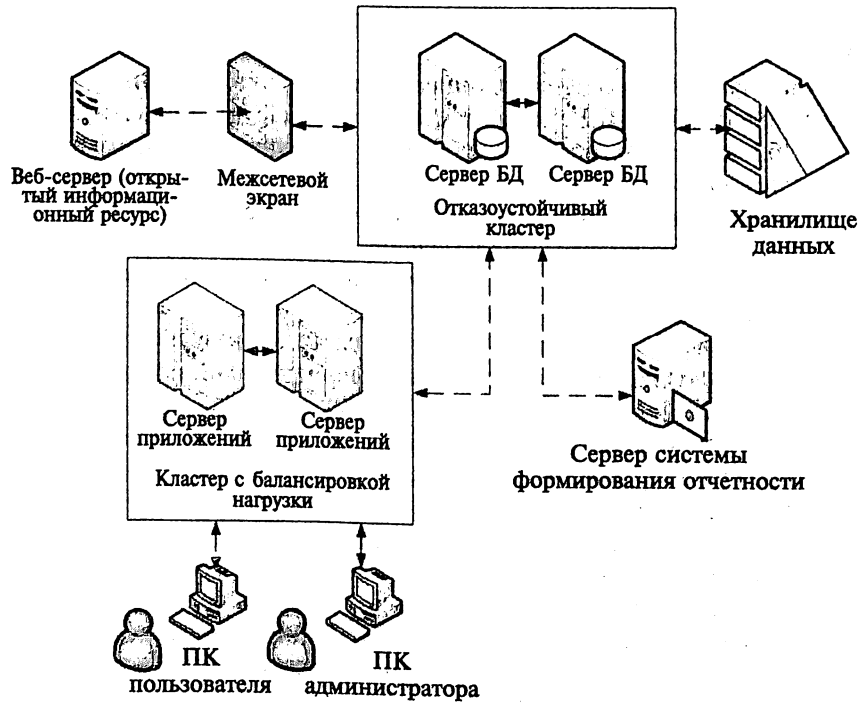


Рис. А1. Структурная схема комплекса технических средств

Серверы БД объединены в отказоустойчивый кластер. Связь между серверами БД и хранилищем данных осуществляется по оптическому каналу. Серверы приложений образуют кластер с балансировкой нагрузки. Серверы БД, серверы приложений и сервер системы формирования отчетности объединены одной локальной сетью с пропускной способностью 100 Мбит [12].

На этапе технического проекта завершается проектирование и начинается разработка ИС с помощью программных инструментариев, предназначенных для этих целей, например с помощью средств быстрой разработки Delphi, Oracle Developer 10g и др.

На этапе «Рабочая документация» каскадной модели ЖЦ разрабатываются программы на основе спецификаций требований для программных модулей и пользовательского интерфейса. В случае приобретения готового программного обеспечения производится его адаптация и интеграция с системой. На всех стадиях создания программных средств осуществляется их тестирование.

На этом же этапе производится разработка программной документации, к которой относятся документы, содержащие сведения, необходимые для разработки, изготовления, сопровождения и эксплуатации программ (руководства операторов, программистов и администраторов, различного рода инструкции). Регламентацию деятельности в процессе подготовки документации определяют стандарты Единой системы программной документации (ЕСПД). В частности, перечень программных документов приводится в стандарте ГОСТ 19.101—77.

Умение создавать программную документацию определяет профессиональный уровень программиста и избавляет разработчиков от необоснованных претензий и большого количества вопросов пользователей.

На этапе «Ввод в действие» выделяются три группы работ: подготовка персонала, пусконаладочные работы и испытания.

Подготовка персонала заключается в обучении различного рода пользователей и обслуживающего персонала по специально созданным учебным материалам обычно с участием разработчиков и в проверке их способности обеспечить функционирование ИС.

Пусконаладочные работы включают автономную наладку технических и программных средств, загрузку информации в базу данных и комплексную наладку всех средств системы.

Испытания системы в полном объеме содержат три стадии, подробно описанные в ГОСТ 34.603—92: ***предварительные испытания, опытная эксплуатация и приемочные испытания.*** Общим для них является создание комиссий для проведения испытаний, программы, методик и протоколов испытаний, а также акта об их результатах.

Предварительные испытания следует выполнять после проведения разработчиком отладки и тестирования поставляемых программных и аппаратных средств системы и представления им соответствующих документов об их готовности к испытаниям, а также после ознакомления персонала ИС с эксплуатационной документацией.

Во время предварительных испытаний осуществляют проверку ИС на работоспособность и соответствие техническому заданию в соответствии с программой и методиками и предварительными испытаниями. По результатам предварительных испытаний производят устранение неисправностей в системе и внесение изменений в документацию техническую и эксплуатационную. В акте о результатах проведенных предварительных испытаниях указываются действия по устранению обнаруженных недостатков, после выполнения которых, возможен переход к проведению опытной эксплуатации.

Опытная эксплуатация заключается в работе с реальными данными без участия разработчиков по программе и методикам испытаний с регистрацией всех ошибок, сбоев и нештатных ситуаций. По итогам опытной эксплуатации производится доработка программного обеспечения и дополнительная наладка технических средств. Работы завершаются оформлением акта о результатах опытной эксплуатации.

Проведение **приемочных испытаний** заключается в комплексной проверке реально функционирующей в полном объеме ИС на соответствие техническому заданию. В случае положительных результатов испытаний оформляется акт о приемке ИС в постоянную эксплуатацию. Этот акт является подтверждением того факта, что разработчик полностью реализовал все положения ТЗ.

Для каждого вида испытания издается **приказ руководителя** со стороны заказчика о составе комиссии и сроках проведения испытаний, разрабатываются программа и методики испытаний. В процессе проведения испытаний оформляются **журналы испытаний**, на их основе составляется **протокол испытаний** (по каждой методике испытаний), который подписывается членами комиссии по проведению испытаний.

На основании протоколов испытаний составляется **Акт результатов испытаний**, в котором указываются обнаруженные в результате испытаний недостатки и ошибки, а также условия проведения последующих испытаний (опытной эксплуатации, приемочных испытаний). После подписания **Акта о завершении приемосдаточных испытаний и результатах выполнения контракта на разработку ИС** все обязательства разработчика перед заказчиком считаются выполненными.

На этапе сопровождения ИС анализируется функционирование ИС, выявляются отклонения эксплуатационных характеристик, устраняются

причины этих отклонений, своевременно выявляются ошибки и нештатные ситуации.

Кроме этого, этап включает процессы и операции, связанные с регистрацией, диагностикой и локализацией ошибок, внесением изменений и тестированием, проведением доработок, тиражированием и распространением новых версий ИС в места ее эксплуатации, что фактически является повторной итерацией почти всех этапов модели ЖЦ. На каждый вид работы оформляется определенная документация: отчет пользователей о выявленных дефектах и предложениях по корректировке версий ИС; журнал выявленных дефектов и предложений по совершенствованию версий ИС; журнал подготовленных и утвержденных корректировок и др.

Оглавление

По вопросам приобретения книг обращайтесь:
Отдел продаж «ИНФРА-М» (оптовая продажа):
127282, Москва, ул. Полярная, д. 31В, стр.1
Тел. (495) 280-15-96; факс (495) 280-36-29
E-mail: books@infra-m.ru

•

Отдел «Книга—почтой»:
тел. (495) 280-15-96 (доб. 246)

ФЗ № 436-ФЗ Издание не подлежит маркировке в соответствии с п. 1 ч. 4 ст. 11

Учебное издание

Коваленко Владимир Васильевич

ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННЫХ СИСТЕМ

УЧЕБНОЕ ПОСОБИЕ

ООО «Издательство Форум»
127282, Москва, ул. Полярная, д. 31В, стр. 1
E-mail: forum-book@yandex.ru
Тел.: (495) 280-15-96

ООО «Научно-издательский центр ИНФРА-М»
127282, Москва, ул. Полярная, д. 31В, стр. 1
Тел.: (495) 280-15-96, 280-33-86. Факс: (495) 280-36-29
E-mail: books@infra-m.ru http://www.infra-m.ru

Подписано в печать 11.12.2018.
Формат 60×90/16. Бумага офсетная. Гарнитура Newton.
Печать цифровая. Усл. печ. л. 20,0.
ППТ10. Заказ № 12422

ТК 162700-980117-170614

Отпечатано в типографии ООО «Научно-издательский центр ИНФРА-М»
127282, Москва, ул. Полярная, д. 31В, стр. 1
Тел.: (495) 280-15-96, 280-33-86. Факс: (495) 280-36-29