Головчинер М.Н.

ПРЕКТИРОВАНИЕ ИНФОРМАЦИОННЫХ СИСТЕМ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ПОДГОТОВКЕ К ГОСУДАРСТВЕННОМУ ЭКЗАМЕНУ

Томск 2009

1. Понятие информационной системы. Классификация ИС

Разнообразие задач, решаемых с помощью ИС, привело к появлению множества разнотипных систем, отличающихся принципами построения и заложенными в них правилами обработки информации.

В качестве классификационных признаков ИС выделим:

- параметры объекта управления (сфера деятельности, масштаб, состав функций управления);
- организационная структура ИС;
- степень интеграции ИС;
- специализация ИС;
- тип хранимых данных;
- степень автоматизации информационных процессов;
- характер обработки;
- характер использования выходной информации;
- уровень управления;
- информационно-технологическая архитектура;
- программно-аппаратная реализация; и др.

1. Сфера деятельности объекта управления:
   - промышленное предприятие;
   - сфера обращения (торговля, банки и кредитные организации);
   - образование, социальная сфера; и др.
2. Границы ИС (масштаб):
   - ИС предприятия (организации);
   - ИС отрасли;
   - международная ИС.
3. Состав функций управления:
   - автоматизация технической подготовки производства;
   - маркетинг и стратегия развития предприятия;
   - технико-экономическое планирование;
   - финансы (бухгалтерский учет, финансовый анализ);
   - материально-техническое обеспечение;
   - оперативно-календарное управление производством;
   - управление сбытом готовой продукции;
   - управление персоналом; и др.
4. Организационная структура:
   - автоматизированное рабочее место (АРМ) управленческого персонала;
   - комплекс взаимосвязанных АРМ.
5. Степень интеграции:
   - локальная ИС (изолированное информационное пространство);
   - частично интегрированная ИС (общее информационное пространство);
   - полностью интегрированная корпоративная ИС.

     Интегрированные (корпоративные) ИС используются для автоматизации всех функций фирмы и охватывают весь цикл работ от планирования деятельности до сбыта продукции. Они включают в себя ряд модулей (подсистем), работающих в едином информационном пространстве и выполняющих функции поддержки соответствующих направлений деятельности.
6. Специализация ИС:
   - ИС менеджмента (IMS - Information Management System или организационно-экономического управления);
   - информационно-поисковые системы (IRS - Information Retrieval System);

- системы автоматизированного обучения (EIS - Education Information System); и др.

Наибольшее распространение получили ИС менеджмента, среди которых по *сфере применения* выделяют:

- АСУП - автоматизированная система управления ресурсами предприятий и организаций (*организационное управление*). Предназначены для автоматизации функций управленческого персонала как промышленных предприятий, так и непромышленных объектов (гостиниц, банков, магазинов и пр.). Основными функциями подобных систем являются: оперативный контроль и регулирование, оперативный учет и анализ, перспективное и оперативное планирование, бухгалтерский учет, управление сбытом, снабжением и другие экономические и организационные задачи;
- АСУТП - автоматизированная система *управления технологическими процессами* производства продукции. Служат для автоматизации функций производственного персонала по контролю и управлению производственными операциями. В таких системах обычно предусматривается наличие развитых средств измерения параметров технологических процессов (температуры, давления, химического состава и т.п.), процедур контроля допустимости значений параметров и регулирования технологических процессов;
- САПР - системы *автоматизированного проектирования* конструкций и технологий производства продукции. Предназначены для автоматизации функций инженеров-проектировщиков, конструкторов, архитекторов, дизайнеров при создании новой техники или технологии. Основными функциями подобных систем являются: инженерные расчеты, создание графической документации (чертежей, схем, планов), создание проектной документации, моделирование проектируемых объектов; и др.

Информационная система менеджмента в качестве компонентов включает в себя другие специализированные ИС, предназначенные для следующих целей:

- автоматизация делопроизводства (OAS - Office Automation System);
- поддержка принятия решений (DDS - Design Support System);
- формирование знаний системы управления (KBS - Knowledge Base System); и др.

В DDS нашли применение технологии:

- оперативного анализа и обработки данных, полученных из **хранилищ данных** (Data Warehouse);
- извлечения информации из данных (Data Mining);
- моделирования бизнес-процессов.

Для выработки стратегии развития предприятия (перспективные направления, планирование, инвестиционное проектирование и пр.) создаются специализированные системы поддержки принятия решений, так называемые **корпоративные стратегические системы (**ESS - Enterprise Strategic System). ESS используют методы статистического анализа и прогнозирования, моделирования данных и бизнес-процессов, имитационного моделирования.

В современных ИС менеджмента значительна роль ИС искусственного интеллекта (AIS -Artificial Intelligence System). Эти ИС:

- поддерживают естественно-языковый интерфейс для пользователей (специалистов по формализации знаний);
- предоставляют методы искусственного интеллекта для решения слабо структурированных и плохо формализованных задач.

Ядром AIS является **база знаний** (KB - Knowledge Base), которая используется для формирования новой информации путем логического вывода.

Для представления экономического объекта и его окружения, исследования его поведения и реакций на внешние события применяются:

- математическое моделирование;
- средства дедуктивных и правдоподобных выводов, полученных на основе неполной или неточной информации.

Примеры (типы) AIS:

- *Экспертные системы* - наиболее распространенный тип AIS; с помощью ЭС на основе реальных данных выдвигается и дается оценка некоторой гипотезы;
- ИС *полнотекстового поиска* (объединяются с реляционными СУБД, образуя новый класс постреляционных СУБД);
- ИС *аналитических вычислений* на основе методов исследования операций, математического моделирования, статистического анализа и прогнозирования; и др.

7. По типу хранимых данных ИС делятся на **фактографические** и **документальные**.
   - *Фактографические* системы предназначены для хранения и обработки структурированных данных в виде чисел и текстов. Над такими данными можно выполнять различные операции.
   - В *документальных* системах информация представлена в виде документов, состоящих из наименований, описаний, рефератов и текстов. Поиск по неструктурированным данным осуществляется с использованием семантических признаков. Отобранные документы предоставляются пользователю, а обработка данных в таких системах практически не производится.
8. По степени автоматизации информационных процессов в системе управления организацией ИС делятся на **ручные**, **автоматические** и **автоматизированные**.
   - *Ручные* ИС характеризуются отсутствием современных технических средств переработки информации и выполнением всех операций человеком.
   - В *автоматических* ИС все операции по переработке информации выполняются без участия человека.
   - *Автоматизированные* ИС предполагают участие в процессе обработки информации и человека, и технических средств, причем главная роль в выполнении рутинных операций обработки данных отводится компьютеру. Именно этот класс систем соответствует современному представлению понятия "информационная система".
9. По характеру обработки данных ИС делятся на информационно-поисковые и информационно-решающие.
   - *Информационно-поисковые* системы производят ввод, систематизацию, хранение, выдачу информации по запросу пользователя без сложных преобразований данных. (Например, ИС библиотечного обслуживания, резервирования и продажи билетов на транспорте, бронирования мест в гостиницах и пр.)
   - *Информационно-решающие* системы осуществляют, кроме того, операции переработки информации по определенному алгоритму.
10. По характеру использования выходной информации такие системы принято делить на управляющие и советующие.
    - *Результирующая* информация управляющих ИС непосредственно трансформируется в принимаемые человеком решения. Для этих систем характерны задачи расчетного характера и обработка больших объемов данных. (Например, ИС планирования производства или заказов, бухгалтерского учета.)
    - *Советующие* ИС вырабатывают информацию, которая принимается человеком к сведению и учитывается при формировании управленческих решений, а не инициирует конкретные действия. Эти системы имитируют интеллектуальные процессы обработки знаний, а не данных. (Например, экспертные системы.)
11. Существует классификация ИС в зависимости от уровня управления, на котором система используется.

- Информационная система *оперативного уровня* - поддерживает исполнителей, обрабатывая данные о сделках и событиях (счета, накладные, зарплата, кредиты, поток сырья и материалов). Информационная система оперативного уровня является связующим звеном между фирмой и внешней средой. Задачи, цели, источники информации и алгоритмы обработки на оперативном уровне заранее определены и в высокой степени структурированы.
- Информационные системы *специалистов* - поддерживают работу с данными и знаниями, повышают продуктивность и производительность работы инженеров и проектировщиков. Задача подобных информационных систем - интеграция новых сведений в организацию и помощь в обработке бумажных документов.
- Информационные системы уровня *менеджмента* - используются работниками среднего управленческого звена для мониторинга, контроля, принятия решений и администрирования. Основные функции этих информационных систем:
  - сравнение текущих показателей с прошлыми;
  - составление периодических отчетов за определенное время, а не выдача отчетов по текущим событиям, как на оперативном уровне;
  - обеспечение доступа к архивной информации и т.д.
- *Стратегическая* информационная система - компьютерная информационная система, обеспечивающая поддержку принятия решений по реализации стратегических перспективных целей развития организации.
  Информационные системы стратегического уровня помогают высшему звену управленцев решать неструктурированные задачи, осуществлять долгосрочное планирование. Основная задача - сравнение происходящих во внешнем окружении изменений с существующим потенциалом фирмы. Они призваны создать общую среду компьютерной телекоммуникационной поддержки решений в неожиданно возникающих ситуациях. Используя самые совершенные программы, эти системы способны в любой момент предоставить информацию из многих источников. Некоторые стратегические системы обладают ограниченными аналитическими возможностями.

12. Информационно-технологическая архитектура:
    - ИС централизованной архитектуры построения (один центр хранения и обработки данных);
    - ИС распределенной архитектуры (компьютерные сети, наличие множества центров обработки и хранения информации);
13. С точки зрения программно-аппаратной реализации можно выделить ряд типовых архитектур ИС:
    - *традиционные* архитектурные решения, основанные на использовании выделенных файл-серверов или серверов баз данных;
    - варианты архитектур корпоративных информационных систем, базирующиеся на технологии Internet (*Intranet-приложения*);
    - архитектура информационной системы, основывающаяся на концепции *"хранилища данных"* (DataWarehouse) - интегрированной информационной среды, включающей разнородные информационные ресурсы.
    - для построения глобальных распределенных информационных приложений используется архитектура интеграции информационно-вычислительных компонентов *на основе объектно-ориентированного подхода.*

Рис. Классификация информационных систем

2. Понятие и структура проекта ИС

Практически каждая фирма, работающая на рынке информационных технологий, заявляет о предоставлении ею неких *консалтинговых услуг*.

Консалтинг - это деятельность специалиста или целой фирмы, занимающихся стратегическим планированием проекта, анализом и формализацией требований к информационной системе, созданием системного проекта, иногда - проектированием приложений. Но все это до этапа собственно программирования или настройки каких-то уже имеющихся комплексных систем управления предприятием, выбор которых и осуществляется на основе системного проекта. В это понятие не входит системная интеграция.

Фактически консультантом выполняется два вида работ.

1. ***Бизнес-консалтинг.*** Иными словами - это элементарное наведение порядка в организации: бизнес-анализ и реструктуризация (реинжиниринг бизнес-процессов - деятельностей, имеющих ценность). Любая организация - это довольно сложный организм, функционирование которого одному человеку просто невозможно понять. Руководство в общих чертах представляет себе общий ход дел, а клерк досконально изучил только то, чем он конкретно занимается, уяснил свою роль в сложившейся системе деловых взаимоотношений. Но как организация функционирует в целом, не знает, как правило, никто. И именно деятельность, направленная на то, чтобы разобраться в функционировании таких организмов, построить соответствующие модели и на их основе

выдвинуть некоторые предложения по поводу улучшения работы некоторых звеньев, а еще лучше - бизнес-процессов, считается бизнес-консалтингом.

2. ***Системный анализ и проектирование.*** Выявление и согласование требований заказчика приводит к пониманию того, что же в действительности необходимо сделать. За этим следует проектирование или выбор готовой системы так, чтобы она в итоге как можно в большей степени удовлетворяла требованиям заказчика.

Кроме того, важный элемент консалтинга - формирование и обучение рабочих групп. Здесь идет речь не только о традиционной учебе, любые проекты, модели должны в итоге кем-то сопровождаться. Поэтому сотрудники предприятия с самого начала участвуют в проекте, обучаясь сопровождать систему. По окончании проекта они способны анализировать и улучшать бизнес-процессы в рамках собственной отдельно взятой организации.

Цели и этапы разработки консалтинговых проектов (далее просто **проектов).**

Проектирование ИС охватывает три основные области:

- проектирование объектов данных, которые будут реализованы в базе данных;
- проектирование программ, экранных форм, отчетов, которые будут обеспечивать выполнение запросов к данным;
- учет 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.

Проектирование информационных систем всегда начинается с определения **цели** проекта. В общем виде цель проекта можно определить как решение набора взаимосвязанных задач, позволяющих обеспечить на момент запуска системы и в течение всего времени ее эксплуатации:

- требуемую функциональность системы и уровень ее адаптивности к изменяющимся условиям функционирования;
- требуемую пропускную способность системы;
- требуемое временя реакции системы на запрос;
- безотказность работы системы;
- необходимый уровень безопасности;
- простоту эксплуатации и поддержки системы.

Основными целями разработки проектов являются:

- представление деятельности предприятия и принятых в нем технологий в виде иерархии диаграмм, обеспечивающих наглядность и полноту их отображения;
- формирование на основании анализа предложений по реорганизации организационно-управленческой структуры;
- упорядочивание информационных потоков (в том числе документооборота) внутри предприятия;
- выработка рекомендаций по построению рациональных технологий работы подразделений предприятия и его взаимодействию с внешним миром;
- анализ требований и проектирование спецификаций корпоративных информационных систем;
- рекомендации и предложения по применимости и внедрению существующих систем управления предприятиями.

Основные этапы разработки проектов:

1. Анализ первичных требований и планирование работ. Данный этап предваряет инициацию работ над проектом. Его основными задачами являются: анализ первичных бизнес-требований, предварительная экономическая оценка проекта, построение план-графика выполнения работ, создание и обучение совместной рабочей группы.

2. Проведение обследования деятельности предприятия. В рамках этого этапа осуществляется:

- предварительное выявление требований, предъявляемых к будущей системе;
- определение организационной и топологической структур предприятия;
- определение перечня целевых задач (функций) предприятия;
- анализ распределения функций по подразделениям и сотрудникам;
- определение перечня применяемых на предприятии средств автоматизации.

При этом выявляются функциональные деятельности каждого из подразделений предприятия и функциональные взаимодействия между ними, информационные потоки внутри подразделений и между ними, внешние по отношению к предприятию объекты и внешние информационные взаимодействия.

По окончании обследования строится и согласуется с заказчиком предварительный вариант функционирования модели предприятия, включающей идентификацию внешних объектов и информационных взаимодействий с ними, а также детализацию до уровня основных деятельностей предприятия и информационных связей между ними.

3. Построение моделей деятельности предприятия.

На данном этапе осуществляется обработка результатов обследования и построение моделей деятельности предприятия следующих типов:

- ***Модели "как есть"***, представляющей собой "снимок" положения дел на предприятии (оргштатная структура, взаимодействия подразделений, принятые технологии, автоматизированные и неавтоматизированные бизнес-процессы и т.д.) на момент обследования и позволяющий понять, что делает и как функционирует данное предприятие с позиций системного анализа, а также на основе автоматической верификации выявить ряд ошибок и узких мест и сформулировать ряд предложений по улучшению ситуации;
- ***Модели "как должно быть"***, интегрирующей перспективные предложения руководства и сотрудников предприятия, экспертов и системных аналитиков и позволяющей сформировать видение новых рациональных технологий работы предприятия.

Переход от модели "как есть" к модели "как должно быть" осуществляется следующими двумя способами:

1) Совершенствование технологий на основе оценки их эффективности. При этом критериями оценки являются стоимостные и временные затраты выполнения бизнес-процессов, дублирование и противоречивость выполнения отдельных задач бизнес-процесса, степень загруженности сотрудников (***"легкий" реинжиниринг***).

2) Радикальное изменение технологий и переосмысление бизнес-процессов (***"жесткий" реинжиниринг***). Например, вместо попыток улучшения бизнес-процесса проверки кредитоспособности клиента, может быть следует задуматься, а нужна ли вообще такая проверка? Возможно затраты на такие проверки каждого из клиентов во много раз превышают убытки, которые может понести компания в отдельных случаях (например, когда клиентов много, а закупок мало).

4. Разработка системного проекта. Данный этап является первой фазой разработки собственно системы автоматизации (точнее фазой анализа требований к системе), на которой требования заказчика уточняются, формализуются и документируются. Фактически на этом этапе дается ответ на вопрос: "Что должна делать будущая система?" На этом этапе определяются:

- архитектура системы, ее функции, внешние условия ее функционирования, распределение аппаратной и программной частями;
- интерфейсы и распределение функций между человеком и системой;
- требования к программным и информационным компонентам системы, необходимые аппаратные ресурсы, требования к базе данных, физические характеристики компонент системы, их интерфейсы;
- состав людей и работ, имеющих отношения к системе;

- ограничения в процессе разработки (директивные сроки завершения отдельных этапов, имеющиеся ресурсы, организационные процедуры и мероприятия, обеспечивающие защиту информации).

Системный проект строится на основе модели "как должно быть" и включает **функциональную модель** будущей системы в соответствии с одним из общеупотребительных стандартов (например, IDEF0 или IDEF3), **информационную модель** (например, в соответствии со стандартом IDEF1X), а также **техническое задание** на создание автоматизированной системы (например, в соответствии с ГОСТ 34.602-89).

5. Разработка предложений по автоматизации предприятия.

На основании системного проекта осуществляется:

- составление перечня автоматизированных рабочих мест (АРМ) и способов взаимодействия между ними;
- анализ применимости существующих систем управления предприятиями для решения требуемых задач и формирование рекомендаций по выбору готовой системы;
- совместное с заказчиком принятие решения о выборе конкретной системы или разработке собственной системы;
- разработка требований к техническим средствам;
- разработка требований к программным средствам;
- разработка предложений по этапам и срокам реализации.

6. Разработка технического проекта.

На данном этапе на основе системного проекта и принятых решений по автоматизации осуществляется проектирование системы. Этот этап подразделяется на два подэтапа:

- проектирование архитектуры системы, включающее разработку структуры и интерфейсов ее компонент, определение информационных потоков между основными компонентами, связей между ними;
- детальное проектирование каждой компоненты, включающее разработку спецификаций каждой компоненты, разработку требований к тестам и плана интеграции компонент, а также построение моделей иерархии программных модулей и проектирование внутренней структуры модулей.

При этом происходит расширение системного проекта за счет его уточнения, за счет построения моделей автоматизированных рабочих мест.

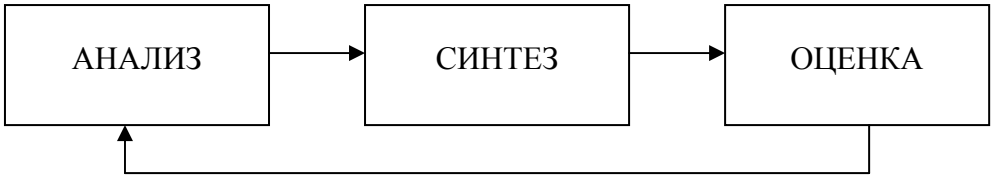
7. Разработка новой системы или настройка существующей системы.

В случае разработки собственной системы последующие этапы являются традиционными: по спецификациям технического проекта осуществляется программирование модулей, их тестирование и отладка.

Настройка существующей системы осуществляется по следующим этапам:

- наполнение системы фактическими данными;
- построение процедур их обработки;
- интеграция процедур внутри автоматизированных рабочих мест;
- интеграция автоматизированных рабочих мест в систему.

Проектирование можно представить как цикл, каждая итерация которого отличается большей детализацией и меньшей общностью



Процесс проектирования ИС требует больших временных, трудовых и материальных затрат, а ошибки при реализации проекта приводят к значительным экономическим потерям. Поэтому важна ***оценка риска*** проекта, при этом рассматривают характеристики трех составляющих:

- заказчика;
- исполнителя;
- проекта.

Характеристики заказчика, влияющие на оценку риска проекта:

- стабильность организационной структуры;
- удовлетворенность заказчика организационной структурой;
- уровень формализации процессов обработки данных в существующей технологии;
- существующий уровень автоматизации процессов сбора и об работки данных;
- уровень подготовки кадров в области автоматизированной технологии обработки данных.

Характеристики исполнителя, влияющие на оценку риска проекта:

- опыт разработки прикладного программного обеспечения (ППО);
- опыт работы с системным программным обеспечением (СПО);
- опыт работы с техническими средствами;
- предполагаемая смена технической и программной среды;
- наличие в группе специалистов в данной предметной области.

Общие показатели проекта, влияющие на оценку его риска:

- уровень охвата автоматизацией процессов обработки данных;
- наличие территориально разнесенных подразделений;
- объем обрабатываемых данных;
- наличие прототипов;
- требования к времени ответа;
- требования к достоверности данных;
- требования к надежности;
- требования к обслуживающему персоналу;
- характер обработки данных (сбор, поиск, представление, оптимизация)

Проектирование информационных систем обычно рассматривается в трех аспектах:

- стадии разработки;
- модели представления;
- уровни детализации.

Стадии разработки определяют в наиболее общей форме состав действий по проектированию ИС, их последовательность и требования к составу и содержанию проектной документации. Стадии разработки регламентируются ГОСТами и отраслевыми стандартами.

Модели представления определяют совокупность понятий (видов элементов и отношений между ними), привлекаемых для описания проектных решений в рамках конкретной предметной области, на определенной стадии разработки, выбранной методики проектирования.

3. Жизненный цикл программного обеспечения ИС и его этапы

Методология проектирования информационных систем описывает процесс создания и сопровождения систем в виде жизненного цикла (ЖЦ) ИС, представляя его как некоторую последовательность стадий и выполняемых на них процессов. Для каждого этапа 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.

Жизненный цикл ИС можно представить как **ряд событий**, происходящих с системой в процессе ее создания и использования.

Каждая из стадий создания системы предусматривает выполнение определенного объема работ, которые представляются в виде **процессов ЖЦ**. Процесс определяется как

совокупность взаимосвязанных действий, преобразующих входные данные в выходные. Описание каждого процесса включает в себя перечень решаемых задач, исходных данных и результатов.

Модель жизненного цикла отражает различные состояния системы, начиная с момента возникновения необходимости в данной ИС и заканчивая моментом ее полного выхода из употребления. Модель жизненного цикла - структура, содержащая процессы, действия и задачи, которые осуществляются в ходе разработки, функционирования и сопровождения программного продукта в течение всей жизни системы, от определения требований до завершения ее использования.

Проектирование информационных систем всегда начинается с определения **цели** проекта. Согласно современной методологии, процесс создания ИС представляет собой процесс построения и последовательного преобразования ряда согласованных моделей на всех этапах жизненного цикла (ЖЦ) ИС. На каждом этапе ЖЦ создаются специфичные для него модели - организации, требований к ИС, проекта ИС, требований к приложениям и т.д. Модели формируются рабочими группами команды проекта, сохраняются и накапливаются в **репозитории** проекта.

Процесс создания ИС делится на ряд этапов (стадий), ограниченных некоторыми временными рамками и заканчивающихся выпуском конкретного продукта (моделей, программных продуктов, документации и пр.).

Обычно выделяют следующие этапы создания ИС:

- формирование требований к системе;
- проектирование;
- реализация;
- тестирование;
- ввод в действие;
- эксплуатация и сопровождение.

Формирование и анализ требований. Целью начальных этапов создания ИС, выполняемых на стадии анализа деятельности организации, является формирование требований к ИС, корректно и точно отражающих цели и задачи организации-заказчика. Фактически на этом этапе дается ответ на вопрос: **"Что должна делать будущая система?"**.

Чтобы специфицировать процесс создания ИС, отвечающей потребностям организации, нужно выяснить и четко сформулировать, в чем заключаются эти потребности. Для этого необходимо определить требования заказчиков к ИС и отобразить их на языке моделей в требования к разработке проекта ИС так, чтобы обеспечить соответствие целям и задачам организации.

Список требований к разрабатываемой системе должен включать:

- совокупность условий, при которых предполагается эксплуатировать будущую систему (аппаратные и программные ресурсы, предоставляемые системе; внешние условия ее функционирования; состав людей и работ, имеющих к ней отношение);
- описание выполняемых системой функций;
- ограничения в процессе разработки (директивные сроки завершения отдельных этапов, имеющиеся ресурсы, организационные процедуры и мероприятия, обеспечивающие защиту информации).

Целью **анализа** является преобразование общих, неясных знаний о требованиях к будущей системе в точные (по возможности) определения. На этом этапе определяются:

- архитектура системы, ее функции, внешние условия, распределение функций между аппаратурой и ПО;
- интерфейсы и распределение функций между человеком и системой;
- требования к программным и информационным компонентам ПО, необходимые аппаратные ресурсы, требования к БД, физические характеристики

Шаг 1. Моделирование бизнес-процессов, протекающих в организации и реализующих ее цели и задачи. Результат - разработка *модели организации*.

Модель организации, описанная в терминах бизнес-процессов и бизнес-функций, позволяет сформулировать основные требования к ИС. Это фундаментальное положение методологии обеспечивает объективность в выработке требований к проектированию системы.
Шаг 2. Множество моделей описания требований к ИС преобразуется в систему моделей, описывающих **концептуальный проект ИС**, включающий:

- модель архитектуры ИС;
- требования к программному обеспечению (ПО);
- требования к информационному обеспечению (ИО).

Шаг 3. Формулирование требований к ИС, в рамках которых:

- формируется **архитектура ПО** и **ИО**;
- выделяются корпоративные БД;
- формируются модели требований к приложениям;
- проводится разработка приложений, их тестирование и интеграция.

Задача формирования требований к ИС является одной из наиболее ответственных, трудно формализуемых и наиболее дорогих и тяжелых для исправления в случае ошибки. Современные инструментальные средства и программные продукты позволяют достаточно быстро создавать ИС по готовым требованиям. Но зачастую эти системы не удовлетворяют заказчиков, требуют многочисленных доработок, что приводит к резкому удорожанию фактической стоимости ИС. Основной причиной такого положения является неправильное, неточное или неполное определение требований к ИС на этапе анализа.

Проектирование

Этап проектирования дает ответ на вопрос **"Как (каким образом) система будет удовлетворять предъявленным к ней требованиям?"**. Задачей этого этапа является исследование структуры системы и логических взаимосвязей ее элементов, причем здесь не рассматриваются вопросы, связанные с реализацией на конкретной платформе. Проектирование определяется как "(итерационный) процесс получения логической модели системы вместе со строго сформулированными целями, поставленными перед нею, а также написания спецификаций физической системы, удовлетворяющей этим требованиям". Обычно этот этап разделяют на два подэтапа:

1. **Проектирование архитектуры ПО**, включающее разработку структуры и интерфейсов компонент, согласование функций и технических требований к компонентам, методам и стандартам проектирования, производство отчетных документов;
2. **Детальное проектирование**, включающее разработку спецификаций каждой компоненты, интерфейсов между компонентами, разработку требований к тестам и плана интеграции компонент.

На этапе проектирования прежде всего формируются **модели данных**. Проектировщики в качестве исходной информации получают результаты анализа. Построение логической и физической моделей данных является основной частью проектирования базы данных. Полученная в процессе анализа информационная модель сначала преобразуется в логическую, а затем в физическую модель данных.
Параллельно с проектированием схемы базы данных выполняется **проектирование процессов**, чтобы получить спецификации (описания) всех модулей ИС. Оба эти процесса проектирования тесно связаны, поскольку часть бизнес-логики обычно реализуется в базе данных (ограничения, триггеры, хранимые процедуры). Главная цель проектирования процессов заключается в отображении функций, полученных на этапе анализа, в модули информационной системы. При проектировании модулей определяют интерфейсы программ: разметку меню, вид окон, горячие клавиши и связанные с ними вызовы.
Конечными продуктами этапа проектирования являются:

- схема базы данных (на основании ER-модели, разработанной на этапе анализа);
- набор спецификаций модулей системы (они строятся на базе моделей функций).

На этапе проектирования осуществляется также разработка **архитектуры ИС**, включающая в себя выбор:

- аппаратной платформы (платформ);
- операционной системы (операционных систем).

В неоднородной ИС могут работать несколько компьютеров на разных аппаратных платформах и под управлением различных операционных систем. Кроме выбора платформы, на этапе проектирования определяются следующие характеристики архитектуры:

- будет ли это архитектура "файл-сервер" или "клиент-сервер";
- будет ли это 3-уровневая архитектура со следующими слоями: сервер, ПО промежуточного слоя (сервер приложений), клиентское ПО;
- будет ли база данных централизованной или распределенной. Если база данных будет распределенной, то какие механизмы поддержки согласованности и актуальности данных будут использоваться;
- будет ли база данных однородной, то есть, будут ли все серверы баз данных продуктами одного и того же производителя . Если база данных не будет однородной, то какое ПО будет использовано для обмена данными между СУБД разных производителей (уже существующее или разработанное специально как часть проекта);.
- будут ли для достижения должной производительности использоваться параллельные серверы баз данных.

В результате деятельности на этапах анализа и проектирования должен быть получен проект системы, содержащий достаточно информации для реализации системы на его основе в рамках бюджета выделенных ресурсов и времени.
Этап проектирования завершается разработкой **технического проекта ИС**.

Реализация
На этапе реализации осуществляется создание программного обеспечения системы, установка технических средств, разработка эксплуатационной документации.

Тестирование
Этап тестирования обычно оказывается распределенным во времени.
Шаг 1. .После завершения разработки отдельного модуля системы выполняют **автономный тест**, который преследует две основные цели:

- обнаружение отказов модуля (жестких сбоев);
- соответствие модуля спецификации (наличие всех необходимых функций, отсутствие лишних функций).

Шаг 2. После того как автономный тест успешно пройдет, модуль включается в состав разработанной части системы и группа сгенерированных модулей проходит **тесты связей**, которые должны отследить их взаимное влияние.
Шаг 3. Группа модулей тестируется на **надежность работы**, то есть проходят:

- тесты имитации отказов системы;
- тесты наработки на отказ.

Первая группа тестов показывает, насколько хорошо система восстанавливается после сбоев программного обеспечения, отказов аппаратного обеспечения.
Вторая группа тестов определяет степень устойчивости системы при штатной работе и позволяет оценить время безотказной работы системы. В комплект тестов устойчивости должны входить тесты, имитирующие пиковую нагрузку на систему.
Шаг 4. Весь комплект модулей проходит **системный тест** - тест внутренней приемки продукта, показывающий уровень его качества. Сюда входят **тесты функциональности** и **тесты надежности** системы.
Шаг 5. Последний тест информационной системы - **приемо-сдаточные испытания**. Такой тест предусматривает показ информационной системы заказчику и должен содержать группу тестов, моделирующих реальные бизнес-процессы, чтобы показать соответствие реализации требованиям заказчика.

4. Эволюция моделей жизненного цикла

Индустрия разработки автоматизированных информационных систем управления зародилась в 1950-х - 1960-х годах и к концу века приобрела вполне законченные формы.

Этап 1. Основным подходом в проектировании ИС был метод "снизу-вверх", когда система создавалась как набор приложений, наиболее важных в данный момент для поддержки деятельности предприятия. Основной целью этих проектов было не создание тиражируемых продуктов, а обслуживание текущих потребностей конкретного учреждения. Такой подход отчасти сохраняется и сегодня. В рамках "лоскутной автоматизации" достаточно хорошо обеспечивается поддержка отдельных функций, но практически полностью отсутствует стратегия развития комплексной системы автоматизации, а объединение функциональных подсистем превращается в самостоятельную и достаточно сложную проблему.

Существуют две основные причины, по которым этот подход терпит неудачу:

1. к выбору инструментальных средств приступают еще до того, как будут определены потребности бизнеса, т. е. здесь выбор программных средств является вопросом технологии, а не бизнеса;
2. этот метод не обеспечивает достаточно тесного взаимодействия заказчиков и бригады разработчиков. При таком подходе, как правило, взаимодействие между бригадой ИТ-специалистов и будущими пользователями вообще отсутствует.

Периодические изменения технологий работы и должностных инструкций, сложности, связанные с разными представлениями пользователей об одних и тех же данных приводили к непрерывным доработкам программных продуктов для удовлетворения все новых и новых пожеланий отдельных работников. Как следствие - и работа программистов, и создаваемые ИС вызывали недовольство руководителей и пользователей системы.

Этап 2. 1960-е -1970-е годы. Проектирование **"сверху-вниз".** Этот этап связан с осознанием того факта, что существует потребность в достаточно стандартных программных средствах автоматизации деятельности различных учреждений и предприятий.

Из всего спектра проблем разработчики выделили наиболее заметные: автоматизацию ведения бухгалтерского аналитического учета и технологических процессов. Системы начали проектироваться "сверху-вниз", т.е. в предположении, что одна программа должна удовлетворять потребности многих пользователей.

Сама идея использования универсальной программы накладывает существенные ограничения на возможности разработчиков по формированию структуры базы данных, экранных форм, по выбору алгоритмов расчета. Заложенные "сверху" жесткие рамки не дают возможности гибко адаптировать систему к специфике деятельности конкретного предприятия:

- учесть необходимую глубину аналитического и производственно-технологического учета,
- включить необходимые процедуры обработки данных,
- обеспечить интерфейс каждого рабочего места с учетом функций и технологии работы конкретного пользователя.

Решение этих задач требует серьезных доработок системы. Таким образом, материальные и временные затраты на внедрение системы и ее доводку под требования заказчика обычно значительно превышают запланированные показатели.

Этап 3. Классическое проектирование ИС берет свое начало в 70-х годах прошлого столетия. **Каскадная** (**водопадная**) модель. Характерна для периода 1970-1985 гг. Основной особенностью данной методики является последовательная организация работ при разбиении структуры ИС на заранее определенный ряд подсистем: организационное, методическое, информационное, программное и аппаратное обеспечения.

Каскадная модель (рис.1) предусматривает последовательное выполнение всех этапов проекта в строго фиксированном порядке. Каждый этап завершается после полного выполнения и документального оформления всех предусмотренных работ.

Каждое приложение при этом подходе представляет собой единый, функционально и информационно независимый блок.

Рис. 1. Каскадная модель ЖЦ ИС

Необходимый набор навыков проектной бригады определяется для каждого этапа и существенно изменяется от одного этапа к другому. В первую очередь определяются требования, которые жестко фиксируются в начале жизненного цикла проекта. Переделка любой части поставляемого изделия после завершения этапа, расценивается как признак низкого качества продукции, полученной на более ранних этапах. Переделка в общем случае связана с изменением, удалением и просто отбрасыванием некоторых уже полученных результатов. Затраты на внесение последующих исправлений в проект обходятся весьма недешево, поскольку они неэффективны. Следует подчеркнуть, что стоимость переделки некоторой проектной задачи в 50—100 раз выше стоимости выполнения той же задачи в рамках проекта.

Каскадный подход хорошо зарекомендовал себя при построении относительно простых ИС, когда в самом начале разработки можно достаточно точно и полно сформулировать все требования к системе.

Можно выделить следующие **положительные стороны** применения каскадного подхода:

- на каждом этапе формируется законченный набор проектной документации, отвечающий критериям полноты и согласованности;
- выполняемые в логической последовательности этапы работ позволяют планировать сроки завершения всех работ и соответствующие затраты.

Основные **недостатки** этого подхода:

Реальный процесс создания системы никогда полностью не укладывается в такую жесткую схему, постоянно возникает потребность в возврате к предыдущим этапам и уточнении или пересмотре ранее принятых решений. В результате реальный процесс создания ИС оказывается соответствующим **поэтапной модели с промежуточным контролем**.

Поэтапная модель с промежуточным контролем (рис. 2). Разработка ИС ведется итерациями с циклами обратной связи между этапами. Межэтапные корректировки позволяют учитывать реально существующее взаимовлияние результатов разработки на различных этапах; время жизни каждого из этапов растягивается на весь период разработки.

Однако и эта схема не позволяет оперативно учитывать возникающие изменения и уточнения требований к системе. Согласование результатов разработки с пользователями производится только в точках, планируемых после завершения каждого этапа работ, а общие требования к ИС зафиксированы в виде технического задания на все время ее создания. Таким образом, пользователи зачастую получают систему, не удовлетворяющую их реальным потребностям.

Спиральная модель процесса разработки (рис. 3)**.** Характерна для периода после 1986 г. и является альтернативой каскадной модели.

Рис. 2. Поэтапная модель с промежуточным контролем

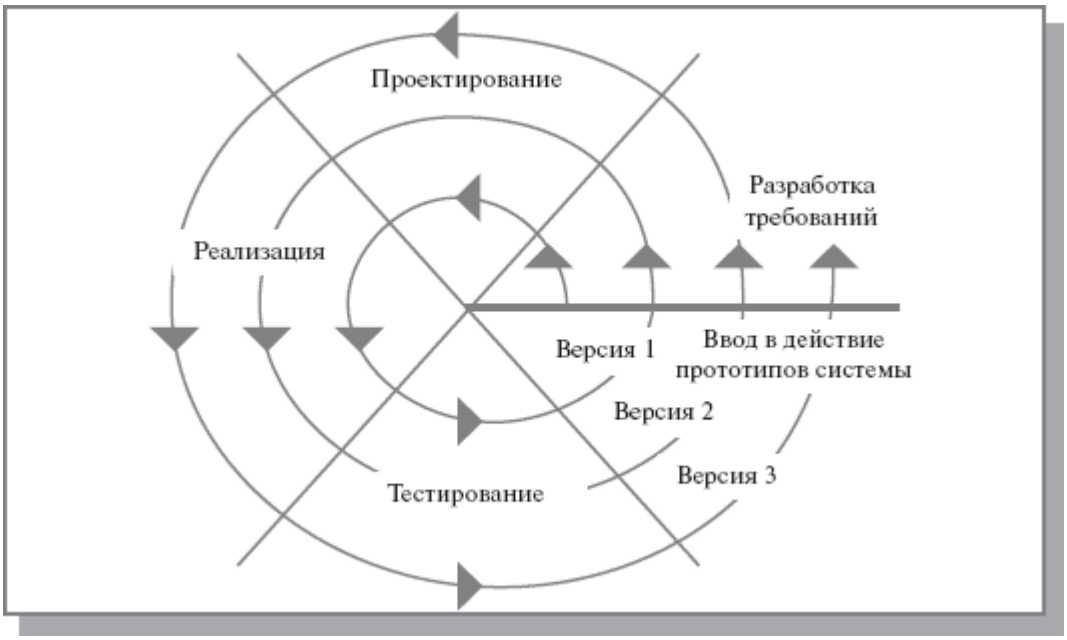


Рис. 3. Спиральная модель ЖЦ ИС

При этом подходе проектные задачи группируются в виде фаз (“витков спирали”), направленных на разработку последовательности поставляемых изделий, каждое из которых со временем наращивает свои функциональные возможности. В пределах фаз разработки конечные продукты обычно разбиваются на более мелкие, но пригодные к эксплуатации компоненты, так что функциональность поставляемых заказчику изделий, прошедших несколько итераций, со временем наращивается. Сущность этого процесса состоит в том, чтобы постоянно предоставлять организации-заказчику реальные, имеющие практическую ценность компоненты информационной системы, которые можно использовать, проанализировать, а затем по результатам анализа выработать изменения к первоначальным требованиям на поставляемое изделие. Сформулированные изменения учитываются на следующей итерации при наращивании функциональных возможностей компонентов, входящих в комплект поставки.

На каждом витке спирали выполняется создание очередной версии продукта, уточняются требования проекта, определяется его качество и планируются работы следующего витка. Каждый виток спирали соответствует созданию работоспособного фрагмента или версии системы. Это позволяет уточнить требования, цели и характеристики проекта, определить качество разработки, спланировать работы следующего витка спирали. Таким образом, углубляются и последовательно конкретизируются детали проекта, и в результате выбирается обоснованный вариант, который удовлетворяет действительным требованиям

заказчика и доводится до реализации. Это обеспечивает обратную связь, которая проясняет потребности заказчика по мере создания изделий.
Итеративная разработка отражает объективно существующий спиральный цикл создания сложных систем. Она позволяет переходить на следующий этап, не дожидаясь полного завершения работы на текущем и решить главную задачу - как можно быстрее показать пользователям системы работоспособный продукт, тем самым, активизируя процесс уточнения и дополнения требований.
В то время как состав проектной бригады на различных этапах водопадной модели подвержен изменениям, спиральная модель ориентирована на стабильный состав бригады разработчиков.
Одно из главных отличий водопадной модели от спиральной состоит в том, что затраты на переделку компонентов при использовании спиральной модели невелики, поскольку в данном случае это обычно связано с добавлением функциональных возможностей к предыдущему поставляемому изделию. Так как фазы спиральной модели определяются в терминах роста функциональных возможностей поставляемых изделий, а не процесса, используемого для создания этих изделий, требования к ним наперед не известны.
Таким образом, можно отметить следующие достоинства спиральной модели:

- накопление и повторное использование программных средств, моделей и прототипов;
- ориентация на развитие и модификацию ПО в процессе его проектирования;
- анализ риска и издержек в процессе проектирования.

В процессе совершенствования появилась схема **непрерывной** разработки ИС (рис.4). Характерной особенностью данной методики стал непрерывный спиральный процесс разработки ИС с **планируемыми** точками передачи в эксплуатацию новых версий и новых функциональных подсистем.

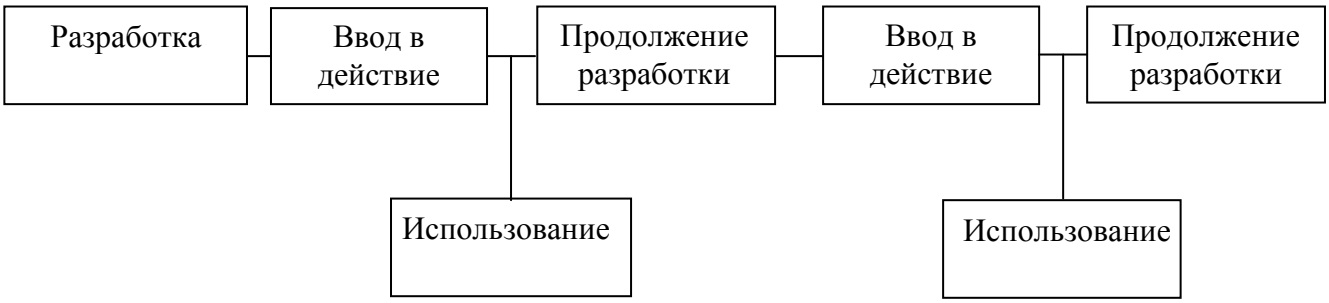


Рис.4. Схема непрерывной разработки

Развитие схемы непрерывной разработки связано с совершенствованием циклических форм проектирования. Примером такого подхода является ускоренный метод проектирования приложений, получивший название «**Быстрое прототипирование**». В проектный цикл дополнительно включаются стадии разработки **макета-прототипа** и его опробирование (рис. 5).

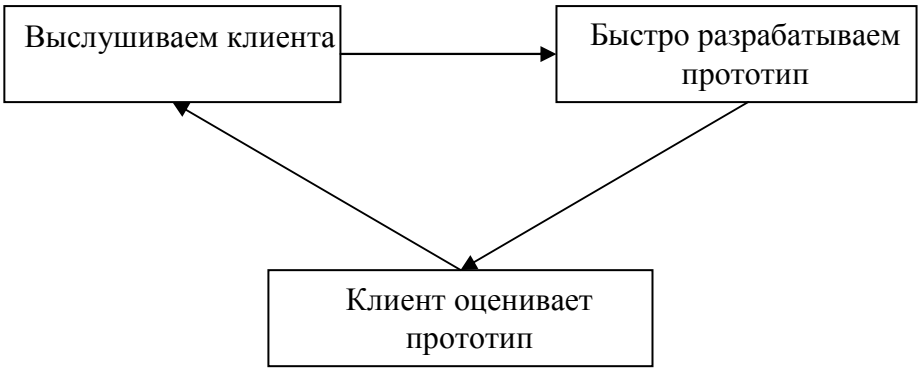


Рис. 5. Схема быстрого прототипирования

Суть этого метода состоит в быстрой разработке части программы и ее оценке в процессе практического использования. Полученный опыт учитывается при повторном проходе цикла разработки. В конечном итоге последний прототип поставляется заказчику в виде готового приложения (или осуществляется переход к другому подходу).
Прототипирование представляет собой проверку концепций, при которой часто приходится отбрасывать частично готовые варианты программы.
Посредством создания прототипов проверяется реализуемость технических решений и степень удовлетворения потребностей заказчика на этапах анализа и проектирования.
Быстрое прототипирование ориентировано на взаимодействие с заказчиком и помогает быстро определять его требования. Оно может быть полезно для демонстрации технических возможностей реализации в тех случаях, когда речь идет о технологии, которая может вызвать затруднения.
Замечание. Итерационная разработка также допускает исключение части кода при изменениях. Но такое исключение не является намеренным.
Основная **проблема спирального цикла** - определение момента перехода на следующий этап. Для ее решения вводятся временные ограничения на каждый из этапов жизненного цикла, и переход осуществляется в соответствии с планом, даже если не вся запланированная работа закончена. Планирование производится на основе статистических данных, полученных в предыдущих проектах, и личного опыта разработчиков.
Недостатками **схемы непрерывной разработки** является жесткость используемых моделей проектирования и **закрытость** создаваемых ИС.
Сложности использования **быстрого прототипирования** состоят в том, что:

- отказаться от написанного кода часто бывает сложно;
- заказчики часто путают успешный прототип с готовым продуктом и не понимают, что прототип предназначен только для демонстрации и может быть лишен устойчивой инфраструктуры; некоторые клиенты рассматривают отбрасывание кода как выбрасывание денег на ветер, им трудно понять, что истинная ценность прототипов состоит в том опыте, который приобретается с их помощью.

Вывод:

1. Каскадная модель предполагает разработку законченных продуктов на каждом этапе: технического задания, технического проекта, программного продукта и пользовательской документации. Разработанная документация позволяет не только определить требования к продукту следующего этапа, но и определить обязанности сторон, объем работ и сроки, при этом окончательная оценка сроков и стоимости проекта производится на начальных этапах, после завершения обследования. Очевидно, что если требования к информационной системе меняются в ходе реализации проекта, а качество документов оказывается невысоким (требования неполны и/или противоречивы), то в действительности использование каскадной модели создает лишь иллюзию определенности и на деле увеличивает риски, уменьшая лишь ответственность участников проекта. При формальном подходе менеджер проекта реализует только те требования, которые содержатся в спецификации, опирается на документ, а не на реальные потребности бизнеса.
2. **Проблемы внедрения** при использовании итерационной модели. В некоторых областях спиральная модель не может применяться, поскольку невозможно использование/тестирование продукта, обладающего неполной функциональностью (например, военные разработки, атомная энергетика и т.д.). Поэтапное итерационное внедрение информационной системы для бизнеса возможно, но сопряжено с организационными сложностями (перенос данных, интеграция систем, изменение бизнес-процессов, учетной политики, обучение пользователей). Трудозатраты при поэтапном итерационном внедрении оказываются значительно выше, а управление проектом требует настоящего искусства. Предвидя указанные сложности, заказчики выбирают каскадную модель, чтобы "внедрять систему один раз".

На практике наибольшее распространение получили каскадная и спиральная модели жизненного цикла. Однако следует отметить, что для больших организаций эти модели, какой бы степени детализации они не достигали, никогда нельзя назвать корпоративными.
Этап 4. В то же время заказчики ИС стали выдвигать все больше требований, направленных на обеспечение возможности комплексного использования корпоративных данных в управлении и планировании своей деятельности. Таким образом, возникла насущная необходимость формирования новой методологии построения информационных систем.
Цель такой методологии заключается в регламентации процесса проектирования ИС и обеспечении управления этим процессом с тем, чтобы гарантировать выполнение требований как к самой ИС, так и к характеристикам процесса разработки. Основными задачами, решению которых должна способствовать методология проектирования корпоративных ИС, являются следующие:

- обеспечивать создание корпоративных ИС, отвечающих целям и задачам организации, а также предъявляемым требованиям по автоматизации деловых процессов заказчика;
- гарантировать создание системы с заданным качеством в заданные сроки и в рамках установленного бюджета проекта;
- поддерживать удобную дисциплину сопровождения, модификации и наращивания системы;
- обеспечивать преемственность разработки, т.е. использование в разрабатываемой ИС существующей информационной инфраструктуры организации (задела в области информационных технологий).

В настоящее время в целом сформировалась идеология и практика применения информационных технологий. Однако необходима организация информационных процессов и технологий, **как системы**, для построения которой целесообразно применить системный подход.
Таким образом, следствием недостатков классических методов проектирования явился переход к **системному** проектированию.
Примечание. Для хорошего плана характерно наличие нескольких промежуточных результатов в виде прототипов или отдельных поставляемых заказчику компонентов, представляющих реальную ценность для организации. Еще одна отличительная черта хорошего плана и умелого управления проектом - возможность получить готовые компоненты прежде, чем существенно изменится модель данных. Т.о., успешный проект по моделированию корпоративных данных обычно завершается вместе с созданием моделей для части предприятия.

5. Организация канонического проектирования ИС. Стадии и этапы

Организация канонического проектирования ИС ориентирована на использование главным образом каскадной модели жизненного цикла ИС. Стадии и этапы работы описаны в стандарте ГОСТ 34.601-90.
В зависимости от сложности объекта автоматизации и набора задач, требующих решения при создании конкретной ИС, стадии и этапы работ могут иметь различную трудоемкость. Допускается объединять последовательные этапы и даже исключать некоторые из них на любой стадии проекта. Допускается также начинать выполнение работ следующей стадии до окончания предыдущей.
Стадии и этапы создания ИС, выполняемые организациями-участниками, прописываются в **договорах** и **технических заданиях** на выполнение работ:
Стадия 1. **Формирование требований к ИС**. На начальной стадии проектирования выделяют следующие этапы работ:

- обследование объекта и обоснование необходимости создания ИС;
- формирование требований пользователей к ИС;
- оформление отчета о выполненной работе и тактико-технического задания на разработку.

Стадия 2. **Разработка концепции ИС**.

- изучение объекта автоматизации;
- проведение необходимых научно-исследовательских работ;
- разработка вариантов концепции ИС, удовлетворяющих требованиям пользователей;
- оформление отчета и утверждение концепции.

Стадия 3. **Техническое задание**.
- разработка и утверждение **технического задания** на создание ИС.

Стадия 4. **Эскизный проект**.
- разработка предварительных проектных решений по системе и ее частям;
- разработка эскизной документации на ИС и ее части.

Стадия 5. **Технический проект**.
- разработка проектных решений по системе и ее частям;
- разработка документации на ИС и ее части;
- разработка и оформление документации на поставку комплектующих изделий;
- разработка заданий на проектирование в смежных частях проекта.

Стадия 6. **Рабочая документация**.
- разработка **рабочей документации** на ИС и ее части;
- разработка и адаптация программ.

Стадия 7. **Ввод в действие**.
- подготовка объекта автоматизации;
- подготовка персонала;
- комплектация ИС поставляемыми изделиями (программными и техническими средствами, программно-техническими комплексами, информационными изделиями);
- проведение *предварительных испытаний*;
- проведение *опытной эксплуатации*;
- проведение *приемочных испытаний*.

Стадия 8. **Сопровождение ИС**.
- выполнение работ в соответствии с гарантийными обязательствами;
- послегарантийное обслуживание.

Обследование - это изучение и диагностический анализ организационной структуры предприятия, его деятельности и существующей системы обработки информации.
Материалы, полученные в результате обследования, используются для:
- обоснования разработки и поэтапного внедрения систем;
- составления технического задания на разработку систем;
- разработки технического и рабочего проектов систем.

На **этапе обследования** целесообразно выделить две составляющие: определение стратегии внедрения ИС и детальный анализ деятельности организации.
Основная задача первого этапа обследования - оценка реального объема проекта, его целей и задач на основе выявленных функций и информационных элементов автоматизируемого объекта высокого уровня. Эти задачи могут быть реализованы или заказчиком ИС самостоятельно, или с привлечением консалтинговых организаций. Этап предполагает тесное взаимодействие с основными потенциальными пользователями системы и бизнес-экспертами. Основная задача взаимодействия - получить полное и однозначное понимание требований заказчика. Как правило, нужная информация может быть получена в результате интервью, бесед или семинаров с руководством, экспертами и пользователями.
По завершении этой стадии обследования появляется возможность определить вероятные технические подходы к созданию системы и оценить затраты на ее реализацию (затраты на аппаратное обеспечение, закупаемое программное обеспечение и разработку нового программного обеспечения).

Результатом **этапа определения стратегии** является документ (***технико-экономическое обоснование проекта*),** где четко сформулировано, что получит заказчик, если согласится финансировать проект, когда он получит готовый продукт (график выполнения работ) и сколько это будет стоить (для крупных проектов должен быть составлен график финансирования на разных этапах работ). В документе желательно отразить не только затраты, но и выгоду проекта, например время окупаемости проекта, ожидаемый экономический эффект (если его удается оценить).

Ориентировочное содержание этого документа:

- ограничения, риски, критические факторы, которые могут повлиять на успешность проекта;
- совокупность условий, при которых предполагается эксплуатировать будущую систему: архитектура системы, аппаратные и программные ресурсы, условия функционирования, обслуживающий персонал и пользователи системы;
- сроки завершения отдельных этапов, форма приемки/сдачи работ, привлекаемые ресурсы, меры по защите информации;
- описание выполняемых системой функций;
- возможности развития системы;
- информационные объекты системы;
- интерфейсы и распределение функций между человеком и системой;
- требования к программным и информационным компонентам ПО, требования к СУБД;
- что не будет реализовано в рамках проекта.

На **этапе детального анализа деятельности** организации изучаются задачи, обеспечивающие реализацию функций управления, организационная структура, штаты и содержание работ по управлению предприятием, а также характер подчиненности вышестоящим органам управления. На этом этапе должны быть выявлены:

- инструктивно-методические и директивные материалы, на основании которых определяются состав подсистем и перечень задач;
- возможности применения новых методов решения задач.

Аналитики собирают и фиксируют информацию в двух взаимосвязанных формах:

- *функции* - информация о событиях и процессах, которые происходят в бизнесе;
- *сущности* - информация о вещах, имеющих значение для организации и о которых что-то известно.

При изучении каждой функциональной задачи управления определяются:

- наименование задачи; сроки и периодичность ее решения;
- степень формализуемости задачи;
- источники информации, необходимые для решения задачи;
- показатели и их количественные характеристики;
- порядок корректировки информации;
- действующие алгоритмы расчета показателей и возможные методы контроля;
- действующие средства сбора, передачи и обработки информации;
- действующие средства связи;
- принятая точность решения задачи;
- трудоемкость решения задачи;
- действующие формы представления исходных данных и результатов их обработки в виде документов;
- потребители результатной информации по задаче.

Одной из наиболее трудоемких, хотя и хорошо формализуемых задач этого этапа является **описание документооборота** организации. При обследовании документооборота составляется схема маршрута движения документов, которая должна отразить:

- количество документов;
- место формирования показателей документа;

- взаимосвязь документов при их формировании;
- маршрут и длительность движения документа;
- место использования и хранения данного документа;
- внутренние и внешние информационные связи;
- объем документа в знаках.

По результатам обследования устанавливается перечень задач управления, решение которых целесообразно автоматизировать, и очередность их разработки.
На этапе обследования следует классифицировать планируемые функции системы по степени важности. Один из возможных форматов представления такой классификации:

- необходимые функции;
- желательные функции;
- возможные функции;
- отсутствующие функции.

Функции первой категории обеспечивают критичные для успешной работы системы возможности.
Реализация функций второй и третьей категорий ограничивается временными и финансовыми рамками: разрабатывается то, что необходимо, а также максимально возможное в порядке приоритета число функций второй и третьей категорий.
Последняя категория функций особенно важна, поскольку необходимо четко представлять границы проекта и набор функций, которые будут отсутствовать в системе.
Модели деятельности организации создаются в двух видах:

- модель ***"как есть"*** - отражает существующие в организации бизнес-процессы;
- модель ***"как должно быть"*** - отражает необходимые изменения бизнес-процессов с учетом внедрения ИС.

На этапе анализа необходимо привлекать к работе группы тестирования для решения следующих задач:

- получения сравнительных характеристик предполагаемых к использованию аппаратных платформ, операционных систем, СУБД, иного окружения;
- разработки плана работ по обеспечению надежности информационной системы и ее тестирования.

Результаты обследования представляют объективную основу для формирования ***технического задания*** на информационную систему.
Техническое задание- это документ, определяющий цели, требования и основные исходные данные, необходимые для разработки автоматизированной системы управления.
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.

Эскизный проект предусматривает разработку предварительных проектных решений по системе и ее частям.
Выполнение стадии эскизного проектирования не является строго обязательной. Если основные проектные решения определены ранее или достаточно очевидны для конкретной ИС и объекта автоматизации, то эта стадия может быть исключена из общей последовательности работ.

Содержание эскизного проекта задается в ТЗ на систему. Как правило, на этапе эскизного проектирования определяются:

- функции ИС;
- функции подсистем, их цели и ожидаемый эффект от внедрения;
- состав комплексов задач и отдельных задач;
- концепция информационной базы и ее укрупненная структура;
- функции системы управления базой данных;
- состав вычислительной системы и других технических средств;
- функции и параметры основных программных средств.

По результатам проделанной работы оформляется, согласовывается и утверждается документация в объеме, необходимом для описания полной совокупности принятых проектных решений и достаточном для дальнейшего выполнения работ по созданию системы.

На основе технического задания (и эскизного проекта) разрабатывается **технический проект** ИС. Технический проект системы - это техническая документация, содержащая общесистемные проектные решения, алгоритмы решения задач, а также оценку экономической эффективности автоматизированной системы управления и перечень мероприятий по подготовке объекта к внедрению.

На этом этапе осуществляется комплекс научно-исследовательских и экспериментальных работ для выбора основных проектных решений и расчет экономической эффективности системы.

В завершение стадии технического проектирования производится разработка документации на поставку серийно выпускаемых изделий для комплектования ИС, а также определяются технические требования и составляются ТЗ на разработку изделий, не изготовляемых серийно.

На стадии "**рабочая документация**" осуществляется создание программного продукта и разработка всей сопровождающей документации. Документация должна содержать все необходимые и достаточные сведения для обеспечения выполнения работ по вводу ИС в действие и ее эксплуатации, а также для поддержания уровня эксплуатационных характеристик (качества) системы. Разработанная документация должна быть соответствующим образом оформлена, согласована и утверждена.

Для ИС, которые являются разновидностью автоматизированных систем, устанавливают следующие основные виды испытаний: предварительные, опытная эксплуатация и приемочные. При необходимости допускается дополнительно проведение других видов испытаний системы и ее частей.

В зависимости от взаимосвязей частей ИС и объекта автоматизации испытания могут быть автономные или комплексные. Автономные испытания охватывают части системы. Их проводят по мере готовности частей системы к сдаче в опытную эксплуатацию. Комплексные испытания проводят для групп взаимосвязанных частей или для системы в целом.

Для планирования проведения всех видов испытаний разрабатывается документ "Программа и методика испытаний". Разработчик документа устанавливается в договоре или ТЗ. В качестве приложения в документ могут включаться тесты или контрольные примеры.

Предварительные испытания проводят для определения работоспособности системы и решения вопроса о возможности ее приемки в опытную эксплуатацию. Предварительные испытания следует выполнять после проведения разработчиком отладки и тестирования поставляемых программных и технических средств системы и представления им соответствующих документов об их готовности к испытаниям, а также после ознакомления персонала ИС с эксплуатационной документацией.

Опытную эксплуатацию системы проводят 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.

6. Краткая характеристика применяемых технологий проектирования

Сравнение существующих методик

В **функциональных моделях** (DFD-диаграммах потоков данных, SADT-диаграммах) главными структурными компонентами являются *функции* (*операции*, действия, работы), которые на диаграммах связываются между собой потоками объектов.
Несомненным достоинством функциональных моделей является реализация структурного подхода к проектированию ИС по принципу "сверху-вниз", когда каждый функциональный блок может быть декомпозирован на множество подфункций и т.д., выполняя, таким образом, модульное проектирование ИС. Для функциональных моделей характерны процедурная строгость декомпозиции ИС и наглядность представления.
При функциональном подходе объектные модели данных в виде ER-диаграмм "объект — свойство — связь" разрабатываются отдельно. Для проверки корректности моделирования предметной области между функциональными и объектными моделями устанавливаются взаимно однозначные связи.
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 снимаются в **объектно-ориентированных моделях**, в которых главным структурообразующим компонентом выступает класс объектов с набором *функций*, которые могут обращаться к атрибутам этого класса.
Для классов объектов характерна иерархия обобщения, позволяющая осуществлять **наследование** не только атрибутов (свойств) объектов от вышестоящего класса объектов к нижестоящему классу, но и *функций* (методов).
В случае наследования *функций* можно абстрагироваться от конкретной реализации процедур (**абстрактные типы данных**), которые отличаются для определенных подклассов ситуаций. Это дает возможность обращаться к подобным программным модулям по общим именам (**полиморфизм**) и осуществлять повторное использование программного кода при модификации программного обеспечения. Таким образом, адаптивность объектно-ориентированных систем к изменению предметной области по сравнению с функциональным подходом значительно выше.
При объектно-ориентированном подходе изменяется и принцип проектирования ИС. Сначала выделяются классы объектов, а далее в зависимости от возможных состояний объектов (жизненного цикла объектов) определяются методы обработки (функциональные процедуры), что обеспечивает наилучшую реализацию динамического поведения информационной системы.
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.
При выборе методики моделирования предметной области обычно в качестве критерия выступает степень ее динамичности. Для более регламентированных задач больше подходят функциональные модели, для более адаптивных *бизнес-процессов* (управления рабочими потоками, реализации динамических запросов к информационным хранилищам) — объектно-ориентированные модели. Однако в рамках одной и той же ИС для

различных классов задач могут требоваться различные виды моделей, описывающих одну и ту же проблемную область. В таком случае должны использоваться комбинированные *модели предметной области*

В **функциональных моделях** (DFD-диаграммах потоков данных, SADT-диаграммах) главными структурными компонентами являются *функции* (*операции*, действия, работы), которые на диаграммах связываются между собой потоками объектов.

Несомненным достоинством функциональных моделей является реализация структурного подхода к проектированию ИС по принципу "сверху-вниз", когда каждый функциональный блок может быть декомпозирован на множество подфункций и т.д., выполняя, таким образом, модульное проектирование ИС. Для функциональных моделей характерны процедурная строгость декомпозиции ИС и наглядность представления.

При функциональном подходе объектные модели данных в виде ER-диаграмм "объект — свойство — связь" разрабатываются отдельно. Для проверки корректности моделирования предметной области между функциональными и объектными моделями устанавливаются взаимно однозначные связи.

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.

При выборе методики моделирования предметной области обычно в качестве критерия выступает степень ее динамичности. Для более регламентированных задач больше подходят функциональные модели, для более адаптивных *бизнес-процессов* (управления рабочими потоками, реализации динамических запросов к информационным хранилищам) — объектно-ориентированные модели. Однако в рамках одной и той же ИС для различных классов задач могут требоваться различные виды моделей, описывающих одну и ту же проблемную область. В таком случае должны использоваться комбинированные *модели предметной области*

При всем разнообразии моделей предметных областей концептуального уровня отсутствуют такие модели, которые бы позволяли в полной мере использовать знания по классификации элементов предметной области для описания свойств ее элементов, и в то же время, сохраняли преимущества традиционных функционального и информационного подходов, основанных на модели данных.

7. Требования, предъявляемые к технологии проектирования ИС. Выбор технологии проектирования

Рациональное построение и оптимизация информационных технологий возможны только на основе использования параметрической модели процесса.

Параметры - измеримые величины, характеризующие структуру процесса и его развитие. Параметры информационных технологий отражают взаимосвязанное множество характеристик процессов. Параметры элементов системы проектирования информационной технологии взаимозависимы.

Рассматривая основные характеристики технологических процессов обработки данных, используются обобщенные показатели с дальнейшей их детализацией на других уровнях анализа системы обработки данных.

К таким параметрам относятся:

- экономический эффект от автоматизации обработки данных;
- капитальные затраты на средства вычислительной и организационной техники;
- стоимость проектирования технических процессов обработки данных;
- ресурсы на проектирование и эксплуатацию системы;
- срок проектирования технологии обработки данных;
- эксплуатационные расходы;
- параметры функциональных задач;

- параметры вычислительной и организационной техники;
- стоимость организации и эксплуатации баз данных или файлов данных;
- параметры структур хранения и стоимость хранения данных;
- время доступа к данным;
- время решения функциональных задач пользователей;
- эффективность методов контроля.

На технологию обработки данных влияют факторы, не зависящие или слабо зависящие от проектировщика - ***нерегулируемые***, и факторы, на которые он может оказать существенное влияние - ***регулируемые*** (управляемые).

К *нерегулируемым* параметрам технологии можно отнести:

- объем входных и выходных данных;
- сложность алгоритма и объем вычислений;
- периодичность и регламентность решения задач;
- степень использования результатов одной задачи в других задачах;
- параметры жестко заданных технических средств и общесистемного программного обеспечения и т.д.

К *регулируемым* параметрам технологии можно отнести:

- выбор характеристик технических средств и программного обеспечения;
- параметры информационного обеспечения;
- методы контроля и защиты данных;
- размещение технических средств;
- последовательность операций технологического процесса.

В процессе выбора регулируемых (управляемых) параметров при проектировании технологии обработки данных хорошим подспорьем является использование методов математического моделирования. Иногда для упрощения задачи приходится рассматривать отдельные фрагменты технологического процесса, осуществляя поиск рациональных решений. Таким методом надо пользоваться очень осторожно, так как частичная оптимизация может оказать отрицательное влияние на общую оптимизацию.

Оценка качества технологических процессов

Проектирование рациональной технологии следует рассматривать как задачу принятия решений. Каждая задача такого типа характеризуется наличием ряда целей и наличием различных путей достижения этих целей с различной эффективностью их реализации. Эффективность реализации различных вариантов технологического процесса должна быть количественно определена, т.е. выражена с помощью определенной величины: критерия эффективности.

Пользуясь этим показателем, можно определить сравнительные достоинства и недостатки различных вариантов организации технологических процессов. Кроме того, углубляясь в сравнительные оценки, необходимо говорить и об эффективности использования тех или иных готовых программных продуктов однотипных или близких по своим функциональным возможностям, будь то табличные процессоры, текстовые редакторы или базы данных.

Анализируя сложность системы, в качестве критерия часто используется отношение **затрат** и **выпуска**. Этот критерий целесообразно применять и при анализе технологии обработки данных. Выпуском при этом можно было бы считать удовлетворение информационных потребностей пользователей. При этом затраты и выпуск должны быть выражены в одних и тех же единицах. Тогда критерий оценки вариантов технологий может быть определен величиной:

$A = B - C$,

где: B - стоимостная оценка выпуска;

C - затраты на разработку (приобретение, модификацию) и функционирование технологии обработки данных.

При этом предпочтение отдается варианту с большим значением А.
В настоящее время, к сожалению, нет достаточно надежных способов определения стоимостной оценки выпуска.
Но, когда для разных технологий имеется одинаковое удовлетворение информационных потребностей пользователей, в качестве критерия эффективности можно принять затраты (Z) . В этом случае выбор вариантов технологий должен осуществляться по минимуму затрат.
Затраты можно разложить на ряд составляющих:
Z = Zr + E + Ze + Zm,
где: Zr - разовые затраты на разработку, отладку, внедрение технологии, приобретение доп. оборудования, обучение персонала и т.д.;
E - коэффициент эффективности капитальных вложений;
Ze - эксплуатационные затраты, связанные с работой по выбранной технологии;
Zm - затраты, связанные с модификацией и адаптацией технологии обработки данных.
Помимо глобального критерия, рассмотренного ранее (эффективность), используются и локальные критерии, одним из которых является время решения задачи на ЭВМ. В настоящее время поставлен и решен целый ряд задач по рациональной и оптимальной технологии обработки данных. Эти задачи связаны с выбором организации информационных массивов, выбором способов обработки данных, в частности выбором методов сортировки, способов разделения задач на модули, поиска информации.
Большое внимание уделяется методам обеспечения достоверности и надежности информации и т.д.
В основе качественной оценки информационной технологии лежит многообразие методов и способов их конструирования. Важнейшим показателем является степень соответствия информационной технологии научно-техническому уровню ее развития.
Другим важнейшим показателем качества информационных технологий является функциональная полнота (F) - отношение областей автоматизированной обработки информации (Qa) к области обработки информации (Qи) для функционирования всей системы управления:
F = Qa / Qи
Показатель своевременности переработки информации (Ксв) определяется числом значений показателей, разработанных в рамках информационной технологии в течение определенного времени (t), и значений показателей, полученных за пределами планового срока их представления (t):
Ксв =(t - D t) / t
Качественной характеристикой информационных технологий являются показатели их надежности. Различают функциональную и адаптивную надежности.
Функциональная - свойство информационных технологий с определенной надежностью реализовать функции информационного программно-технологического обеспечения, технического и эргономического обеспечения.
Адаптивная - свойство информационной технологии реализовывать свои функции при их изменении в пределах установленных при проектировании границ.

8. Системы автоматизированного проектирования ИС: CASE-технологии. Оценка и выбор CASE-средств

Под **CASE-технологией** будем понимать комплекс программных средств, поддерживающих процессы создания и сопровождения программного обеспечения, включая анализ и формулировку требований, проектирование, генерацию кода, тестирование, документирование, обеспечение качества, конфигурационное управление и управление проектом (CASE может обеспечивать поддержку только в заданных функциональных областях или в широком диапазоне функциональных областей).

В связи с наличием двух подходов к проектированию программного обеспечения существуют CASE-технологии, ориентированные на **структурный** подход, **объектно-ориентированный** под ход, а также комбинированные.
Исходя из основных положений объектно-ориентированного подхода, рассмотрим концепцию идеального объектно-ориентированного CASE-средства.
Классическая постановка задачи разработки программной системы (**инжиниринг**) представляет собой спиральный цикл итеративного чередования этапов объектно-ориентированного анализа, проектирования и реализации (программирования).
В реальной практике в большинстве случаев имеется предыстория в виде совокупности разработанных и внедренных программ, которые целесообразно использовать при разработке новой системы. Процесс проектирования в таком случае основан на **реинжиниринге** программных кодов, при котором путем анализа текстов программ восстанавливается исходная модель программной системы.
Современные CASE-средства поддерживают процессы инжиниринга и автоматизированного реинжиниринга.
Идеальное объектно-ориентированное CASE-средство (рис.1.) должно содержать четыре основных блока: **анализ**, **проектирование**, **разработка** и **инфраструктура**.
Основные требования к **блоку анализа**:

- возможность выбора выводимой на экран информации из всей совокупности данных, описывающих модели;
- согласованность диаграмм при хранении их в репозитории;
- внесение комментариев в диаграммы и соответствующую документацию для фиксации проектных решений;
- возможность динамического моделирования в терминах событий;
- поддержка нескольких нотаций (хотя бы три нотации — Г. Буча, И.Джекобсона и OMT).

Основные требования к **блоку проектирования**:

- поддержка всего процесса проектирования приложения;
- возможность работы с библиотеками, средствами поиска и выбора;
- возможность разработки пользовательского интерфейса;
- поддержка стандартов OLE, ActiveX и доступ к библиотекам HTML или Java;
- поддержка разработки распределенных или двух- и трехзвенных клиент-серверных систем (работа с CORBA, DCOM, Internet).

Основные требования к **блоку реализации**:

- генерация кода полностью из диаграмм;
- возможность доработки приложений в клиент-серверных CASE-средствах типа Power Builder;
- реинжиниринг кодов и внесение соответствующих изменений в модель системы;
- наличие средств контроля, которые позволяют выявлять несоответствие между диаграммами и генерируемыми кодами и обнаруживать ошибки как на стадии проектирования, так и на стадии реализации.

Основные требования к **блоку инфраструктуры**:

- наличие репозитория на основе базы данных, отвечающего за генерацию кода, реинжиниринг, отображение кода на диаграммах, а также обеспечивающего соответствие между моделями и программными кодами;
- обеспечение командной работы (многопользовательской работы и управление версиями) и реинжиниринга.

<table>
<tr><td colspan="2">Анализ</td><td colspan="2">Проектирование</td><td>Реализация</td></tr>
<tr><td colspan="2">Возможность добавлять пояснительные надписи к диаграммам и в документацию</td><td colspan="2">Возможность создавать различные представления и скрывать ненужные в данный момент слои системы</td><td>Возможность просматривать и выбирать элементы и бизнес-объекты для использования в системе</td><td>Возможность генерировать заготовки программного кода на нескольких объектно-ориентированных языках</td></tr>
<tr><td colspan="3">Среда для создания диаграмм разнообразных моделей</td><td>Возможность создания пользовательского интерфейса (поддержка OLE, ActiveX, HTML)</td><td>Возможность проверки кода на синтаксическую корректность</td></tr>
<tr><td>Поддержка различных нотаций</td><td colspan="2">Возможность динамического моделирования событий в системе</td><td>Возможности определения бизнес-модели и бизнес-правил</td><td>Возможность генерировать код для 4GL и клиент-серверных продуктов.</td></tr>
<tr><td>Возможность генерации документации для печати</td><td colspan="2">Возможность динамической коррекции одной диаграммы из другой</td><td>Возможность связи с объектно-ориентированными базами данных и распределенными модулями(поддержка CORBA, DCOM, IIOP, HTML)</td><td></td></tr>
<tr><td colspan="5">Инфраструктура</td></tr>
<tr><td colspan="3">Контроль версий. Блокирование и согласование частей системы при групповой разработке</td><td>Репозиторий</td><td>Возможность реинжиниринга программного кода, 4GL, клиент-серверных систем в диаграммы моделей</td></tr>
</table>

Рис. 1. Идеальное объектно-ориентированное CASE-средство

Сравнительный анализ CASE-систем показывает, что на сегодняшний день одним из наиболее приближенных к идеальному варианту CASE-средств является семейство Rational Rose фирмы Rational Software Corporation. Следует отметить, что именно здесь работают авторы унифицированного языка моделирования Г. Буч, Д. Рамбо и И. Джекобсон, под руководством которых ведется разработка нового CASE-средства, поддерживающего UML.

Выделим основные критерии оценки и выбора CASE-средств.

1. Функциональные характеристики:
   - среда функционирования; проектная среда, программное обеспечение/технические средства, технологическая среда;
   - функции, ориентированные на фазы жизненного цикла: моделирование, реализация, тестирование;
   - общие функции: документирование, управление конфигурацией, управление проектом;

2. Надежность;
3. Простота использования;
4. Эффективность;
5. Сопровождаемость;
6. Переносимость;
7. Общие критерии (стоимость, затраты, эффект внедрения, характеристики поставщика).

Входной информацией для процесса оценки является:

- определение пользовательских потребностей;
- цели и ограничения проекта;
- данные о доступных CASE-средствах;
- список критериев, используемых в процессе оценки.

Элементы процесса включают:

- цели, предположения и ограничения, которые могут уточняться в ходе процесса;
- потребности пользователей, отражающие количественные и качественные требования пользователей к CASE-средствам;
- критерии, определяющие набор параметров, в соответствии с которыми производится оценка и принятие решения о выборе;
- формализованные результаты оценок одного или более средств;
- рекомендуемое решение (обычно либо решение о выборе, либо дальнейшая оценка).

Процесс оценки и/или выбора может быть начат только тогда, когда лицо, группа или организация полностью определила для себя конкретные потребности и формализовала их в виде количественных и качественных требований в заданной предметной области. Термин "пользовательские требования" далее означает именно такие формализованные требования.

Определение списка критериев основано на пользовательских требованиях и включает:

- выбор критериев для использования из приведенного далее перечня;
- определение дополнительных критериев;
- определение области использования каждого критерия (оценка, выбор или оба процесса);
- определение одной или более метрик для каждого критерия оценки;
- назначение веса каждому критерию при выборе.

Критерии оценки и выбора

Критерии формируют базис для процессов оценки и выбора и могут принимать различные формы, включая:

- числовые меры в широком диапазоне значений, например, объем требуемой памяти;
- числовые меры в ограниченном диапазоне значений, например, простота освоения, выраженная в баллах от 1 до 5;
- двоичные меры (истина/ложь, да/нет);
- меры, которые могут принимать одно или более из конечных множеств значений, например, платформы, для которых поддерживается CASE-средство.

Типичный процесс оценки и/или выбора может использовать набор критериев различных типов.

Каждый критерий должен быть выбран и адаптирован экспертом с учетом особенностей конкретного процесса. В большинстве случаев только некоторые из множества описанных ниже критериев оказываются приемлемыми для использования, при этом также добавляются дополнительные критерии. Выбор и уточнение набора используемых критериев является критическим шагом в процессе оценки и/или выбора.

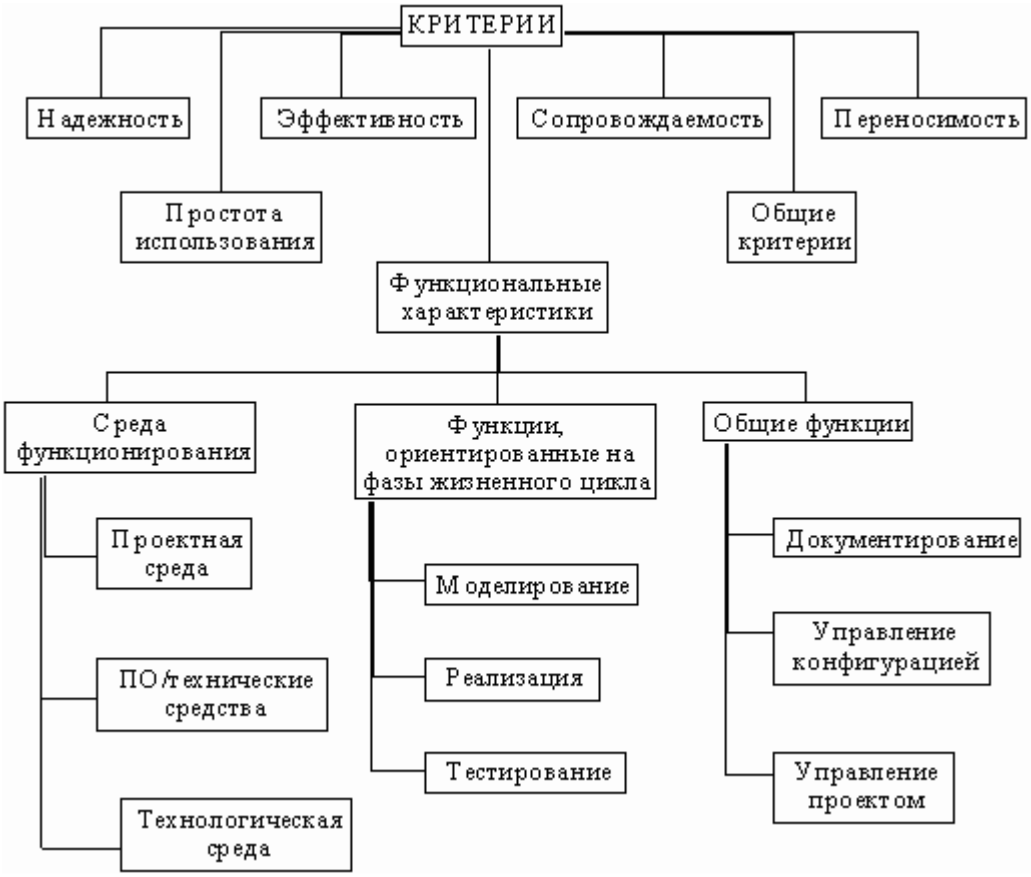


Рис. 2. Функциональные характеристики

Критерии первого класса предназначены для определения функциональных характеристик CASE-средства. Они в свою очередь подразделяются на ряд групп и подгрупп.

1. Среда функционирования:

Проектная среда:

- *поддержка процессов жизненного цикла.* Определяет набор процессов ЖЦ, которые поддерживает CASE-средство. • *область применения.* Примерами являются системы обработки транзакций, системы реального времени, информационные системы и т.д.
- *размер поддерживаемых приложений.* Определяет ограничения на такие величины, как количество строк кода, уровней вложенности, размер базы данных, количество элементов данных, количество объектов конфигурационного управления.

ПО/технические средства:

- *требуемые технические средства.* Оборудование, необходимое для функционирования CASE-средства, включая тип процессора, объем оперативной и дисковой памяти.
- *поддерживаемые технические средства.* Элементы оборудования, которые могут использоваться CASE-средством, например, устройства ввода/вывода.
- *требуемое ПО.* ПО, необходимое для функционирования CASE-средства, включая операционные системы и графические оболочки.
- *поддерживаемое ПО.* Программные продукты, которые могут использоваться CASE-средством.

Технологическая среда:

- *соответствие стандартам технологической среды.* Такие стандарты касаются языка, базы данных, репозитория, коммуникаций, графического интерфейса пользователя, документации, разработки, управления конфигурацией, безопасности, стандартов обмена информацией и интеграции по данным, по управлению и по пользовательскому интерфейсу.

- *совместимость с другими средствами.* Способность к взаимодействию с другими средствами, включая непосредственный обмен данными (примерами таких средств являются текстовые процессоры, базы данных и другие CASE-средства). Возможность преобразования репозитория или его части в стандартный формат для обработки другими средствами.
- *поддерживаемая методология.* Набор методов и методик, поддерживаемых CASE-средством. Примерами являются структурный или объектно-ориентированный анализ и проектирование.
- *поддерживаемые языки.* Все языки, используемые CASE-средством. Примерами таких языков являются языки программирования (Кобол, Ада, С), языки баз данных и языки запросов (DDL, SQL), графические языки (Postscript, HPGL), языки спецификации проектных требований и интерфейсы операционных систем (языки управления заданиями).

2. Функции, ориентированные на фазы жизненного цикла:

Моделирование:

Данные критерии определяют способность выполнения функций, необходимых для спецификации требований к ПО и преобразованию их в проект:

- *построение диаграмм.* Возможность создания и редактирования диаграмм различных типов, представляющих интерес для пользователя.
- *графический анализ.* Возможность анализа графических объектов, а также хранения и представления проектной информации в графическом представлении. В большинстве случаев графические анализаторы интегрированы со средствами построения диаграмм.
- *ввод и редактирование спецификаций требований и проектных спецификаций.* К спецификациям такого рода относятся описания функций, данных, интерфейсов, структуры, качества, производительности, технических средств, среды, затрат и графиков.
- *язык спецификации требований и проектных спецификаций.* Возможность импорта, экспорта и редактирования спецификаций с использованием формального языка.
- *моделирование данных.* Возможность ввода и редактирования информации, описывающей элементы данных системы и их отношения.
- *моделирование процессов.* Возможность ввода и редактирования информации, описывающей процессы системы и их отношения.
- *проектирование архитектуры ПО.* Проектирование логической структуры ПО - структуры модулей, интерфейсов и др.
- *имитационное моделирование.* Возможность динамического моделирования различных аспектов функционирования системы на основе спецификаций требований и/или проектных спецификаций, включая внешний интерфейс и производительность (например, время отклика, коэффициент использования ресурсов и пропускную способность).
- *прототипирование.* Возможность проектирования и генерации предварительного варианта всей системы или ее отдельных компонент на основе спецификаций требований и/или проектных спецификаций.
- *генерация экранных форм.* Возможность генерации экранных форм на основе спецификаций требований и/или проектных спецификаций.
- *возможность трассировки.* Возможность сквозного анализа функционирования системы от спецификации требований до конечных результатов (установления и отслеживания соответствий и связей между функциональными и другими внешними требованиями к ИС, техническими решениями и результатами проектирования). Прямая трассировка (проверка учета всех требований) и обратная трассировка (поиск проектных решений, не связанных ни с какими внешними требованиями).

- *синтаксический и семантический контроль проектных спецификаций.* Контроль синтаксиса диаграмм и типов их элементов, контроль декомпозиции функций, проверка спецификаций на полноту и непротиворечивость.
- *другие виды анализа.* Конкретные дополнительные виды анализа могут включать алгоритмы, потоки данных, нормализацию данных, использование данных, пользовательский интерфейс.
- *автоматизированное проектирование отчетов.*

Реализация:

Реализация затрагивает функции, связанные с созданием исполняемых элементов системы (программных кодов) или модификацией существующей системы. Многие из критериев зависят от конкретных языков.

Тестирование:

Критерии тестирования включают следующие:

- *описание тестов.* Типичные возможности включают генерацию тестовых данных, алгоритмов тестирования, требуемых результатов и т.д.
- *фиксация и повторение действий оператора.* Возможность фиксировать данные, вводимые оператором с помощью клавиатуры, мыши и т.д., редактировать их и воспроизводить в тестовых примерах.
- *автоматический запуск тестовых примеров.*
- *регрессионное тестирование.* Возможность повторения и модификации ранее выполненных тестов для определения различий в системе и/или среде.
- *автоматизированный анализ результатов тестирования.* Типичные возможности включают сравнение ожидаемых и реальных результатов, сравнение файлов, статистический анализ результатов и др.
- *анализ тестового покрытия.* Оснащенность средствами контроля исходного кода и анализ тестового покрытия. Проверяются, в частности, обращения к операторам, процедурам и переменным.
- *анализ производительности.* Возможность анализа производительности программ. Анализируемые параметры производительности могут включать использование центрального процессора, памяти, обращения к определенным элементам данных и/или сегментам кода, временные характеристики и т.д.
- *анализ исключительных ситуаций в процессе тестирования.*
- *динамическое моделирование среды.* В частности, возможность автоматически генерировать моделируемые входные данные системы.

3. Общие функции:

Приведенные ниже критерии определяют функции CASE-средств, охватывающие всю совокупность фаз ЖЦ. Поддержка всех этих функций осуществляется посредством репозитория.

Документирование:

- *редактирование текстов и графики.* Возможность вводить и редактировать данные в текстовом и графическом формате.
- *редактирование с помощью форм.* Возможность поддерживать формы, определенные пользователями, вводить и редактировать данные в соответствии с формами.
- *возможности издательских систем.*
- *поддержка функций и форматов гипертекста.*
- *соответствие стандартам документирования.*
- *автоматическое извлечение данных из репозитория и генерация документации по спецификациям пользователя.*

Управление конфигурацией:

- *контроль доступа и изменений.* Возможность контроля доступа на физическом уровне к элементам данных и контроля изменений. Контроль доступа включает возможности

определения прав доступа к компонентам, а также извлечения элементов данных для модификации, блокировки доступа к ним на время модификации и помещения обратно в репозиторий.

- *отслеживание модификаций.* Фиксация и ведение журнала всех модификаций, внесенных в систему в процессе разработки или сопровождения.
- *управление версиями.* Ведение и контроль данных о версиях системы и всех ее коллективно используемых компонентах.
- *учет состояния объектов конфигурационного управления.* Возможность получения отчетов о всех последовательных версиях, содержимом и состоянии различных объектов конфигурационного управления.
- *генерация версий и модификаций.* Поддержка пользовательского описания последовательности действий, требуемых для формирования версий и модификаций, и автоматическое выполнение этих действий.
- *архивирование.* Возможность автоматического архивирования элементов данных для последующего использования.

Управление проектом:

- *управление работами и ресурсами.* Контроль и управление процессом проектирования ИС в терминах структуры заданий и назначения исполнителей, последовательности их выполнения, завершенности отдельных этапов проекта и проекта в целом. Возможность поддержки плановых данных, фактических данных и их анализа. Типичные данные включают графики (с учетом календаря, рабочих часов, выходных и др.), компьютерные ресурсы, распределение персонала, бюджет и др.
- *оценка.* Возможность оценивать затраты, график и другие проектные параметры, вводимые пользователями.
- *управление процедурой тестирования.* Поддержка управления процедурами и программой тестирования, например, управления расписанием планируемых процедур, фиксация и запись результатов тестирования, генерация отчетов и т.д.
- *управление качеством.* Ввод соответствующих данных, их анализ и генерация отчетов.
- *корректирующие действия.* Поддержка управления корректирующими действиями, включая обработку сообщений о проблемных ситуациях.

9. Системы автоматизированного проектирования ИС: RAD-технологии

RAD (rapid application development) - быстрая разработка приложений. Это метод разработки систем, основанный на функциональной организации бригад. Представляет собой бригадную методологию разработки ИС.

Эти новые группы, или бригады, избавились от узости мышления, присущей иерархической организации; теперь они могли сосредоточиться на выполнении общей задачи и формировались из разнообразных специалистов, представляющих разные иерархические ветви организационной структуры.

Методология RAD использует:

- бригадные подходы, такие, как JAD;
- спиральную модель;
- такие понятия разработки, как тесные временные рамки и цикличность.

Методология JAD (joint application development) - совместная разработка приложений. Идея (Кроуфорд, Норрис - 1977 г.) - собрать всех заинтересованных участников и достичь консенсуса. Возникновение методологии JAD совпало по времени с переходом корпоративной Америки от организации с командно-административной иерархией к идеологии функциональных групп или бригад.

Еще один фактор, заставивший ИТ-сообщество обратиться к использованию методологии JAD - формирование более искушенных пользователей. По мере того, как ПК получали

все большее распространение, бизнес-клиенты становились технологически все более образованными.
Применение методов JAD служит гарантией того, что ИТ-специалисты работают над проектом совместно с организацией-заказчиком. Т.о., посредством применения процессов JAD осуществляется раскрытие стратегических информационных потребностей организации для ответа на жизненно важные вопросы, связанные с ведением бизнеса.
Тесные временные рамки представляют собой один из способов управления проектным бюджетом за счет ограничения масштабов проекта и привязки нужд заказчиков к проектным требованиям. Именно этой цели и служат ограничения, накладываемые на продолжительность каждого цикла. Тесные временные рамки концентрируют бригаду разработчиков на приоритетах проекта, а также препятствуют «размыванию» границ проекта и стимулируют распределение функциональных приоритетов, устанавливая дату, до которой необходимо завершить изготовление комплекта поставки. При этом требования с более низким приоритетом могут быть выполнены в следующих итерациях.
Цикличность означает, что поставляемое изделие совершенствуется в течение нескольких фаз.
Примечание Для хорошего плана характерно наличие нескольких промежуточных результатов в виде прототипов или отдельных поставляемых заказчику компонентов, представляющих реальную ценность для организации. Еще одна отличительная черта хорошего плана и умелого управления проектом - возможность получить готовые компоненты прежде, чем существенно изменится модель данных.
Т.о., успешный проект по моделированию корпоративных данных обычно завершается вместе с созданием моделей для части предприятия.
Весь путь построения приложения складывается из 3-х итерактивных этапов:

1) Этап выбора проекта, сферы применения системы и предметной области автоматизации.
2) Этап пилотного проекта или проверка концепции.
3) Этап изготовления.

На каждом шаге этих 3-х этапов применяется подход, который условно можно обозначить как «планирование, выполнение, проверка, действие».

1. Сначала создается план для каждого этапа.
2. Проектная бригада осуществляет разработку поставляемого изделия в соответствии с планом.
3. Несколько раз в ходе реализации плана достигнутый прогресс и состояние изделия сравниваются с запланированными.
4. На основании результатов проверки бригада исправляет упущения в плане или устраняет дефекты, возникшие в процессе его выполнения.

10. Системный подход к процессу проектирования. Принципы и средства структурного анализа

Идеи, лежащие в основе структурных методов
Методы структурного анализа и проектирования стремятся преодолеть сложность больших систем путем расчленения их на части ("черные ящики") и иерархической организации этих черных ящиков. Выгода в использовании черных ящиков заключается в том, что их пользователю не требуется знать, как они работают, необходимо знать лишь его входы и выходы, а также его назначение (т.е. функцию, которую он выполняет).
В окружающем нас мире черные ящики встречаются в большом количестве.
Проиллюстрируем преимущества систем, составленных из них, на примере музыкального центра.

- *Конструирование системы черных ящиков существенно упрощается.* Намного легче разработать магнитофон или проигрыватель, если не беспокоиться о создании встроенного усилительного блока.

- *Облегчается тестирование таких систем.* Если появляется плохой звук одной из колонок, можно поменять колонки местами. Если неисправность переместилась с колонкой, то именно она подлежит ремонту; если нет, тогда проблема в усилителе, магнитофоне или местах их соединения.
- *Имеется возможность простого ре-конфигурирования системы черных ящиков.* Если колонка неисправна, то Вы можете отправить ее в ремонтную мастерскую, а сами пока продолжать слушать свои записи в моно-режиме.
- *Облегчается доступность для понимания и освоения.* Можно стать специалистом по магнитофонам без углубленных знаний о колонках.
- *Увеличивается удобство при модификации.* Вы можете приобрести колонки более высокого качества и более мощный усилитель, но это совсем не означает, что Вам необходим больших размеров проигрыватель.

Таким образом, первым шагом упрощения сложной системы является ее разбиение на черные ящики, при этом такое разбиение должно удовлетворять следующим критериям:

- каждый черный ящик должен реализовывать единственную функцию системы;
- функция каждого черного ящика должна быть легко понимаема независимо от сложности ее реализации (например, в системе управления ракетой может быть черный ящик для расчета места ее приземления: несмотря на сложность алгоритма, функция черного ящика очевидна - "расчет точки приземления");
- связь между черными ящиками должна вводиться только при наличии связи между соответствующими функциями системы (например, в бухгалтерии один черный ящик необходим для расчета общей заработной платы служащего, а другой для расчета налогов - необходима связь между этими черными ящиками: размер заработанной платы требуется для расчета налогов);
- связи между черными ящиками должны быть простыми, насколько это возможно, для обеспечения независимости между ними.

Второй важной идеей, лежащей в основе структурных методов, является идея иерархии. Для того, чтобы понять сложную систему недостаточно разбить ее на части, необходимо эти части организовать определенным образом, а именно в виде иерархических структур. Все сложные системы Вселенной организованы в иерархии. Да и сама она включает галактики, звездные системы, планеты, ..., молекулы, атомы, элементарные частицы. Человек при создании сложных систем также подражает природе. Любая организация имеет директора, заместителей по направлениям, иерархию руководителей подразделений, рядовых служащих.

Наконец, третий момент: структурные методы широко используют графические нотации, также служащие для облегчения понятия сути сложных систем. Известно, что "одна картинка стоит тысячи слов".

Структурные методы также позволяют дополнить "картинки" любой информацией, которая не может быть отражена при использовании соответствующей графической нотации.

Принципы структурного анализа

Анализ требований разрабатываемой системы является важнейшим среди всех этапов ЖЦ. Он оказывает существенное влияние на все последующие этапы, являясь в то же время наименее изученным и понятным процессом. На этом этапе, во-первых, необходимо понять, что предполагается сделать, а во-вторых, задокументировать это, т.к. если требования не зафиксированы и не сделаны доступными для участников проекта, то они вроде бы и не существуют. При этом язык, на котором формулируются требования, должен быть достаточно прост и понятен заказчику.

Во многих аспектах системный анализ является наиболее трудной частью разработки. Нижеследующие проблемы, с которыми сталкивается системный аналитик, взаимосвязаны (и это является одной из главных причин их трудной разрешимости):

- аналитику сложно получить исчерпывающую информацию для оценки требований к системе с точки зрения заказчика;
- заказчик, в свою очередь, не имеет достаточной информации о проблеме обработки данных для того, чтобы судить, что является выполнимым, а что нет;
- аналитик сталкивается с чрезмерным количеством подробных сведений, как о предметной области, так и о новой системе;
- спецификация системы из-за объема и технических терминов часто непонятна для заказчика;
- в случае понятности спецификации для заказчика, она будет являться недостаточной для проектировщиков и программистов, создающих систему.

Конечно, применение известных аналитических методов снимает некоторые из перечисленных проблем анализа, однако эти проблемы могут быть существенно облегчены за счет применения современных структурных методов, среди которых центральное место занимают методологии структурного анализа.

Структурным анализом принято называть метод исследования системы, которое начинается с ее общего обзора и затем детализируется, приобретая иерархическую структуру со все большим числом уровней. Для таких методов характерно:

- разбиение на уровни абстракции с ограничением числа элементов на каждом из уровней (обычно от 3 до 6-7);
- ограниченный контекст, включающий лишь существенные на каждом уровне детали; дуальность данных и операций над ними;
- использование строгих формальных правил записи;
- последовательное приближение к конечному результату.

Все методологии структурного анализа базируются на ряде общих принципов, часть из которых регламентирует организацию работ на начальных этапах ЖЦ, а часть используется при выработке рекомендаций по организации работ. В качестве двух базовых принципов используются следующие: принцип "**разделяй и властвуй**" и принцип **иерархического упорядочивания**.

Принцип "**разделяй и властвуй**" является принципом решения трудных проблем путем разбиения их на множество меньших независимых задач, легких для понимания и решения.

Принцип **иерархического упорядочивания** декларирует, что устройство этих частей также существенно для понимания. Понимаемость проблемы резко повышается при организации ее частей в древовидные иерархические структуры, т.е. система может быть понята и построена по уровням, каждый из которых добавляет новые детали.

Однако стоит отметить, что остальные принципы структурного анализа является не менее важными. Отметим основные принципы:

1) **Принцип абстрагирования** заключается в выделении существенных с некоторых позиций аспектов системы и в отвлечении от несущественных с целью представления проблемы в простом общем виде.

2) **Принцип формализации** заключается в необходимости строгого методического подхода к решению проблемы.

3) **Принцип упрятывания** заключается в упрятывании несущественной на конкретном этапе информации: каждая часть "знает" только необходимую ей информацию.

4) **Принцип концептуальной общности** заключается в следовании единой философии на всех этапах ЖЦ (структурный анализ - структурное проектирование - структурное программирование - структурное тестирование).

5) **Принцип полноты** заключается в контроле на присутствие лишних элементов.

6) **Принцип непротиворечивости** заключается в обоснованности и согласованности элементов.

7) **Принцип логической независимости** заключается в концентрации внимания на логическом проектировании для обеспечения независимости от физического проектирования.
8) **Принцип независимости данных** заключается в том, что модели данных должны быть проанализированы и спроектированы независимо от процессов их логической обработки, а также от их физической структуры и распределения.
9) **Принцип структурирования данных** заключается в том, что данные должны быть структурированы и иерархически организованы.
10) **Принцип доступа конечного пользователя** заключается в том, что пользователь должен иметь средства доступа к базе данных, которые он может использовать непосредственно (без программирования).
Соблюдение указанных принципов необходимо при организации работ на начальных этапах ЖЦ независимо от типа разрабатываемого ПО и используемых при этом методологий.

Средства структурного анализа и их взаимоотношения

Для целей моделирования систем вообще, и структурного анализа в частности, используются три группы средств, иллюстрирующих:

- функции, которые система должна выполнять;
- отношения между данными;
- зависящее от времени поведение системы (аспекты реального времени).

Среди всего многообразия средств решения данных задач в методологиях структурного анализа наиболее часто и эффективно применяемыми являются следующие:

- DFD (Data Flow Diagrams) - **диаграммы потоков данных** совместно со **словарями и спецификациями процессов или миниспецификациями**;
- ERD (Entity-Relationship Diagrams) - **диаграммы "сущность-связь"**;
- STD (State Transition Diagrams) - **диаграммы переходов состояний**.

Все они содержат графические и текстовые средства моделирования: первые - для удобства демонстрирования основных компонент модели, вторые - для обеспечения точного определения ее компонент и связей.

11. SADT - технология структурного анализа и проектирования

Методологии структурного и системного анализа и проектирования

Методология структурного анализа и проектирования ПО определяет шаги работы, которые должны быть выполнены, их последовательность, правила распределения и назначения операций и методов.
В настоящее время успешно используются такие методологии, как **SADT** (Structure Analysis and Design Technique**), структурный системный анализ Гейна-Сарсона**, **структурный анализ и проектирование Йодана/Де Марко**, развитие **систем Джексона** и другие.
Перечисленные структурные методологии жестко регламентируют фазы анализа требований и проектирования спецификаций.
Несмотря на достаточно широкий спектр используемых методов и диаграммных техник, большинство методологий базируется на следующей "классической" совокупности:

- Диаграммы потоков данных в нотации Йодана/Де Марко или Гейна-Сарсона, обеспечивающие анализ требований и функциональное проектирование информационных систем;
- Расширения Хатли и Уорда-Меллора для проектирования систем реального времени, основанные на диаграммах переходов состояний, таблицах решений, картах и схемах потоков управления;
- Диаграммы "сущность-связь" (в нотации Чена или Баркера) для проектирования структур данных, схем БД, форматов файлов как части всего проекта;

- Структурные карты Джексона и/или Константайна для проектирования межмодульных взаимодействий и внутренней структуры модулей.

Разработка ПО основана на модели ВХОД-ОБРАБОТКА-ВЫХОД: данные входят в систему, обрабатываются или преобразуются и выходят из системы. Такая модель используется во всех структурных методологиях. При этом важен порядок построения модели.

Традиционный **процедурно-ориентированный** подход регламентирует первичность проектирования функциональных компонент по отношению к проектированию структур данных: требования к данных раскрываются через функциональные требования.

При подходе, **ориентированном на данные**, вход и выход являются наиболее важными - структуры данных определяются первыми, а процедурные компоненты являются производными от данных.

SADT - технология структурного анализа и проектирования

SADT - одна из самых известных методологий анализа и проектирования информационных систем, введенная в 1973 году Россом.

С точки зрения SADT модель может основываться либо на функциях системы, либо на ее предметах (планах, данных, оборудовании, информации и т.д.). Соответствующие модели принято называть **функциональными моделями** и **моделями данных**.

Функциональная модель представляет с нужной степенью подробности систему активностей, которые в свою очередь отражают свои взаимоотношения через предметы системы.

Модели данных представляют собой подробное описание предметов системы. Полная методология SADT заключается в построении моделей обеих типов для более точного описания сложной системы. Однако в настоящее время широкое применение нашли только функциональные модели.

Методология SADT представляет собой совокупность методов, правил и процедур, предназначенных для построения функциональной модели объекта какой-либо предметной области. Функциональная модель SADT отображает функциональную структуру объекта, т.е. производимые им действия и связи между этими действиями. Основные элементы этой методологии основываются на следующих концепциях:

- графическое представление блочного моделирования. Графика блоков и дуг SADT-диаграммы отображает функцию в виде блока, а интерфейсы входа/выхода представляются дугами, соответственно входящими в блок и выходящими из него. Взаимодействие блоков друг с другом описываются посредством интерфейсных дуг, выражающих "ограничения", которые в свою очередь определяют, когда и каким образом функции выполняются и управляются;
- строгость и точность. Выполнение правил SADT требует достаточной строгости и точности, не накладывая в то же время чрезмерных ограничений на действия аналитика.

Правила SADT включают:

- ограничение количества блоков на каждом уровне декомпозиции (правило 3-6 блоков);
- связность диаграмм (номера блоков);
- уникальность меток и наименований (отсутствие повторяющихся имен);
- синтаксические правила для графики (блоков и дуг);
- разделение входов и управлений (правило определения роли данных).
- отделение организации от функции, т.е. исключение влияния организационной структуры на функциональную модель.

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

Диаграммы - главные компоненты модели, все функции ИС и интерфейсы на них представлены как **блоки** и **дуги**. Место соединения дуги с блоком определяет тип интерфейса. Управляющая информация входит в блок сверху, в то время как информация, которая подвергается обработке, показана с левой стороны блока, а результаты выхода

показаны с правой стороны. Механизм (человек или автоматизированная система), который осуществляет операцию, представляется дугой, входящей в блок снизу (рис. 1).

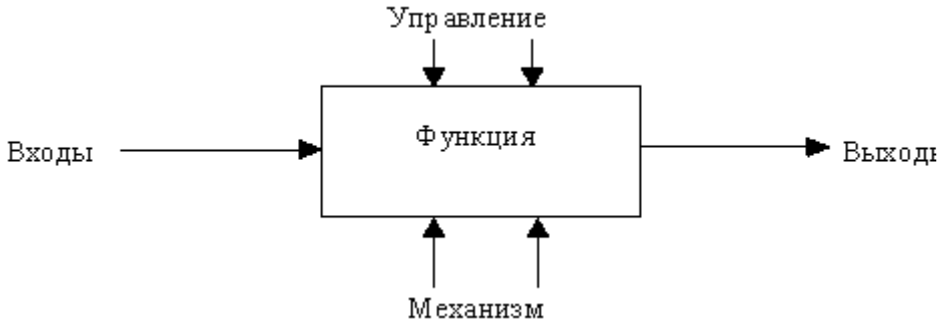


Рис. 1. Функциональный блок и интерфейсные дуги

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

На рис.2. , где приведены четыре диаграммы и их взаимосвязи, показана структура SADT-модели. Каждый компонент модели может быть декомпозирован на другой диаграмме. Каждая диаграмма иллюстрирует "внутреннее строение" блока на родительской диаграмме.

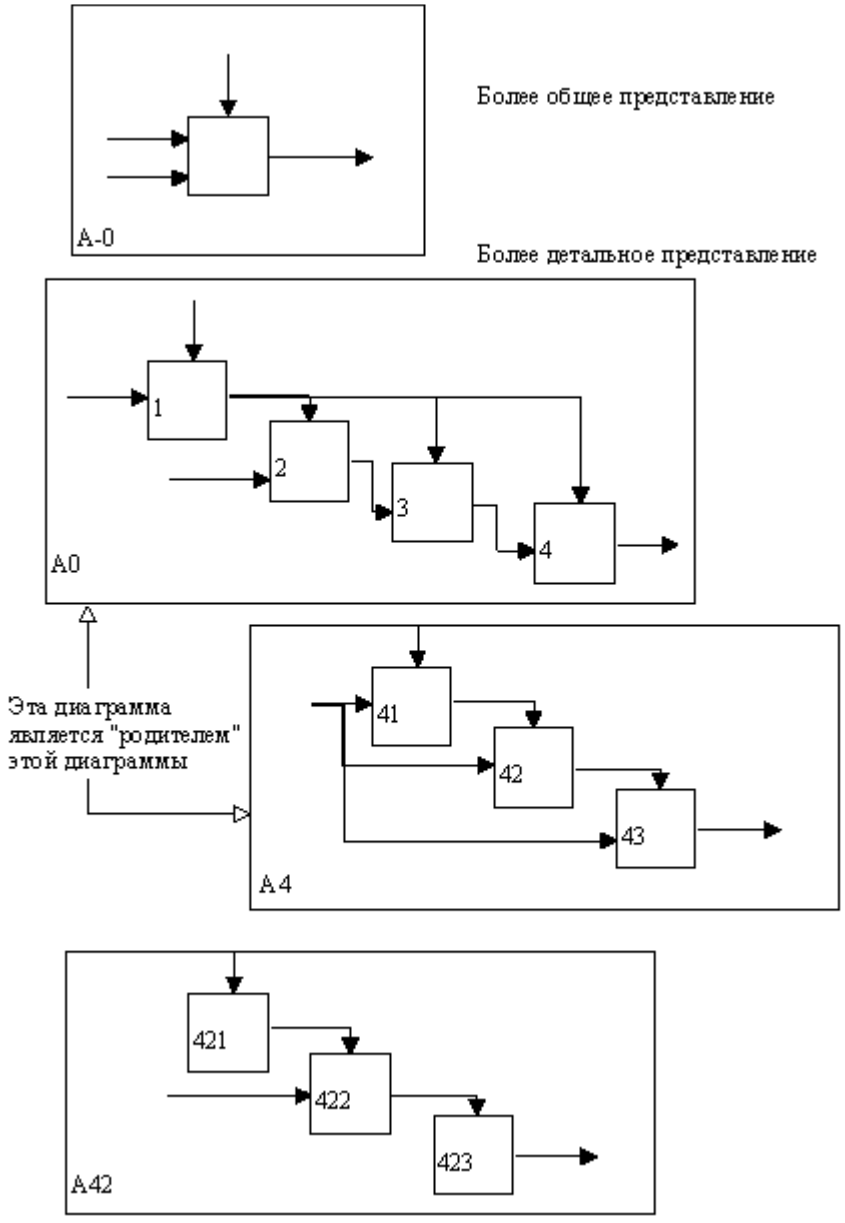


Рис. 2.. Структура SADT-модели. Декомпозиция диаграмм

Иерархия диаграмм

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

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

Во всех случаях каждая подфункция может содержать только те элементы, которые входят в исходную функцию. Кроме того, модель не может опустить какие-либо элементы, т.е., как уже отмечалось, родительский блок и его интерфейсы обеспечивают контекст. К нему нельзя ничего добавить, и из него не может быть ничего удалено.

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

Дуги, входящие в блок и выходящие из него на диаграмме верхнего уровня, являются точно теми же самыми, что и дуги, входящие в диаграмму нижнего уровня и выходящие из нее, потому что блок и диаграмма представляют одну и ту же часть системы.

На рис.3-5 представлены различные варианты выполнения функций и соединения дуг с блоками.

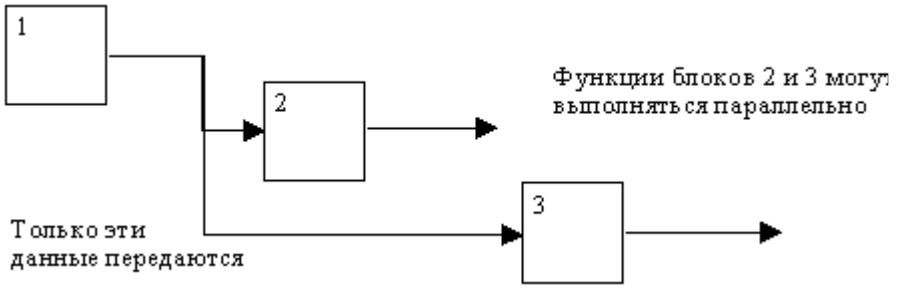


Рис. 3. Одновременное выполнение

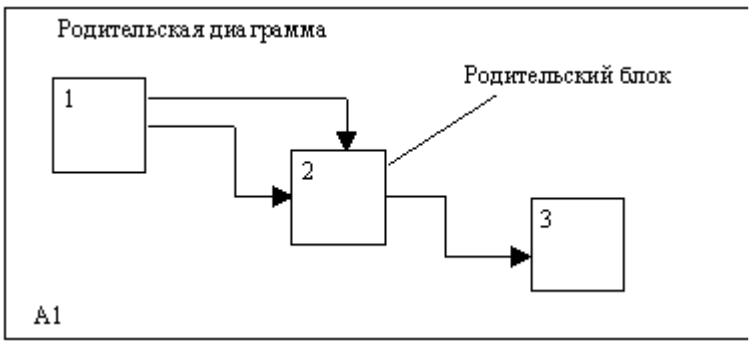


Рис. 4. Соответствие должно быть полным и непротиворечивым

Некоторые дуги присоединены к блокам диаграммы обоими концами, у других же один конец остается неприсоединенным. Неприсоединенные дуги соответствуют входам, управлениям и выходам родительского блока. Источник или получатель этих пограничных дуг может быть обнаружен только на родительской диаграмме. Неприсоединенные концы должны соответствовать дугам на исходной диаграмме. Все граничные дуги должны продолжаться на родительской диаграмме, чтобы она была полной и непротиворечивой.

На SADT-диаграммах не указаны явно ни последовательность, ни время. Обратные связи, итерации, продолжающиеся процессы и перекрывающиеся (по времени) функции могут быть изображены с помощью дуг. Обратные связи могут выступать в виде комментариев, замечаний, исправлений и т.д. (рис.5.).

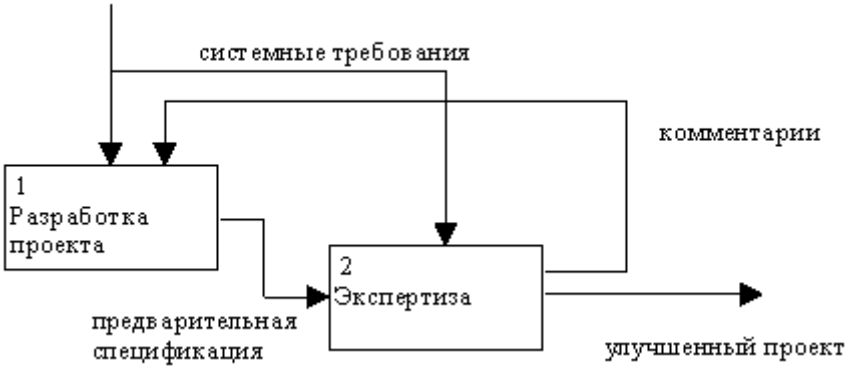


Рис. 5. Пример обратной связи

Как было отмечено, механизмы (дуги с нижней стороны) показывают средства, с помощью которых осуществляется выполнение функций. Механизм может быть человеком, компьютером или любым другим устройством, которое помогает выполнять данную функцию (рис.6.).

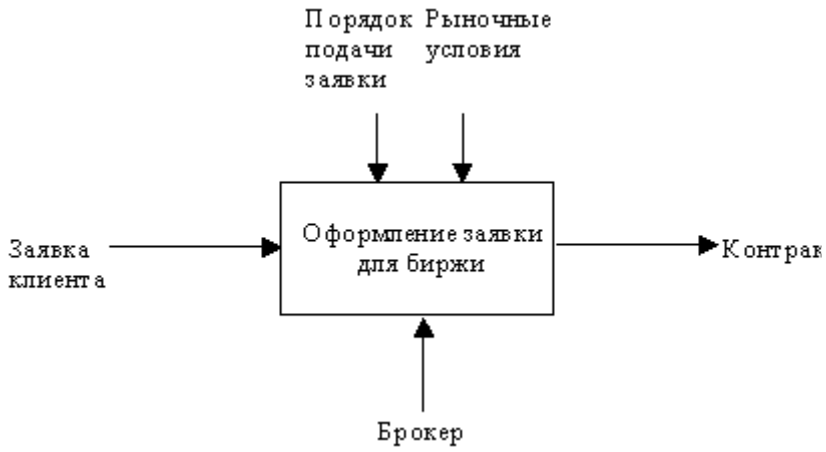


Рис. 6.. Пример механизма

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

Для того, чтобы указать положение любой диаграммы или блока в иерархии, используются номера диаграмм. Например, A21 является диаграммой, которая детализирует блок 1 на диаграмме A2. Аналогично, A2 детализирует блок 2 на диаграмме A0, которая является самой верхней диаграммой модели. На рис.7 показано типичное дерево диаграмм.

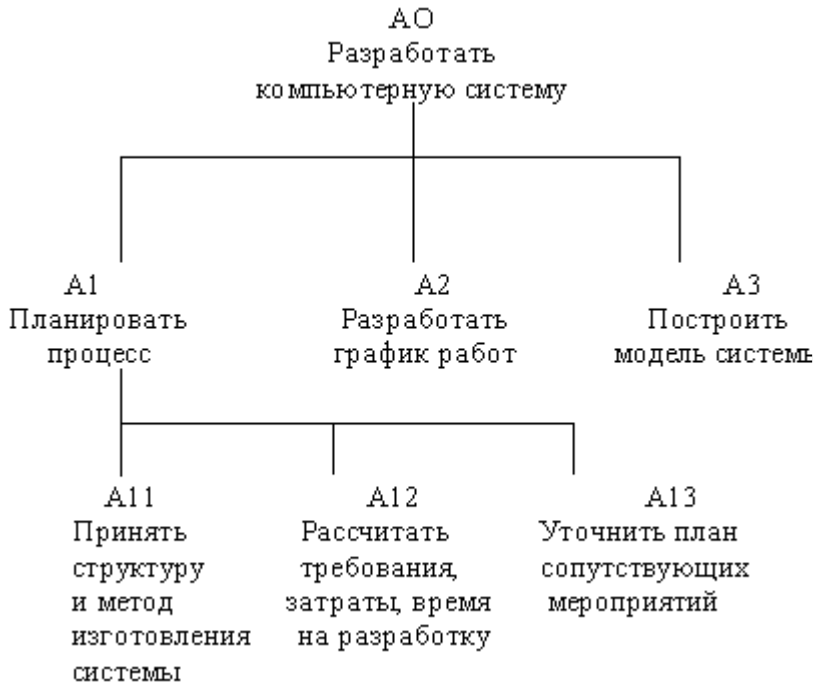


Рис. 7. Иерархия диаграмм

Синтаксис диаграмм

Как уже отмечалось, каждая SADT-диаграмма содержит блоки и дуги. Блоки изображают функции системы, дуги связывают блоки и отображают взаимодействия и взаимосвязи между ними. Диаграмме дается название, которое располагается внизу.

Поскольку блоки отображают функции системы, то в названии блоков используют глаголы или глагольные обороты (рассчитать, выполнить задание и т.д.).

Функциональные блоки изображаются прямоугольниками. Блоки на SADT-диаграмме никогда не размещаются случайным образом. Они размещаются по степени важности, как ее понимает автор диаграммы. В SADT этот относительный порядок называется ***доминированием***. Доминирование понимается как влияние, которое один блок оказывает на другие блоки диаграммы. Например, самым доминирующим блоком диаграммы может быть первый из требуемой последовательности функций. Наиболее доминирующий блок обычно располагается в верхнем левом углу диаграммы, а наименее доминирующий - в правом нижнем углу диаграммы. В результате имеем ступенчатую схему.

Таким образом, SADT-диаграмма составлена из блоков, связанных дугами, которые определяют, как блоки влияют друг на друга. Это влияние может выражаться либо в передаче выходной информации к другой функции для дальнейшего преобразования, либо в выработке управляющей информации, предписывающей, что именно должна выполнять другая функция. Поэтому SADT-диаграммы нельзя отнести к блок-схемам или диаграммам потоков данных. Скорее всего, это предписывающие диаграммы, представляющие входные-выходные преобразования и указывающие правила этих преобразований.

В методологии SADT требуется только пять типов взаимосвязей между блоками для описания их отношений: управление, вход, обратная связь по управлению, обратная связь по входу, выход-механизм. Рассмотрим на примере.

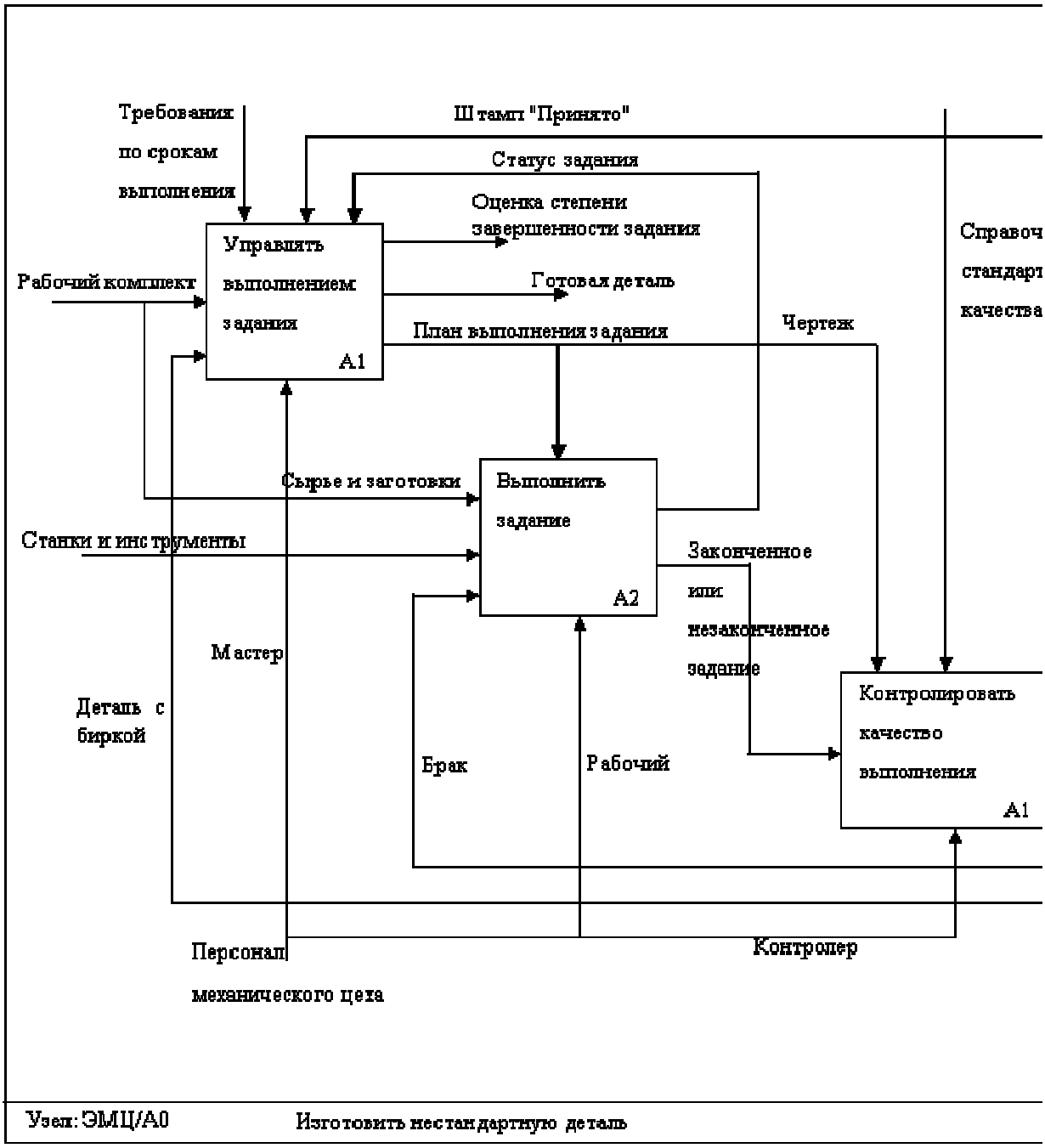


Рис. 8. Пример SADT-диаграммы

Связи по управлению и входу являются простейшими, поскольку они отражают прямые воздействия, которые интуитивно понятны и очень просты. Отношение управления возникает тогда, когда выход одного блока непосредственно влияет на блок с меньшим доминированием. Пример: блок "управлять выполнением задания" влияет на блок "выполнить задание" в соответствии с планом выполнения задания.

Отношение входа возникает тогда, когда выход одного блока становится входом для блока с меньшим доминированием. Пример: выход "законченное или незаконченное задание" является входом функции "контролировать качество выполнения" при выполнении функции "выполнить задание".

Обратная связь по управлению и обратная связь по входу являются более сложными, поскольку они представляют собой итерацию или рекурсию. А именно выходы одной функции влияют на будущее выполнение других функций, которые в последствии влияют на исходную функцию.

Обратная связь по управлению возникает тогда, когда выход некоторого блока влияет на блок с большим доминированием. Пример: функция "управлять выполнением задания" ограничивает действие функции "контролировать качество выполнения" с помощью чертежа, в котором указаны разрешенные допуски. Кроме того, дуга штамп "принято", являющаяся выходом блока "контролировать качество выполнения" организует работу блока "управлять выполнением задания", поскольку именно штамп "принято" указывает, что задание завершено. Таким образом, штамп "принято" влияет на будущую деятельность блока "управлять выполнением задания", поэтому соответствующая дуга направлена назад. Связь по входной обратной связи имеет место тогда, когда выход одного блока становится входом другого блока с большим доминированием. Пример: задания, отвергнутые функцией "контролировать качество выполнения", отсылаются на вход блока "выполнить задание" в качестве брака.

Синтаксис моделей и работа с ними

Понятие цели системы. SADT-модель дает полное, точное и адекватное описание системы, имеющее конкретное назначение. Это назначение называют **целью** системы. Таким образом, целью модели является получение ответов на совокупность вопросов. Если модель отвечает не на все вопросы или ее ответы недостаточно точны, то говорят о том, что модель не достигла своей цели. Определяя модель таки образом, SADT закладывает основы практического моделирования.

Точка зрения модели.

С определением модели тесно связана позиция, с которой наблюдается система и создается ее модель. Эта позиция и называется "точкой зрения" данной модели. "Точку зрения" лучше всего представлять себе как место (позицию) человека или объекта, в которое надо встать, чтобы увидеть систему в действии.

Построим контекстную диаграмму модели изготовления нестандартной детали (рис.9). Определим для начала цель и точку зрения модели.

Цель: определить, какие функции должны быть включены в процесс изготовления нестандартной детали и как эти функции взаимосвязаны между собой.

Точка зрения: лучше всего описать все функции может начальник цеха, в котором изготавливаются нестандартные детали.

SADT-модели развиваются в процессе структурной декомпозиции сверху вниз. Сначала декомпозируется один блок, являющийся границей модели. Название диаграммы совпадает с названием декомпозируемого блока. В методологии SADT идентифицируется каждая диаграмма данной модели посредством того, что называется "номер узла". Номер узла для контекстной диаграммы имеет следующий вид: название модели или аббревиатура, косая черта, заглавная буква A (activity в функциональных диаграммах), дефис и ноль.

Номером узла диаграммы, декомпозирующей контекстную диаграмму, является тот же номер узла, но без дефиса.

Создание функциональных моделей и диаграмм

Сбор информации

Рассмотрим методы, которые использует SADT-аналитик для изучения предметной области и технологии получения от экспертов сведений о системе, подлежащих описанию. Обычно на практике эту технологию называют сбором данных, а в информатике она известна как опрос (интервьюирование) или извлечение знаний.

Обычно источниками информации служат эксперты. Существует множество различных стратегий для извлечения информации из этих источников. Наиболее используемые стратегии:

- Чтение документов;
- Наблюдение за выполняемыми операциями;
- Анкетирование;
- Использование собственных знаний;
- Составление описания.

Документы - наиболее хороший источник информации, потому что они чаще всего доступны и их можно "опрашивать" в удобном для себя темпе. Чтение документов - прекрасный способ получить первоначальное представление о системе и сформулировать вопросы к экспертам.
Наблюдение за работой моделируемой системы - хорошая стратегия получения информации. Во время работы системы очень часто возникают вопросы, которые никогда бы не появились в результате чтения документов или разговоров с экспертами.
Анкетирование проводится для того, чтобы опросить большие группы экспертов в сжатые сроки. Анкетирование при опросе экспертов позволяет выявить, какие части системы более всего нуждаются в улучшении.

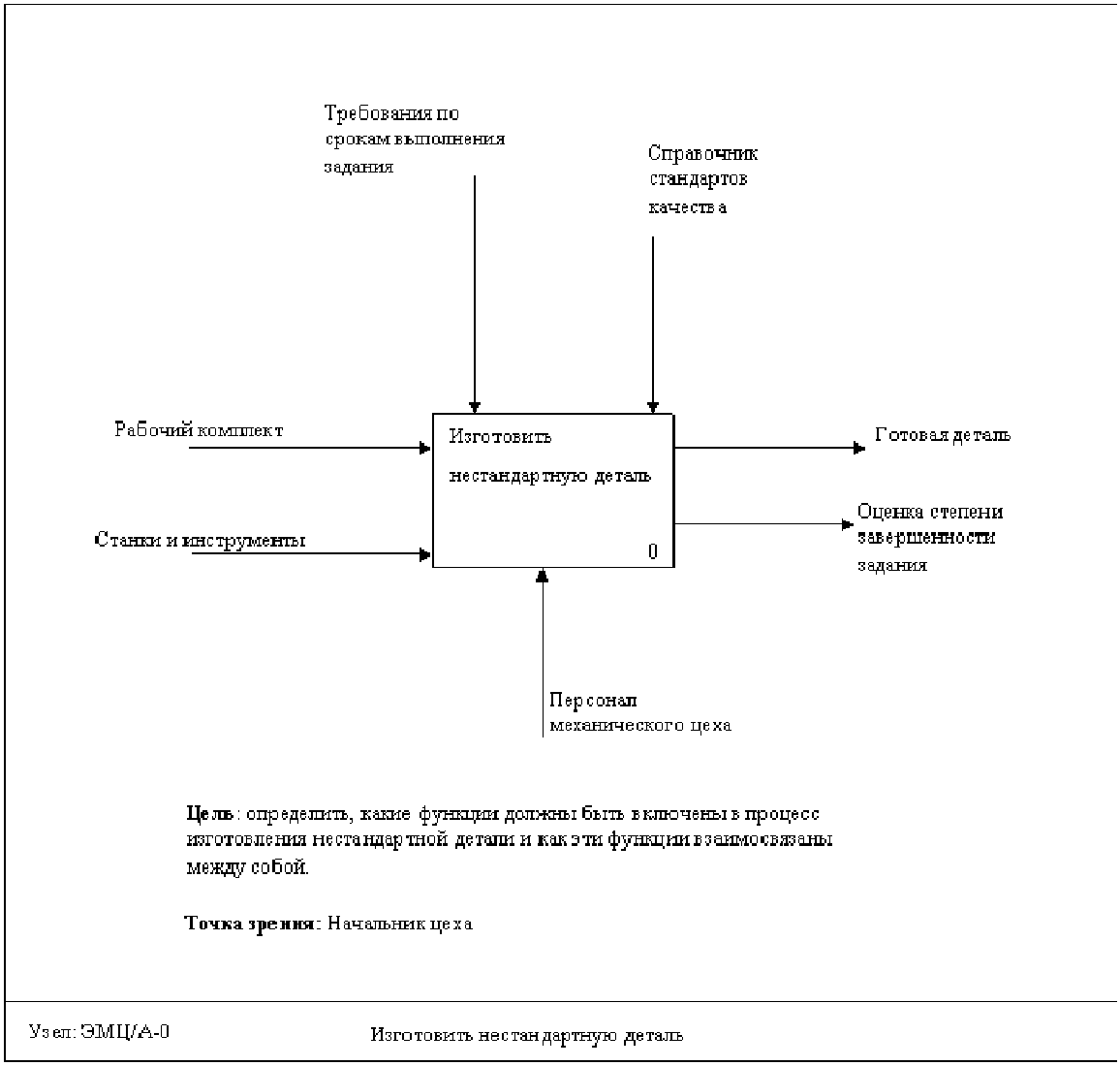


Рис.9. Контекстная диаграмма

Использование собственных знаний чаще всего доступно очень опытным аналитикам, которые исследовали большое число систем определенного типа, а потому они обладают фундаментальными знаниями в соответствующей предметной области.
Еще одна полезная стратегия - придумать описание и дать его экспертам для корректировки. Придуманные описания могут дать альтернативные схемы функционирования системы - схемы.

Дополнения к диаграммам и моделям

Диаграммы законченной SADT-модели упорядоченно организуют все важные компоненты и детали системы. Опытные аналитики создают различные дополнения. Дополнения и

уточнения, которые не входят в сами диаграммы, обогащают информационное содержание модели. SADT-диаграммы могут быть дополнены информацией в виде текстов, рисунков и глоссариев.
Текст обычно представляет собой рассказ об одной из части диаграммы.
Рисунки - это картинки, поясняющие отдельные моменты.
Глоссарий - набор определений объектов и функций, представленных на диаграмме. Глоссарий используется для того, чтобы собрать вместе и определить новые понятия, которые вводятся диаграммой, декомпозирующей блок, особенно если это первая декомпозиция родительского блока. Для функциональных SADT-диаграмм такими понятиями могут быть либо новые функции, либо новые объекты, представляемые дугами, либо декомпозиция внешних дуг.

12. DFD - функциональная методика потоков данных

На сегодняшний день существует два основных подхода к разработке информационных систем, отличающихся критериями декомпозиции. Первый подход, получивший название функционально-модульного или структурного, определяется принципом алгоритмической декомпозиции. В соответствии с этим принципом осуществляется разделение функций ИС на модули по функциональной принадлежности, и каждый модуль реализует один из этапов общего процесса. Такой традиционный функционально-модульный подход к проектированию ИС, получивший название модель «водопада», предусматривает строго последовательный порядок действий.
Диаграммы потоков данных (DFD - Data Flow Diagram**)** являются основным средством моделирования ***функциональных требований*** проектируемой системы. Целью методики является построение модели рассматриваемой системы в виде **диаграммы потоков данных**, обеспечивающей правильное описание выходов (отклика системы в виде данных) при заданном воздействии на вход системы (подаче сигналов через внешние интерфейсы). С их помощью функциональные требования разбиваются на ***функциональные компоненты*** (процессы) и представляются в виде сети, связанной потоками данных. Главная цель таких средств - продемонстрировать, как каждый процесс преобразует свои входные данные в выходные, а также выявить отношения между этими процессами.
Логическая модель DFD показывает внешние по отношению к системе источники (адресаты) данных, идентифицирует логические функции (процессы) и группы элементов данных, связывающие одну функцию с другой (потоки), а также идентифицирует хранилища (накопители) данных, к которым осуществляется доступ. Структуры потоков данных и определения их компонент хранятся и анализируются в **словаре данных**.
Каждая логическая функция (процесс) может быть детализирована с помощью DFD нижнего уровня; когда дальнейшая детализация перестает быть полезной, переходят к выражению логики функции при помощи спецификации процесса (миниспецификации). Фактически миниспецификации представляют собой алгоритмы описания задач, выполняемых процессами: множество всех миниспецификаций является полной спецификацией системы.
Содержимое каждого хранилища также сохраняют в словаре данных, модель данных хранилища раскрывается с помощью ERD. В случае наличия реального времени DFD дополняется средствами описания зависящего от времени поведения системы, раскрывающимися с помощью STD. Эти связи показаны на рис.1.
Перечисленные средства дают полное описание системы; таким образом, строится логическая функциональная спецификация - подробное описание того, что должна делать система, освобожденное насколько это возможно от рассмотрения путей реализации. Это дает проектировщику четкое представление о конечных результатах, которые следует достигать.

Для изображения DFD традиционно используются две различные нотации: Йодана (Yourdon) и Гейна-Сарсона (Gane-Sarson). Далее при построении примеров используется нотация Йодана.

Основные символы диаграммы

Основные символы (компоненты) DFD изображены на рис.2.

На диаграммах функциональные требования представляются с помощью *процессов* и *хранилищ*, связанных *потоками данных*.

Потоки данных являются механизмами, использующимися для моделирования передачи информации (или даже физических компонент) из одной части системы в другую. Важность этого объекта очевидна: он дает название целому инструменту. Потоки на диаграммах обычно изображаются именованными стрелками, ориентация которых указывает направление движения информации.

Иногда информация может двигаться в одном направлении, обрабатываться и возвращаться назад в ее источник. Такая ситуация может моделироваться либо двумя различными потоками, либо одним - двунаправленным.

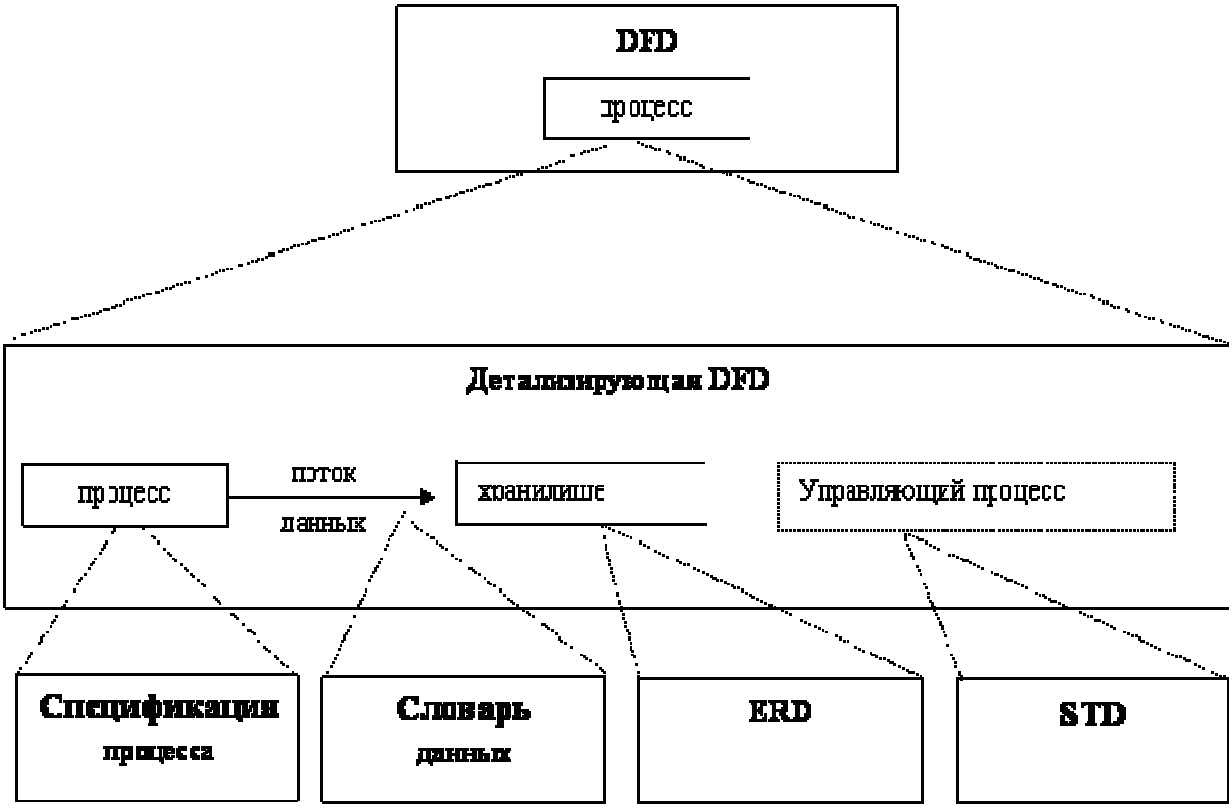


Рис.1. Компоненты логической модели

Назначение **процесса** состоит в преобразовании входных потоков данных в выходные в соответствии с действием, задаваемым именем процесса. Это имя должно содержать глагол в неопределенной форме с последующим дополнением (например, ВЫЧИСЛИТЬ МАКСИМАЛЬНУЮ ВЫСОТУ). Кроме того, каждый процесс должен иметь уникальный номер для ссылок на него внутри диаграммы. Этот номер может использоваться совместно с номером диаграммы для получения уникального индекса процесса во всей модели.

Хранилище (накопитель) данных позволяет на определенных участках определять данные, которые будут сохраняться в памяти между процессами. Фактически хранилище представляет "срезы" потоков данных во времени. Информация, которую оно содержит, может использоваться в любое время после ее определения, при этом данные могут

выбираться в любом порядке. Имя хранилища должно идентифицировать его содержимое и быть существительным. В случае, когда поток данных входит или выходит в/из хранилища и его структура соответствует структуре хранилища, он должен иметь то же самое имя, которое нет необходимости отражать на диаграмме.

Внешняя сущность (**терминатор**) представляет сущность вне контекста системы, являющуюся источником или приемником системных данных. Ее имя должно содержать существительное, например, СКЛАД ТОВАРОВ. Предполагается, что объекты, представленные такими узлами, не должны участвовать ни в какой обработке.

Контекстная диаграмма и детализация процессов

Декомпозиция DFD осуществляется на основе процессов: каждый процесс может раскрываться с помощью DFD нижнего уровня.

Важную специфическую роль в модели играет специальный вид DFD - **контекстная диаграмма**, моделирующая систему наиболее общим образом. Контекстная диаграмма отражает интерфейс системы с внешним миром, а именно, информационные потоки между системой и внешними сущностями, с которыми она должна быть связана. Она идентифицирует эти внешние сущности, а также, как правило, единственный процесс, отражающий главную цель или природу системы насколько это возможно. И хотя контекстная диаграмма выглядит тривиальной, несомненная ее полезность заключается в том, что она устанавливает границы анализируемой системы. Каждый проект должен иметь ровно одну контекстную диаграмму, при этом нет необходимости в нумерации единственного ее процесса.

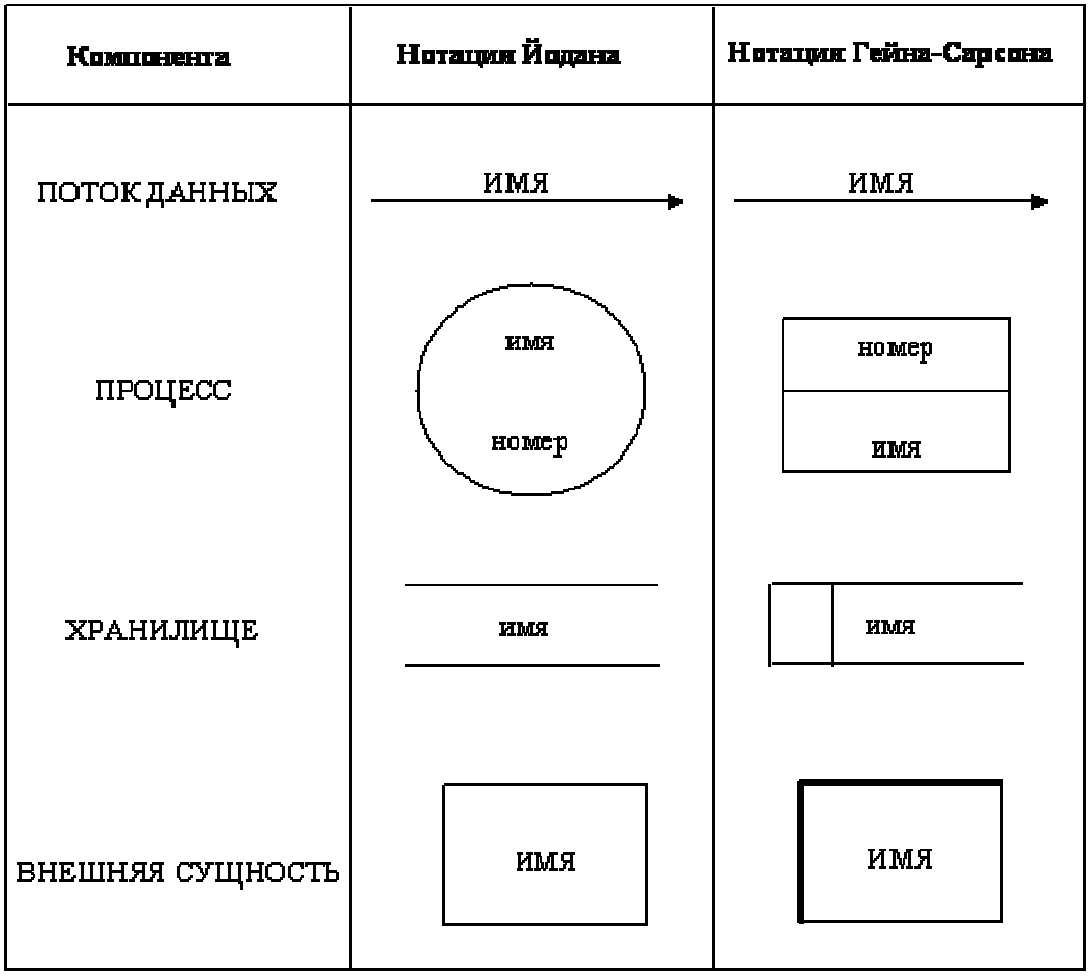


Рис.2. Основные компоненты диаграммы потоков данных

DFD первого уровня строится как декомпозиция процесса, который присутствует на контекстной диаграмме.
Построенная диаграмма первого уровня также имеет множество процессов, которые в свою очередь могут быть декомпозированы в DFD нижнего уровня. Таким образом, строится иерархия DFD с контекстной диаграммой в корне дерева. Этот процесс декомпозиции продолжается до тех пор, пока процессы могут быть эффективно описаны с помощью коротких (до одной страницы) миниспецификаций обработки (спецификаций процессов).
При таком построении иерархии DFD каждый процесс более низкого уровня необходимо соотнести с процессом верхнего уровня. Обычно для этой цели используются структурированные номера процессов. Так, например, если мы детализируем процесс номер 2 на диаграмме первого уровня, раскрывая его с помощью DFD, содержащей три процесса, то их номера будут иметь следующий вид: 2.1, 2.2 и 2.3. При необходимости можно перейти на следующий уровень, т.е. для процесса 2.2 получим 2.2.1, 2.2.2. и т.д.
Проиллюстрируем контекстную диаграмму на примере.
Пример. Рассмотрим процесс СДАТЬ ЭКЗАМЕН. У нас есть две сущности СТУДЕНТ и ПРЕПОДАВАТЕЛЬ. Опишем потоки данных, которыми обменивается наша проектируемая система с внешними объектами.

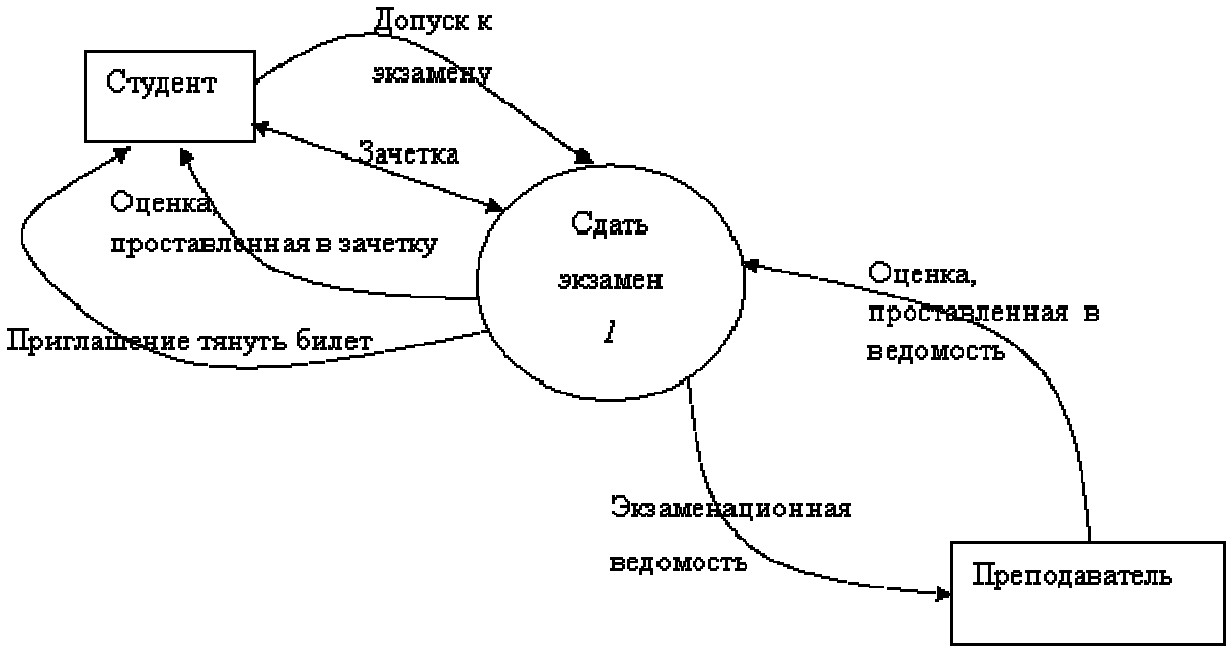


Рис. 3. DFD-диаграмма процесса "Сдать экзамен"

Со стороны сущности СТУДЕНТ опишем информационные потоки. Для сдачи экзамена необходимо, чтобы у СТУДЕНТА была ЗАЧЕТКА, чтобы он имел ДОПУСК К ЭКЗАМЕНУ. Результатом сдачи экзамена, т.е. выходными потоками будут ОЦЕНКА ЗА ЭКЗАМЕН и ЗАЧЕТКА, в которую будет проставлена ОЦЕНКА.
Со стороны сущности ПРЕПОДАВАТЕЛЬ информационные потоки следующие. ЭКЗАМЕНАЦИОННАЯ ВЕДОМОСТЬ согласно которой будет известно, что СТУДЕНТ допущен до экзамена, а также официальна бумага, куда будет занесен результат экзамена, т.е. ОЦЕНКА ЗА ЭКЗАМЕН, ПРОСТАВЛЕННАЯ В ВЕДОМОСТЬ.
Теперь детализируем процесс 1.СДАТЬ ЭКЗАМЕН. Этот процесс будет содержать следующие процессы:
1.1. Вытянуть билет
1.2. Подготовиться к ответу
1.3. Ответы на билет

1.4. Проставление оценки

Декомпозиция данных и соответствующие расширения диаграмм потоков данных
Индивидуальные данные в системе часто являются независимыми. Однако иногда необходимо иметь дело с несколькими независимыми данными одновременно. Например, в системе имеются потоки ЯБЛОКИ, АПЕЛЬСИНЫ и ГРУШИ. Эти потоки могут быть сгруппированы с помощью введения нового потока ФРУКТЫ. Для этого необходимо определить формально поток ФРУКТЫ как состоящий из нескольких элементов-потомков. В свою очередь поток ФРУКТЫ сам может содержаться в потоке-предке ЕДА вместе с потоками ОВОЩИ, МЯСО и др. Такие потоки, объединяющие несколько потоков, получили название **групповых**.
Обратная операция, расщепление потоков на подпотоки, осуществляется с использованием группового узла, позволяющего расщепить поток на любое число подпотоков. При расщеплении также необходимо формально определить подпотоки в словаре данных.
Аналогичным образом осуществляется и декомпозиция потоков через границы диаграмм, позволяющая упростить детализирующую DFD. Пусть имеется поток ФРУКТЫ, входящий в детализируемый процесс. На детализирующей этот процесс диаграмме потока ФРУКТЫ может не быть вовсе, но вместо него могут быть потоки ЯБЛОКИ и АПЕЛЬСИНЫ (как будто бы они переданы из детализируемого процесса). В этом случае должно существовать определение потока ФРУКТЫ, состоящего из подпотоков ЯБЛОКИ и АПЕЛЬСИНЫ, для целей балансирования.

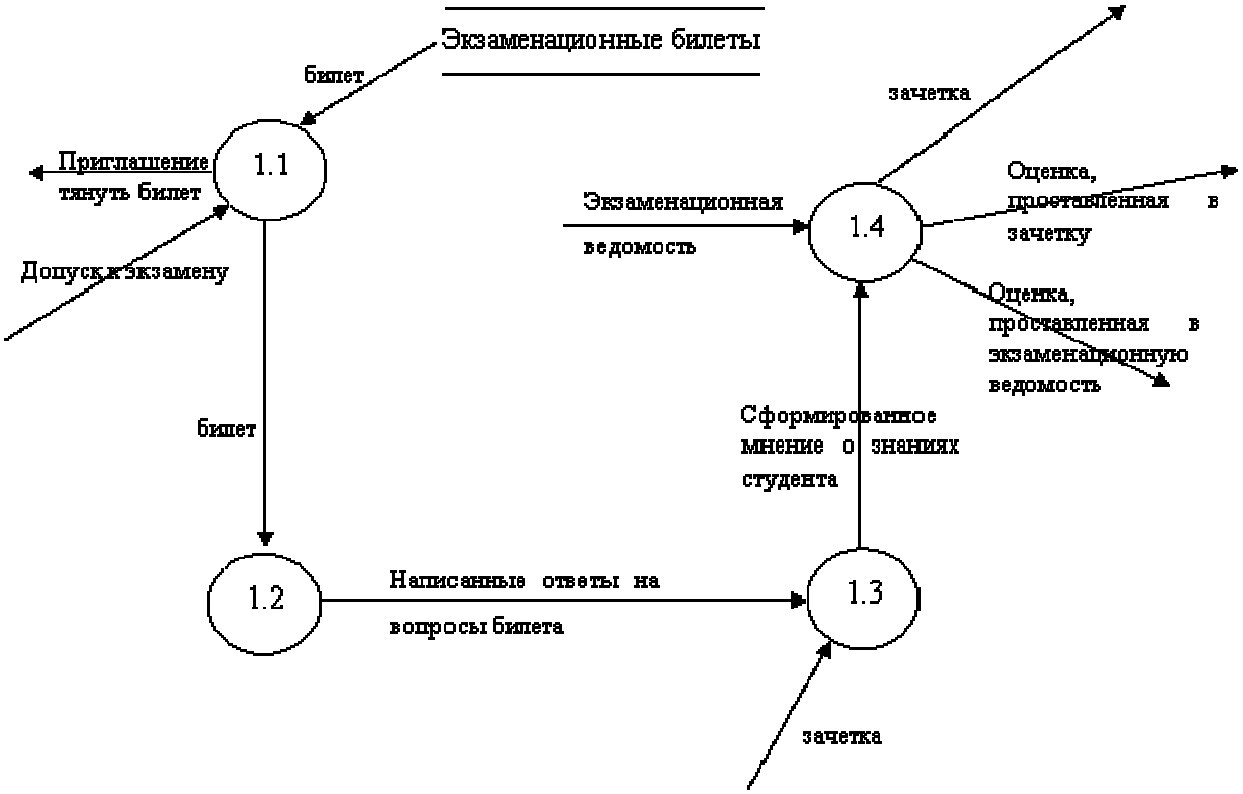


Рис. 4. Декомпозиция 1-го уровня

Применение этих операций над данными позволяет обеспечить структуризацию данных, увеличивает наглядность и читабельность диаграмм.
Процесс построения DFD.

1. Создание так называемой **основной диаграммы** типа "***звезда***", на которой представлен моделируемый процесс и все внешние сущности, с которыми он взаимодействует. В

случае сложного основного процесса он сразу представляется в виде декомпозиции на ряд взаимодействующих процессов.

Внешние сущности выделяются по отношению к основному процессу. Для их определения необходимо выделить поставщиков и потребителей основного процесса, т.е. все объекты, которые взаимодействуют с основным процессом. На этом этапе описание взаимодействия заключается в выборе глагола, дающего представление о том, как внешняя сущность использует основной процесс или используется им. Например, основной процесс – "учет обращений граждан", внешняя сущность – "граждане", описание взаимодействия – "подает заявления и получает ответы". Этот этап является принципиально важным, поскольку именно он определяет границы моделируемой системы. Для всех внешних сущностей строится **таблица событий**, описывающая их взаимодействие с основным потоком. Таблица событий включает в себя наименование внешней сущности, событие, его тип (типичный для системы или исключительный, реализующийся при определенных условиях) и реакцию системы.

2. Декомпозиция основного процесса на набор взаимосвязанных процессов, обменивающихся потоками данных. Сами потоки не конкретизируются, определяется лишь характер взаимодействия. Декомпозиция завершается, когда процесс становится простым, т.е.:
   - процесс имеет два-три входных и выходных потока;
   - процесс может быть описан в виде преобразования входных данных в выходные;
   - процесс может быть описан в виде последовательного алгоритма.
3. Определение полной таблицы событий; выделяются потоки данных, которыми обмениваются процессы и внешние сущности. Простейший способ их выделения заключается в анализе таблиц событий. События преобразуются в потоки данных от инициатора события к запрашиваемому процессу, а реакции – в обратный поток событий. После построения входных и выходных потоков аналогичным образом строятся внутренние потоки. Для их выделения для каждого из внутренних процессов выделяются поставщики и потребители информации. Если поставщик или потребитель информации представляет процесс сохранения или запроса информации, то вводится хранилище данных, для которого данный процесс является интерфейсом.
4. После построения потоков данных диаграмма должна быть проверена на полноту и непротиворечивость. Полнота диаграммы обеспечивается, если в системе нет "повисших" процессов, не используемых в процессе преобразования входных потоков в выходные. Непротиворечивость системы обеспечивается выполнением наборов формальных правил о возможных типах процессов: на диаграмме не может быть потока, связывающего две внешние сущности – это взаимодействие удаляется из рассмотрения; ни одна сущность не может непосредственно получать или отдавать информацию в хранилище данных – хранилище данных является пассивным элементом, управляемым с помощью интерфейсного процесса; два хранилища данных не могут непосредственно обмениваться информацией - они должны быть объединены.

Содержимое словаря данных

Для каждого потока данных в словаре необходимо хранить имя потока, его тип и атрибуты. Информация по каждому потоку состоит из ряда словарных статей, каждая из которых начинается с ключевого слова - заголовка соответствующей статьи, которому предшествует символ "@".

По типу потока в словаре содержится информация, идентифицирующая:

- простые (элементарные) или групповые (комплексные) потоки;
- внутренние (существующие только внутри системы) или внешние (связывающие систему с другими системами) потоки;
- потоки данных или потоки управления;

- непрерывные (принимающие любые значения в пределах определенного диапазона) или дискретные (принимающие определенные значения) потоки.

К **преимуществам** методики DFD относятся:

- возможность однозначно определить внешние сущности, анализируя потоки информации
- внутри и вне системы;
- возможность проектирования сверху вниз, что облегчает построение модели "как должно быть";
- наличие спецификаций процессов нижнего уровня, что позволяет преодолеть логическую незавершенность функциональной модели и построить полную функциональную спецификацию разрабатываемой системы.

К недостаткам модели можно отнести:

- необходимость искусственного ввода управляющих процессов, поскольку управляющие воздействия (потоки) и управляющие процессы с точки зрения DFD ничем не отличаются от обычных;
- отсутствие понятия времени, т.е. отсутствие анализа временных промежутков при преобразовании данных (все ограничения по времени должны быть введены в спецификациях процессов);
- недостаточная обратная связь, присущая каскадной модели; ориентация на функционально-модульный подход увеличивает вероятность потери контроля над решением возникающих проблем.

13. ERD - методика проектирования информационной базы

Диаграммы "сущность-связь"

Диаграммы "сущность-связь" (ERD) предназначены для разработки моделей данных и обеспечивают стандартный способ определения данных и отношений между ними. Фактически с помощью ERD осуществляется детализация хранилищ данных проектируемой системы, а также документируются сущности системы и способы их взаимодействия, включая идентификацию объектов, важных для предметной области (сущностей), свойств этих объектов (атрибутов) и их отношений с другими объектами (связей).

Данная нотация была введена Ченом (Chen) и получила дальнейшее развитие в работах Баркера (Barker). Нотация Чена предоставляет богатый набор средств моделирования данных, включая собственно ERD, а также диаграммы атрибутов и диаграммы декомпозиции. Эти диаграммные техники используются прежде всего для проектирования реляционных баз данных (хотя также могут с успехом применяться и для моделирования как иерархических, так и сетевых баз данных).

Сущности, отношения и связи в нотации Чена

Сущность представляет собой множество экземпляров реальных или абстрактных объектов (людей, событий, состояний, идей, предметов и т.п.), обладающих общими атрибутами или характеристиками. Любой объект системы может быть представлен только одной сущностью, которая должна быть уникально идентифицирована. При этом имя сущности должно отражать тип или класс объекта, а не его конкретный экземпляр (например, АЭРОПОРТ, а не ВНУКОВО).

Отношение в самом общем виде представляет собой связь между двумя и более сущностями. Именование отношения осуществляется с помощью грамматического оборота глагола (ИМЕЕТ, ОПРЕДЕЛЯЕТ, МОЖЕТ ВЛАДЕТЬ и т.п.).

Другими словами, сущности представляют собой базовые типы информации, хранимой в базе данных, а отношения показывают, как эти типы данных взаимоувязаны друг с другом. Введение подобных отношений преследует две основополагающие цели:

- обеспечение хранения информации в единственном месте (даже если она используется в различных комбинациях);
- использование этой информации различными приложениями.

Символы ERD, соответствующие сущностям и отношениям, приведены на рис. 1.

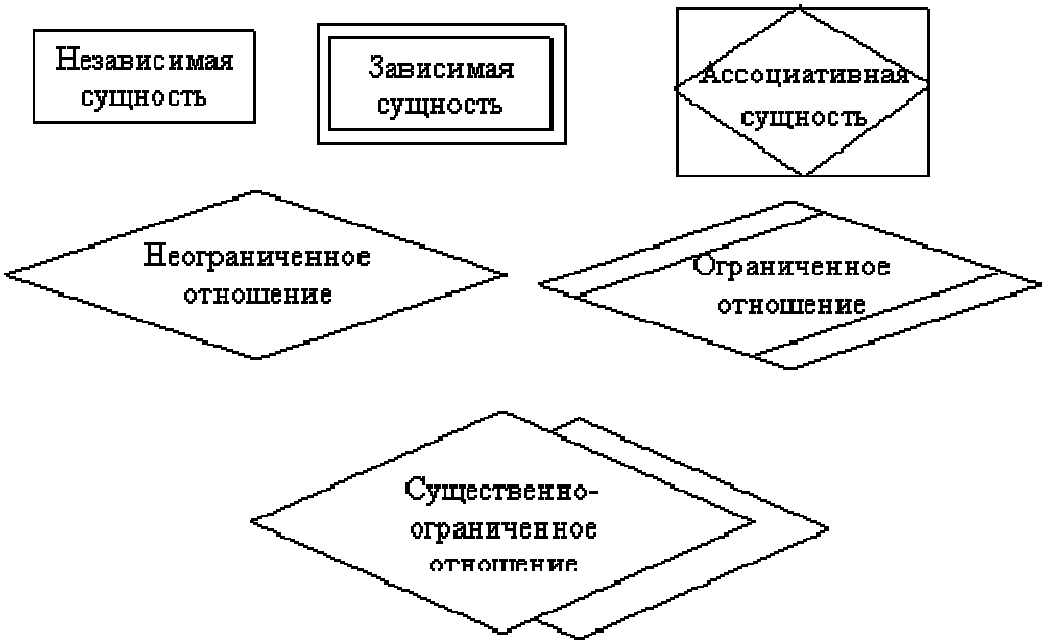


Рис.1. Символы ERD в нотации Чена

Независимая сущность представляет независимые данные, которые всегда присутствуют в системе. При этом отношения с другими сущностями могут как существовать, так и отсутствовать.

В свою очередь, **зависимая сущность** представляет данные, зависящие от других сущностей в системе. Поэтому она должна всегда иметь отношения с другими сущностями.

Ассоциированная сущность представляет данные, которые ассоциируются с отношениями между двумя и более сущностями.

Неограниченное (обязательное) отношение представляет собой безусловное отношение, т.е. отношение, которое всегда существует до тех пор, пока существуют относящиеся к делу сущности.

Ограниченное (необязательное) отношение представляет собой условное отношение между сущностями.

Существенно-ограниченное отношение используется, когда соответствующие сущности взаимно зависимы в системе.

Для идентификации требований, в соответствии с которыми сущности вовлекаются в отношения, используются **СВЯЗИ**. Каждая связь соединяет сущность и отношение и может быть направлена только от отношения к сущности.

Значение связи характеризует ее тип и, как правило, выбирается из следующего множества:

{"0 или 1", "0 или более", "1", "1 или более", "p:q" (диапазон)}.

Пара значений связей, принадлежащих одному и тому же отношению, определяет тип этого отношения. Практика показала, что для большинства приложений достаточно использовал следующие типы отношений:

1) **1*1 (один-к-одному)**. Отношения данного типа используются, как правило, на верхних уровнях иерархии модели данных, а на нижних уровнях встречаются сравнительно редко.

2) **1*n (один-к-многим)**. Отношения данного типа являются наиболее часто используемыми.

3) **n*m (многие-к-многим)**. Отношения данного типа обычно используются на ранних этапах проектирования с целью прояснения ситуации. В дальнейшем каждое из таких отношений должно быть преобразовано в комбинацию отношений типов 1 и 2 (возможно, с добавлением вспомогательных сущностей и введением новых отношений).

Диаграммы атрибутов

Каждая сущность обладает одним или несколькими **атрибутами**, которые однозначно идентифицируют каждый экземпляр сущности. При этом любой атрибут может быть определен как ключевой.

Детализация сущности осуществляется с использованием **диаграмм атрибутов**, которые раскрывают ассоциированные сущностью атрибуты. Диаграмма атрибутов состоит из детализируемой *сущности*, соответствующих *атрибутов* и *доменов*, описывающих области значений атрибутов. На диаграмме каждый атрибут представляется в виде связи между сущностью и соответствующим доменом, являющимся графическим представлением множества возможных значений атрибута. Все атрибутные связи имеют значения на своем окончании. Для идентификации ключевого атрибута используется подчеркивание имени атрибута.

Категоризация сущностей

Сущность может быть разделена и представлена в виде двух или более сущностей-категорий, каждая из которых имеет общие атрибуты и/или отношения, которые определяются однажды на верхнем уровне и наследуются на нижнем. Сущности-категории могут иметь и свои собственные атрибуты и/или отношения, а также, в свою очередь, могут быть декомпозированы своими сущностями-категориями на следующем уровне. Расщепляемая на категории сущность получила название **общей сущности** (отметим, что на промежуточных уровнях декомпозиции одна и та же сущность может быть как общей сущностью, так и сущностью-категорией).

Для демонстрации декомпозиции сущности на категории используются **диаграммы категоризации**. Такая диаграмма содержит общую сущность, две и более сущности-категории и специальный **узел-дискриминатор**, который описывает способы декомпозиции сущностей (см. рис. 2).

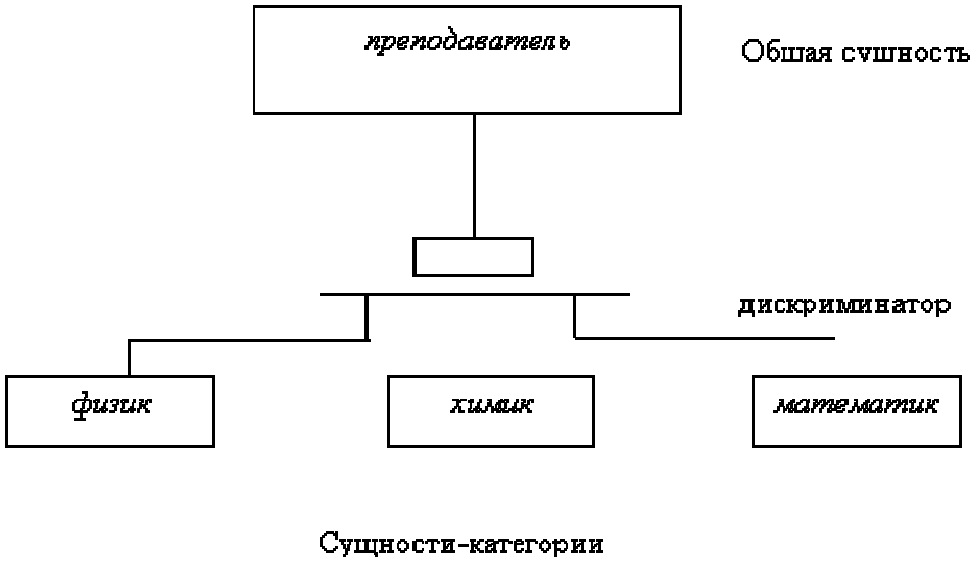


Рис. 2. Узел-дискриминатор

Существуют 4 возможных типа дискриминатора:

1) **Полное и обязательное вхождение E/M** (exclusive/mandatory) - сущность должна быть одной и только одной из следуемых категорий.

2) **Полное и необязательное вхождение E/O** (exclusive/optional) - сущность может быть одной и только одной из следуемых категорий.

3) **Неполное и обязательное вхождение I/M** (inclusive/mandatory) - сущность должна быть, по крайней мере, одной из следуемых категорий.
4) **Неполное и необязательное вхождение I/O** (inclusive/optional) - сущность может быть, по крайней мере, одной из следуемых категорий.

Построение модели

Разработка ERD включает следующие этапы:
1) Идентификация сущностей, их атрибутов, а также первичных и альтернативных ключей.
2) Идентификация отношений между сущностями и указаний типов отношений.
3) Разрешение неспецифических отношений (отношений n*m).
Этап 1 является определяющим при построении модели, его исходной информацией служит содержимое хранилищ данных определяемое входящими и выходящими в/из него потоками данных. Первоначально осуществляется анализ хранилища, включающий сравнение содержимого входных и выходных потоков и создание на основе этого сравнения варианта схемы хранилища, описание структур данных, содержащихся во входных и выходных потоках.
Затем, если полученная схема обладает избыточностью, то происходит упрощение схемы путем нормализации (удаления повторяющихся групп). Концепции и методы нормализации были разработаны Коддом, установившим существование трех типов нормализованных схем, называемых в порядке уменьшения сложности **первой, второй и третьей нормальной формой (1НФ, 2НФ, 3НФ)**.
Пример НФ. Рассмотрим все три нормальные формы на примере Группы Студентов. У Студента ключом является реквизит Номер (№ личного дела), к описательным реквизитам относятся: Фамилия (Фамилия студента), Имя (Имя студента), Отчество (Отчество студента), Дата (Дата рождения), Группа (N группы). Если отсутствует реквизит Номер, то для однозначного определения характеристик конкретного студента необходимо использование составного ключа из трех реквизитов: Фамилия + Имя + Отчество.

Первая нормальная форма

Отношение называется нормализованным или приведенным к первой нормальной форме, если все его атрибуты простые (далее неделимы). Преобразование отношения к первой нормальной форме может привести к увеличению количества реквизитов (полей) отношения и изменению ключа.
Например, отношение Студент=(Номер, Фамилия, Имя, Отчество, Дата, Группа) находится в первой нормальной форме.

Вторая нормальная форма

Чтобы рассмотреть вопрос приведения отношений ко второй нормальной форме, необходимо дать пояснения к таким понятиям, как функциональная зависимость и полная функциональная зависимость.
Описательные реквизиты информационного объекта логически связаны с общим для них ключом, эта связь носит характер функциональной зависимости реквизитов.
Функциональная зависимость реквизитов - зависимость, при которой в экземпляре сущности (в нашем случае, сущность - это студент) определенному значению ключевого реквизита соответствует только одно значение описательного реквизита.
Такое определение функциональной зависимости позволяет при анализе всех взаимосвязей реквизитов предметной области выделить самостоятельные информационные объекты.
В случае составного ключа вводится понятие функционально полной зависимости.

Функционально полная зависимость неключевых атрибутов заключается в том, что каждый неключевой атрибут функционально зависит от ключа, но не находится в функциональной зависимости ни от какой части составного ключа.
Отношение будет находиться во второй нормальной форме, если оно находится в первой нормальной форме, и каждый неключевой атрибут функционально полно зависит от составного ключа.
Отношение Студент = (Номер, Фамилия, Имя, Отчество, Дата, Группа) находится в первой и во второй нормальной форме одновременно, так как описательные реквизиты однозначно определены и функционально зависят от ключа Номер. Отношение Успеваемость = (Номер, Фамилия, Имя, Отчество, Дисциплина, оценка) находится в первой нормальной форме и имеет составной ключ Номер+Дисциплина. Это отношение не находится во второй нормальной форме, так как атрибуты Фамилия, Имя, Отчество не находятся в полной функциональной зависимости с составным ключом отношения.

Третья нормальная форма

Понятие третьей нормальной формы основывается на понятии нетранзитивной зависимости.
Транзитивная зависимость наблюдается в том случае, если один из двух описательных реквизитов зависит от ключа, а другой описательный реквизит зависит от первого описательного реквизита.
Отношение будет находиться в третьей нормальной форме, если оно находится во второй нормальной форме, и каждый неключевой атрибут нетранзитивно зависит от первичного ключа.
Пример. Если в состав описательных реквизитов Студент включить фамилию старосты группы (Староста), которая определяется только номером группы, то одна и та же фамилия старосты будет многократна повторяться в разных экземплярах Студентов. В этом случае наблюдаются затруднения в корректировке фамилии старосты в случае назначения нового старосты, а также неоправданный расход памяти для хранения дублированной информации.
Для устранения транзитивной зависимости описательных реквизитов необходимо провести "расщепление" исходной сущности. В результате расщепления часть реквизитов удаляется из исходной сущности и включается в состав другой (возможно, вновь созданных) сущности.
Пример. Студент группы представляется в виде совокупности правильно структурированных информационных объектов (Студент и Группа), реквизитный состав которых тождественен исходному объекту. Отношение Студент = (Номер, Фамилия, Имя, Отчество, Дата, Группа) находится одновременно в первой, второй и третьей нормальной форме.
ЗНФ является наиболее простым способом представления данных, отражающим здравый смысл. Построив ЗНФ, мы фактически выделяем базовые сущности предметной области.
Этап 2 служит для выявления и определения отношений между сущностями, а также для идентификации типов отношений. Некоторые отношения на данном этапе могут быть неспецифическими (n*m - многие-ко-многим). Такие отношения потребуют дальнейшей детализации на этапе 3.
Определение отношений включает выявление связей, для этого отношение должно быть проверено в обоих направлениях следующим образом: выбирается экземпляр одной из сущностей и определяется, сколько различных экземпляров второй сущности может быть связано с ним и наоборот.
Пример. Связь "1-к-1". Человек и его паспорт. Связь "1-ко-многим": Человек и телефоны.
Этап 3 предназначен для разрешения неспецифических отношений. Для этого каждое неспецифическое отношение преобразуется в два специфических отношения с введением новых (а именно, ассоциативных) сущностей. Пример. Студент может изучать много Предметов, а Предмет может изучаться многими Студентами. Мы не можем определить,

какой Студент изучает какой Предмет. Следовательно, у нас есть неспецифическое отношение, которое можно расщепить на два специфических отношения, введя ассоциативную сущность Изучение_Предмета. Каждый экземпляр введенной сущности связан с одним Студентом и одним Предметом (рис.3.).

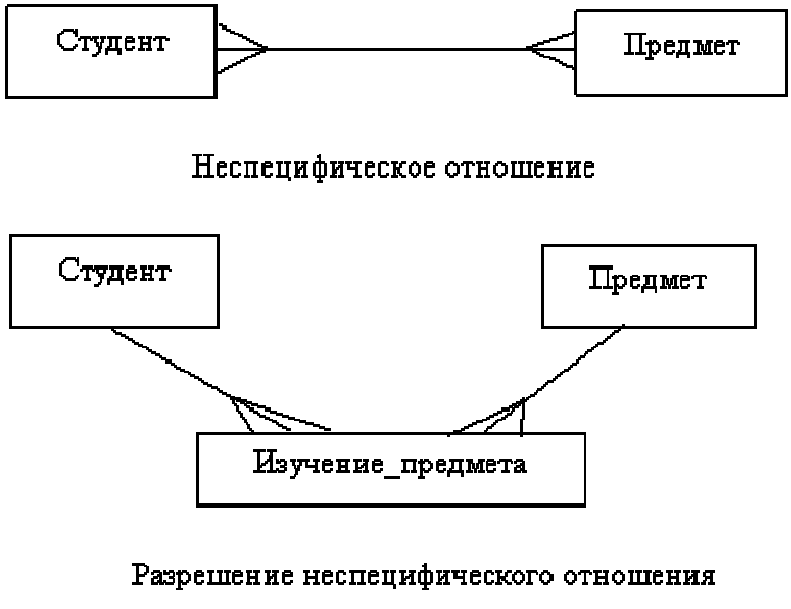


Рис. 3. Пример разрешения неспецифического отношения

14. STD - методика спецификации управления

Спецификации управления предназначены для моделирования и документирования аспектов систем, зависящих от времени или реакции на событие. Они позволяют осуществлять декомпозицию управляющих процессов и описывают отношения между входными и выходными управляющими потоками на управляющем процессе-предке. Для этой цели обычно используются **диаграммы переходов состояний (STD)**.

С помощью STD можно моделировать последующее функционирование системы на основе ее предыдущего и текущего функционирования. Моделируемая система в любой заданный момент времени находится точно в одном из конечного множества состояний. С течением времени она может изменить свое состояние, при этом переходы между состояниями должны быть точно определены.

STD состоит из следующих объектов:

Состояние - может рассматриваться как условие устойчивости для системы. Находясь в определенном состоянии, мы имеем достаточно информации о прошлой истории системы, чтобы определить очередное состояние в зависимости от текущих входных событий. Имя состояния должно отражать реальную ситуацию, в которой находится система, например, НАГРЕВАНИЕ, ОХЛАЖДЕНИЕ и т.п.

Начальное состояние - узел STD, являющийся стартовой точкой для начального системного перехода. STD имеет ровно одно начальное состояние, соответствующее состоянию системы после ее инсталляции, но перед началом реальной обработки, а также любое (конечное) число завершающих состояний.

Переход определяет перемещение моделируемой системы из одного состояния в другое. При этом имя перехода идентифицирует событие, являющееся причиной перехода и управляющее им. Это событие обычно состоит из управляющего потока (сигнала), возникающего как во внешнем мире, так и внутри моделируемой системы при выполнении некоторого условия (например СЧЕТЧИК=999 или КНОПКА НАЖАТА). Следует отметить, что, вообще говоря, не все события необходимо вызывают переходы из

отдельных состояний. С другой стороны, одно и то же событие не всегда вызывает переход в то же самое состояние.
Таким образом, **УСЛОВИЕ** представляет собой событие (или события), вызывающее переход и идентифицируемое именем перехода. Если в условии участвует входной управляющий поток управляющего процесса-предка, то имя потока должно быть заключено в кавычки, например, "ПАРОЛЬ"=666, где ПАРОЛЬ - входной поток.
Кроме условия с переходом может связываться действие или ряд действий, выполняющихся, когда переход имеет место. То есть **ДЕЙСТВИЕ** - это операция, которая может иметь место при выполнении перехода. Если действие необходимо для выбора выходного управляющего потока, то имя этого потока должно заключаться в кавычки, например:

"ВВЕДЕННАЯ КАРТА " = TRUE,
где ВВЕДЕННАЯ КАРТА - выходной поток.

На STD состояния представляются узлами, а переходы дугами (рис.1.). Условия (по-другому называемые **стимулирующими событиями**) идентифицируются именем перехода и возбуждают выполнение перехода. Действия или отклики на события привязываются к переходам и записываются под соответствующим условием. Начальное состояние на диаграмме должно иметь входной переход, изображаемый потоком из подразумеваемого стартового узла (иногда этот стартовый узел изображается небольшим квадратом и привязывается к входному состоянию).

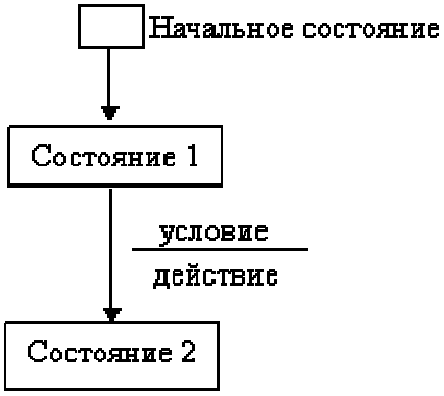


Рис. 1. Пример STD узлов

В ситуации, когда число состояний и/или переходов велико, для проектирования спецификаций управления могут использоваться **таблицы** или **матрицы переходов состояний**.
Первая колонка таблицы содержит список всех состояний проектируемой системы, во второй колонке для каждого состояния приведены все условия, вызывающие переходы в другие состояния, а в третьей колонке - совершаемые при этих переходах действия. Четвертая колонка содержит соответствующие имена состояний, в которые осуществляется переход из рассматриваемого состояния при выполнении определенного условия.
Матрица переходов состояний содержит по вертикали перечень состояний системы, а по горизонтали список условий. Каждый ее элемент содержит список действий, а также имя состояния, в которое осуществляется переход.

15. Методы задания спецификации процессов

Спецификация процесса (СП) используется для описания функционирования процесса в случае отсутствия необходимости детализировать его с помощью DFD (т.е. если он достаточно невелик, и его описание может занимать до одной страницы текста). Фактически СП представляют собой алгоритмы описания задач, выполняемых процессами: множество всех СП является полной спецификацией системы. СП содержат номер и/или имя процесса, списки входных и выходных данных и тело (описание)

процесса, являющееся спецификацией алгоритма или операции, преобразующей входные потоки данных в выходные. Известно большое число разнообразных методов, позволяющих задать *тело* процесса - соответствующий язык может варьироваться от **структурированного естественного языка** (или **псевдокода)** до **визуальных языков проектирования** (типа **FLOW-форм и диаграмм Насси-Шнейдермана**) и **формальных компьютерных языков**.

Независимо от используемой нотации спецификация процесса должна начинаться с ключевого слова (например, **@СПЕЦПРОЦ**). Требуемые входные и выходные данные должны быть специфицированы следующим образом:

@ВХОД = <имя символа данных>
@ВЫХОД = <имя символа данных>
@ВХОДВЫХОД = <имя символа данных>,
где <имя символа данных> - соответствующее имя из Словаря Данных.

Эти ключевые слова должны использоваться перед определением СП, например,

@ВХОД = СЛОВА ПАМЯТИ
@ВЫХОД = ХРАНИМЫЕ ЗНАЧЕНИЯ
@СПЕЦПРОЦ
Для всех СЛОВ ПАМЯТИ выполнить:
Распечатать ХРАНИМЫЕ ЗНАЧЕНИЯ
@

Ситуация, когда символ данных является одновременно входным и выходным, может быть описана двумя способами: либо символ описывается два раза с помощью **@ВХОД** и **@ВЫХОД**, либо один раз с помощью **@ВХОДВЫХОД**.

Иногда в СП задаются **пред**- и **пост**-условия выполнения данного процесса. В пред-условии записываются объекты, значения которых должны быть истинны перед началом выполнения процесса, что обеспечивает определенные гарантии безопасности для пользователя. Аналогично, в случае наличия пост-условия гарантируется, что значения всех входящих в него объектов будут истинны при завершении процесса.

Спецификации должны удовлетворять следующим требованиям:

- для каждого процесса нижнего уровня должна существовать одна и только одна спецификация;
- спецификация должна определять способ преобразования входных потоков в выходные;
- нет необходимости (на данном этапе) определять метод реализации этого преобразования;
- спецификация должна стремиться к ограничению избыточности - не следует переопределять то, что уже было определено на диаграмме или в словаре данных;
- набор конструкций для построения спецификации должен быть простым и стандартным.

Рассмотрим некоторые наиболее часто используемые методы задания спецификаций процессов.

Структурированный естественный язык

Структурированный естественный язык применяется для читабельного, строгого описания спецификаций процессов. Он является разумной комбинацией строгости языка программирования и читабельности естественного языка и состоит из подмножества слов, организованных в определенные логические структуры, арифметических выражений и диаграмм.

В состав языка входят следующие основные символы:

- глаголы, ориентированные на действие и применяемые к объектам;
- термины, определенные на любой стадии проекта ПО (например, задачи, процедуры, символы данных и т.п.);
- предлоги и союзы, используемые в логических отношениях;
- общеупотребительные математические, физические и технические термины;

- арифметические уравнения;
- таблицы, диаграммы, графы и т.п.;
- комментарии.

Управляющие структуры языка имеют один вход и один выход. К ним относятся:
1) последовательная конструкция:
ВЫПОЛНИТЬ функция1
ВЫПОЛНИТЬ функция2
ВЫПОЛНИТЬ функция3
2) конструкция выбора:
ЕСЛИ <условие> **ТО**
ВЫПОЛНИТЬ функция1
ИНАЧЕ
ВЫПОЛНИТЬ функция2
КОНЕЦЕСЛИ
3)итерация:
ДЛЯ <условие>
ВЫПОЛНИТЬ функция
КОНЕЦДЛЯ
или
ПОКА <условие>
ВЫПОЛНИТЬ функция
КОНЕЦПОКА
При использовании структурированного естественного языка приняты следующие соглашения:
1) Логика процесса выражается в виде комбинации последовательных конструкций, конструкций выбора и итераций.
2) Ключевые слова **ЕСЛИ, ВЫПОЛНИТЬ, ИНАЧЕ** и т.д. должны быть написаны заглавными буквами.
3) Слова или фразы, определенные в словаре данных, должны быть написаны заглавными буквами.
4) Глаголы должны быть активными, недвусмысленными и ориентированными на целевое действие (заполнить, вычислить, извлечь, а не модернизировать, обработать).
5) Логика процесса должна быть выражена четко недвусмысленно.

Таблицы решений

Структурированный естественный язык неприемлем да некоторых типов преобразований. Например, если действие зависит от нескольких переменных, которые в совокупности могут продуцировать большое число комбинаций, то его описание будет слишком запутанным и с большим числом уровней вложенности. Для описания подобных действий традиционно используются таблицы и деревья решений.
Проектирование спецификаций процессов с помощью **таблиц решений (ТР)** заключается в задании матрицы, отображающей множество входных **условий** во множество **действий**.
Таблица Решений состоит из двух частей. Верхняя часть таблицы используется для определения условий. Обычно условие является **ЕСЛИ**-частью оператора **ЕСЛИ-ТО** и требует ответа "да-нет". Однако иногда в условии может присутствовать и ограниченное множество значений, например, ***является ли длина строки большей, меньшей или равной граничному значению?***
Нижняя часть Таблицы Решений используется для определения действий, т.е. **ТО**-части оператора **ЕСЛИ-ТО**. Так, в конструкции **ЕСЛИ** ИДЕТ ДОЖДЬ, **ТО** РАСКРЫТЬ ЗОНТ. ИДЕТ ДОЖДЬ является условием, а РАСКРЫТЬ ЗОНТ - действием.
Левая часть Таблицы Решений содержит собственно описание условий и действий, а в правой части перечисляются все возможные комбинации условий и, соответственно,

указывается, *какие* конкретно действия и *в какой* последовательности выполняются, когда определенная комбинация условий имеет место.
Поясним вышесказанное на примере спецификации процесса выбора верхней одежды из корзины с вещами. При выборе верхней одежды необходимо руководствоваться следующими правилами:
1) *если очередная вещь является верхней одеждой, то взять и положить в свою сумку;*
2) *если своя сумка полная, то закончить поиск верхней одежды;*
3) *если корзина с вещами пуста, то закончить поиск;*
4) *иначе поместить вещь в контейнер для просмотренных вещей.*
Таблица решений для данного примера выглядит следующим образом (таблица 1):

Таблица 1

	УСЛОВИЯ	1	2	3	4	5	6	7	8
C1	id_wear(c)	Д	Н	Д	Н	Д	Н	Д	Н
C2	Full_bag()	Н	Д	Д	Д	Н	Н	Д	Н
C3	Clear_kor()	Н	Д	Н	Н	Д	Д	Д	Н
	ДЕЙСТВИЯ								
D1	Put_bag(c)	1				1			
D2	End_search()		1	1	1	2	1	1	
D3	Put_kont(c)								1

Заметим, что если выполняется условие C2, то нет необходимости в проверке условий C1 и C3. Поэтому комбинации 2,3,4 и 7 могут быть заменены обобщающей комбинацией (-,Д,-), где "-" означает любую из возможных альтернатив (в нашем случае, Д или Н). Тогда мы получим редуцированную таблицу решений:

Таблица 2

	УСЛОВИЯ	1	2	3	4	5
C1	id_wear(c)	Д	-	Д	Н	Н
C2	Full_bag()	Н	Д	Н	Н	Н
C3	Clear_kor()	Н	-	Д	Д	Н
	ДЕЙСТВИЯ					
D1	Put_bag(c)	1		1		
D2	End_search()		1	2	1	
D3	Put_kont(c)					1

Построение Таблицы Решений рекомендуется осуществлять по следующим шагам:
1. Идентифицировать все условия (или переменные) в спецификации. Идентифицировать все значения, которые каждая переменная может иметь.
2. Вычислить число комбинаций условий. Если все условия являются бинарными, то существует 2**N комбинаций N переменных.
3. Идентифицировать каждое из возможных действий, которые могут вызываться в спецификации.

4. Построить пустую таблицу, включающую все возможные условия и действия, а также номера комбинаций условий.
5. Выписать и занести в таблицу все возможные комбинации условий.
6. Редуцировать комбинации условий.
7. Проверить каждую комбинацию условий и идентифицировать соответствующие выполняемые действия.
8. Выделить комбинации условий, для которых спецификация не указывает список выполняемых действий.
9. Обсудить построенную таблицу.

Визуальные языки проектирования спецификаций

Визуальные языки проектирования базируются на основных идеях структурного программирования и позволяют определять потоки управления с помощью специальных иерархически организованных схем.

Одним из наиболее известных подходов к визуальному проектированию спецификаций является подход с использованием **FLOW-форм**. Каждый символ FLOW-формы имеет вид прямоугольника и может быть вписан в любой внутренний прямоугольник любого другого символа. Символы помечаются с помощью предложений на естественном языке или с использованием математической нотации.

Символы FLOW-форм приведены на рис. 1. Каждый символ является блоком обработки. Каждый прямоугольник внутри любого символа также представляет собой блок обработки.

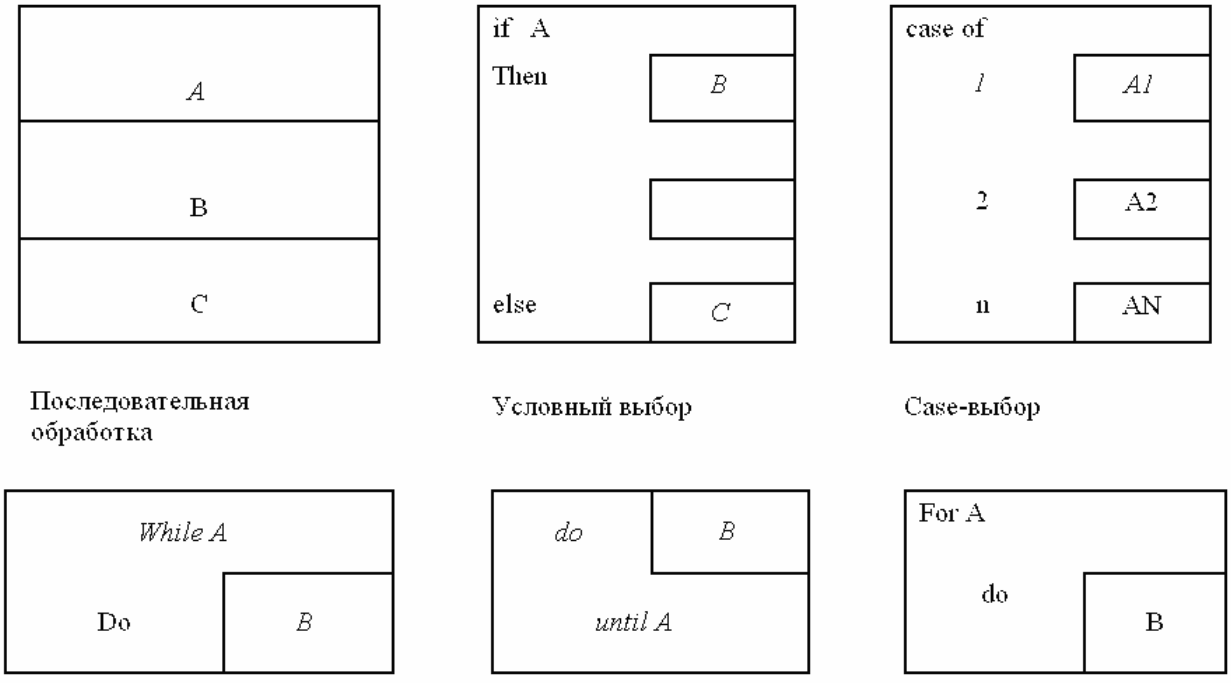


Рис. 7 Символы FLOW-форм.

Дальнейшее развитие FLOW-формы получили в диаграммах Насси-Шнейдермана. На этих диаграммах символы последовательной обработки и цикла изображаются также. В символах условного выбора и case-выбора само условие располагается в верхнем треугольнике, а вбираемые варианты - на нижних сторонах треугольника, а блоки обработки располагаются под выбираемыми вариантами. Диаграмма Насси-Шенейдермана представлена на рис.2.

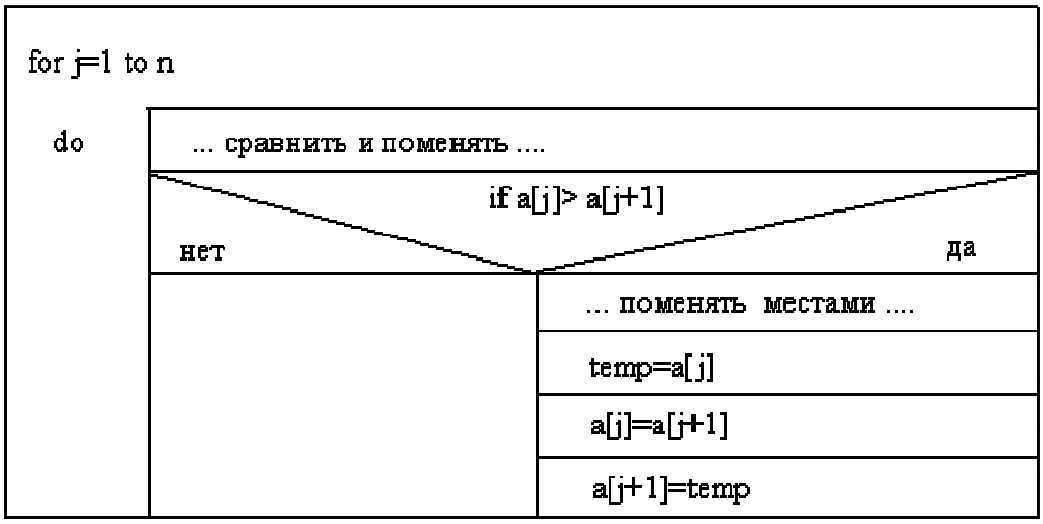


Рис. 2. Диаграмма Насси-Шнейдермана

16. Функционально-модульный подход при декомпозиции предметной области

Процесс бизнес-моделирования может быть реализован в рамках различных методик, отличающихся прежде всего своим подходом к тому, что представляет собой моделируемая организация. В соответствии с различными представлениями об организации методики принято делить на объектные и функциональные (структурные). *Объектные методики* рассматривают моделируемую организацию как набор взаимодействующих объектов – производственных единиц. Объект определяется как осязаемая реальность – предмет или явление, имеющие четко определяемое поведение. Целью применения данной методики является выделение объектов, составляющих организацию, и распределение между ними ответственностей за выполняемые действия. ***Функциональные методики***, наиболее известной из которых является методика IDEF, рассматривают организацию как набор *функций*, преобразующий поступающий поток информации в выходной поток. Процесс преобразования информации потребляет определенные ресурсы. Основное отличие от *объектной методики* заключается в четком отделении *функций* (методов обработки данных) от самих данных.

С точки зрения бизнес-моделирования каждый из представленных подходов обладает своими преимуществами. Объектный подход позволяет построить более устойчивую к изменениям систему, лучше соответствует существующим структурам организации. Функциональное моделирование хорошо показывает себя в тех случаях, когда организационная структура находится в процессе изменения или вообще слабо оформлена. Подход от выполняемых *функций* интуитивно лучше понимается исполнителями при получении от них информации об их текущей работе.

Функциональная методика IDEF0

Методологию IDEF0 можно считать следующим этапом развития хорошо известного графического языка описания функциональных систем SADT (Structured Analysis and Design Teqnique). Исторически IDEF0 как стандарт был разработан в 1981 году в рамках обширной программы автоматизации промышленных предприятий, которая носила обозначение ICAM (Integrated Computer Aided Manufacturing). Семейство стандартов IDEF унаследовало свое обозначение от названия этой программы (IDEF=Icam DEFinition), и последняя его редакция была выпущена в декабре 1993 года Национальным Институтом по Стандартам и Технологиям США (NIST).

Целью методики является построение функциональной схемы исследуемой системы, описывающей все необходимые процессы с точностью, достаточной для однозначного моделирования деятельности системы.
В основе методологии лежат четыре основных понятия: функциональный блок, интерфейсная дуга, декомпозиция, глоссарий.
Функциональный блок (Activity Box) представляет собой некоторую конкретную *функцию* в рамках рассматриваемой системы. По требованиям стандарта название каждого функционального блока должно быть сформулировано в глагольном наклонении (например, "производить услуги"). На диаграмме функциональный блок изображается прямоугольником (рис. 1). Каждая из четырех сторон функционального блока имеет свое определенное значение (роль), при этом:

- верхняя сторона имеет значение "Управление" (Control);
- левая сторона имеет значение "Вход" (Input);
- правая сторона имеет значение "Выход" (Output);
- нижняя сторона имеет значение "Механизм" (Mechanism).

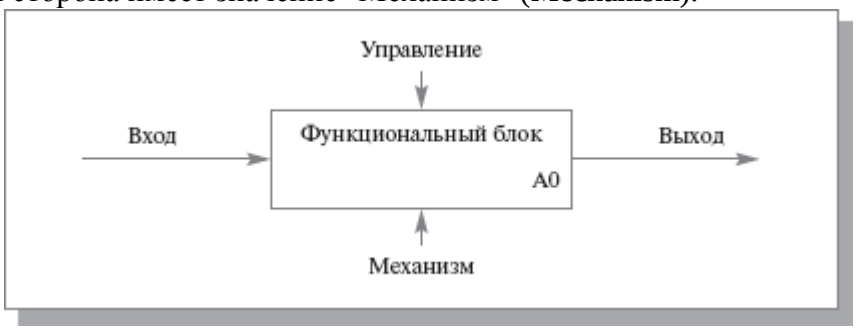


Рис. 1. Функциональный блок

Интерфейсная дуга (Arrow) отображает элемент системы, который обрабатывается функциональным блоком или оказывает иное влияние на *функцию*, представленную данным функциональным блоком. Интерфейсные дуги часто называют потоками или стрелками.
С помощью интерфейсных дуг отображают различные объекты, в той или иной степени определяющие процессы, происходящие в системе. Такими объектами могут быть элементы реального мира (детали, вагоны, сотрудники и т.д.) или потоки данных и информации (документы, данные, инструкции и т.д.).
В зависимости от того, к какой из сторон функционального блока подходит данная интерфейсная дуга, она носит название "входящей", "исходящей" или "управляющей". Необходимо отметить, что любой функциональный блок по требованиям стандарта должен иметь, по крайней мере, одну управляющую интерфейсную дугу и одну исходящую. Это и понятно – каждый процесс должен происходить по каким-то правилам (отображаемым управляющей дугой) и должен выдавать некоторый результат (выходящая дуга), иначе его рассмотрение не имеет никакого смысла.
Обязательное наличие управляющих интерфейсных дуг является одним из главных отличий стандарта IDEF0 от других методологий классов DFD (Data Flow Diagram) и WFD (Work Flow Diagram).
Декомпозиция (Decomposition) является основным понятием стандарта IDEF0. Принцип декомпозиции применяется при разбиении сложного процесса на составляющие его *функции*. При этом уровень детализации процесса определяется непосредственно разработчиком модели.
Декомпозиция позволяет постепенно и структурированно представлять модель системы в виде иерархической структуры отдельных диаграмм, что делает ее менее перегруженной и легко усваиваемой.
Последним из понятий IDEF0 является глоссарий (Glossary). Для каждого из элементов IDEF0 — диаграмм, функциональных блоков, интерфейсных дуг — существующий стандарт подразумевает создание и поддержание набора соответствующих определений,

ключевых слов, повествовательных изложений и т.д., которые характеризуют объект, отображенный данным элементом. Этот набор называется глоссарием и является описанием сущности данного элемента. Глоссарий гармонично дополняет наглядный графический язык, снабжая диаграммы необходимой дополнительной информацией.
Модель IDEF0 всегда начинается с представления системы как единого целого – одного функционального блока с интерфейсными дугами, простирающимися за пределы рассматриваемой области. Такая диаграмма с одним функциональным блоком называется контекстной диаграммой.
В пояснительном тексте к контекстной диаграмме должна быть указана цель (Purpose) построения диаграммы в виде краткого описания и зафиксирована точка зрения (Viewpoint).
Определение и формализация цели разработки IDEF0-модели является крайне важным моментом. Фактически цель определяет соответствующие области в исследуемой системе, на которых необходимо фокусироваться в первую очередь.
Точка зрения определяет основное направление развития модели и уровень необходимой детализации. Четкое фиксирование точки зрения позволяет разгрузить модель, отказавшись от детализации и исследования отдельных элементов, не являющихся необходимыми, исходя из выбранной точки зрения на систему. Правильный выбор точки зрения существенно сокращает временные затраты на построение конечной модели.
Выделение *подпроцессов*. В процессе декомпозиции функциональный блок, который в контекстной диаграмме отображает систему как единое целое, подвергается детализации на другой диаграмме. Получившаяся диаграмма второго уровня содержит функциональные блоки, отображающие главные подфункции функционального блока контекстной диаграммы, и называется дочерней (Child Diagram) по отношению к нему (каждый из функциональных блоков, принадлежащих дочерней диаграмме, соответственно называется дочерним блоком – Child Box). В свою очередь, функциональный блок — предок называется родительским блоком по отношению к дочерней диаграмме (Parent Box), а диаграмма, к которой он принадлежит – родительской диаграммой (Parent Diagram). Каждая из подфункций дочерней диаграммы может быть далее детализирована путем аналогичной декомпозиции соответствующего ей функционального блока. В каждом случае декомпозиции функционального блока все интерфейсные дуги, входящие в данный блок или исходящие из него, фиксируются на дочерней диаграмме. Этим достигается структурная целостность IDEF0–модели.
Иногда отдельные интерфейсные дуги высшего уровня не имеет смысла продолжать рассматривать на диаграммах нижнего уровня, или наоборот — отдельные дуги нижнего отражать на диаграммах более высоких уровней – это будет только перегружать диаграммы и делать их сложными для восприятия. Для решения подобных задач в стандарте IDEF0 предусмотрено понятие туннелирования. Обозначение "туннеля" (Arrow Tunnel) в виде двух круглых скобок вокруг начала интерфейсной дуги обозначает, что эта дуга не была унаследована от функционального родительского блока и появилась (из "туннеля") только на этой диаграмме. В свою очередь, такое же обозначение вокруг конца (стрелки) интерфейсной дуги в непосредственной близи от блока–приемника означает тот факт, что в дочерней по отношению к этому блоку диаграмме эта дуга отображаться и рассматриваться не будет. Чаще всего бывает, что отдельные объекты и соответствующие им интерфейсные дуги не рассматриваются на некоторых промежуточных уровнях иерархии, – в таком случае они сначала "погружаются в туннель", а затем при необходимости "возвращаются из туннеля".
Обычно IDEF0-модели несут в себе сложную и концентрированную информацию, и для того, чтобы ограничить их перегруженность и сделать удобочитаемыми, в стандарте приняты соответствующие ограничения сложности.

Рекомендуется представлять на диаграмме от трех до шести функциональных блоков, при этом количество подходящих к одному функциональному блоку (выходящих из одного функционального блока) интерфейсных дуг предполагается не более четырех.
Стандарт IDEF0 содержит набор процедур, позволяющих разрабатывать и согласовывать модель большой группой людей, принадлежащих к разным областям деятельности моделируемой системы. Обычно процесс разработки является итеративным и состоит из следующих условных этапов:

- Создание модели группой специалистов, относящихся к различным сферам деятельности предприятия. Эта группа в терминах IDEF0 называется авторами (Authors). Построение первоначальной модели является динамическим процессом, в течение которого авторы опрашивают компетентных лиц о структуре различных процессов, создавая модели деятельности подразделений. При этом их интересуют ответы на следующие вопросы:

Что поступает в подразделение "на входе"?

- Какие *функции* и в какой последовательности выполняются в рамках подразделения?
- Кто является ответственным за выполнение каждой из *функций*?
- Чем руководствуется исполнитель при выполнении каждой из *функций*?
- Что является результатом работы подразделения (на выходе)?

На основе имеющихся положений, документов и результатов опросов создается черновик (Model Draft) модели.

- Распространение черновика для рассмотрения, согласований и комментариев. На этой стадии происходит обсуждение черновика модели с широким кругом компетентных лиц (в терминах IDEF0 — читателей) на предприятии. При этом каждая из диаграмм черновой модели письменно критикуется и комментируется, а затем передается автору. Автор, в свою очередь, также письменно соглашается с критикой или отвергает ее с изложением логики принятия решения и вновь возвращает откорректированный черновик для дальнейшего рассмотрения. Этот цикл продолжается до тех пор, пока авторы и читатели не придут к единому мнению.
- Официальное утверждение модели. Утверждение согласованной модели происходит руководителем рабочей группы в том случае, если у авторов модели и читателей отсутствуют разногласия по поводу ее адекватности. Окончательная модель представляет собой согласованное представление о предприятии (системе) с заданной точки зрения и для заданной цели.

Наглядность графического языка IDEF0 делает модель вполне читаемой и для лиц, которые не принимали участия в проекте ее создания, а также эффективной для проведения показов и презентаций. В дальнейшем на базе построенной модели могут быть организованы новые проекты, нацеленные на производство изменений в модели.

17. Объектно-ориентированный подход при декомпозиции предметной области. Принципы построения объектной модели

Объектно-ориентированная методика

Принципиальное отличие между функциональным и объектным подходом заключается в способе декомпозиции системы. Объектно-ориентированный подход использует **объектную** декомпозицию, при этом статическая структура описывается в терминах **объектов и связей** между ними, а поведение системы описывается в терминах **обмена сообщениями** между объектами. Целью методики является построение **бизнес-модели** организации, позволяющей перейти от модели сценариев использования к модели, определяющей отдельные объекты, участвующие в реализации бизнес-функций.
Концептуальной основой объектно-ориентированного подхода является **объектная модель**, которая строится с учетом следующих принципов:

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

Основными понятиями объектно-ориентированного подхода являются объект и класс. **Объект — предмет или явление, имеющее четко определенное поведение и обладающие состоянием, поведением и индивидуальностью.** Структура и поведение схожих объектов определяют общий для них класс. **Класс – это множество объектов, связанных общностью структуры и поведения.** Следующую группу важных понятий объектного подхода составляют **наследование и полиморфизм**.
Наследование означает построение новых классов на основе существующих с возможностью добавления или переопределения данных и методов.
Для классов объектов характерна **иерархия обобщения**, позволяющая осуществлять **наследование** не только атрибутов (свойств) объектов от вышестоящего класса объектов к нижестоящему классу, но и *функций* (методов).
Понятие **полиморфизм** может быть интерпретировано как способность класса принадлежать более чем одному типу.
В случае наследования **функций** можно абстрагироваться от конкретной реализации процедур (**абстрактные типы данных**), которые отличаются для определенных подклассов ситуаций. Это дает возможность обращаться к подобным программным модулям по общим именам (**полиморфизм**) и осуществлять повторное использование программного кода при модификации программного обеспечения. Таким образом, адаптивность объектно-ориентированных систем к изменению предметной области по сравнению с функциональным подходом значительно выше.
Важным качеством объектного подхода является согласованность моделей деятельности организации и моделей проектируемой информационной системы от стадии формирования требований до стадии реализации. По объектным моделям может быть прослежено отображение реальных сущностей моделируемой предметной области (организации) в объекты и классы информационной системы.
Большинство существующих методов объектно-ориентированного подхода включают **язык моделирования** и **описание процесса моделирования**.
Процесс – это описание шагов, которые необходимо выполнить при разработке проекта. В качестве *языка моделирования* объектного подхода используется унифицированный язык моделирования **UML**, который содержит стандартный набор диаграмм для моделирования.
Объектно-ориентированный подход обладает следующими преимуществами:

- Объектная декомпозиция дает возможность создавать модели меньшего размера путем использования общих механизмов, обеспечивающих необходимую экономию выразительных средств. Использование объектного подхода существенно повышает уровень унификации разработки и пригодность для повторного использования, что ведет к созданию среды разработки и переходу к сборочному созданию моделей.
- Объектная декомпозиция позволяет избежать создания сложных моделей, так как она предполагает эволюционный путь развития модели на базе относительно небольших подсистем.
- Объектная модель естественна, поскольку ориентированна на человеческое восприятие мира.

К недостаткам объектно-ориентированного подхода относятся высокие начальные затраты. Этот подход не дает немедленной отдачи. Эффект от его применения сказывается после разработки двух–трех проектов и накопления повторно используемых компонентов. Диаграммы, отражающие специфику объектного подхода, менее наглядны.
При объектно-ориентированном подходе изменяется и принцип проектирования ИС. Сначала выделяются классы объектов, а далее в зависимости от возможных состояний

объектов (жизненного цикла объектов) определяются методы обработки (функциональные процедуры), что обеспечивает наилучшую реализацию динамического поведения информационной системы.
Объектно-ориентированная технология проектирования ИС включает в себя следующие компоненты:
• технологию конструирования концептуальной объектно-ориентированной модели предметной области;
• инструментальные средства спецификации проектных решений;
• библиотеки типовых компонентов модели предметной области;
• типовые проектные решения для ряда функциональных областей.
В основу объектно-ориентированной технологии проектирования ИС положены разработка, анализ и спецификация концептуальной объектно-ориентированной модели предметной области.
Концептуальная объектно-ориентированная модель предметной области является основой проекта и реализации системы и обеспечивает:
• необходимый уровень формализации описания проектных решений;
• высокий уровень абстрагирования, типизации и параметризации проектных решений;
• компактность описания;
• удобство сопровождения готовой системы.
Отличительными чертами предлагаемой методологии являются следующие:
• наличие единого методологически обоснованного ядра, обеспечивающего открытость технологии для модификации, расширения и создания новых моделей представления проектных решений;
• наличие единого формального аппарата анализа проектных решений для используемых моделей представления;
Отличительными чертами предлагаемой технологии являются:
• совместное рассмотрение информационных, материальных и финансовых потоков;
• первичная и вторичная классификация объектов предметной области с обязательным указанием оснований классификации;
• наличие конструктивных методик декомпозиции и агрегирования компонентов проекта, использующих результаты классификации;
• наличие формальных методов оценки связности и сцепления компонентов проекта;
• использование функциональной модели данных с атрибутами — функциями доступа и атрибутами — категориями в качестве основы концептуальной модели данных.
Объектно-ориентированный подход обладает следующими **преимуществами**:

- Объектная декомпозиция дает возможность создавать модели меньшего размера путем использования общих механизмов, обеспечивающих необходимую экономию выразительных средств. Использование объектного подхода существенно повышает уровень унификации разработки и пригодность для повторного использования, что ведет к созданию среды разработки и переходу к сборочному созданию моделей.
- . «Чистый» объектный подход (Гради Буч) уже на ранних стадиях требует представлять данные о классификации в виде диаграмм классов. Это слишком жесткое требование.
- Выделение иерархии классов требует проведения объемного и тонкого анализа различных аспектов взаимосвязей объектов предметной области. В рамках самого объектного подхода подобных методик нет.
- Объектная декомпозиция позволяет избежать создания сложных моделей, так как она предполагает эволюционный путь развития модели на базе относительно небольших подсистем.
- Объектная модель естественна, поскольку ориентированна на человеческое восприятие мира.

К **недостаткам** объектно-ориентированного подхода можно отнести:

- высокие начальные затраты; этот подход не дает немедленной отдачи, эффект от его применения сказывается после разработки двух–трех проектов и накопления повторно используемых компонентов;
- по наглядности представления модели пользователю-заказчику объектно-ориентированные модели явно уступают функциональным моделям.

В настоящее время господствующим направлением проектирования ИС является объектно-ориентированная технология как основа создания открытых, гибких, многофункциональных систем для различных предметных областей.

Объектно-ориентированная технология проектирования ИС предоставляет мощную, гибкую, универсальную концептуальную основу для конструирования информационно-управляющих систем в различных областях хозяйственной деятельности и управления, сочетающую использование моделей современной логистики, объектного подхода к компонентам предметной области, современных инструментальных средств визуального программирования и СУБД с SQL-интерфейсом.

Объектно-ориентированная технология проектирования ИС включает в себя следующие компоненты:

• технологию конструирования концептуальной объектно-ориентированной модели предметной области;
• инструментальные средства спецификации проектных решений;
• библиотеки типовых компонентов модели предметной области;
• типовые проектные решения для ряда функциональных областей.

В основу объектно-ориентированной технологии проектирования ИС положены разработка, анализ и спецификация концептуальной объектно-ориентированной модели предметной области.

Концептуальная объектно-ориентированная модель предметной области является основой проекта и реализации системы и обеспечивает:

• необходимый уровень формализации описания проектных решений;
• высокий уровень абстрагирования, типизации и параметризации проектных решений;
• компактность описания;
• удобство сопровождения готовой системы.

Отличительными чертами предлагаемой методологии являются следующие:

• наличие единого методологически обоснованного ядра, обеспечивающего открытость технологии для модификации, расширения и создания новых моделей представления проектных решений;
• наличие единого формального аппарата анализа проектных решений для используемых моделей представления;

Отличительными чертами предлагаемой технологии являются:

• совместное рассмотрение информационных, материальных и финансовых потоков;
• первичная и вторичная классификация объектов предметной области с обязательным указанием оснований классификации;
• наличие конструктивных методик декомпозиции и агрегирования компонентов проекта, использующих результаты классификации;
• наличие формальных методов оценки связности и сцепления компонентов проекта;
• использование функциональной модели данных с атрибутами — функциями доступа и атрибутами — категориями в качестве основы концептуальной модели данных.

При всем разнообразии моделей предметных областей концептуального уровня отсутствуют такие модели, которые бы позволяли в полной мере использовать знания по классификации элементов предметной области для описания свойств ее элементов, и в то же время, сохраняли преимущества традиционных функционального и информационного подходов, основанных на модели данных. «Чистый» объектный подход (Гради Буч) уже на ранних стадиях требует представлять данные о классификации в виде диаграмм классов. Это слишком жесткое требование. Выделение иерархии классов требует

проведения объемного и тонкого анализа различных аспектов взаимосвязей объектов предметной области. В рамках самого объектного подхода подобных методик нет. С другой стороны, попытки совместить чистый объектный подход с традиционными подходами (Салли Шлеер) оказываются неудачными, так как последние рассматриваются не как обоснование решений объектного подхода, а как средство моделирования последнего.

18. Состав и требования к информационному обеспечению ИС

Информационное обеспечение ИС включает два комплекса: ***внемашинное*** информационное обеспечение (*классификаторы* технико-экономической информации, документы, методические инструктивные материалы) и ***внутримашинное*** информационное обеспечение (макеты/экранные формы для ввода первичных данных в ЭВМ или вывода результатной информации, структуры *информационной базы*: входных, выходных файлов, базы данных).

Таким образом, Информационное обеспечение ИС можно определить как совокупность единой **системы классификации**, унифицированной **системы документации** и **информационной базы**.

К информационному обеспечению предъявляются следующие общие требования:

- информационное обеспечение должно быть достаточным для поддержания всех автоматизируемых функций объекта;
- для кодирования информации должны использоваться принятые у заказчика *классификаторы*;
- для кодирования входной и выходной информации, которая используется на высшем уровне управления, должны быть использованы *классификаторы* этого уровня;
- должна быть обеспечена совместимость с информационным обеспечением систем, взаимодействующих с разрабатываемой системой;
- формы документов должны отвечать требованиям корпоративных стандартов заказчика (или унифицированной *системы документации*);
- структура документов и экранных форм должна соответствовать характеристиками терминалов на рабочих местах конечных пользователей;
- графики формирования и содержание информационных сообщений, а также используемые аббревиатуры должны быть общеприняты в этой предметной области и согласованы с заказчиком;
- в ИС должны быть предусмотрены средства контроля входной и результатной информации, обновления данных в информационных массивах, контроля целостности *информационной базы*, защиты от несанкционированного доступа.

Основной компонентой ***внемашинного*** информационного обеспечения ИС является система документации, применяемая в процессе управления экономическим объектом. Под документом понимается определенная совокупность сведений, используемая при решении технико-экономических задач, расположенная на материальном носителе в соответствии с установленной формой.

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

Отличительной особенностью системы экономической документации является большое разнообразие видов документов.

Существующие системы документации, характерные для неавтоматизированных ИС, отличаются большим количеством разных типов форм документов, большим объемом потоков документов и их запутанностью, дублированием информации в документах и работ по их обработке и, как следствие, низкой достоверностью получаемых результатов.

Для того чтобы упростить систему документации, используют следующие два подхода:

- проведение унификации и стандартизации документов;

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

Унификация документов выполняется путем введения единых форм документов. Таким образом, вводится единообразие в наименования показателей, единиц измерения и терминов, в результате чего получается унифицированная система документации.
Унифицированная система документации (УСД) — это рационально организованный комплекс взаимосвязанных документов, который отвечает единым правилам и требованиям и содержит информацию, необходимую для управления некоторым экономическим объектом
Любой тип УСД должен удовлетворять следующим требованиям:

- документы, входящие в состав УСД, должны разрабатываться с учетом их использования в системе взаимосвязанных ЭИС;
- УСД должна содержать полную информацию, необходимую для оптимального управления тем объектом, для которого разрабатывается эта система;
- УСД должна быть ориентирована на использование средств вычислительной техники для сбора, обработки и передачи информации;
- УСД должна обеспечить информационную совместимость ЭИС различных уровней;
- все документы, входящие в состав разрабатываемой УСД, и все реквизиты-признаки в них должны быть закодированы с использованием международных, общесистемных или локальных ***классификаторов***.

Основной частью **внутримашнного** информационного обеспечения является ***информационная база***. *Информационная база* (ИБ) — это совокупность данных, организованная определенным способом и хранимая в памяти вычислительной системы в виде файлов, с помощью которых удовлетворяются информационные потребности управленческих процессов и решаемых задач.
Все файлы ИБ можно классифицировать по следующим признакам:

- **по этапам обработки** (входные, базовые, результатные);
- **по типу носителя** (на промежуточных носителях — гибких магнитных дисках и магнитных лентах и на основных носителях — жестких магнитных дисках, магнитооптических дисках и др.);
- **по составу информации** (файлы с оперативной информацией и файлы с постоянной информацией);
- **по назначению** (по типу функциональных подсистем);
- **по типу логической организации** (файлы с линейной и иерархической структурой записи, реляционные, табличные);
- **по способу физической организации** (файлы с последовательным, индексным и прямым способом доступа).

Входные файлы создаются с первичных документов для ввода данных или обновления базовых файлов.
Файлы с **результатной информацией** предназначаются для вывода ее на печать или передачи по каналам связи и не подлежат долговременному хранению.
К числу **базовых файлов**, хранящихся в *информационной базе*, относят основные, рабочие, промежуточные, служебные и архивные файлы.
Основные файлы должны иметь однородную структуру записей и могут содержать записи с оперативной и условно-постоянной информацией. **Оперативные файлы** могут создаваться на базе одного или нескольких входных файлов и отражать информацию одного или нескольких первичных документов. **Файлы с условно-постоянной информацией** могут содержать справочную, расценочную, табличную и другие виды информации, изменяющейся в течение года не более чем на 40%, а следовательно, имеющие коэффициент стабильности не менее 0,6.
Файлы **со справочной информацией** должны отражать все характеристики элементов материального производства (материалы, сырье, основные фонды, трудовые ресурсы и

т.п.). Как правило, справочники содержат информацию *классификаторов* и дополнительные сведения об элементах Материальной сферы, например о ценах. Нормативно-расценочные файлы должны содержать данные о нормах расхода и расценках на выполнение операций и услуг. Табличные файлы содержат сведения об экономических показателях, считающихся постоянными в течение длительного времени (например, процент удержания, отчисления и пр.). Плановые файлы содержат плановые показатели, хранящиеся весь плановый период.

Рабочие файлы создаются для решения конкретных задач на базе основных файлов путем выборки части информации из нескольких основных файлов с целью сокращения времени обработки данных.

Промежуточные файлы отличаются от рабочих файлов тем, что они образуются в результате решения экономических задач, подвергаются хранению с целью дальнейшего использования для решения других задач. Эти файлы, так же как и рабочие файлы, при высокой частоте обращений могут быть также переведены в категорию основных файлов.

Служебные файлы предназначаются для ускорения поиска информации в основных файлах и включают в себя справочники, индексные файлы и каталоги.

Архивные файлы содержат ретроспективные данные из основных файлов, которые используются для решения аналитических, например прогнозных, задач. Архивные данные могут также использоваться для восстановления *информационной базы* при разрушениях.

Организация хранения файлов в *информационной базе* должна отвечать следующим требованиям:

- полнота хранимой информации для выполнения всех функций управления и решения экономических задач;
- целостность хранимой информации, т. е. обеспечение непротиворечивости данных при вводе информации в ИБ;
- своевременность и одновременность обновления данных во всех копиях данных;
- гибкость системы, т.е. адаптируемость ИБ к изменяющимся информационным потребностям;
- реализуемость системы, обеспечивающая требуемую степень сложности структуры ИБ;
- релевантность ИБ, под которой подразумевается способность системы осуществлять поиск и выдавать информацию, точно соответствующую запросам пользователей;
- удобство языкового интерфейса, позволяющее быстро формулировать запрос к ИБ;
- разграничение прав доступа, т.е. определение для каждого пользователя доступных типов записей, полей, файлов и видов операций над ними.

Существуют следующие **способы организации ИБ**:

- совокупность ***локальных файлов***, поддерживаемых функциональными пакетами прикладных программ;
- ***интегрированная база данных***, основывающаяся на использовании универсальных программных средств загрузки, хранения, поиска и ведения данных, т.е. системы управления базами данных (СУБД).

Локальные файлы вследствие специализации структуры данных под задачи обеспечивают, как правило, более быстрое время обработки данных. Однако недостатки организации локальных файлов, связанные с большим дублированием данных в информационной системе и, как следствие, несогласованностью данных в разных приложениях, а также негибкостью доступа к информации, перекрывают указанные преимущества. Поэтому организация локальных файлов может применяться только в специализированных приложениях, требующих очень высокой скорости реакции при импорте необходимых данных.

Интегрированная ИБ, т.е. база данных (БД) — это совокупность взаимосвязанных, хранящихся вместе данных при такой минимальной избыточности, которая допускает их использование оптимальным образом для множества приложений.
Централизация управления данными с помощью СУБД обеспечивает совместимость этих данных, уменьшение синтаксической и семантической избыточности, соответствие данных реальному состоянию объекта, разделение хранения данных между пользователями и возможность подключения новых пользователей. Но централизация управления и интеграция данных приводят к проблемам другого характера: необходимости усиления контроля вводимых данных, необходимости обеспечения соглашения между пользователями по поводу состава и структуры данных, разграничения доступа и секретности данных.
Основными способами организации БД являются создание централизованных и распределенных БД. Основным критерием выбора способа организации ИБ является достижение минимальных трудовых и стоимостных затрат на проектирование структуры ИБ, программного обеспечения системы ведения файлов, а также на перепроектирование ИБ при возникновении новых задач.
Кроме ИБ***, внутримашинное*** информационное обеспечение включает макеты (экранные формы) для ввода первичных данных в ЭВМ или вывода результатной информации, и структуры информационной базы: входных, выходных файлов, базы данных.
Под электронными формами документов понимается не изображение бумажного документа, а изначально электронная (безбумажная) технология работы; она предполагает появление бумажной формы только в качестве твердой копии документа.
Электронная форма документа (ЭД) — это страница с пустыми полями, оставленными для заполнения пользователем. Формы могут допускать различный тип входной информации и содержать командные кнопки, переключатели, выпадающие меню или списки для выбора.
Технология обработки электронных документов требует использования специализированного программного обеспечения — программ управления документооборотом, которые зачастую встраиваются в корпоративные ИС.
Проектирование форм электронных документов, т.е. создание шаблона формы с помощью программного обеспечения проектирования форм, обычно включает в себя выполнение следующих шагов:

- создание структуры ЭД — подготовка внешнего вида с помощью графических средств проектирования;
- определение содержания формы ЭД, т.е. выбор способов, которыми будут заполняться поля. Поля могут быть заполнены вручную или посредством выбора значений из какого-либо списка, меню, базы данных;
- определения перечня макетов экранных форм — по каждой задаче проектировщик анализирует «постановку» каждой задачи, в которой приводятся перечни используемых входных документов с оперативной и постоянной информацией и документов с результатной информацией;
- определение содержания макетов — выполняется на основе анализа состава реквизитов первичных документов с постоянной и оперативной информацией и результатных документов.

Работа заканчивается программированием разработанных макетов экранных форм и их апробацией.
К недостаткам электронных документов можно отнести неполную юридическую проработку процесса их утверждения или подписания.

19. Внемашинное информационное обеспечение. Основные понятия и способы классификации информации

Основной компонентой **внемашинного информационного обеспечения** ИС является **система документации**, применяемая в процессе управления экономическим объектом.
Под **документом** понимается определенная совокупность сведений, используемая при решении технико-экономических задач, расположенная на материальном носителе в соответствии с установленной формой.
Система документации — это совокупность взаимосвязанных форм документов, регулярно используемых в процессе управления экономическим объектом. Отличительной особенностью системы экономической документации является большое разнообразие видов документов.
Существующие системы документации, характерные для неавтоматизированных ИС, отличаются большим количеством разных типов форм документов, большим объемом потоков документов и их запутанностью, дублированием информации в документах и работ по их обработке и, как следствие, низкой достоверностью получаемых результатов. Для того чтобы упростить систему документации, используют следующие два подхода:

- проведение **унификации** и **стандартизации** документов;
- введение **безбумажной технологии**, основанной на использовании электронных документов и новых информационных технологий их обработки.

Унификация документов выполняется путем введения единых форм документов. Таким образом, вводится единообразие в наименования показателей, единиц измерения и терминов, в результате чего получается *унифицированная* система документации.
Унифицированная система документации (УСД) — это рационально организованный комплекс взаимосвязанных документов, который отвечает единым правилам и требованиям и содержит информацию, необходимую для управления некоторым экономическим объектом. По уровням управления, они делятся на *межотраслевые* и *отраслевые* системы документации, и системы документации *локального* уровня, т. е. обязательные для использования в рамках предприятий или организаций.
Любой тип УСД должен удовлетворять следующим требованиям:

- документы, входящие в состав УСД, должны разрабатываться с учетом их использования в системе взаимосвязанных ЭИС;
- УСД должна содержать полную информацию, необходимую для оптимального управления тем объектом, для которого разрабатывается эта система;
- УСД должна быть ориентирована на использование средств вычислительной техники для сбора, обработки и передачи информации;
- УСД должна обеспечить информационную совместимость ЭИС различных уровней;
- все документы, входящие в состав разрабатываемой УСД, и все реквизиты-признаки в них должны быть закодированы с использованием международных, общесистемных или локальных ***классификаторов***.

Для того чтобы обеспечить эффективный поиск, обработку на ЭВМ и передачу по каналам связи технико-экономической информации, ее необходимо представить в цифровом виде. С этой целью ее нужно сначала упорядочить (классифицировать), а затем формализовать (закодировать) с использованием **классификатора**.
Классификация – это разделение множества объектов на подмножества по их сходству или различию в соответствии с принятыми методами. Классификация фиксирует закономерные связи между классами объектов. Под объектом понимается любой предмет, процесс, явление материального или нематериального свойства.
Система классификации позволяет сгруппировать объекты и выделить определенные классы, которые будут характеризоваться рядом общих свойств. Таким образом, система классификации - это совокупность правил распределения объектов множества на подмножества.

Признак классификации - свойство или характеристика объекта классификации, которое позволяет установить его сходство или различие с другими объектами классификации. Например, признак «роль предприятия-партнера в отношении деятельности объекта автоматизации» позволяет разделить все предприятия на две группы (на два подмножества): «поставщики» и «потребители».

Классификационная группировка - множество или подмножество, объединяющее часть объектов классификации по одному или нескольким признакам.

Классификатор — это документ, с помощью которого осуществляется формализованное описание информации в ИС, содержащей наименования объектов, наименования классификационных группировок и их кодовые обозначения.

По сфере действия выделяют следующие виды *классификаторов*: *международные*, *общегосударственные* (*общесистемные*), *отраслевые* и *локальные*.

- **Международные классификаторы** входят в состав Системы международных экономических стандартов (СМЭС) и обязательны для передачи информации между организациями разных стран мирового сообщества.
- **Общегосударственные (общесистемные) классификаторы** обязательны для организации процессов передачи и обработки информации между экономическими системами государственного уровня внутри страны.
- **Отраслевые классификаторы** используют для выполнения процедур обработки информации и передачи ее между организациями внутри отрасли.
- **Локальные классификаторы** используют в пределах отдельных предприятий.

Каждая система классификации характеризуется следующими свойствами:

- гибкостью системы;
- емкостью системы;
- степенью заполненности системы.

Гибкость системы — это способность допускать включение новых признаков, объектов без разрушения структуры классификатора. Необходимая гибкость определяется временем жизни системы.

Емкость системы — это наибольшее количество классификационных группировок, допускаемое в данной *системе классификации*.

Степень заполненности системы определяется как частное от деления фактического количества группировок на величину емкости системы.

В настоящее время чаще всего применяются два типа систем классификации: *иерархическая* и *многоаспектная*.

При использовании **иерархического** метода классификации происходит «последовательное разделение множества объектов на подчиненные, зависимые классификационные группировки». Получаемая на основе этого процесса классификационная схема имеет иерархическую структуру. В ней первоначальный объем классифицируемых объектов разбивается на подмножества по какому-либо признаку и детализируется на каждой следующей ступени *классификации*. Обобщенное изображение иерархической классификационной схемы представлено на рис.**9.**1.

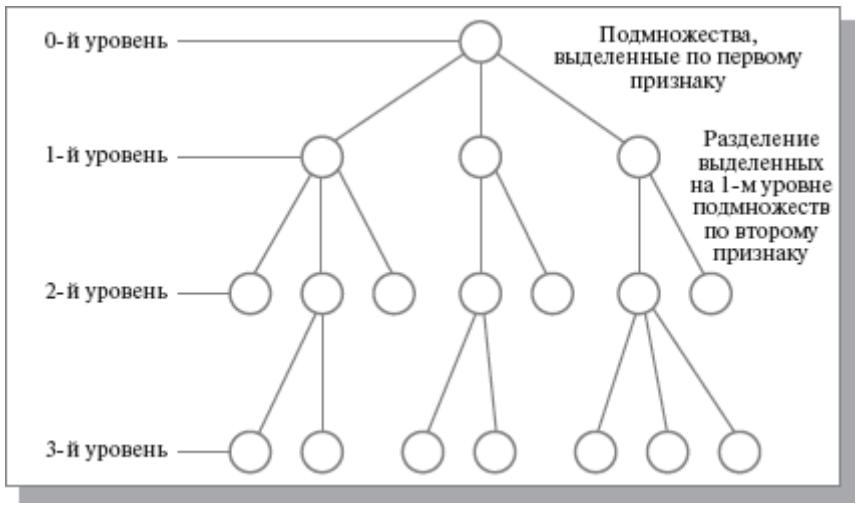


Рис. 1. Иерархическая классификационная схема

Характерными особенностями иерархической системы являются:

- возможность использования неограниченного количества признаков классификации;
- соподчиненность признаков классификации, что выражается разбиением каждой классификационной группировки, образованной по одному признаку, на множество классификационных группировок по нижестоящему (подчиненному) признаку.

Таким образом, классификационные схемы, построенные на основе иерархического принципа, имеют неограниченную емкость, величина которой зависит от **глубины** классификации (числа ступеней деления) и **количества объектов** классификации, которое можно расположить на каждой ступени. Количество же объектов на каждой ступени классификации определяется основанием кода, то есть числом знаков в выбранном алфавите кода. (Например, если алфавит – двузначные десятичные цифры, то можно на одном уровне разместить 100 объектов). Выбор необходимой глубины классификации и структуры кода зависит от характера объектов классификации и характера задач, для решения которых предназначен классификатор.

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

Серьезным **недостатком** иерархического метода классификации является жесткость классификационной схемы. Она обусловлена заранее установленным выбором признаков классификации и порядком их использования по ступеням классификации. Это ведет к тому, что при изменении состава объектов классификации, их характеристик или характера решаемых при помощи классификатора задач требуется коренная переработка классификационной схемы. Гибкость этой системы обеспечивается только за счет ввода большой избыточности в ветвях, что приводит к слабой заполненности структуры классификатора. Поэтому при разработке классификаторов следует учитывать, что иерархический метод классификации более предпочтителен для объектов с относительно стабильными признаками и для решения стабильного комплекса задач.

Пример применения иерархической классификации объектов в корпоративной ИС приведены на рис. 2. Использование приведенной модели позволяет выполнить кодирование информации о соответствующих объектах, а также использовать процедуры обобщения при обработке данных (при анализе затрат на заработную плату — по принадлежности работника к определенной службе, при анализе затрат на производство — по группам материалов: по металлу, по покупным комплектующим и пр.).

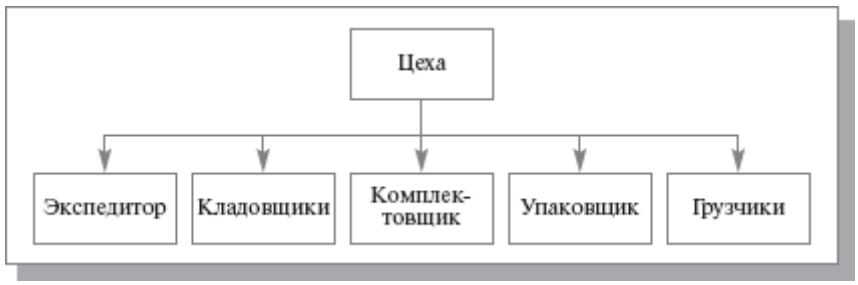


Рис. 2. Организационная структура подразделения предприятия-цеха отгрузки

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

Фасетная система
Фасет — это аспект классификации, который используется для образования независимых классификационных группировок.
Под **фасетным методом** классификации понимается «параллельное разделение множества объектов на независимые классификационные группировки». При этом методе классификации заранее жесткой классификационной схемы и конечных группировок не создается. Разрабатывается лишь система таблиц признаков объектов классификации, называемых фасетами. При необходимости создания классификационной группировки для решения конкретной задачи осуществляется выборка необходимых признаков из фасетов и их объединение в определенной последовательности. Общий вид фасетной классификационной схемы представлен на рис. 3.

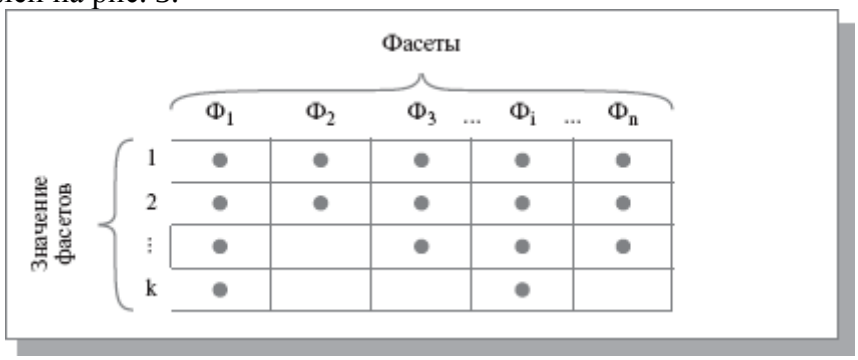


Рис. 3. Схема признаков фасетной классификации
Внутри фасета значения признаков могут просто перечисляться по некоторому порядку или образовывать сложную иерархическую структуру, если существует соподчиненность выделенных признаков.
К **преимуществам** данной системы следует отнести большую емкость системы и высокую степень гибкости, поскольку при необходимости можно вводить дополнительные фасеты и изменять их место в формуле. При изменении характера задач или характеристик объектов классификации разрабатываются новые фасеты или дополняются новыми признаками уже существующие фасеты без коренной перестройки структуры всего классификатора.
К **недостаткам**, характерным для данной системы, можно отнести сложность структуры и низкую степень заполненности системы.

Дескрипторная система
Дескриптор — ключевое слово, определяющее некоторое понятие, которое формирует описание объекта и дает принадлежность этого объекта к классу, группе и т.д.; это термин естественного языка (слово или словосочетание), используемый при описании документов

или показателей, который имеет самостоятельный смысл и неделим без изменения своего значения. С помощью списка дескрипторов можно достаточно полно и точно отразить содержание документов или показателей.
Для поиска показателей и документов по набору содержательных признаков используется информационный **язык дескрипторного типа**, которой характеризуется совокупностью терминов (дескрипторов) и набором отношений между терминами.
Для того чтобы обеспечить точность и однозначность поиска с помощью дескрипторного языка, необходимо предварительно определить все постоянные отношения между терминами: родовидовые, отношения синонимии, омонимии и полисемии, а также ассоциативные отношения.
Все выделенные отношения явно описываются в систематическом словаре понятий — тезаурусе, который разрабатывается с целью проведения индексирования документов, показателей и информационных запросов.
В современных классификационных схемах часто одновременно используются оба метода классификации. Это снижает влияние недостатков методов классификации и расширяет возможность использования в информационном обеспечении управления.
В качестве примера использования комбинированных схем классификации в корпоративных ИС можно привести следующую модель описания продукции предприятия.
Принята иерархическая классификация выпускаемой продукции по следующему ряду уровней:

- семейство продуктов;
- группа продуктов;
- серия продуктов.

Однако эта система классификации не обеспечивает идентификацию **любого** выпускаемого изделия. Для каждой единицы продукта должны указываться следующие атрибуты (фасеты):

- код серии продукта;
- конфигурационные параметры;
- свойства.

Код серии продукта – алфавитно-цифровой код, однозначно идентифицирующий отдельный продукт.
Конфигурационные параметры – свойства, значения которых могут быть различными в зависимости от потребностей пользователей.
Свойства – предопределенные характеристики отдельных продуктов, которые не могут меняться для одного и того же продукта.
Допустимые варианты записи кода серии для различных продуктов показаны на рис. 4.
Признаки фасета «Конфигурационные параметры» для одного семейства продуктов приведены в таблице 1.
Рассмотренные выше *системы классификации* хорошо приспособлены для организации поиска с целью последующей логической и арифметической обработки информации на ЭВМ, но лишь частично решают проблему содержательного поиска информации при принятии управленческих решений.

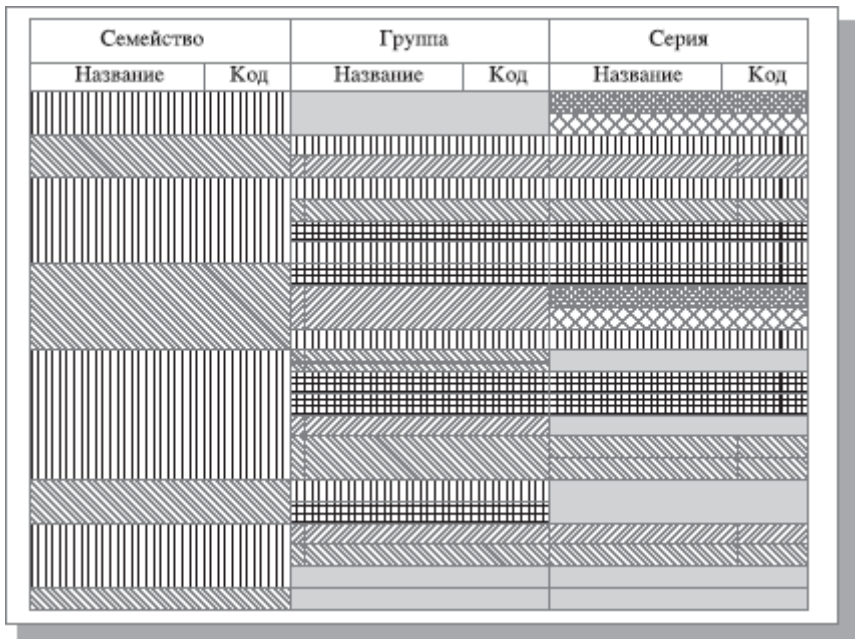


Рис.4. Варианты записи кода серии продукта (серым цветом отмечены неиспользуемые элементы кода)

Таблица 1. Признаки фасета «Конфигурационные параметры» для одного семейства продуктов

Продукты и модификации	**Характеристики**	
	Общие для семейства	**Специальные для отдельны моделей**
Датчики разности давлений	• Искробезопасное исполнение • Взрывозащищенное исполнение • Исполнение по материалам • Климатическое исполнение • Предел допускаемой основной погрешности • Верхний предел измерений • Код выходного сигнала • Состав комплекта монтажных частей	• Предельно допустимое рабочее избыточное давление
Датчики абсолютного давления, избыточного давления, разрежения, давления-разрежения		• Измеряемый параметр

20. Внутримашинное информационное обеспечение. Принципы проектирования системы документооборота

При построении эффективной автоматизированной системы первым этапом является исследование и формализация бизнес-процессов деятельности банка или предприятия. Т.е. описание системы ведения делопроизводства с целью эффективного использования информации для достижения поставленных задач и решения проблем, стоящих перед организацией. Организация работы с документами (будь то платежные или конструкторско-технологические документы) является важной составной частью процессов управления и принятия управленческих решений, существенно влияющей на оперативность и качество управления. Процесс принятия управленческого решения состоит из:

- Получения информации;
- Переработка информации;

- Анализа, подготовки и принятия решения.

Все эти этапы самым тесным образом связаны с документационным обеспечением процессов управления, проектирования и производства. Если на предприятии отсутствует четкая организация работы с документами, то, как следствие этого, закономерно появление документов низкого качества, как в оформлении, так и в полноте и ценности содержащейся в них информации, увеличение сроков их обработки. Это приводит к ухудшению качества управления и увеличению сроков принятия решений и числу неверных решений. С ростом масштабов предприятия и численности его сотрудников вопрос об эффективности документационного обеспечения управления становится все более актуальным. Основные проблемы, возникающие при этом, выглядят примерно так:

- руководство теряет целостную картину происходящего;
- структурные подразделения, не имея информации о деятельности друг друга, перестают слаженно осуществлять свою деятельность. Неизбежно падает качество обслуживания клиентов и способность организации поддерживать внешние контакты;
- это приводит к падению производительности и вызывает ощущение недостатка в ресурсах: людских, технических, коммуникационных и т.д.;
- приходится расширять штат, вкладывать деньги в оборудование новых рабочих мест, помещения, коммуникации, обучение новых сотрудников;
- для производственных предприятий увеличение штата может повлечь изменение технологии производства, что потребует дополнительных инвестиций;
- оказывается, что штат увеличен, производительность упала, производство требует инвестиций, соответственно возникает потребность в увеличении оборотного капитала, что может потребовать новых кредитов и уменьшить плановую прибыль.

В итоге предприятие перестает расти интенсивно и дальнейшее расширение происходит чисто экстенсивным путем за счет ранее созданной прибыли.

Почему же сегодня, когда для организации документооборота (в дальнейшем под этим термином мы будем понимать документооборот любых документов: конструкторских, технологических, финансовых, организационных и т.п.) предлагается множество самых различных средств автоматизации, документооборот часто организован плохо, даже на относительно небольших предприятиях? Ответ, независимо от степени автоматизации предприятия и его типа, может быть один - *отсутствие или игнорирование* ***модели организации документооборота*** *неизбежно приведет к тому, что старые проблемы останутся нерешенными.* При этом, если по состоянию делопроизводства в организации был "ручной" хаос, то результатом автоматизации будет "компьютерный" хаос.

Когда на Западе, а теперь и в России схлынула первая волна увлечения системами автоматизации документооборота, оказалось, ***что без должной оценки возможностей пользователя, исследования бизнес - процессов его предприятия трудно ожидать эффекта от внедрения любых систем документооборота.*** При этом совсем неважно, как планируется или уже реализован документооборот: вручную или путем автоматизации с помощью мощных западных либо отечественных пакетов - всегда на первом месте должна быть четкая стратегия, направленная на упорядочение бизнес - процессов

Основные понятия электронного документооборота

Документ - это совокупность трех составляющих

- *Физическая регистрация* информации.
- *Форма представления* информации
- *Активизация* определенной *деятельности*.

Пример: Вы написали заявление на отпуск и передали его в отдел кадров. Так появился документ. Что же превратило чистый лист бумаги в документ? Во-первых, *информация, представленная в виде текста*. Во-вторых, *текст в форме заявления*. И, в-третьих, бумагу готовили с расчетом на *последующую деятельность* сотрудников отдела кадров.

Теперь предположим, что вы обратились в отдел кадров с устным заявлением об отпуске. Можно ли назвать документом эту процедуру? Устная беседа не была зафиксирована физически, она не поддается точному воспроизведению и, следовательно, документом не является.

Именно некоторая деятельность и превращает информацию в документ. Но документ перестает существовать, если в дальнейшем не подразумевает процедуры обработки. При этом форма документа тесно связана с характером дальнейшей деятельности, она порождает необходимость документов. Так родилась бюрократия - неизбежный спутник цивилизации.

Документ - слабоструктурированная совокупность блоков или объектов информации, понятная человеку. В общем случае обойтись без документов пока нельзя. Сам по себе документ, независимо от того, обычная ли это бумага или электронный бланк, проблем корпорации не решает - первичны бизнес-процессы и четкий контроль за выполнением проекта.

Бюрократическая технология - это технология взаимодействия людей, служб и подразделений внутри и вне организации. Не будет технологии - возникнет анархия. Если работник не знает что ему надо делать, он делает то, что считает нужным, а не то, что требует тот или иной бизнес-процесс предприятия. Сама бюрократия неизбежна, опасность представляет отрыв реальных целей предприятия от работы текущей системы документооборота.

Собственно документооборот может быть двух типов:

- **универсальный** - автоматизирующий существующие информационные потоки слабоструктурированной информации. Справедливо было бы его называть аморфным или беспорядочным документооборотом;
- **операционный** - ориентированный на работу с документами, содержащими операционную атрибутику, вместе с которой ведется слабоструктурированная информация.

Кроме собственно документов важен еще **регламент** работы с ними. Любой опытный менеджер может подтвердить, что работа не по регламенту порой отнимает намного больше времени, чем собственно производственная деятельность. Дублирование документов, их потеря, навязчивый способ их распространения, а также запутанный порядок их прохождения могут существенно усложнить работу, повысив вероятность допущения ошибки вследствие, например, потери нужной информации.

Итак, документ занимает определенное место в процессе некоторой деятельности на границе разделяемых функций исполнения. Поэтому правильно рассматривать документ как *инструмент распределения функций* между работниками

Преимущества электронного документооборота

К основным преимуществам электронного документооборота можно отнести следующие:

- Полный контроль за перемещением и эволюцией документа, регламентация доступа и способ работы пользователей с различными документами и их отдельными частями.
- Уменьшение расходов на управление за счет высвобождения (на 90% и более) людских ресурсов, занятых различными видами обработки бумажных документов, снижение бюрократической волокиты за счет маршрутизированного перемещения документов и жесткого контроля за порядком и сроками прохождения документов.
- Быстрое создание новых документов из уже существующих.
- Поддержка одновременной работы многих пользователей с одним и тем же документом, предотвращение его потери или порчи.
- Сокращение времени поиска нужных документов.

Использование автоматизированной информационной системы (АИС) может рассматриваться в качестве базы для общего совершенствования управления

предприятием. При этом управление предприятием реализует следующие основные функции:

- обслуживание клиентов;
- разработка продукции;
- учет и контроль за деятельностью предприятия;
- финансовое обеспечение деятельности предприятия и т.д.

Модели информационного пространства предприятия

Комплексная автоматизация этих функции требует создания *единого информационного пространства* предприятия, в котором сотрудники и руководство могут осуществлять свою деятельность, руководствуясь едиными правилами представления и обработки информации в документном и бездокументном виде.

Для этого в рамках предприятия требуется создать единую информационную *систему по управлению информацией* или *единую систему управления документами*, включающую возможности:

- удаленной работы, когда члены одного коллектива могут работать в разных комнатах здания или в разных зданиях;
- доступа к информации, когда разные пользователи должны иметь доступ к одним и тем же данным без потерь в производительности и независимо от своего местоположения в сети;
- средств коммуникации, например: электронная почта, факс, печать документов;
- сохранения целостности данных в общей базе данных;
- полнотекстового и реквизитного поиска информации;
- открытость системы, когда пользователи должны иметь доступ к привычным средствам создания документов и к уже существующим документам, созданным в других системах;
- защищенность информации;
- удобства настройки на конкретные задачи пользователей;
- масштабируемости системы для поддержки роста организаций и защиты вложенных инвестиций и т.д.

Начальным этапом создания такой системы является построение *модели предметной области* или другими словами *модели документооборота*.

Определенные направления автоматизации документооборота определяют трехмерное пространство свойств, где по некоторой траектории движется любой программный продукт данного класса, проходя различные стадии в своем развитии (рис.1.).

Первая ось *(F*) характеризует уровень организации хранения **фактографической** информации, которая привязана к специфике конкретного рода деятельности компании или организации. Например: при закупке материальных ценностей происходит оформление товарно-сопроводительных документов (накладных, приемо-передаточных актов, приходных складских ордеров и т.д.), регистрируемых в качестве операционных документов, атрибутика которых очень важна для принятия управленческих решений. Информация из операционных документов используется при сложной аналитической и синтетической обработке, и, в частном случае, может быть получена пользователем через систему отчетов.

Рис.1. Трехмерное пространство свойств.

Вторая ось *(D)* - **полнотекстовые документы**, отражает необходимость организации взаимодействия: формирование и передача товаров, услуг или информации как внутри корпорации, так и вне ее. В этих документах наряду с фактографической информацией содержится слабо структурированная информация, не подлежащая автоматизированной аналитической обработке, такая, например, как форс-мажерные факторы и порядок предъявления претензий при нарушении условий договора. Все взаимоотношения между субъектами бизнеса сопровождаются документами, которые становятся осязаемым отражением результата взаимодействия.

Третья ось *(R)* вносит в пространство документооборота третье измерение - **регламент процессов** прохождения документов, а именно: описание того какие процедуры, когда и как должны выполняться. Основа для позиционирования относительно данной оси - набор формальных признаков (атрибутов) и перечень выполнения операций.

Точка в пространстве *(F, D, R)* определяет состояние системы документооборота и имеет координаты (f,d,r), где f,d и r принадлежат множествам F,D и R, соответственно. Положение этой точки зависит от уровня развития и стадии внедрения системы документооборота на предприятии, а также от его специфики и самих масштабов бизнеса. Представив модель документооборота именно таким образом, можно, например, зная текущее положение дел с организацией делопроизводства на каждом конкретном предприятии, четко представить, в каких направлениях нужно двигаться дальше, чего недостает в текущий момент и каким образом органично использовать уже существующие системы автоматизации. Например, в одном из московских банков был накоплен большой массив фактографических данных, для обработки которых использовалась современная СУБД, развернутая на мощных, отказоустойчивых серверах - все, казалось бы, должно быть отлично. Однако при работе с внутренними документами наблюдалось дублирование информации: возникали ситуации, когда "никто вроде бы и не виноват", а банк время от времени лишается выгодных клиентов. Причина в том, что точка, отражающая положение системы документооборота для этой организации, имела достаточно большие координаты по оси "F" и, возможно, по оси "D", однако значение координаты по оси "R" было близко к нулю. Конкретным решением в этом случае может быть рассмотрение вопроса о внедрении системы управления регламентом. При этом не надо пока заботится о СУБД (ось "F") или электронных архивах (ось "D") - речь идет только об изменении значения координат по оси "R".

В общем случае, как уже отмечалось, процесс автоматизации делопроизводства на предприятии можно представить в виде кривой в трехмерном пространстве координат F,D,R. Причем, чем круче эта кривая, тем быстрее идет процесс модернизации, а чем больше значения всех трех координат - тем выше уровень автоматизации на корпорации и, как следствие, тем меньше у нее проблем с организацией своей собственной деятельности.

Новое качество, с которым сегодня ассоциируется возросший интерес к системам электронного документооборота, связано с использованием инструментальных систем,

предназначенных для хранения, регистрации, поиска документов, а также для управления регламентом. Чаще ошибочно под новым качеством сегодня понимается простое внедрение отличной от ранее используемой технологии работы с документами, например локальной сети вместо дискет, переносимых с одного компьютера на другой. Вряд ли в этом случае уместно говорить о новом качестве управления предприятием.
Рассмотрим в качестве примера основные шаги по разработке функциональной модели типовой автоматизированной банковской системы (АБС). Среди архитектур реализации АБС на сегодня выделяют АБС пяти поколений: первые были предназначены для решения задач автоматизации бухгалтерских операций, АБС второго и третьего поколений также реализовывали элементы автоматизированного документооборота, АБС четвертого поколения строились по классической схеме трехзвенной клиент-серверной архитектуры и имели **модульную структуру**, реализующую концепции **docflow**. АБС пятого поколения являются информационными системами, реализующими полнофункциональную модель **workflow** - автоматизации **бизнес-процессов** (рис.2).

Рис.2. Поколения АБС и их характеристики.

Рис.3.Приницпы реализации интеллектуальной инфраструктуры банка.

Схема функциональных бизнес-потоков типовой АБС четвертого поколения представлена на рис.4., а функциональная схема технологических потоков операций на рис.5.

Рис.4. Функциональная схема типовой АБС 4-го поколения.

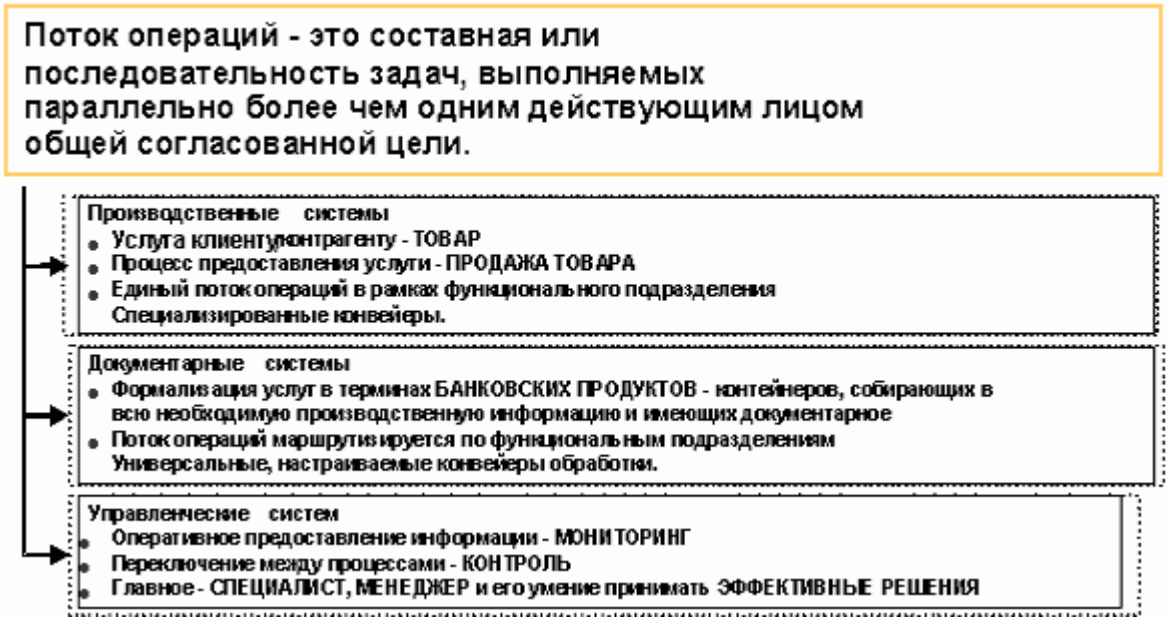


Рис.5. Функциональная схема технологических потоков операций.

Назначения большинства модулей АБС (рис.4) достаточны понятны, единственное следует уточнить, что под интернет-банкингом понимается автоматизированная система, реализующая технологии busines-to-customers (клиентское обслуживание, включая электронные платежи физических лиц) и bisnes-to-bisnes (обеспечение работы дистрибьюторско-дилерской сети корпораций в режиме on-line коммерции).

Разработав функциональную схему бизнес-потоков, структурировав ее в соответствии с модульным принципом построения любой системы управления и выделив базовые технологические потоки операций на каждом из уровне модульности (итог: перечень

документов и операций, обрабатываемых на конкретном рабочем месте, маршруты их движения, контроля и управления), можно приступать к следующему этапу проектирования информационной системы.

Выводы

Координаты точки, характеризующей сбалансированную систему документооборота (бизнес - модель функционирования предприятия), должны иметь ненулевые значения, а в идеале быть примерно одинаковы - соответствовать друг другу. Главное - не автоматизация как таковая, а оптимизация потоков документов и интегральность.

Нет большого смысла говорить о жизненном цикле документов без связи с основными бизнес-процессами предприятия. Система автоматизации документооборота, функционирующая в отрыве от всех слоев, будет мертва, мало того, она может нанести вред: запутать и без того неуправляемые бизнес - процессы, отвлечь персонал от выполнения основной работы ради поддержания системы автоматизации документооборота, по сути дела, ничего не автоматизирующего. И, как следствие, раздувание штатного расписания и дискредитация самой идеи автоматизации делопроизводства. Знакомая ситуация - все работники заняты, работа кипит, однако если рассмотреть две разные фирмы, имеющие равный доход, занимающиеся одним бизнесом, то штат сотрудников у одной из них будет вдвое больше, чем у другой.

21. Внутримашинное информационное обеспечение. Построение модели данных

Работа проектировщиков базы данных в значительной степени зависит от качества ***информационной модели***. Информационная модель не должна содержать никаких непонятных конструкций, которые нельзя реализовать в рамках выбранной СУБД. Следует отметить, что информационная модель создается для того, чтобы на ее основе можно было построить модель данных, то есть должна учитывать особенности реализации выбранной СУБД. Если те или иные особенности СУБД не позволяют отразить в модели данных то, что описывает информационная модель, значит, надо менять информационную модель.

Построение **логической** и **физической** моделей данных является основной частью проектирования базы данных. Полученная в процессе анализа информационная модель сначала преобразуется в логическую, а затем в физическую модель данных. После этого для разработчиков информационной системы создается пробная база данных. С ней начинают работать разработчики кода. В идеале к моменту начала разработки модель данных должна быть устойчива. Проектирование базы данных не может быть оторвано от проектирования модулей и приложений, поскольку бизнес-правила могут создавать объекты в базе данных, например серверные ограничения, а также хранимые процедуры и триггеры, - в этом случае часто говорят, что часть бизнес-логики переносится в базу данных. Проектирование модели данных для каждой СУБД содержит свои особенности, проектные решения, которые дают хороший результат для одной СУБД, но могут оказаться совершенно неприемлемыми для другой.

Общими для проектирования моделей данных являются следующие задачи:

- выявление нереализуемых или необычных конструкций в ER-модели и в определениях сущностей;
- разрешение всех дуг, подтипов и супертипов;
- изучение возможных, первичных, внешних ключей, описание ссылочной целостности (в зависимости от реализации декларативно или с использованием триггеров);
- проектирование и реализация денормализации базы данных в целях повышения производительности;
- определение части бизнес-логики, которую следует реализовать в базе данных (пакеты, хранимые процедуры);

- реализация ограничений (ограничений и триггеров), отражающих все централизованно определенные бизнес-правила, генерация ограничений и триггеров;
- определение набора бизнес-правил, которые не могут быть заданы как ограничения в базе данных;
- определение необходимых индексов, кластеров (если таковые реализованы в СУБД), определение горизонтальной фрагментации таблиц (если это реализовано в СУБД);
- оценка размеров всех таблиц, индексов, кластеров;
- определение размеров табличных пространств и особенностей их размещения на носителях информации, определение спецификации носителей информации для промышленной системы (например, тип raid-массивов, их количество, какие табличные пространства на них размещаются), определение размеров необходимых системных табличных пространств (например, системного каталога, журнала транзакций, временного табличного пространства и т.п.);
- определение пользователей базы данных, их уровней доступа, разработка и внедрение правил безопасности доступа, аудита (если это необходимо), создание пакетированных привилегий (в зависимости от реализации СУБД это роли или группы), синонимов;
- разработка топологии базы данных в случае распределенной базы данных, определение механизмов доступа к удаленным данным.

Схема базы данных

Схема базы данных содержит описание всех объектов базы данных: пользователей, их привилегий, таблиц, представлений, индексов, кластеров, ограничений, пакетов, хранимых процедур, триггеров и т.п. При этом создаются не только определения этих объектов, но и сами объекты, с которыми потом работают разработчики.

ER-модель и ее отображение на схему данных. Результат этапа анализа - построение информационной модели. Казалось бы, дело это простое: сущности становятся таблицами, а атрибуты сущностей - столбцами таблиц; ключи становятся первичными ключами, для возможных ключей определяется ограничение unique, внешние ключи становятся декларациями ссылочной целостности. Аналитики, как правило, не вникают в особенности реализации той или иной СУБД, поэтому при проектировании схемы базы данных проектировщик сталкивается с конструкциями в информационной модели, которые не реализуемы или трудно реализуемы в выбранной СУБД.

Приведем несколько примеров ограничений реализации СУБД:

В информационной модели описаны три сущности - A, B, C. Сущности B и C содержат внешние ключи, ссылающиеся на сущность A. В СУБД поддерживается возможность определения внешнего ключа только для первичного ключа, а для возможного ключа определить декларативную ссылочную целостность нельзя. В этом случае отображение ER-модели на физическую модель данных невозможно без изменения информационной модели.

В информационной модели определен атрибут, представляющий собой строку длиной в 500 символов. По этому атрибуту часто осуществляется поиск в информационной системе; объем данных велик. В СУБД можно индексировать строки символов не длиннее чем 128 или 256 символов. Если осуществлять поиск без индекса, то время ответа информационной системы существенно превышает допустимое, вследствие чего придется изменить описание сущности.

Две диаграммы потока данных описывают различные бизнес-процессы, работающие над одними и теми же данными. Допустим, первая диаграмма описывает выписку товара со склада, а вторая - сложный отчет, отражающий состояние склада. Один процесс интенсивно модифицирует данные, второй работает в режиме чтения данных, но требует согласованности данных в течение длительного времени. Каждый из процессов

описывается транзакцией над данными. В СУБД уровни изолированности транзакций реализованы так, что читающие транзакции конфликтуют с модифицирующими транзакциями. Это приводит к остановке выписки товаров со склада на время выполнения отчета, что неприемлемо для заказчика. Здесь может потребоваться очень серьезное изменение информационной модели.

Подобных примеров, когда не только ER-модель, но и другие продукты анализа не могут быть перенесены автоматически на модель данных, можно привести множество. Каждый такой случай инициирует изменение информационной модели. Решение проблемы определяется возможностями СУБД, выбранной для реализации проекта. Если проблем, не разрешаемых в рамках данной СУБД, накапливается очень много, то проектировщики могут поставить вопрос о смене СУБД. Такой вопрос поднимается именно на стадии проектирования, поскольку если уже разработчики столкнутся с подобными проблемами, то цена смены СУБД будет выше. Ясно, что одинаковых СУБД не бывает: то, что хорошо работает в одной, может плохо работать или вообще не работать в другой, несмотря на уверения производителей СУБД в поддержке стандартов SQL.

Вопросы производительности информационной системы также влияют на отображение ER-модели на модель данных.

Типы данных

Как правило, СУБД поддерживают небольшой набор базовых типов данных: числовые типы (целые, вещественные с плавающей и фиксированной точкой), строки (символов и байт), дата и время, BLOB (и его разновидности, например BLOB-поля для хранения только текста). В информационной модели каждому атрибуту соответствует домен. Поскольку не все реализации СУБД поддерживают домены, то в этом случае при определении модели данных ограничения домена описывают как ограничения столбца таблицы (если такое возможно); в частности используют триггеры.

Следует отметить, что при определении типов столбцов таблиц нужно учитывать, какие типы данных поддерживаются в словаре данных СУБД.

Индексы, кластеры

В правильно спроектированной базе данных каждая таблица содержит первичный ключ, что означает наличие индекса.

Отметим, что если используется составной индекс, то поиск по всем атрибутам, входящим в индекс, начиная со второго, будет медленным. Допустим, определен индекс index1(id1, id2), в этом случае поиск значений, удовлетворяющих условию id2=1, будет медленным (не исключено, что оптимизатор вообще не будет использовать этот индекс для обработки данного условия и будет принято решение о полном сканировании данных), а поиск значений, удовлетворяющих условию id1=1 and id2=1, будет быстрым. Данные особенности следует учитывать при определении индексов в схеме базы данных, а именно:

- индексировать нужно атрибуты, по которым наиболее часто осуществляется поиск или соединение. Наличие индекса замедляет операции модификации, но ускоряет поиск;
- наличие индекса обязательно, если для атрибута или набора атрибутов указано ограничение unique. Такие индексы СУБД создает автоматически, если в описании таблицы указаны ограничения unique;
- индекс может быть использован для выборки данных в заданном порядке. В этом случае не вызывается процесс сортировки ответа, а используется уже готовый индекс;
- атрибуты, входящие во внешний ключ, также следует индексировать, если СУБД не делает эту операцию автоматически при декларации внешнего ключа;
- в некоторых СУБД поддерживаются хеш-индексы, например для кластеров. Такие индексы эффективно используются при поиске на равенство.

Кластеризация - это попытка разместить рядом в одном физическом блоке данных те строки, доступ к которым осуществляется при помощи одинаковых значений ключа. Индексные кластеры, например, удобно использовать для хранения родительской и дочерних строк таблиц, связанных ссылочной целостностью. Кластеры удобно определять для тех наборов атрибутов, соединение по которым проводится наиболее часто, поскольку это увеличивает скорость поиска. Следует отметить, что в реализациях СУБД существуют жесткие ограничения на количество кластеров для таблицы, как правило, это один кластер. Особенности реализации кластеров в СУБД необходимо учитывать при проектировании критичных по времени выполнения модулей.

Хранение объектов данных

Это одна из самых сложных задач проектирования схемы базы данных, для решения которой привлекаются администраторы баз данных. Универсальных решений, которые подошли бы для любой СУБД, не существует. На проектирование схемы базы данных влияют следующие параметры, общие для большинства СУБД:

- размер табличных пространств для хранения таблиц;
- размер табличных пространств для хранения индексов;
- размер табличных пространств для хранения BLOB;
- кластеры и их параметры;
- размер словаря данных, включая код всех хранимых процедур, функций, триггеров, пакетов, статического SQL (реализован только в DB2);
- управляющие файлы;
- файлы журнала;
- интенсивность потока запросов, модифицирующих данные и индексы;
- фалы временных табличных пространств (для хранения временных таблиц, которые строятся, например, при выполнении group by, а также других временных объектов);
- интенсивность потока запросов, инициирующих создание временных таблиц;
- потоки транзакций read-write, read-only, объем модифицируемых и считываемых ими данных, характеристики параллельной работы транзакций (какие и сколько их);
- количество приложений, работающих параллельно с базой данных;
- количество соединений с базой данных для каждого приложения;
- файлы параметров старта ядра СУБД;
- загрузочные модули ядра СУБД и утилиты СУБД;
- входные и выходные данные, генерируемые пользовательскими программами;
- скрипты управления СУБД.

Защита данных

Стратегия защиты определяется на этапе анализа, а на этапе проектирования предстоит реализовать эту стратегию, спроектировав соответствующие структуры в схеме базы данных и модули. Большинство СУБД имеют развитые средства дискреционной защиты, а ряд СУБД имеют встроенные подсистемы аудита, что освобождает от необходимости создания собственных средств защиты.

Обычно СУБД предоставляют набор пакетированных привилегий для управления данными, например:

- **connect**, которая разрешает соединение с базой данных;
- **resource**, которая дополнительно разрешает создание собственных объектов базы данных, **dba**, которая позволяет выполнять функции администратора конкретной базы данных, и др.

Дискреционная защита предполагает разграничение доступа к объектам данных (таблиц, представлений, и т.п.), а не собственно к данным, которые хранятся в этих объектах. Дискреционная защита также обеспечивает создание пользовательских пакетированных

привилегий - ролей или групп привилегий. В этом случае набор привилегий на те или иные объекты данных назначается группе или роли, а затем эта группа или роль назначается пользователю; таким образом, пользователь получает привилегии на выполнение тех или иных операций над объектами данных косвенно - через группу или роль.
К сфере защиты данных относятся также сохранность данных и восстановление их после сбоя системы. Для обеспечения бесперебойной работы часто применяют архивирование базы данных и журнала транзакций, а в случае отказа системы при следующем старте операции над данными восстанавливают по журналу транзакций (например, производят их откат до определенного момента времени). Применяют также методы горячего резервирования, когда работают два сервера: основной, обрабатывающий запросы пользователей, и резервный, который продолжает работу основного сервера в случае его отказа. Состояние хранилищ данных на основном и резервном серверах согласовано и поддерживается СУБД автоматически, что позволяет проектировщикам не разрабатывать собственные механизмы репликации данных.

22. Этапы проектирования ИС с применением UML

Унифицированный язык объектно-ориентированного моделирования Unified Modeling Language (UML) Существует достаточное количество инструментальных средств, поддерживающих с помощью *UML* жизненный цикл информационных систем. Мощный толчок к разработке этого направления информационных технологий дало распространение *объектно-ориентированных* языков программирования в конце 1980-х — начале 1990-х годов. Пользователям хотелось получить единый язык моделирования, который объединил бы в себе всю мощь объектно-ориентированного подхода и давал бы четкую модель системы, отражающую все ее значимые стороны.
Создание *UML* началось в октябре 1994 г., когда Джим Рамбо и Гради Буч из Rational Software Corporation стали работать над объединением своих методов OMT и Booch.
UML представляет собой объектно-ориентированный язык моделирования, обладающий следующими основными характеристиками:

- является *языком визуального моделирования*, который обеспечивает разработку репрезентативных моделей для организации взаимодействия заказчика и разработчика ИС, различных групп разработчиков ИС;
- содержит механизмы расширения и специализации базовых концепций языка.

UML — это стандартная нотация визуального моделирования программных систем, принятая консорциумом Object Managing Group (OMG) осенью 1997 г., и на сегодняшний день она поддерживается многими ***объектно-ориентированными*** CASE-продуктами.
UML включает внутренний набор средств моделирования («**ядро**»), которые сейчас приняты во многих методах и средствах моделирования. Пользователям языка предоставлены возможности:

- строить модели на основе средств ядра без использования механизмов расширения для большинства типовых приложений;
- добавлять при необходимости новые элементы и условные обозначения, если они не входят в ядро, или специализировать компоненты, систему условных обозначений (нотацию) и ограничения для конкретных предметных областей.

Синтаксис и семантика основных объектов UML
Классы — это базовые элементы любой объектно-ориентированной системы. Классы представляют собой описание совокупностей однородных объектов (типов объектов) с присущими им свойствами — атрибутами, операциями, отношениями и семантикой.
Операция — реализация функции, которую можно запросить у любого объекта класса. Операция показывает, что можно сделать с объектом. Исполнение операции часто связано с обработкой и изменением значений атрибутов объекта, а также изменением состояния объекта.

На рис.1 приведено графическое изображение класса «Заказ» в нотации UML.

Рис.1. Изображение класса в UML

Словарь языка UML включает три вида строительных блоков:

- сущности;
- отношения;
- диаграммы.

Сущности в UML - это абстракции, являющиеся основными элементами модели. В UML имеется четыре типа сущностей:

- **структурные** - это имена существительные в моделях на языке UML. Как правило, они представляют собой статические части модели, соответствующие концептуальным или физическим элементам системы. Существует семь разновидностей структурных сущностей: Класс, Интерфейс, Кооперация, Прецедент, Активный класс, Компонент, Узел;
- **поведенческие** - являются динамическими составляющими модели UML. Это глаголы языка: они описывают поведение модели во времени и пространстве. Существует всего два основных типа поведенческих сущностей: Взаимодействие и Автомат;
- **группирующие** - являются организующими частями модели UML. Это блоки, на которые можно разложить модель. Есть только одна первичная группирующая сущность, а именно - **пакет**;
- **аннотационные** - пояснительные части модели UML. Это комментарии для дополнительного описания, разъяснения или замечания к любому элементу модели. Имеется только один базовый тип аннотационных элементов - **примечание**.

Отношения связывают различные сущности.

Диаграмма в UML - это графическое представление набора элементов, изображаемое чаще всего в виде связанного графа с вершинами (сущностями) и ребрами (отношениями). Диаграммы рисуют для визуализации системы с разных точек зрения. Теоретически диаграммы могут содержать любые комбинации сущностей и отношений. На практике, однако, применяется сравнительно небольшое количество типовых комбинаций.

В UML выделяют следующие типы диаграмм.

Диаграммы классов (class diagrams) **.** Классы в UML изображаются на ***диаграммах классов***, которые позволяют описать систему в статическом состоянии — определить типы объектов системы и различного рода статические связи между ними, другими словами, диаграммы классов представляют логическую модель базовой структуры системы.

Между классами возможны различные отношения, представленные на рис. 2:

- ***зависимости***, которые описывают существующие между классами отношения использования;
- ***обобщения***, связывающие обобщенные классы *со* специализированными;
- ***ассоциации***, отражающие структурные отношения между объектами классов.

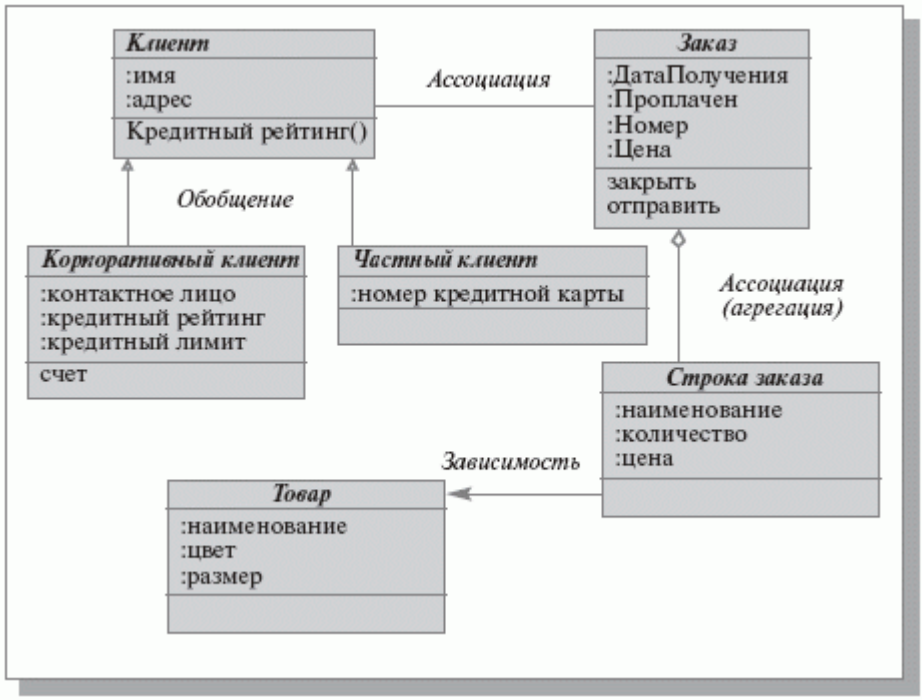


Рис. 2. Отображение связей между классами

Зависимостью называется отношение использования, согласно которому изменение в спецификации одного элемента (например, класса «товар») может повлиять на использующий его элемент (класс «строка заказа»). Часто зависимости показывают, что один класс использует другой в качестве аргумента.

Обобщение — это отношение между общей сущностью (родителем — класс «клиент») и ее конкретным воплощением (потомком — классы «корпоративный клиент» или «частный клиент»).

Ассоциация — это отношение, показывающее, что объекты одного типа неким образом связаны с объектами другого типа («клиент» может сделать «заказ»). Если между двумя классами определена ассоциация, то можно перемещаться от объектов одного класса к объектам другого. При необходимости направление навигации может задаваться стрелкой. Допускается задание ассоциаций на одном классе. В этом случае оба конца ассоциации относятся к одному и тому же классу. Это означает, что с объектом некоторого класса можно связать другие объекты из того же класса. Ассоциации может быть присвоено имя, описывающее семантику отношений. Каждая ассоциация имеет *две роли*, которые могут быть отражены на диаграмме (рис. 3). *Роль* ассоциации обладает свойством *множественности*, которое показывает, сколько соответствующих объектов может участвовать в данной связи.

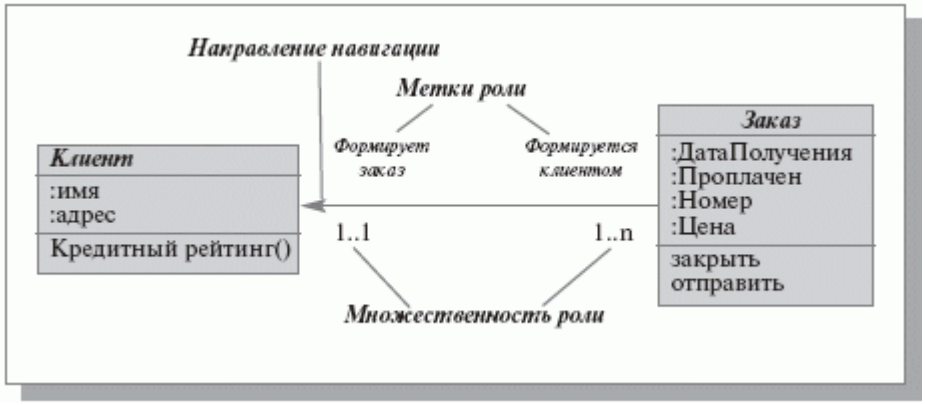


Рис. 3. Свойства ассоциации

Рис. 3 иллюстрирует модель формирования заказа. Каждый заказ может быть создан единственным клиентом (множественность роли 1.1). Каждый клиент может создать один

и более заказов (множественность роли 1..n). Направление навигации показывает, что каждый заказ должен быть «привязан» к определенному клиенту.
Такого рода ассоциация является простой и отражает отношение между равноправными сущностями, когда оба класса находятся на одном концептуальном уровне и ни один не является более важным, чем другой. Если приходится моделировать отношение типа «часть-целое», то используется специальный тип ассоциации — *агрегирование*. В такой ассоциации один из классов имеет более высокий ранг (целое — класс «заказ», рис. 2) и состоит из нескольких меньших по рангу классов (частей — класс «строка заказа»). В UML используется и более сильная разновидность агрегации — *композиция*, в которой объект-часть может принадлежать только единственному целому. В композиции жизненный цикл частей и целого совпадают, любое удаление целого обязательно захватывает и его части.
Для ассоциаций можно задавать атрибуты и операции, создавая по обычным правилам UML классы ассоциаций.
Диаграммы использования. Диаграммы использования описывают функциональность ИС, которая будет видна пользователям системы. «Каждая функциональность» изображается в виде «*прецедентов использования*» (use case) или просто ***прецедентов***.
Прецедент — это типичное взаимодействие пользователя с системой, которое при этом:

- описывает видимую пользователем функцию,
- может представлять различные уровни детализации,
- обеспечивает достижение конкретной цели, важной для пользователя.

Прецедент обозначается на диаграмме овалом, связанным с пользователями, которых принято называть действующими лицами (актерами, actors). Действующие лица используют систему (или используются системой) в данном прецеденте. Действующее лицо выполняет некоторую **роль** в данном прецеденте. На диаграмме изображается только одно действующее лицо, однако реальных пользователей, выступающих в данной роли по отношению к ИС, может быть много. Список всех прецедентов фактически определяет функциональные требования к ИС, которые лежат в основе разработки технического задания на создание системы.
Диаграммы прецедентов (диаграммы вариантов использования, use case diagrams**).**
Диаграммы прецедентов – это обобщенная модель функционирования системы в окружающей среде. На диаграммах прецедентов, кроме связей между действующими лицами и прецедентами, возможно использование еще двух видов связей между прецедентами: «использование» и «расширение» (рис. 4).
Связь типа «*расширение*» применяется, когда один прецедент подобен другому, но несет несколько большую функциональную нагрузку. Ее следует применять при описании изменений в нормальном поведении системы.
Связь типа «*использование*» позволяет выделить некий фрагмент поведения системы и включать его в различные прецеденты без повторного описания.
На рис. 4 показано, что при исполнении прецедента «формирование заказа» возможно использование информации из предыдущего заказа, что позволит не вводить все необходимые данные. А при исполнении прецедентов «оценить риск сделки» и «согласовать цену» необходимо выполнить одно и то же действие — рассчитать стоимость заказа.

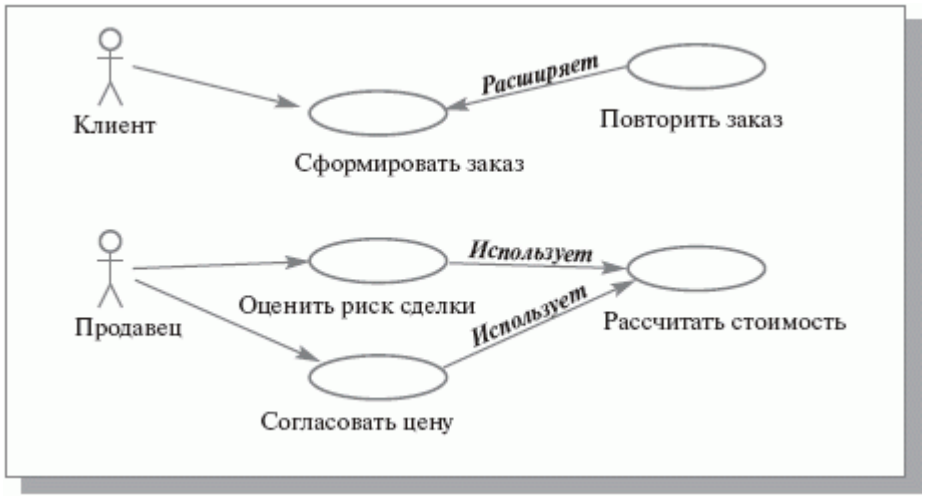


Рис. 4. Связи на диаграммах прецедентов

Динамические аспекты поведения системы отражаются приведенными ниже диаграммами.

Диаграммы взаимодействия (interaction diagrams). В *UML диаграммы классов* не содержат сообщений, которые усложняют их чтение. Поток сообщений между объектами выносится на *диаграммы взаимодействия*. Как правило, *диаграмма взаимодействия* охватывает поведение объектов в рамках одного варианта использования. Другими словами, диаграммы взаимодействия – модель процесса обмена сообщениями между объектами. Сообщения обозначаются ярлыками возле стрелок, они могут иметь нумерацию и показывать возвращаемые значения.

Существуют два вида диаграмм взаимодействия: *диаграммы последовательностей* и *кооперативные диаграммы*.

Диаграммы последовательностей (sequence diagrams)**.** Этот вид диаграмм используется для точного определения логики сценария выполнения прецедента. Диаграммы последовательностей отображают типы объектов, взаимодействующих при исполнении прецедентов, сообщения, которые они посылают друг другу, и любые возвращаемые значения, ассоциированные с этими сообщениями. Прямоугольники на вертикальных линиях показывают «время жизни» объекта. Линии со стрелками и надписями названий методов означают вызов метода у объекта.

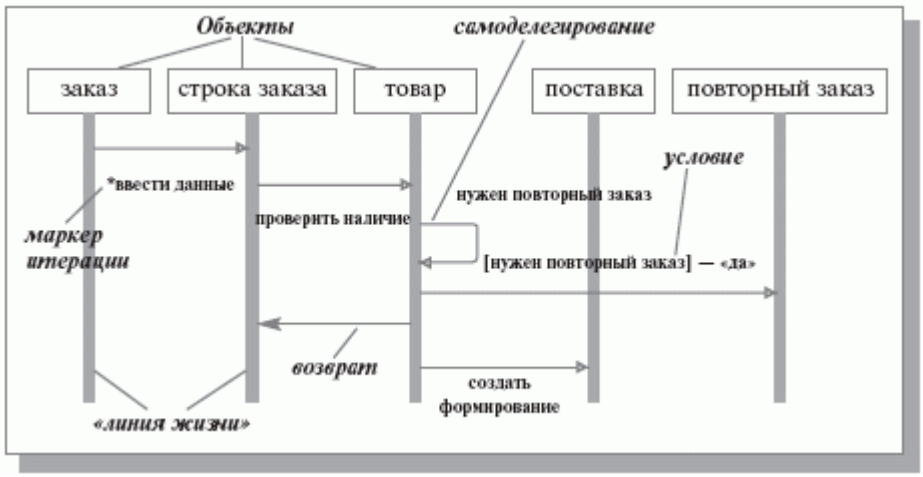


Рис. 5. Диаграмма последовательности обработки заказа

- вводятся строки заказа;
- по каждой строке проверяется наличие товара;
- если запас достаточен — инициируется поставка;
- если запас недостаточен — инициируется дозаказ (повторный заказ).

Рис. 6. Кооперативная диаграмма прохождения заказа

Сообщения появляются в той последовательности, как они показаны на диаграмме — сверху вниз. Если предусматривается отправка сообщения объектом самому себе (самоделегирование), то стрелка начинается и заканчивается на одной «линии жизни». На диаграммы может быть добавлена управляющая информация: описание условий, при которых посылается сообщение; признак многократной отправки сообщения (маркер итерации); признак возврата сообщения.

Кооперативные диаграммы (collaboration diagrams)**.** На кооперативных диаграммах объекты (или *классы*) показываются в виде прямоугольников, а стрелками обозначаются сообщения, которыми они обмениваются в рамках одного варианта использования. Временная последовательность сообщений отражается их нумерацией.

Диаграммы состояний (statechart diagrams)**. И**спользуются для описания поведения сложных систем. Они определяют все возможные состояния, в которых может находиться объект, а также процесс смены состояний объекта в результате некоторых ***событий***, то есть диаграммы состояний – это модель динамического поведения системы и ее компонентов при переходе из одного состояния в другое. Эти диаграммы обычно используются для описания поведения одного объекта в нескольких прецедентах. Прямоугольниками представляются состояния, через которые проходит объект во время своего поведения. Состояниям соответствуют определенные значения атрибутов объектов. Стрелки представляют переходы от одного состояния к другому, которые вызываются выполнением некоторых функций объекта. Имеется также два вида *псевдо-состояний*: начальное состояние, в котором находится только что созданный объект, и конечное состояние, которое объект не покидает, как только туда перешел.

Переходы имеют метки, которые синтаксически состоят из трех необязательных частей (см. рис. 7):

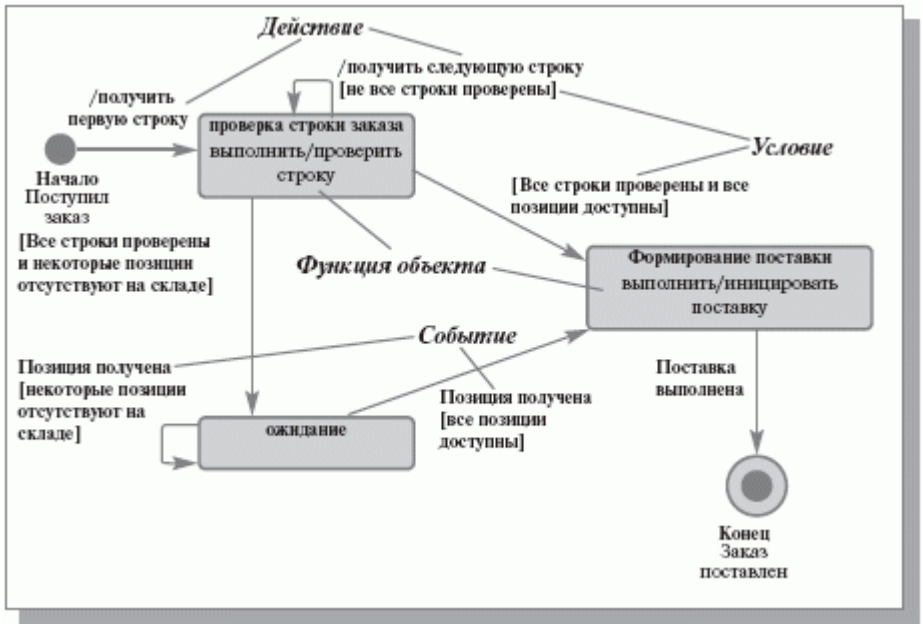


Рис. 7. Диаграмма состояний объекта «заказ»

<Событие> <[Условие]> < / Действие>.

На диаграммах также отображаются функции, которые выполняются объектом в определенном состоянии. Синтаксис метки деятельности:

выполнить/< деятельность >.

Диаграммы деятельности. Диаграмма деятельности — это частный случай *диаграммы состояний*. Диаграммы видов деятельности (диаграммы деятельностей, activity diagrams) – модель бизнес-процесса или поведения системы в рамках прецедента.

На диаграмме деятельности представлены переходы потока управления от одной деятельности к другой внутри системы. Этот вид диаграмм обычно используется для описания поведения, включающего в себя множество *параллельных процессов*.

Основными элементами диаграмм деятельности являются (рис. 8):

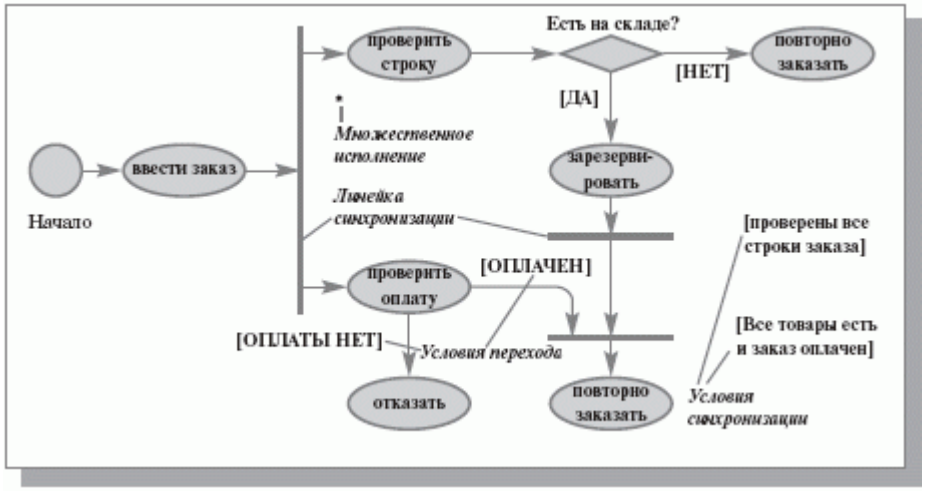


Рис. 8. Диаграмма деятельности — обработка заказа

- овалы, изображающие действия объекта;
- линейки синхронизации, указывающие на необходимость завершить или начать несколько действий (модель логического условия «И»);
- ромбы, отражающие принятие решений по выбору одного из маршрутов выполнения процесса (модель логического условия «ИЛИ»);
- стрелки — отражают последовательность действий, могут иметь метки условий.

На диаграмме деятельности могут быть представлены действия, соответствующие нескольким вариантам использования. На таких диаграммах появляется множество начальных точек, поскольку они отражают теперь реакцию системы на множество внешних событий. Таким образом, диаграммы деятельности позволяют получить

полную картину поведения системы и легко оценивать влияние изменений в отдельных вариантах использования на конечное поведение системы.
Любая деятельность может быть подвергнута дальнейшей декомпозиции и представлена в виде отдельной диаграммы деятельности или спецификации (словесного описания).
Диаграммы базы данных (database diagrams) — модель структуры базы данных, отображает таблицы, столбцы, ограничения и т.п.
Диаграммы компонентов. Позволяют изобразить модель системы на физическом уровне. Элементами диаграммы являются компоненты — физические замещаемые модули системы.
Каждый компонент является полностью независимым элементом системы. Для различных типов компонентов предусмотрены соответствующие стереотипы в языке *UML*.
Компонентом может быть любой достаточно крупный модульный объект, такой как таблица или экстент базы данных, подсистема, бинарный исполняемый файл, готовая к использованию система или приложение.
Таким образом, *диаграмму компонентов* можно рассматривать как *диаграмму классов* в более крупном (менее детальном) масштабе. Компонент, как правило, представляет собой физическую упаковку логических элементов, таких как *классы*, интерфейсы и кооперации.
Основное назначение *диаграмм компонентов* — разделение системы на элементы, которые имеют стабильный интерфейс и образуют единое целое. Это позволяет создать ядро системы, которое не будет меняться в ответ на изменения, происходящие на уровне подсистем.
На рис. 9 показана упрощенная схема элементов фрагмента корпоративной системы. «Коробки» представляют собой компоненты — приложения или внутренние подсистемы. Пунктирные линии отражают зависимости между компонентами.

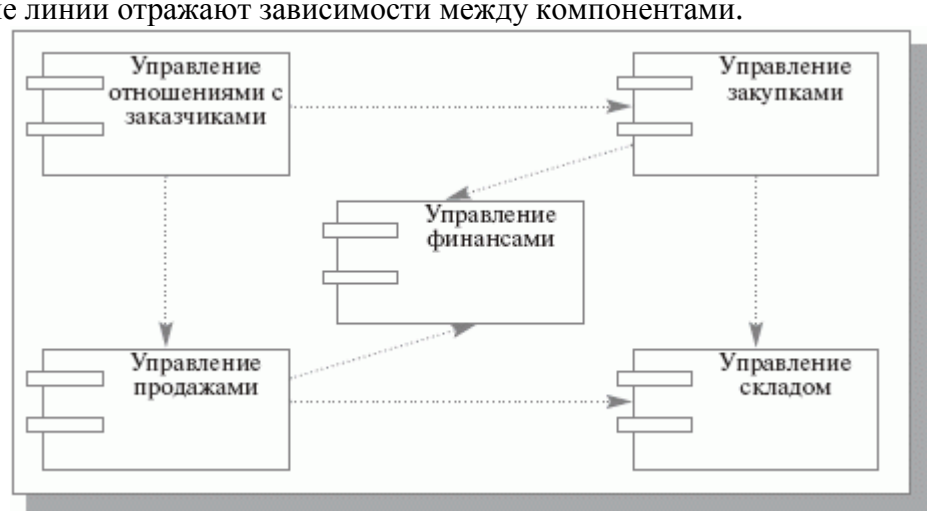


Рис. 9. Диаграмма компонентов фрагмента КИС

Каждый компонент диаграммы при необходимости документируется с помощью более детальной диаграммы компонентов, диаграммы сценариев или диаграммы классов.
Диаграммы развертывания (**диаграммы размещения**, deployment diagrams) – модель физической архитектуры системы, отображает аппаратную конфигурацию ИС.
Диаграмма пакетов содержит ***пакеты*** **классов** и зависимости между ними. Зависимость между двумя пакетами имеет место в том случае, если изменения в определении одного элемента влекут за собой изменения в другом. По отношению к пакетам можно использовать механизм обобщения.
Пакеты представляют собой универсальный механизм организации элементов в группы. В пакет можно поместить диаграммы различного типа и назначения. В отличие от компонентов, существующих во время работы программы, пакеты носят чисто концептуальный характер, то есть существуют только во время разработки. Изображается пакет в виде папки с закладкой, содержащей, как правило, только имя и иногда — описание содержимого.

Этапы проектирования ИС с применением UML

На этапе создания **концептуальной модели** ИС для описания:

- *бизнес-деятельности* используются ***модели бизнес-прецедентов*** и диаграммы ***видов деятельности,***
- *бизнес-объектов* – ***модели бизнес-объектов*** и диаграммы ***последовательностей***.

На этапе создания **логической модели** ИС описание:

- *требований к системе* задается в виде модели и описания ***системных прецедентов***,
- *предварительное проектирование* осуществляется с использованием диаграмм ***классов***, диаграмм ***последовательностей*** и диаграмм ***состояний***.

На этапе создания **физической модели** детальное проектирование выполняется с использованием диаграмм ***классов***, диаграмм ***компонентов***, диаграмм ***развертывания***.

На рис. 10. показаны отношения между различными видами диаграмм UML. Указатели стрелок можно интерпретировать как отношение "является источником входных данных для..." (например, диаграмма прецедентов является источником данных для диаграмм видов деятельности и последовательности). Приведенная схема является наглядной иллюстрацией итеративного характера разработки моделей с использованием UML.

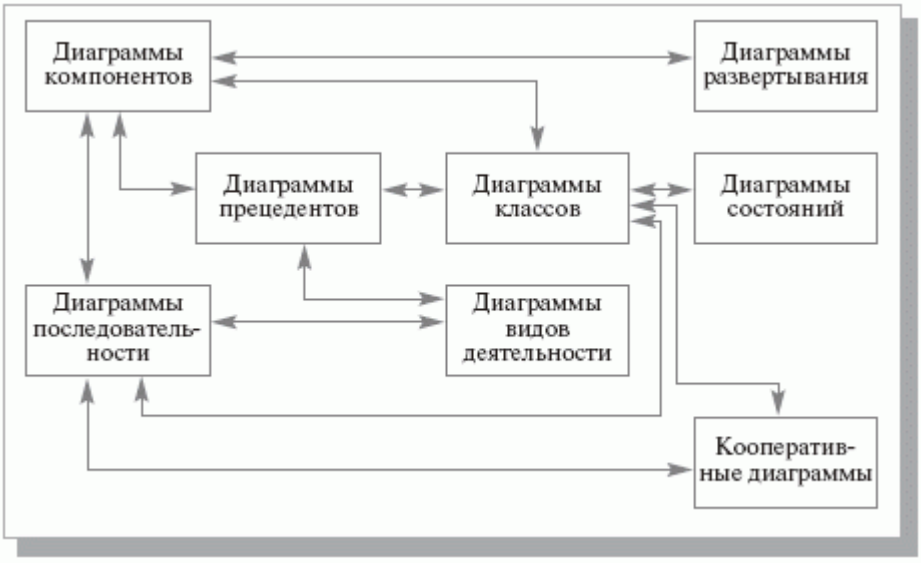


Рис. 10. Взаимосвязи между диаграммами UML

Таким образом, UML-модель включает в себя следующие аспекты.

1. **Структурный:**
   - диаграммы, идентифицирующие бизнес-процессы и бизнес-транзакции, их взаимосвязь, соподчиненность и взаимодействие;
   - диаграммы, описывающие структуру предметной области и иерархическую структуру организации.
2. **Статический**:
   - диаграммы, отражающие совокупность взаимосвязанных объектов, т.е. рассматривающие логическую структуру предметной области, ее внутренние концепции, иерархию объектов и статические связи между ними, структуры данных и объектов;
   - диаграммы, отражающие технологические ресурсы организации.
3. **Динамический**:
   - диаграммы, описывающие поведение (жизненный цикл) бизнес-процессов в их взаимодействии во времени и в пространстве с привязкой к используемым ресурсам и получаемым результатам.

23. Понятие типового проекта, предпосылки типизации. Методы типового проектирования ИС

Типовое проектирование ИС предполагает создание системы из готовых типовых элементов. Основополагающим требованием для применения методов *типового проектирования* является возможность декомпозиции проектируемой ИС на множество составляющих компонентов (подсистем, комплексов задач, программных модулей и т.д.). Для реализации выделенных компонентов выбираются имеющиеся на рынке *типовые проектные решения*, которые настраиваются на особенности конкретного предприятия.
Типовое проектное решение (ТПР)- это тиражируемое (пригодное к многократному использованию) проектное решение.
Принятая классификация *ТПР* основана на уровне декомпозиции системы. Выделяются следующие классы *ТПР*:

- **элементные** *ТПР* - типовые решения по задаче или по отдельному виду обеспечения задачи (информационному, программному, техническому, математическому, организационному);
- **подсистемные** *ТПР* - в качестве элементов типизации выступают отдельные подсистемы, разработанные с учетом функциональной полноты и минимизации внешних информационных связей;
- **объектные** *ТПР* - типовые отраслевые проекты, которые включают полный набор функциональных и обеспечивающих подсистем ИС.

Каждое типовое решение предполагает наличие, кроме собственно функциональных элементов (программных или аппаратных), документации с детальным описанием *ТПР* и процедур настройки в соответствии с требованиями разрабатываемой системы.
Основные особенности различных классов *ТПР* приведены в таблице 1.

Таблица 1. Достоинства и недостатки ТПР

Класс ТПР	**Реализация ТПР**	**Достоинства**	**Недостатки**
Элементные *ТПР*	Библиотеки методо-ориентированных программ	Обеспечивается применение модульного подхода к проектированию и документированию ИС	• большие затраты времени на сопряжение разнородных элементов вследствие информационной, программной и технической несовместимости ; • большие затраты времени на доработку *ТПР* отдельных элементов
Подсистемные *ТПР*	Пакеты прикладных программ	Достигается высокая степень интеграции элементов ИС; Позволяют осуществлять: • модульное проектирование; • параметрическую настройку программных компонентов на различные объекты управления ; Обеспечивают: • сокращение затрат на проектирование и программмирование взаимосвязанных компонентов;	*ТПР* недостаточна с позиции непрерывного инжиниринга деловых процессов; Возникают проблемы в комплексировании разных функциональных подсистем, особенно в случае использования решений нескольких производителей программного обеспечения

		• хорошее документирование отображаемых процессов обработки информации	
Объектные *ТПР*	Отраслевые проекты ИС	Комплексирование всех компонентов ИС за счет методологического единства и информационной, программной и технической совместимости; Открытость архитектуры - позволяет устанавливать *ТПР* на разных программно-технических платформах; Масштабируемость - допускает конфигурацию ИС для переменного числа рабочих мест; Конфигурируемость — позволяет выбирать необходимое подмножество компонентов	Проблемы привязки типового проекта к конкретному объекту управления, что вызывает в некоторых случаях даже необходимость изменения организационно-экономической структуры объекта автоматизации

Для реализации *типового проектирования* используются два подхода: **параметрически-ориентированное** и **модельно-ориентированное проектирование**.

Параметрически-ориентированное проектирование включает следующие этапы: определение критериев оценки пригодности пакетов прикладных программ (ППП) для решения поставленных задач, анализ и оценка доступных ППП по сформулированным критериям, выбор и закупка наиболее подходящего пакета, настройка параметров (доработка) закупленного ППП.

Критерии оценки ППП делятся на следующие группы:

- назначение и возможности пакета;
- отличительные признаки и свойства пакета;
- требования к техническим и программным средствам;
- документация пакета;
- факторы финансового порядка;
- особенности установки пакета;
- особенности эксплуатации пакета;
- помощь поставщика по внедрению и поддержанию пакета;
- оценка качества пакета и опыт его использования;
- перспективы развития пакета.

Внутри каждой группы критериев выделяется некоторое подмножество частных показателей, детализирующих каждый из десяти выделенных аспектов анализа выбираемых ППП.

Числовые значения показателей для конкретных ППП устанавливаются экспертами по выбранной шкале оценок (например, 10-балльной). На их основе формируются групповые оценки и комплексная оценка пакета (путем вычисления средневзвешенных значений). Нормированные взвешивающие коэффициенты также получаются экспертным путем.

Модельно-ориентированное проектирование заключается в адаптации состава и характеристик типовой ИС в соответствии с моделью объекта автоматизации.

Технология проектирования в этом случае должна обеспечивать единые средства для работы как с моделью типовой ИС, так и с моделью конкретного предприятия.
Типовая ИС в специальной базе метаинформации - репозитории - содержит модель объекта автоматизации, на основе которой осуществляется конфигурирование программного обеспечения. Таким образом, *модельно-ориентированное проектирование ИС* предполагает, прежде всего, построение модели объекта автоматизации с использованием специального программного инструментария. Возможно также создание системы на базе *типовой модели ИС* из репозитория, который поставляется вместе с программным продуктом и расширяется по мере накопления опыта проектирования информационных систем для различных отраслей и типов производства.
Репозиторий содержит *базовую (ссылочную) модель ИС*, *типовые (референтные) модели* определенных классов ИС, модели конкретных ИС предприятий.
Базовая модель ИС в репозитории содержит описание бизнес-функций, бизнес-процессов, бизнес-объектов, бизнес-правил, организационной структуры, которые поддерживаются программными модулями типовой ИС.
Типовые модели описывают конфигурации информационной системы для определенных отраслей или типов производства.
Модель конкретного предприятия строится либо путем выбора фрагментов основной или типовой модели в соответствии со специфическими особенностями предприятия, либо путем автоматизированной адаптации этих моделей в результате экспертного опроса.
Построенная модель предприятия в виде метаописания хранится в репозитории и при необходимости может быть откорректирована. На основе этой модели автоматически осуществляется конфигурирование и настройка информационной системы.
Бизнес-правила определяют условия корректности совместного применения различных компонентов ИС и используются для поддержания целостности создаваемой системы.
Модель бизнес-функций представляет собой иерархическую декомпозицию функциональной деятельности предприятия.
Модель бизнес-процессов отражает выполнение работ для функций самого нижнего уровня модели бизнес-функций. Для отображения процессов используется модель управления событиями (EPC - Event-driven Process Chain). Именно модель бизнес-процессов позволяет выполнить настройку программных модулей - приложений информационной системы в соответствии с характерными особенностями конкретного предприятия.
Модели бизнес-объектов используются для интеграции приложений, поддерживающих исполнение различных бизнес-процессов.
Модель организационной структуры предприятия представляет собой иерархическую структуру подчинения подразделений и персонала.
Внедрение типовой информационной системы начинается с анализа требований к конкретной ИС, которые выявляются на основе результатов предпроектного обследования объекта автоматизации. Для оценки соответствия этим требованиям программных продуктов может использоваться описанная выше методика оценки ППП. После выбора программного продукта на базе имеющихся в нем референтных моделей строится предварительная модель ИС, в которой отражаются все особенности реализации ИС для конкретного предприятия. Предварительная модель является основой для выбора *типовой модели* системы и определения перечня компонентов, которые будут реализованы с использованием других программных средств или потребуют разработки с помощью имеющихся в составе типовой ИС инструментальных средств.
Реализация типового проекта предусматривает выполнение следующих операций:

- установку глобальных параметров системы;
- задание структуры объекта автоматизации;
- определение структуры основных данных;
- задание перечня реализуемых функций и процессов;

- описание интерфейсов;
- описание отчетов;
- настройку авторизации доступа;
- настройку системы архивирования.

24. Принципы и особенности проектирования корпоративных ИС

Внедрение корпоративных информационных систем как основы для комплексной автоматизации деятельности предприятий направлено на поддержку принятия управленческих решений менеджерами высшего звена корпорации. А это предполагает, что предварительно должны быть решены задачи автоматизации рабочих мест, связанных с выполнением текущих производственных функций и оперативным управлением производственными процессами на уровне нижнего и среднего звена менеджеров.
До последнего времени существовало два подхода к решению задачи комплексной автоматизации деятельности предприятия:

- Поэтапная разработка корпоративной системы собственными силами (включая использование готовых или заказных программных продуктов сторонних фирм и организаций, позволяющих автоматизировать отдельные рабочие места или производственные процессы) и
- Внедрение готовой информационной системы корпоративного уровня

Преимущество первого подхода состояло в том, что в создаваемой собственными силами системе в наибольшей степени можно было учесть потребности и специфику работы конкретного предприятия. Хотя, следует отметить, не всегда это качество является достоинством - автоматизация плохо организованных бизнес-процессов способна только ухудшить ситуацию на предприятии. Поэтому, разработке информационной системы должен предшествовать анализ, а если необходимо, то и реинжиниринг производственной деятельности. Кроме того, "эволюционный" характер постепенных улучшений с возможностью поэтапного финансирования разработок во многих случаях выглядит более привлекательно по сравнению с риском кардинальных преобразований и значительных затрат, связанных с внедрением готовых систем. К сожалению, этот путь решения проблемы автоматизации оказывается слишком растянут во времени, часто превращаясь в "постоянный процесс разработки", когда разработчики не успевают за изменениями, происходящими в организации.
Корпорации, располагающие необходимыми финансовыми средствами, отдают предпочтение готовым программным системам. Однако, успех от внедрения такой системы в значительной степени зависит от готовности (и возможности) корпорации работать по "правилам", диктуемым приобретаемой информационной системой. "Готовая" информационная система имеет модульную архитектуру и процесс внедрения такой системы может быть выполнен по этапам - начиная с модулей, автоматизирующих наиболее критичные участки работы. При этом, обеспечивается "целостность" системы, позволяющая воспользоваться на соответствующих рабочих местах новыми функциями подключаемых модулей.
Опыт разработки "готовых" информационных систем позволил сформировать новый подход к созданию корпоративных информационных систем, основанный на "сборке" систем из программных "компонент" различных фирм-производителей. Компонентная архитектура корпоративных информационных систем стала возможной благодаря поддержке ведущими производителями программного обеспечения общих стандартов на проектирование, разработку и технологию компонентной "сборки" информационных систем, реализуемых на различных программно-аппаратных платформах.
На современном этапе развития информационных технологий компонентная технология создания корпоративных информационных систем выглядит наиболее привлекательной и перспективной. Действительно, она объединяет гибкость в выборе необходимых компонент информационной системы, свойственную разработке системы собственными

силами, с надежностью кода и функциональной полнотой, проверенными многократным использованием, характерным для коммерческих программных продуктов. Более того, компонентная технология позволяет оперативно вносить изменения в существующую информационную систему, не нарушая ее работоспособности. При этом новые приложения могут работать с новыми модулями, а старые - с прежними модулями, которые остаются в системе. Снимается проблема "унаследованных" систем - нет необходимости их замены для изменения или расширения функциональности, а значит уменьшаются затраты на сопровождение и модернизацию информационной системы. Для того, чтобы компонентная архитектура информационных систем стала реальностью, необходимы три условия:

- наличие методологии анализа и проектирования информационных систем, обеспечивающих компонентную разработку и "сборку" систем;
- сформированный рынок готовых программных компонент, поддерживающих общие стандарты на технологию разработки и "сборки" компонент;
- стандартные компоненты программного обеспечения "инфраструктуры" информационной системы, поддерживающие взаимодействие между компонентами системы.

Компонентная технология проектирования и разработки информационных систем на сегодняшний день располагает необходимым арсеналом средств - начиная от инструментов визуального анализа и моделирования, поддерживающих существующие средства разработки, и кончая широким выбором библиотек готовых компонент, включая компоненты "инфраструктуры" для различных программно-аппаратных платформ. А это значит, что информационные технологии стоят на пороге появления "конструкторов" готовых систем, состоящих из наборов компонент от различных производителей. Особенно сильно тенденция к созданию многокомпонентных систем проявилась в технологии интернет/интранет.

Intranet-приложение - это корпоративная система, для организации которой используются механизмы Internet. Intranet-система может основываться на локальной сети компьютеров, собственной корпоративной глобальной сети или виртуальной корпоративной подсети Internet. Различают несколько типов Intranet-систем, для реализации каждого из которых, вообще говоря, применяются разные средства:

1. **Коммуникационные** Intranet-системы предназначены главным образом для связывания территориально разнесенных подразделений корпорации, уменьшая потребность в многочисленных выделенных линиях связи. При реализации систем этого типа следует обращать особое внимание на эффективность, соответствие стандартам и управляемость системы.
2. **Интегрирующие** Intranet-системы служат для интеграции разнородных существующих коммуникационных и обрабатывающих корпоративных подсистем. С этой точки зрения достоинством Intranet-системы является поддержка общего интерфейса доступа к "унаследованным" системам и установление связи между ними за счет понимаемого всеми гипертекстового представления информации.
3. Если от Intranet-системы требуется обеспечение широкого доступа к большим объемам информации, в частности, мультимедийной, то особое внимание требуется уделить выбору **базового сервера баз данных**. Нужно учитывать возможности сервера по части управления очень большими данными и поддержки сложных типов данных.
4. Intranet-системы с упрощенной для пользователей процедурой доступа обычно основываются на механизме **электронной подписи**. Такие системы должны быть особенно надежно защищены от внешнего мира.

Конечно, в общем случае информационная Intranet-система может включать свойства каждого из перечисленных типов, и в этом случае при ее разработке придется учитывать все требования.

Основные понятия Intranet

Поскольку для разработки Intranet-систем используются методы и средства Internet, и главным образом, технология WWW, то и понятия и термины Internet и Intranet совпадают. Приведем некоторые из них:

HTTP (HyperText Transfer Protocol) - протокол обмена гипертекстовой информацией;

URL (Universal Resource Locator) - универсальный локатор ресурсов. Используется в качестве универсальной схемы адресации ресурсов в сети.

HTML (HyperText Markup Language) - язык гипертекстовой разметки документов. Специальная форма подготовки документов для их опубликования в World Wide Web.

CGI (Common Gateway Interface) - спецификация на формат обмена данными между сервером протокола HTTP и прикладной программой.

API (Application Program Interface) - в данном контексте это спецификация, определяющая правила обмена данными между сервером и программным модулем, который должен быть включен в состав сервера.

VRML (Virtual Reality Modeling Language) - язык описания трехмерных сцен и взаимодействия трехмерных объектов.

Javaapplets - мобильные (независимые от архитектуры "железа") программные коды, написанные на языке программирования Java.

Java - объектно-ориентированный язык программирования, разработанный компанией Sun Microsystems и используемый в качестве основного средства мобильного программирования.

MIME (Multipurpose Internet Mail Exchange) - формат почтового сообщения Internet. В данном контексте стандарт MIME используется для установления соответствия между типом информационного файла, именем этого файла и программой просмотра этого файла.

CCI (Common Client Interface) - спецификация обмена данными меду прикладной программой и браузером Mosaic. В случае применения программного обеспечения, выполненного согласно CCI, браузер превращается в сервер-посредник для программного обеспечения пользователя.

Возможные архитектуры Intranet-приложений

Технологии Internet обеспечивают широкий диапазон возможных архитектур Intranet-систем. Краткое рассмотрение некоторых вариантов.

1. **Решения, ориентированные на клиентскую часть системы**

Видимо, это наиболее тривиальная архитектура. Аналогично тому, как это было в файл-серверных решениях, вся прикладная часть системы находится в клиенте, который взаимодействует с разнообразными серверами Internet (электронной почты, ftp и т.д.) и серверами, управляющими файлами и/или базами данных. Клиент должен быть достаточно "толстым", чтобы быть в состоянии уметь работать с разными видами серверов (для каждого из них требуется индивидуальная клиентская часть) и одновременно выполнять прикладную обработку данных. Серверы могут быть разной толщины в зависимости от своей функциональной ориентированности (естественно, что сервер электронной почты нуждается в существенно меньшем числе ресурсов, чем мощный сервер баз данных) (рис. 1.).

2. **Трехзвенные архитектуры (Web-ориентированные)**

Ориентация на использование Web-технологии позволяет одновременно добиться двух эффектов. Во-первых, за счет использования CGI или API можно перенести на сторону сервера часть логики приложения. Во-вторых, используя технику шлюзования Web-сервера (опять же применяя CGI-шлюзы или API) можно работать через Web-сервер (в стандартном интерфейсе) с другими серверами. Клиента можно сделать очень "тонким", Web-сервер будет достаточно "толстым", а уж остальные такими, как велит судьба (рис. 2).

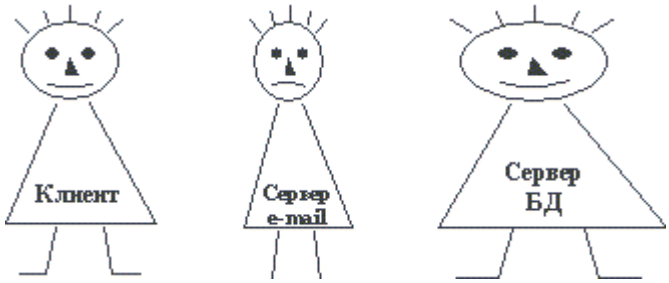


Рис. 1. "Толстый" клиент и серверы разной толщины в Intranet-системах, ориентированных на клиента

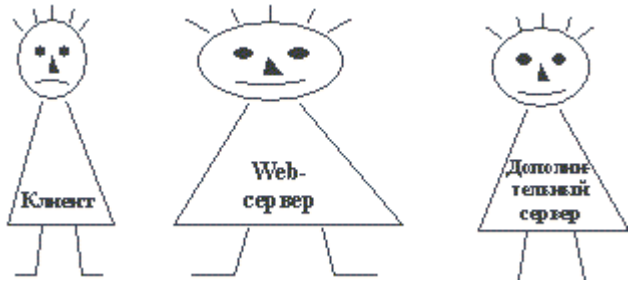


Рис. 2. "Тонкий" клиент, "толстый" Web-сервер и сравнительно "стройный" дополнительный сервер

3. **Решения, основанные на использовании языка Java**

Язык Java можно использовать для программирования Java-апплетов, которые выполняются на стороне клиента, и Java-приложений, выполняемых на стороне сервера. Естественно, клиент, приспособленный к выполнению Java-апплетов, становится несколько толще. Что же касается использования Java-программ на стороне сервера, то большее значение может иметь сравнительная надежность этого языка (в том смысле, что интерпретируемая Java-программа с меньшей вероятностью может нанести вред серверу).

25. Оценка качества информационной системы

К наиболее часто упоминаемым факторам, определяющим успех программных проектов, относятся следующие:

- своевременная поставка;
- достижение целей проекта;
- установление тесных связей и взаимопонимание с заказчиками.

Качество ИС связано с дефектами, заложенными на этапе проектирования и проявляющимися в процессе эксплуатации. Любые свойства ИС, в том числе и дефектологические, могут проявляться лишь во взаимодействии с внешней средой, включающей технические средства, персонал, информационное и программное окружение.

В зависимости от целей исследования и этапов жизненного цикла ИС дефектологические свойства разделяют на дефектогенность, дефектабельность и дефектоскопичность.

Дефектогенность определяется влиянием следующих факторов:

• численностью разработчиков ИС, их профессиональными и психофизиологическими характеристиками;

• условиями и организацией процесса разработки ИС;

• характеристиками инструментальных средств и компонент ИС;

• сложностью задач, решаемых ИС;

• степенью агрессивности внешней среды (потенциальной возможностью внешней среды вносить преднамеренные дефекты, на пример, воздействие вирусов).

Дефектабельность характеризует наличие дефектов ИС и определяется их количеством и местонахождением. Другими факторами, влияющими на дефектабельность являются:
• структурно-конструктивные особенности ИС;
• интенсивность и характеристики ошибок, приводящих к дефектам.
Дефектоскопичность характеризует возможность проявления дефектов в виде отказов и сбоев в процессе отладки, испытаний или эксплуатации. На дефектоскопичность влияют:
• количество, типы и характер распределения дефектов в ИС;
• устойчивость ИС к проявлению дефектов;
• характеристики средств контроля и диагностики дефектов;
• квалификация обслуживающего персонала.
Оценка качества ИС является крайне сложной задачей в виду многообразия интересов пользователей. Поэтому невозможно предложить одну универсальную меру качества и приходится использоватъ ряд характеристик, охватывающих весь спектр предъявляемых требований.
Наиболее близки к задачам оценки качества ИС модели качества программного обеспечения, являющегося одной из важных составных частей ИС. В настоящее время используется несколько абстрактных моделей качества программного обеспечения, основанных на определениях **характеристики качества**, **показателя качества**, **критерия** и **метрики**.
Определим понятие **критерия** как независимого атрибута ИС или процесса ее создания. При оценке качества ИС прежде всего определяются **характеристики качества**, в числе которых могут быть, например, общая полезность, исходная полезность, удобство эксплуатации. Характеристика оценивается в соответствии с некоторым критерием. Далее формируются **показатели**, к числу которых могут быть отнесены: практичность, целостность, корректность, удобство обслуживания, оцениваемость, гибкость, адаптируемость, мобильность, возможность взаимодействия. Каждому показателю качества ставится в соответствие группа критериев. Для указанных выше показателей ниже приведены возможные критерии. Надо отметить, что один и тот же критерий может характеризовать несколько показателей:
практичность — работоспособность, возможность обучения, коммуникативность, объем ввода, скорость ввода-вывода;
целостность — регулирование доступа, контроль доступа;
эффективность — эффективность использования памяти, эффективность функционирования;
корректность — трассируемость, завершенность, согласованность;
надежность — точность, устойчивость к ошибкам, согласованность, простота;
удобство обслуживания — согласованность, простоту, краткость, информативность, модульность;
оцениваемость — простоту, наличие измерительных средств, информативность, модульность;
гибкость — распространяемость, общность, информатированность, модульность;
адаптируемость — общность, информативность, модульность, аппаратную независимость, программную независимость;
мобильность — информативность, модульность, аппаратную не зависимость, программную независимость;
возможность взаимодействия — модульность, увифицируемость процедур связи, *унифицируемость данных*.
С помощью **метрик** можно дать количественную или качественную оценку качества ИС. Различают следующие виды метрик и шкал для измерения критериев.
Первый тип — метрики, которые используют интервальную шкалу, характеризуемую относительными величинами или реально измеряемыми физическими показателями,

например, временем наработки на отказ, вероятностью ошибки, объемом информации и др.

Второй тип — метрики, которым соответствует порядковая шкала, позволяющая ранжировать характеристики путем сравнения с опорными значениями.

Третий тип — метрики, которым соответствуют номинальная или категорированная шкала, определяющая наличие рассматриваемого свойства или признака у рассматриваемого объекта без учета градаций по этому признаку. Так, например, интерфейс может быть «простым для понимания», «умеренно простым», «сложным для понимания».

Таким образом, с помощью критерия может быть измерена характеристика качества ИС на основе той или иной метрики. Совокупность нескольких критериев определяет показатель качества, формируемый исходя из требований, предъявляемых к ИС

Конструктивные критерии предназначены для оценки компонент ИС, не зависящих от целевого назначения.

В настоящее время наибольшее распространение получила иерархическая модель взаимосвязи компонент качества ИС. Развитием иерархического подхода является представленная на рис. 1. модель классификации критериев качества информационных систем. С помощью функциональных критериев оценивается степень выполнения ИС основных целей или задач.

Одним из путей обеспечения качества ИС является сертификация. В США Радиотехническая комиссия по аэронавтике в своем руководящем документе определяет процесс сертификации следующим образом: «Сертификация — процесс официального утверждения государственным полномочным органом ... выполняемой функции системы ... путем удостоверения, что функция ... удовлетворяет всем требованиям заказчика, а также государственным нормативным документам». К сожалению, в настоящее время не существует стандартов, полностью удовлетворяющих оценке качества ИС. В западно-европейских странах имеется ряд стандартов, определяющих основы сертификации программных систем. Существующая в нашей стране система нормативно-технических документов относит программное обеспечение к «продукции производственно-технического назначения», которая рассматривается как материальный объект. Однако программное обеспечение является скорее абстрактной нематериальной сферой. Существующие ГОСТы явно устарели и являются неполными.

Р и с. 1. Модель классификации критериев качества информационных систем