ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ

Федеральное государственное образовательное бюджетное

учреждение высшего профессионального образования

«САНКТ-ПЕТЕРБУРГСКИЙ

ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ

им. проф. М. А. БОНЧ-БРУЕВИЧА»

А.Н.Губин

Современные методы проектирования информационных систем

Краткий конспект лекций

Раздел 2. Основные технологии проектирования ИС.

САНКТ-ПЕТЕРБУРГ

2015

Содержание

Раздел 4. Типизация проектных решений.

4.1. Особенности технологии типового проектирования ИС.

4.2. Основные методы типового проектирования ИС.

4.3. RAD – технология проектирования.

Раздел 5. Управление проектами ИС.

5.1. Жизненный цикл ИС.

5.2. Расширение и обновление ИС.

5.3. Сопровождение, контроль эффективности и качества ИС.

5.4. Мониторинг безопасности ИС.

5.5. Перспективы и основные направления развития ИС и средств их проектирования.

Раздел 2. Основные технологии проектирования ИС.

2.1. Структурный подход к проектированию ИС.

Сущность структурного подхода к разработке ИС заключается в ее декомпозиции (разбиении) на автоматизируемые функции: система разбивается на функциональные подсистемы, которые в свою очередь делятся на подфункции, подразделяемые на задачи и так далее [1, 2, 3].

Процесс разбиения продолжается вплоть до получения конкретных процедур. При этом разрабатываемая система сохраняет целостное представление, в котором все составляющие компоненты взаимоувязаны.

В тоже время при разработке системы "снизу-вверх" от отдельных задач ко всей системе целостность теряется, возникают проблемы при информационной стыковке отдельных компонентов ИС.

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

- принцип "разделяй " - принцип решения сложных проблем путем их разбиения на множество меньших независимых задач, легких для понимания и решения;
- принцип иерархического упорядочивания - принцип организации составных частей проблемы в иерархические древовидные структуры с добавлением новых деталей на каждом уровне.

Выделение двух базовых принципов не означает, что остальные принципы являются второстепенными, поскольку игнорирование любого из них может привести к непредсказуемым последствия. Основными из этих принципов являются:

- принцип абстрагирования - заключается в выделении существенных аспектов системы и отвлечения от несущественных;
- принцип формализации - заключается в необходимости строгого подхода к решению проблемы;
- принцип непротиворечивости - заключается в обоснованности и согласованности элементов;
- принцип структурирования данных - заключается в том, что данные должны быть структурированы и иерархически организованы.

В структурном анализе используются в основном две группы средств, иллюстрирующих функции, выполняемые системой и отношения между данными.

Каждой группе средств соответствуют определенные виды моделей (диаграмм), наиболее распространенными среди которых являются следующие:

- SADT (Structured Analysis and Design Technique) модели и соответствующие функциональные диаграммы;
- DFD (Data Flow Diagrams) диаграммы потоков данных;
- ERD (Entity-Relationship Diagrams) диаграммы "сущность-связь" .

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

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

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 (рисунок 2.1).

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

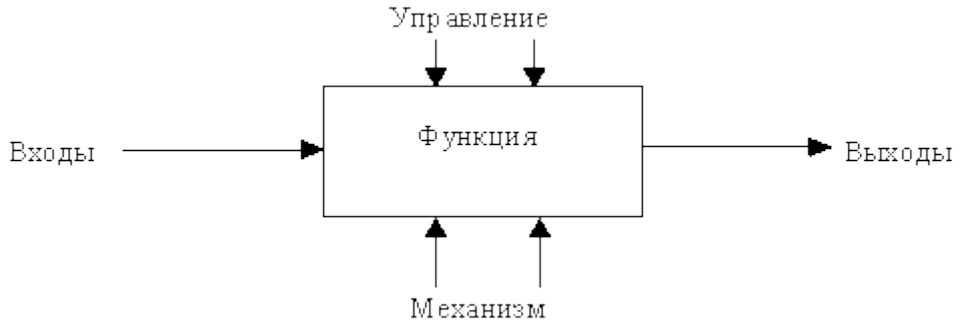

Рис. 2.1. Функциональный блок и интерфейсные дуги

На рисунке 2.2, 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.

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.

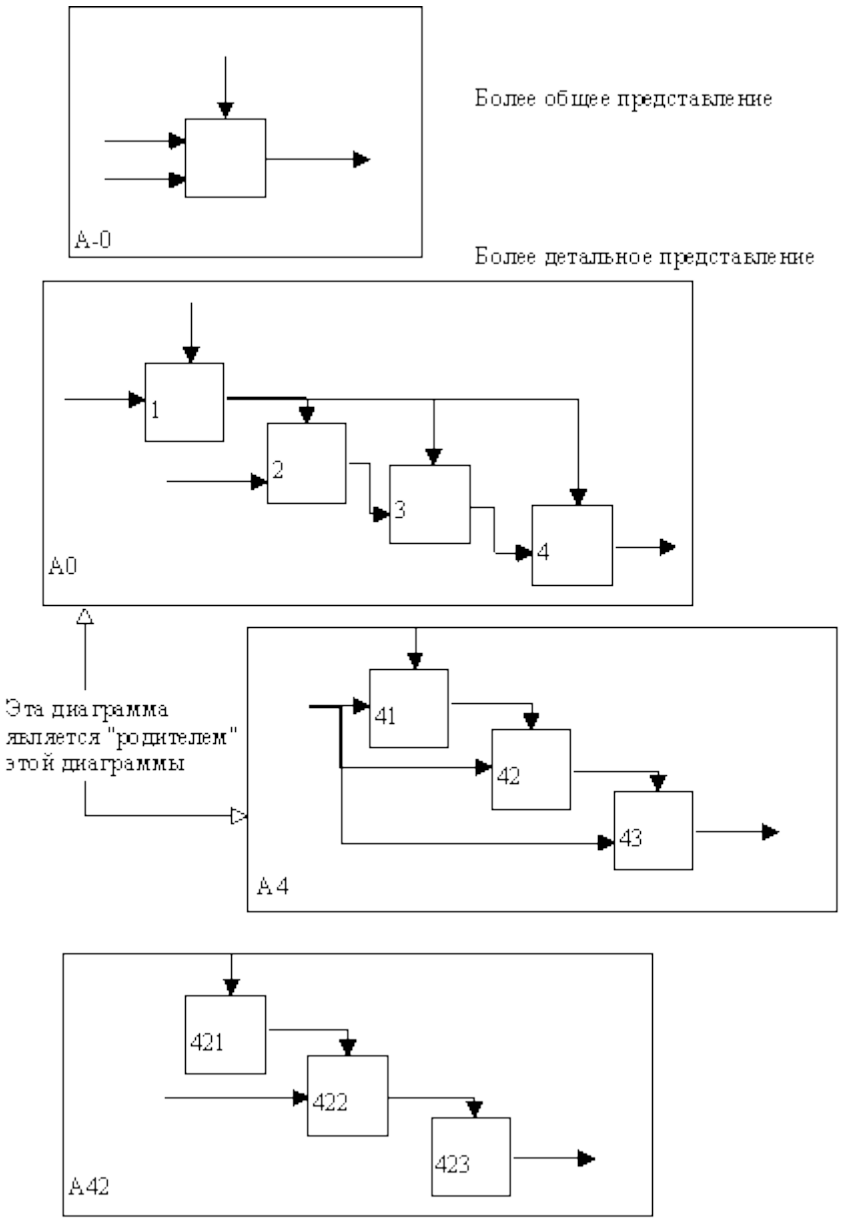

Рис. 2.2. Структура SADT-модели. Декомпозиция диаграмм

На рисунках 2.3 - 2.5 представлены различные варианты выполнения функций и соединения дуг с блоками.

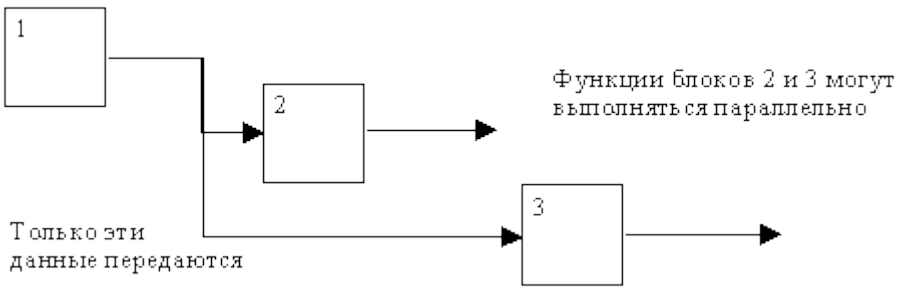

Рис. 2.3. Одновременное выполнение

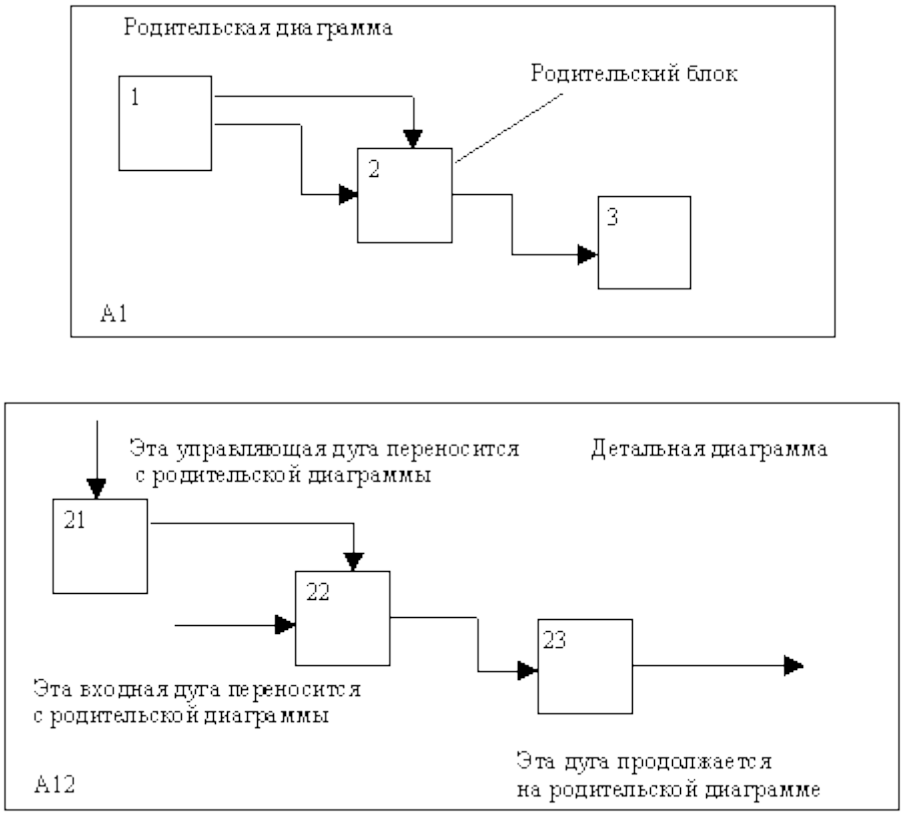

Рис. 2.4. Соответствие должно быть полным и непротиворечивым

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. (рисунок 2.5).

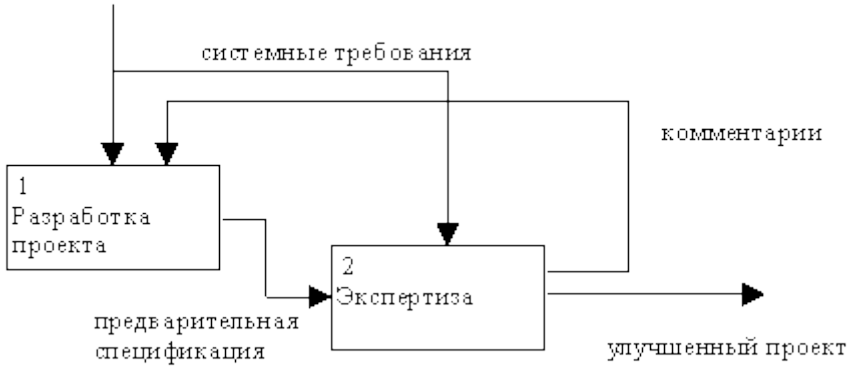

Рис. 2.5. Пример обратной связи

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 (рисунок 2.6).

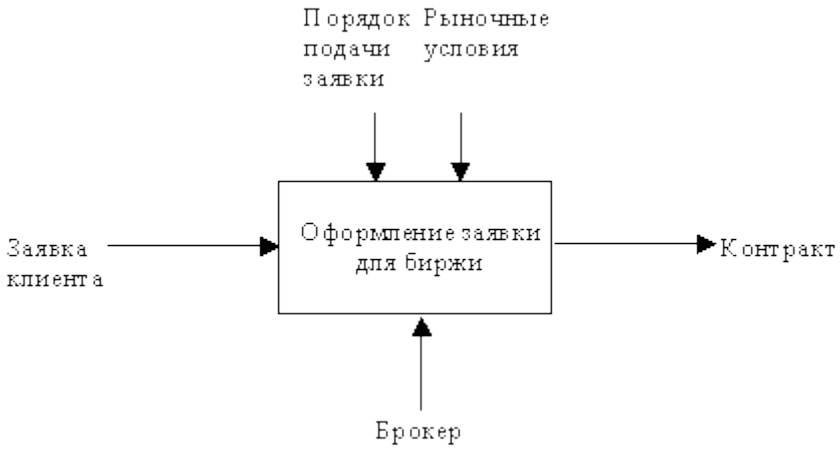

Рис. 2.6. Пример механизма

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 рисунке 2.7 показано типичное дерево диаграмм.

Рис. 2.7. Иерархия диаграмм

Одним из важных моментов при проектировании ИС с помощью методологии SADT является точная согласованность типов связей между функциями. Различают, по крайней мере, семь типов связывания (табл. 2.1).

Значимость связей различного типа

Таблица 2.1

Тип связи	**Относительная значимость**
Случайная	0
Логическая	1
Временная	2
Процедурная	3
Коммуникационная	4
Последовательная	5
Функциональная	6

Ниже каждый тип связи кратко определен и проиллюстрирован с помощью типичного примера из SADT.

(0) Тип случайной связности: наименее желательный.

Случайная связность возникает, когда конкретная связь между функциями мала или полностью отсутствует. Это относится к ситуации, когда имена данных на SADT-дугах в одной диаграмме имеют малую связь друг с другом. Крайний вариант этого случая показан на рисунке 2.8.

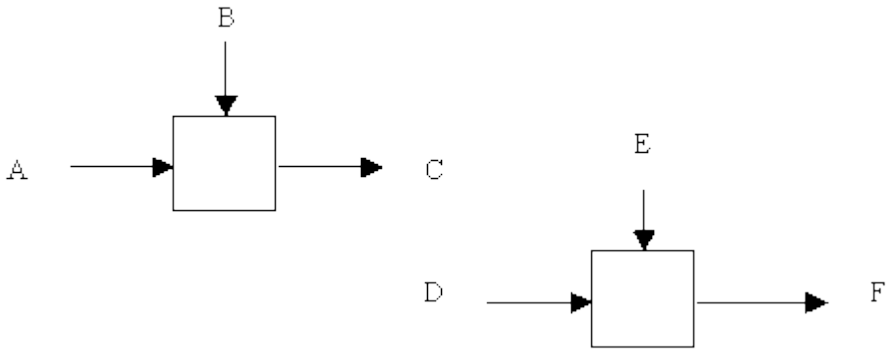

Рис. 2.8. Случайная связность

(1) Тип логической связности. Логическое связывание происходит тогда, когда данные и функции собираются вместе вследствие того, что они попадают в общий класс или набор элементов, но необходимых функциональных отношений между ними не обнаруживается.

(2) Тип временной связности. Связанные по времени элементы возникают вследствие того, что они представляют функции, связанные во времени, когда данные используются одновременно или функции включаются параллельно, а не последовательно.

(3) Тип процедурной связности. Процедурно-связанные элементы появляются сгруппированными вместе вследствие того, что они выполняются в течение одной и той же части цикла или процесса. Пример процедурно-связанной диаграммы приведен на рисунке 2.9.

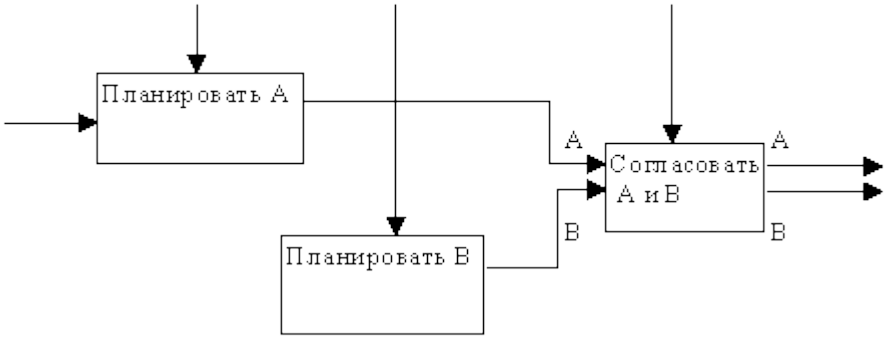

Рис. 2.9. Процедурная связность

(4) Тип коммуникационной связности. Диаграммы демонстрируют коммуникационные связи, когда блоки группируются вследствие того, что они используют одни и те же входные данные и/или производят одни и те же выходные данные (рисунок 2.10).

(5) Тип последовательной связности. На диаграммах, имеющих последовательные связи, выход одной функции служит входными данными для следующей функции. Связь между элементами на диаграмме является более тесной, чем на рассмотренных выше уровнях связок, поскольку моделируются причинно-следственные зависимости (рисунок 2.11).

(6) Тип функциональной связности. Диаграмма отражает полную функциональную связность, при наличии полной зависимости одной функции от другой. Диаграмма, которая является чисто функциональной, не содержит чужеродных элементов, относящихся к последовательному или более слабому типу связности. Одним из способов определения функционально-связанных диаграмм является рассмотрение двух блоков, связанных через управляющие дуги, как показано на рисунке 2.12.

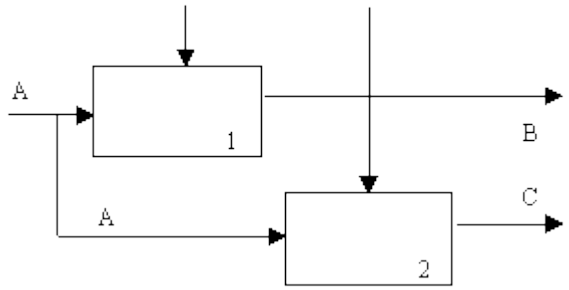

Рис. 2.10. Коммуникационная связность

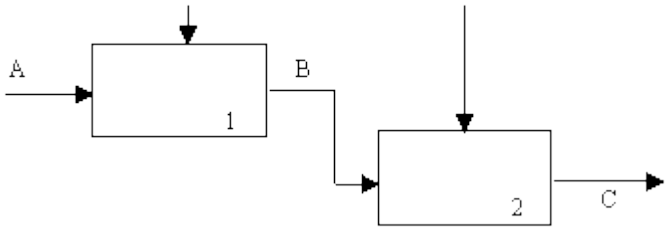

Рис. 2.11. Последовательная связность

В математических терминах необходимое условие для простейшего типа функциональной связности, показанной на рисунке 2.12, имеет следующий вид:

```
C = g(B) = g(f(A))
```

Ниже в таблице 2.2 представлены все типы связей, рассмотренные выше. Важно отметить, что уровни 4-6 устанавливают типы связностей, которые разработчики считают важнейшими для получения диаграмм хорошего качества.

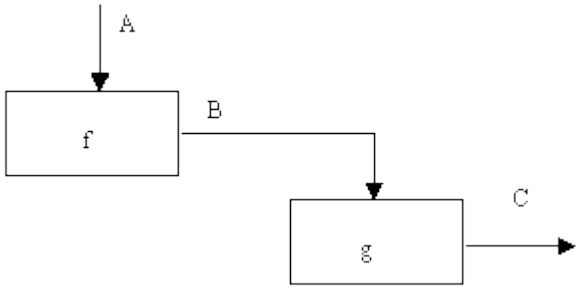

Рис. 2.12. Функциональная связность

Таблица 2.2

Значимость	Тип связности	Для функций	Для данных
0	Случайная	Случайная	Случайная
1	Логическая	Функции одного и того же множества или типа (например, "редактировать все входы")	Данные одного и того же множества или типа
2	Временная	Функции одного и того же периода времени (например, "операции инициализации")	Данные, используемые в каком-либо временном интервале
3	Процедурная	Функции, работающие в одной и той же фазе или итерации (например, "первый проход компилятора")	Данные, используемые во время одной и той же фазы или итерации
4	Коммуникационная	Функции, использующие одни и те же данные	Данные, на которые воздействует одна и та же деятельность
5	Последовательная	Функции, выполняющие последовательные преобразования одних и тех же данных	Данные, преобразуемые последовательными функциями
6	Функциональная	Функции, объединяемые для выполнения одной функции	Данные, связанные с одной функцией

2.2. Особенности современных методов и средств проектирования ИС, основанных на CASE-технологии.

Моделирование потоков данных (процессов)

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

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

- внешние сущности;
- системы/подсистемы;
- процессы;
- накопители данных;
- потоки данных.

Внешние сущности

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

часть процессов ИС может быть вынесена за пределы диаграммы и представлена как внешняя сущность.

Внешняя сущность обозначается квадратом (рисунок 2.13), расположенным как бы "над" диаграммой и бросающим на нее тень, для того, чтобы можно было выделить этот символ среди других обозначений:

Рис. 2.13. Внешняя сущность

Системы и подсистемы

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

Подсистема (или система) на контекстной диаграмме изображается следующим образом (рисунок 2.14).

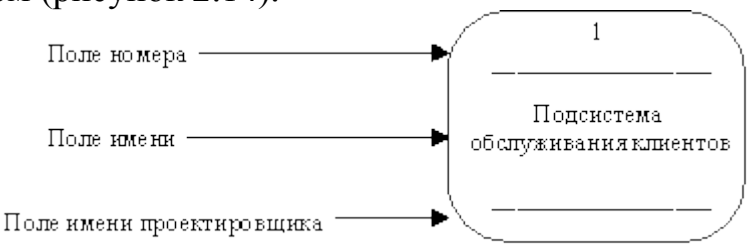

Рис. 2.14. Подсистема

Номер подсистемы служит для ее идентификации. В поле имени вводится наименование подсистемы в виде предложения с подлежащим и соответствующими определениями и дополнениями.

Процессы

Процесс представляет собой преобразование входных потоков данных в выходные в соответствии с определенным алгоритмом. Физически процесс может быть реализован различными способами: это может быть подразделение организации (отдел), выполняющее обработку входных документов и выпуск отчетов, программа, аппаратно реализованное логическое устройство и т.д.

Процесс на диаграмме потоков данных изображается, как показано на рисунке 2.15.

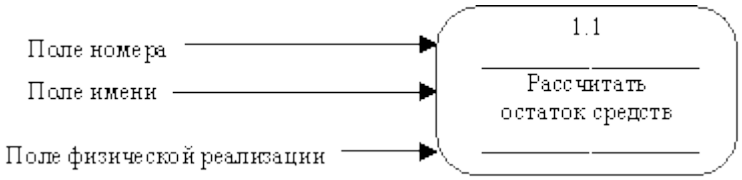

Рис. 2.15. Процесс

Номер процесса служит для его идентификации. В поле имени вводится наименование процесса в виде предложения с активным недвусмысленным глаголом в неопределенной форме (вычислить, рассчитать, проверить, определить, создать, получить), за которым следуют существительные в винительном падеже, например:

- "Ввести сведения о клиентах";
- "Выдать информацию о текущих расходах";
- "Проверить кредитоспособность клиента".

Использование таких глаголов, как "обработать", "модернизировать" или "отредактировать" означает, как правило, недостаточно глубокое понимание данного процесса и требует дальнейшего анализа.

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

Потоки данных

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

Поток данных на диаграмме изображается линией, оканчивающейся стрелкой, которая показывает направление потока (рисунок 2.17). Каждый поток данных имеет имя, отражающее его содержание.

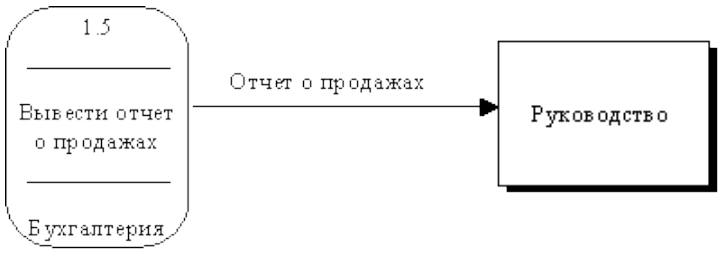

Рис. 2.17. Поток данных

Построение иерархии диаграмм потоков данных

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

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

- наличие большого количества внешних сущностей (десять и более);
- распределенная природа системы;
- многофункциональность системы с уже сложившейся или выявленной группировкой функций в отдельные подсистемы.

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

Иерархия контекстных диаграмм определяет взаимодействие основных функциональных подсистем проектируемой ИС как между собой, так и с внешними входными и выходными потоками данных и внешними объектами (источниками и приемниками информации), с которыми взаимодействует ИС.

Разработка контекстных диаграмм решает проблему строгого определения функциональной структуры ИС на самой ранней стадии ее проектирования, что особенно важно для сложных многофункциональных

систем, в разработке которых участвуют разные организации и коллективы разработчиков.

После построения контекстных диаграмм полученную модель следует проверить на полноту исходных данных об объектах системы и изолированность объектов (отсутствие информационных связей с другими объектами).

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

- правило балансировки - означает, что при детализации подсистемы или процесса детализирующая диаграмма в качестве внешних источников/приемников данных может иметь только те компоненты (подсистемы, процессы, внешние сущности, накопители данных), с которыми имеет информационную связь детализируемая подсистема или процесс на родительской диаграмме;
- правило нумерации - означает, что при детализации процессов должна поддерживаться их иерархическая нумерация. Например, процессы, детализирующие процесс с номером 12, получают номера 12.1, 12.2, 12.3 и т.д.

Миниспецификация (описание логики процесса) должна формулировать его основные функции таким образом, чтобы в дальнейшем специалист, выполняющий реализацию проекта, смог выполнить их или разработать соответствующую программу.

Миниспецификация является конечной вершиной иерархии ДПД. Решение о завершении детализации процесса и использовании миниспецификации принимается аналитиком исходя из следующих критериев:

- наличия у процесса относительно небольшого количества входных и выходных потоков данных (2-3 потока);
- возможности описания преобразования данных процессом в виде последовательного алгоритма;
- выполнения процессом единственной логической функции преобразования входной информации в выходную;
- возможности описания логики процесса при помощи миниспецификации небольшого объема (не более 20-30 строк).

При построении иерархии ДПД переходить к детализации процессов следует только после определения содержания всех потоков и накопителей данных, которое описывается при помощи структур данных. Структуры

данных конструируются из элементов данных и могут содержать альтернативы, условные вхождения и итерации. Условное вхождение означает, что данный компонент может отсутствовать в структуре. Альтернатива означает, что в структуру может входить один из перечисленных элементов. Итерация означает вхождение любого числа элементов в указанном диапазоне. Для каждого элемента данных может указываться его тип (непрерывные или дискретные данные). Для непрерывных данных может указываться единица измерения (кг, см и т.п.), диапазон значений, точность представления и форма физического кодирования. Для дискретных данных может указываться таблица допустимых значений.

После построения законченной модели системы ее необходимо верифицировать (проверить на полноту и согласованность). В полной модели все ее объекты (подсистемы, процессы, потоки данных) должны быть подробно описаны и детализированы. Выявленные недетализированные объекты следует детализировать, вернувшись на предыдущие шаги разработки. В согласованной модели для всех потоков данных и накопителей данных должно выполняться правило сохранения информации: все поступающие куда-либо данные должны быть считаны, а все считываемые данные должны быть записаны.

Case-метод Баркера

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

Наиболее распространенным средством моделирования данных являются диаграммы "сущность-связь" (ERD). С их помощью определяются важные для предметной области объекты (сущности), их свойства (атрибуты) и отношения друг с другом (связи). ERD непосредственно используются для проектирования реляционных баз данных.

Нотация ERD была впервые введена П. Ченом (Chen) и получила дальнейшее развитие в работах Баркера [8]. Метод Баркера будет излагаться на примере моделирования деятельности компании по торговле автомобилями. Ниже приведены выдержки из интервью, проведенного с персоналом компании.

Главный менеджер: одна из основных обязанностей - содержание автомобильного имущества. Он должен знать, сколько заплачено за машины и каковы накладные расходы. Обладая этой информацией, он может установить нижнюю цену, за которую мог бы продать данный экземпляр. Кроме того, он несет ответственность за продавцов и ему нужно знать, кто что продает и сколько машин продал каждый из них.

Продавец: ему нужно знать, какую цену запрашивать и какова нижняя цена, за которую можно совершить сделку. Кроме того, ему нужна основная информация о машинах: год выпуска, марка, модель и т.п.

Администратор: его задача сводится к составлению контрактов, для чего нужна информация о покупателе, автомашине и продавце, поскольку именно контракты приносят продавцам вознаграждения за продажи.

Первый шаг моделирования - извлечение информации из интервью и выделение сущностей.

Сущность (Entity) - реальный либо воображаемый объект, имеющий существенное значение для рассматриваемой предметной области, информация о котором подлежит хранению (рисунок 2.18).

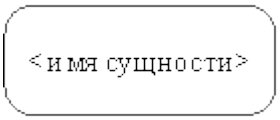

Рис. 2.18. Графическое изображение сущности

Каждая сущность должна обладать уникальным идентификатором. Каждый экземпляр сущности должен однозначно идентифицироваться и отличаться от всех других экземпляров данного типа сущности. Каждая сущность должна обладать некоторыми свойствами:

- каждая сущность должна иметь уникальное имя, и к одному и тому же имени должна всегда применяться одна и та же интерпретация. Одна и та же интерпретация не может применяться к различным именам, если только они не являются псевдонимами;
- сущность обладает одним или несколькими атрибутами, которые либо принадлежат сущности, либо наследуются через связь;
- сущность обладает одним или несколькими атрибутами, которые однозначно идентифицируют каждый экземпляр сущности;
- каждая сущность может обладать любым количеством связей с другими сущностями модели.

Обращаясь к приведенным выше выдержкам из интервью, видно, что сущности, которые могут быть идентифицированы с главным менеджером - это автомашины и продавцы. Продавцу важны автомашины и связанные с их продажей данные. Для администратора важны покупатели, автомашины, продавцы и контракты. Исходя из этого, выделяются 4 сущности (автомашина, продавец, покупатель, контракт), которые изображаются на диаграмме следующим образом (рисунок 2.19).

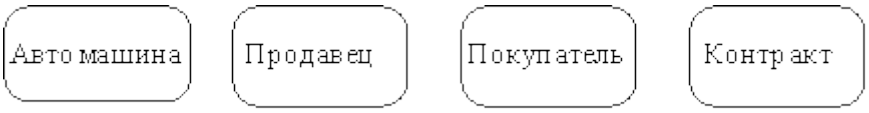

Рис. 2.19. Идентифицированные сущности

Следующим шагом моделирования является идентификация связей.

Связь (Relationship) - поименованная ассоциация между двумя сущностями, значимая для рассматриваемой предметной области. Связь - это ассоциация между сущностями, при которой, как правило, каждый экземпляр одной сущности, называемой родительской сущностью, ассоциирован с произвольным (в том числе нулевым) количеством экземпляров второй сущности, называемой сущностью-потомком, а каждый экземпляр сущности-потомка ассоциирован в точности с одним экземпляром сущности-родителя. Таким образом, экземпляр сущности-потомка может существовать только при существовании сущности родителя.

Связи может даваться имя, выражаемое грамматическим оборотом глагола и помещаемое возле линии связи. Имя каждой связи между двумя данными сущностями должно быть уникальным, но имена связей в модели не обязаны быть уникальными. Имя связи всегда формируется с точки зрения родителя, так что предложение может быть образовано соединением имени сущности-родителя, имени связи, выражения степени и имени сущности-потомка.

Например, связь продавца с контрактом может быть выражена следующим образом:

- продавец может получить вознаграждение за 1 или более контрактов;
- контракт должен быть инициирован ровно одним продавцом.

Степень связи и обязательность графически изображаются следующим образом (рисунок 2.20).

Рис. 2.20. Основные виды связи

Таким образом, 2 предложения, описывающие связь продавца с контрактом, графически будут выражены следующим образом (рисунок 2.21).

Рис. 2.21. Связь Продавец-Контракт

Описав также связи остальных сущностей, получим следующую схему (рисунок 2.22).

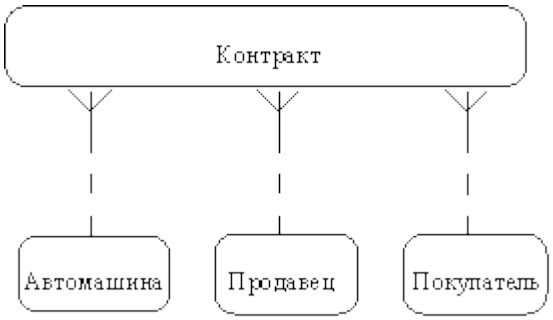

Рис. 2.22. Общая схема связей с контрактом

Последним шагом моделирования является идентификация атрибутов.

Атрибут - любая характеристика сущности, значимая для рассматриваемой предметной области и предназначенная для квалификации, идентификации, классификации, количественной характеристики или выражения состояния сущности. Атрибут представляет тип характеристик или свойств, ассоциированных со множеством реальных или абстрактных объектов (людей, мест, событий, состояний, идей, пар предметов и т.д.). Экземпляр атрибута - это определенная характеристика отдельного элемента множества. Экземпляр атрибута определяется типом характеристики и ее значением, называемым значением атрибута. В ER-модели атрибуты ассоциируются с конкретными сущностями. Таким образом, экземпляр сущности должен обладать единственным определенным значением для ассоциированного атрибута.

Атрибут может быть либо обязательным, либо необязательным (рисунок 2.23). Обязательность означает, что атрибут не может принимать неопределенных значений (null values). Атрибут может быть либо описательным (т.е. обычным дескриптором сущности), либо входить в состав уникального идентификатора (первичного ключа).

Уникальный идентификатор - это атрибут или совокупность атрибутов и/или связей, предназначенная для уникальной идентификации каждого экземпляра данного типа сущности. В случае полной идентификации каждый

экземпляр данного типа сущности полностью идентифицируется своими собственными ключевыми атрибутами, в противном случае в его идентификации участвуют также атрибуты другой сущности-родителя (рисунок 2.24).

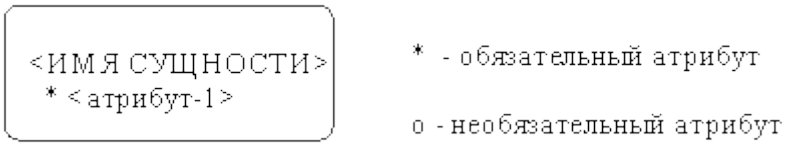

Рис. 2.23. Идентификация сущности

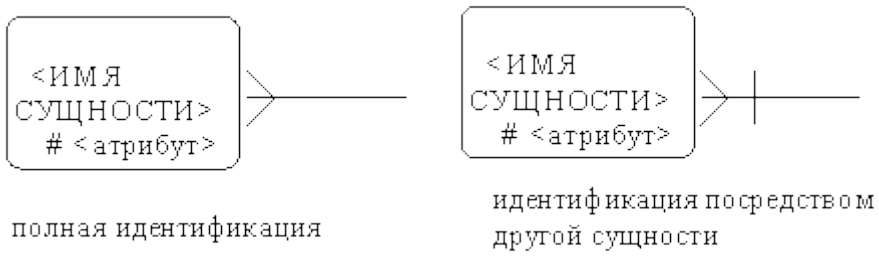

Рис. 2.24. Виды идентификации сущностей

Каждый атрибут идентифицируется уникальным именем, выражаемым грамматическим оборотом существительного, описывающим представляемую атрибутом характеристику. Атрибуты изображаются в виде списка имен внутри блока ассоциированной сущности, причем каждый атрибут занимает отдельную строку. Атрибуты, определяющие первичный ключ, размещаются наверху списка и выделяются знаком "#".

Каждая сущность должна обладать хотя бы одним возможным ключом. Возможный ключ сущности - это один или несколько атрибутов, чьи значения однозначно определяют каждый экземпляр сущности. При существовании нескольких возможных ключей один из них обозначается в качестве первичного ключа, а остальные - как альтернативные ключи.

С учетом имеющейся информации дополним построенную ранее диаграмму (рисунок 2.25).

Помимо перечисленных основных конструкций модель данных может содержать ряд дополнительных.

Подтипы и супертипы: одна сущность является обобщающим понятием для группы подобных сущностей (рисунок 2.26).

Взаимно исключающие связи: каждый экземпляр сущности участвует только в одной связи из группы взаимно исключающих связей (рисунок 2.27).

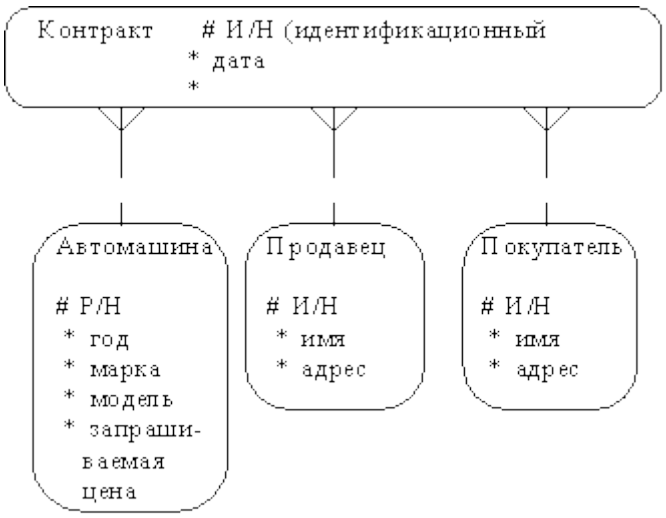

Рис. 2.25. Общая схема связей с контрактом с учетом идентификации сущностей

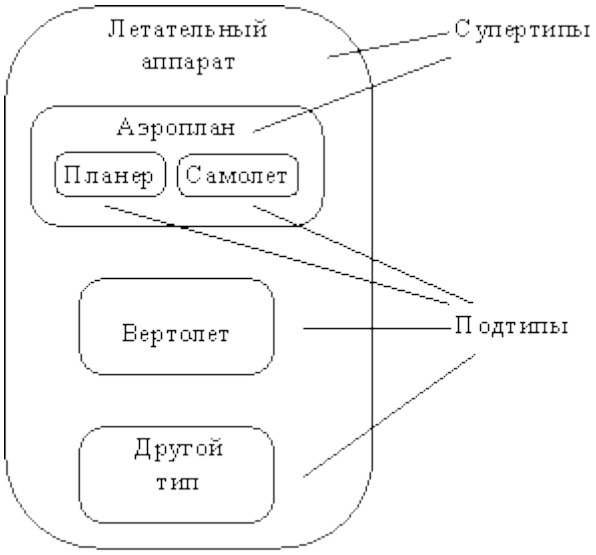

Рис. 2.26. Подтипы и супертипы сущностей

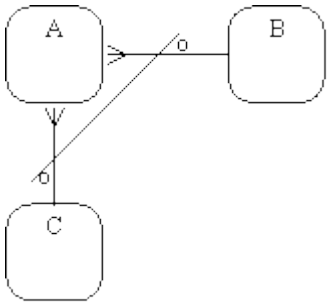

Рис. 2.27. Взаимно исключающие связи

Рекурсивная связь: сущность может быть связана сама с собой (рисунок 2.28).

Неперемещаемые (non-transferrable) связи: экземпляр сущности не может быть перенесен из одного экземпляра связи в другой (рисунок 2.29).

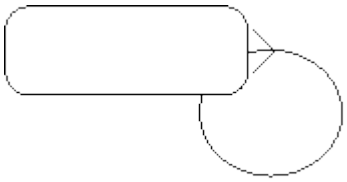

Рис. 2.28. Рекурсивная связь

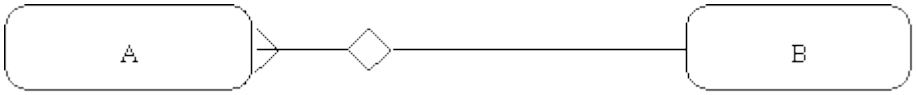

Рис. 2.29. Неперемещаемая связь

Методология IDEF1

Метод IDEF1, разработанный Т.Рэмей (T.Ramey), также основан на подходе П.Чена и позволяет построить модель данных, эквивалентную реляционной модели в третьей нормальной форме. В настоящее время на основе совершенствования методологии IDEF1 создана ее новая версия - методология IDEF1X. IDEF1X разработана с учетом таких требований, как простота изучения и возможность автоматизации. IDEF1X-диаграммы используются рядом распространенных CASE-средств (в частности, ERwin, Design/IDEF).

Сущность в методологии IDEF1X является независимой от идентификаторов или просто независимой, если каждый экземпляр сущности может быть однозначно идентифицирован без определения его отношений с другими сущностями. Сущность называется зависимой от идентификаторов или просто зависимой, если однозначная идентификация экземпляра сущности зависит от его отношения к другой сущности (рисунок 2.30).

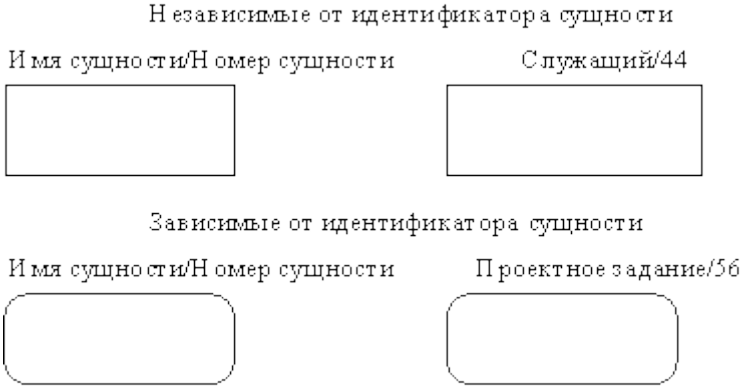

Рис. 2.30. Сущности

Каждой сущности присваивается уникальное имя и номер, разделяемые косой чертой "/" и помещаемые над блоком.

Связь может дополнительно определяться с помощью указания степени или мощности (количества экземпляров сущности-потомка, которое может существовать для каждого экземпляра сущности-родителя). В IDEF1X могут быть выражены следующие мощности связей:

- каждый экземпляр сущности-родителя может иметь ноль, один или более связанных с ним экземпляров сущности-потомка;
- каждый экземпляр сущности-родителя должен иметь не менее одного связанного с ним экземпляра сущности-потомка;
- каждый экземпляр сущности-родителя должен иметь не более одного связанного с ним экземпляра сущности-потомка;
- каждый экземпляр сущности-родителя связан с некоторым фиксированным числом экземпляров сущности-потомка.

Если экземпляр сущности-потомка однозначно определяется своей связью с сущностью-родителем, то связь называется идентифицирующей, в противном случае - неидентифицирующей.

Связь изображается линией, проводимой между сущностью-родителем и сущностью-потомком с точкой на конце линии у сущности-потомка. Мощность связи обозначается как показано на рис. 2.31 (мощность по умолчанию - N).

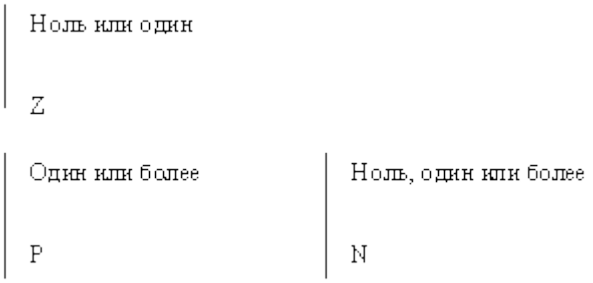

Рис. 2.31. Мощность связи

Идентифицирующая связь между сущностью-родителем и сущностью-потомком изображается сплошной линией (рисунок 2.32). Сущность-потомок в идентифицирующей связи является зависимой от идентификатора сущностью. Сущность-родитель в идентифицирующей связи может быть как независимой, так и зависимой от идентификатора сущностью (это определяется ее связями с другими сущностями).

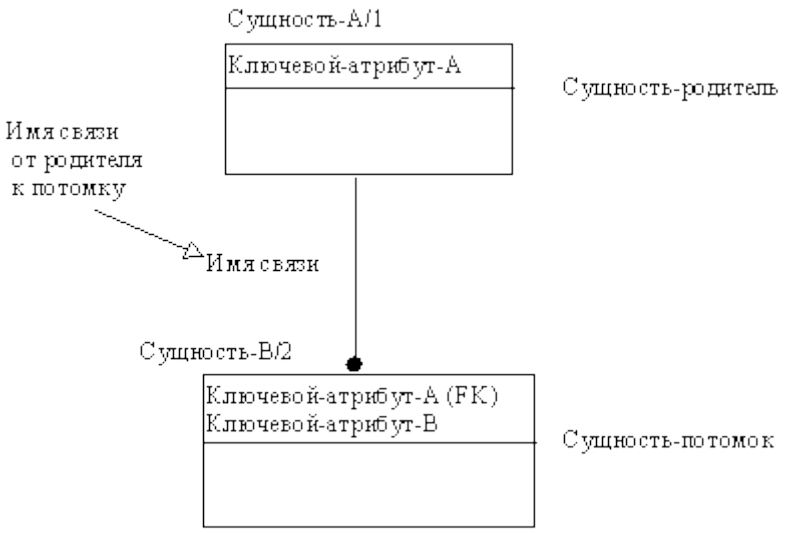

Рис. 2.32. Идентифицирующая связь

Пунктирная линия изображает неидентифицирующую связь (рисунок 2.33). Сущность-потомок в неидентифицирующей связи будет независимой от идентификатора, если она не является также сущностью-потомком в какой-либо идентифицирующей связи.

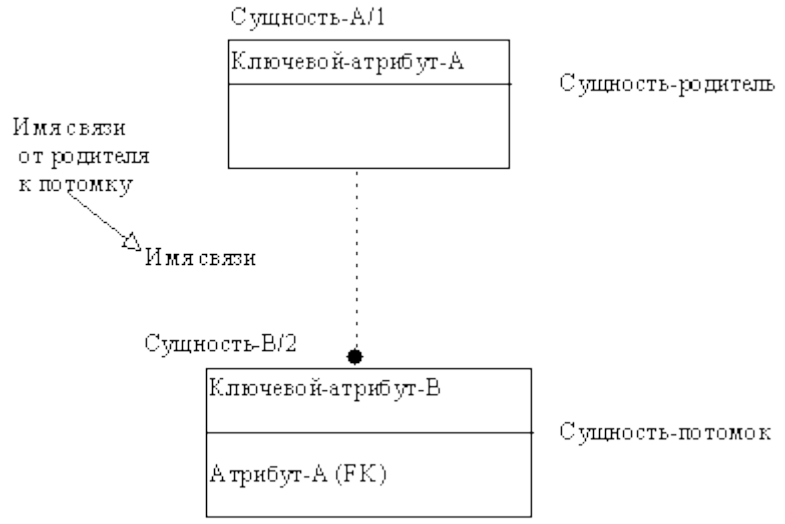

Рис. 2.33. Неидентифицирующая связь

Атрибуты изображаются в виде списка имен внутри блока сущности. Атрибуты, определяющие первичный ключ, размещаются наверху списка и отделяются от других атрибутов горизонтальной чертой (рисунок 2.34).

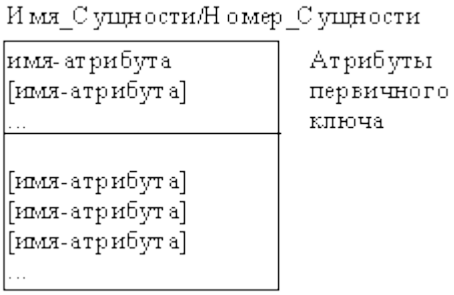

Рис. 2.34. Атрибуты и первичные ключи

Сущности могут иметь также внешние ключи (Foreign Key), которые могут использоваться в качестве части или целого первичного ключа или неключевого атрибута. Внешний ключ изображается с помощью помещения внутрь блока сущности имен атрибутов, после которых следуют буквы FK в скобках (рисунок 2.35).

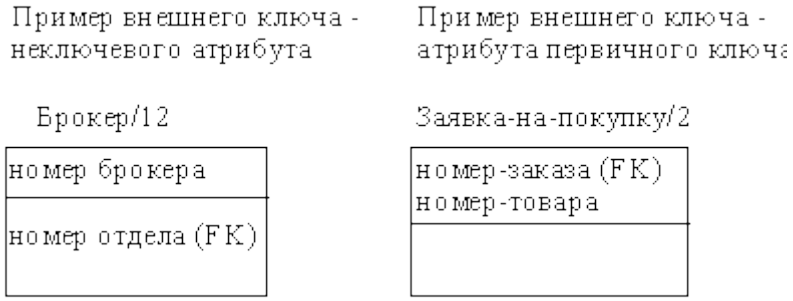

Рис. 2.35. Примеры внешних ключей

Подход, используемый в CASE-средстве Vantage Team Builder

В CASE-средстве Vantage Team Builder (Westmount I-CASE) [14] используется один из вариантов нотации П. Чена. На ER-диаграммах сущность обозначается прямоугольником, содержащим имя сущности (рисунок 2.36), а связь - ромбом, связанным линией с каждой из взаимодействующих сущностей. Числа над линиями означают степень связи.

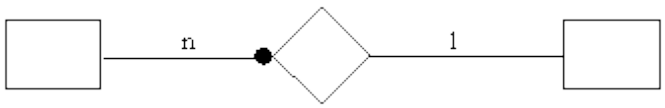

Рис. 2.36. Обозначение сущностей и связей

Связи являются многонаправленными и могут иметь атрибуты (за исключением ключевых). Выделяют два вида связей:

- необязательная связь (optional);

- слабая связь (weak).

В **необязательной связи** (рисунок 2.37) могут участвовать не все экземпляры сущности.

Рис. 2.37. Необязательная связь

В отличие от необязательной связи в **полной (total)** связи участвуют все экземпляры хотя бы одной из сущностей. Это означает, что экземпляры такой связи существуют только при условии существования экземпляров другой сущности. Полная связь может иметь один из 4-х видов: обязательная связь, слабая связь, связь "супертип-подтип" и ассоциативная связь.

Обязательная (mandatory) связь описывает связь между "независимой" и "зависимой" сущностями. Все экземпляры зависимой ("обязательной") сущности могут существовать только при наличии экземпляров независимой ("необязательной") сущности, т.е. экземпляр "обязательной" сущности может существовать только при условии существования определенного экземпляра "необязательной" сущности.

В примере (рисунок 2.38) подразумевается, что каждый автомобиль имеет по крайней мере одного водителя, но не каждый служащий управляет машиной.

Рис. 2.38. Обязательная связь

В **слабой связи** существование одной из сущностей, принадлежащей некоторому множеству ("слабой") зависит от существования определенной сущности, принадлежащей другому множеству ("сильной"), т.е. экземпляр "слабой" сущности может быть идентифицирован только посредством экземпляра "сильной" сущности. Ключ "сильной" сущности является частью составного ключа "слабой" сущности.

Слабая связь всегда является бинарной и подразумевает обязательную связь для "слабой" сущности. Сущность может быть "слабой" в одной связи и "сильной" в другой, но не может быть "слабой" более, чем в одной связи. Слабая связь может не иметь атрибутов.

Пример на рисунке 2.39: ключ (номер) строки в документе может не быть уникальным и должен быть дополнен ключом документа.

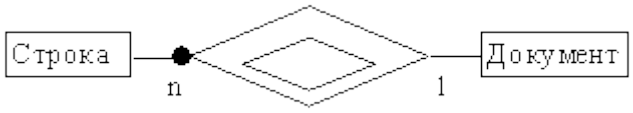

Рис. 2.39. Слабая связь

Связь "супертип-подтип" изображена на рисунке 2.40. Общие характеристики (атрибуты) типа определяются в сущности-супертипе, сущность-подтип наследует все характеристики супертипа. Экземпляр подтипа существует только при условии существования определенного экземпляра супертипа. Подтип не может иметь ключа (он импортирует ключ из супертипа). Сущность, являющаяся супертипом в одной связи, может быть подтипом в другой связи. Связь супертипа не может иметь атрибутов.

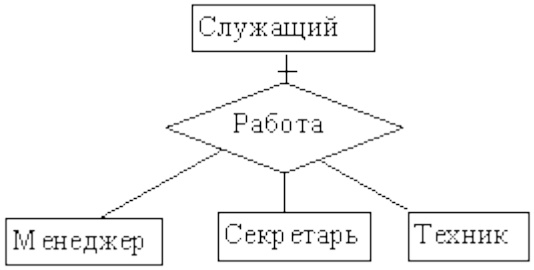

Рис. 2.40. Связь "супертип-подтип"

В **ассоциативной связи** каждый экземпляр связи (ассоциативный объект) может существовать только при условии существования определенных экземпляров каждой из взаимосвязанных сущностей. Ассоциативный объект - объект, являющийся одновременно сущностью и связью. Ассоциативная связь - это связь между несколькими "независимыми" сущностями и одной "зависимой" сущностью. Связь между независимыми сущностями имеет атрибуты, которые определяются в зависимой сущности. Таким образом, зависимая сущность определяется в терминах атрибутов связи между остальными сущностями.

В примере на рисунке 2.41 самолет выполняет посадку на взлетную полосу в заданное время при определенной скорости и направлении ветра. Поскольку эти характеристики применимы только к конкретной посадке, они являются атрибутами посадки, а не самолета или взлетной полосы. Пилот, выполняющий посадку, связан гораздо сильнее с конкретной посадкой, чем с самолетом или взлетной полосой.

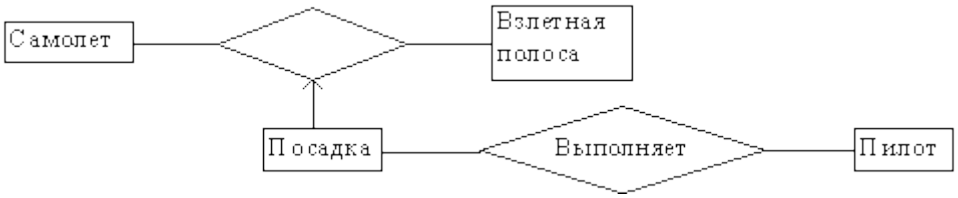

Рис. 2.41. Ассоциативная связь

Первичный ключ каждого типа сущности помечается звездочкой (*).

ER-диаграмма должна подчиняться следующим правилам:

- каждая сущность, каждый атрибут и каждая связь должны иметь имя (связь супертипа или ассоциативная связь может не иметь имени);
- имя сущности должно быть уникально в рамках модели данных;
- имя атрибута должно быть уникально в рамках сущности;
- имя связи должно быть уникально, если для нее генерируется таблица БД;
- каждый атрибут должен иметь определение типа данных;
- сущность в необязательной связи должна иметь ключевой атрибут. То же самое относится к сильной сущности в слабой связи, супертипу в связи "супертип-подтип" и необязательной сущности в обязательной (полной) связи;
- подтип в связи "супертип-подтип" не может иметь ключевой атрибут;
- в ассоциативной или слабой связи может быть только одна ассоциативная (слабая) сущность;
- связь не может быть одновременно обязательной, "супертип-подтип" или ассоциативной.

Пример использования структурного подхода

Описание предметной области

В данном примере используется методология Yourdon [1], реализованная в CASE-средстве Vantage Team Builder [1].

В качестве предметной области используется описание работы видеобиблиотеки, которая получает запросы на фильмы от клиентов и ленты, возвращаемые клиентами. Запросы рассматриваются администрацией видеобиблиотеки с использованием информации о клиентах, фильмах и лентах. При этом проверяется и обновляется список арендованных лент, а также проверяются записи о членстве в библиотеке. Администрация контролирует также возвраты лент, используя информацию о фильмах, лентах и список арендованных лент, который обновляется. Обработка запросов на

фильмы и возвратов лент включает следующие действия: если клиент не является членом библиотеки, он не имеет права на аренду. Если требуемый фильм имеется в наличии, администрация информирует клиента об арендной плате. Однако, если клиент просрочил срок возврата имеющихся у него лент, ему не разрешается брать новые фильмы. Когда лента возвращается, администрация рассчитывает арендную плату плюс пени за несвоевременный возврат.

Видеобиблиотека получает новые ленты от своих поставщиков. Когда новые ленты поступают в библиотеку, необходимая информация о них фиксируется. Информация о членстве в библиотеке содержится отдельно от записей об аренде лент.

Администрация библиотеки регулярно готовит отчеты за определенный период времени о членах библиотеки, поставщиках лент, выдаче определенных лент и лентах, приобретенных библиотекой.

Организация проекта

Весь проект разделяется на 4 фазы: анализ, глобальное проектирование (проектирование архитектуры системы), детальное проектирование и реализация (программирование).

На фазе анализа строится модель среды (Environmental Model). Построение модели среды включает:

- анализ поведения системы (определение назначения ИС, построение начальной контекстной диаграммы потоков данных (DFD) и формирование матрицы списка событий (ELM), построение контекстных диаграмм);
- анализ данных (определение состава потоков данных и построение диаграмм структур данных (DSD), конструирование глобальной модели данных в виде ER-диаграммы).

Назначение ИС определяет соглашение между проектировщиками и заказчиками относительно назначения будущей ИС, общее описание ИС для самих проектировщиков и границы ИС. Назначение фиксируется как текстовый комментарий в "нулевом" процессе контекстной диаграммы.

Например, в данном случае назначение ИС формулируется следующим образом: ведение базы данных о членах библиотеки, фильмах, аренде и поставщиках. При этом руководство библиотеки должно иметь возможность получать различные виды отчетов для выполнения своих задач.

Перед построением контекстной DFD необходимо проанализировать внешние события (внешние объекты), оказывающие влияние на функционирование библиотеки. Эти объекты взаимодействуют с ИС путем информационного обмена с ней.

Из описания предметной области следует, что в процессе работы библиотеки участвуют следующие группы людей: клиенты, поставщики и руководство. Эти группы являются внешними объектами. Они не только взаимодействуют с системой, но также определяют ее границы и изображаются на начальной контекстной DFD как терминаторы (внешние сущности).

Начальная контекстная диаграмма изображена на рисунке 2.42. В отличие от нотации Gane/Sarson внешние сущности обозначаются обычными прямоугольниками, а процессы - окружностями.

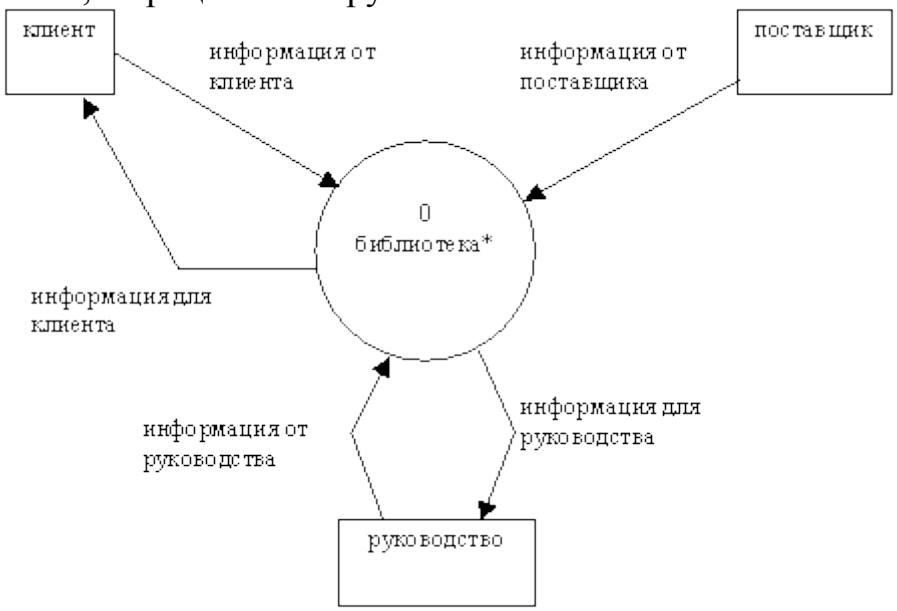

Рис. 2.42. Начальная контекстная диаграмма

Список событий строится в виде матрицы (ELM) и описывает различные действия внешних сущностей и реакцию ИС на них. Эти действия представляют собой внешние события, воздействующие на библиотеку. Различают следующие типы событий (табл. 2.3):

Основные типы управления

Таблица 2.3

Аббревиатура	**Тип**
NC	Нормальное управление
ND	Нормальные данные
NCD	Нормальное управление/данные
TC	Временное управление
TD	Временные данные
TCD	Временное управление/данные

Все действия помечаются как нормальные данные. Эти данные являются событиями, которые ИС воспринимает непосредственно, например, изменение адреса клиента, которое должно быть сразу зарегистрировано. Они появляются в DFD в качестве содержимого потоков данных.

Матрица событий

Таблица 2.4

	Описание	Тип	Реакция
	Клиент желает стать членом библиотеки	ND	Регистрация клиента в качестве члена библиотеки
	Клиент сообщает об изменении адреса	ND	Регистрация измененного адреса клиента
	Клиент запрашивает аренду фильма	ND	Рассмотрение запроса
	Клиент возвращает фильм	ND	Регистрация возврата
	Руководство предоставляет полномочия новому поставщику	ND	Регистрация поставщика
	Поставщик сообщает об изменении адреса	ND	Регистрация измененного адреса поставщика
	Поставщик направляет фильм в библиотеку	ND	Получение нового фильма
	Руководство запрашивает новый отчет	ND	Формирование требуемого отчета для руководства

Для завершения анализа функционального аспекта поведения системы строится полная контекстная диаграмма, включающая диаграмму нулевого уровня. При этом процесс "библиотека" декомпозируется на 4 процесса, отражающие основные виды административной деятельности библиотеки. Существующие "абстрактные" потоки данных между терминаторами и процессами трансформируются в потоки, представляющие обмен данными на более конкретном уровне. Список событий показывает, какие потоки существуют на этом уровне: каждое событие из списка должно формировать некоторый поток (событие формирует входной поток, реакция - выходной поток). Один "абстрактный" поток может быть разделен на более чем один "конкретный" поток.

Основные потоки данных

Таблица 2.5

Потоки на диаграмме верхнего уровня	Потоки на диаграмме нулевого уровня
Информация от клиента	Данные о клиенте, Запрос об аренде
Информация для клиента	Членская карточка, Ответ на запрос об аренде
Информация от руководства	Запрос отчета о новых членах, Новый поставщик, Запрос отчета о поставщиках, Запрос отчета об аренде, Запрос отчета о фильмах

Информация для руководства	Отчет о новых членах, Отчет о поставщиках, Отчет об аренде, Отчет о фильмах
Информация от поставщика	Данные о поставщике, Новые фильмы

На приведенной DFD (рисунок 2.43) накопитель данных "библиотека" является глобальным или абстрактным представлением хранилища данных.

Анализ функционального аспекта поведения системы дает представление об обмене и преобразовании данных в системе. Взаимосвязь между "абстрактными" потоками данных и "конкретными" потоками данных на диаграмме нулевого уровня выражается в диаграммах структур данных (рисунок 2.44).

На фазе анализа строится глобальная модель данных, представляемая в виде диаграммы "сущность-связь" (рисунок 2.45).

Между различными типами диаграмм существуют следующие взаимосвязи:

- ELM-DFD: события - входные потоки, реакции - выходные потоки
- DFD-DSD: потоки данных - структуры данных верхнего уровня
- DFD-ERD: накопители данных - ER-диаграммы
- DSD-ERD: структуры данных нижнего уровня - атрибуты сущностей

На фазе проектирования архитектуры строится предметная модель. Процесс построения предметной модели включает в себя:

- детальное описание функционирования системы;
- дальнейший анализ используемых данных и построение логической модели данных для последующего проектирования базы данных;
- определение структуры пользовательского интерфейса, спецификации форм и порядка их появления;
- уточнение диаграмм потоков данных и списка событий, выделение среди процессов нижнего уровня интерактивных и неинтерактивных, определение для них миниспецификаций.

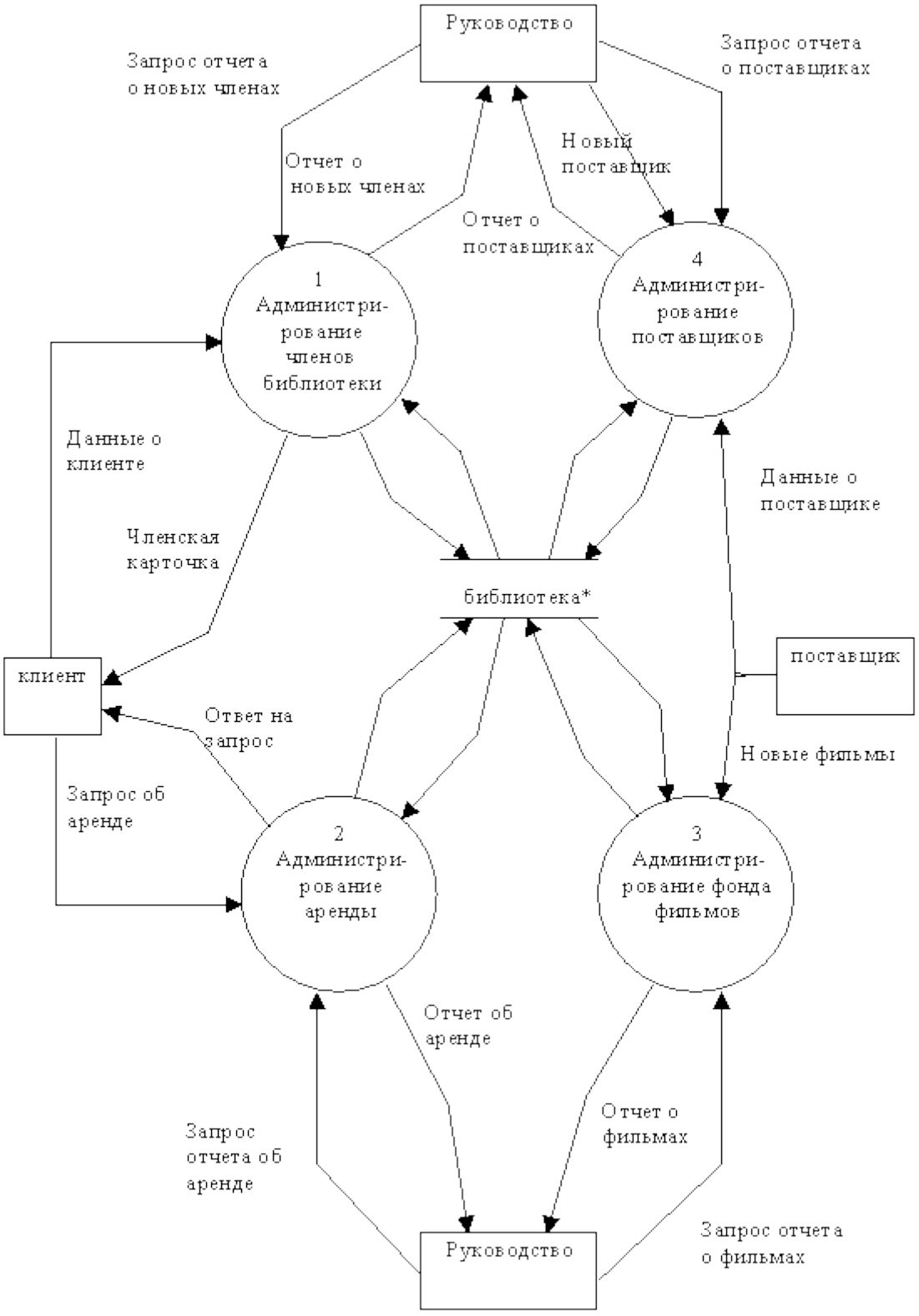

Рис. 2.43. Контекстная диаграмма

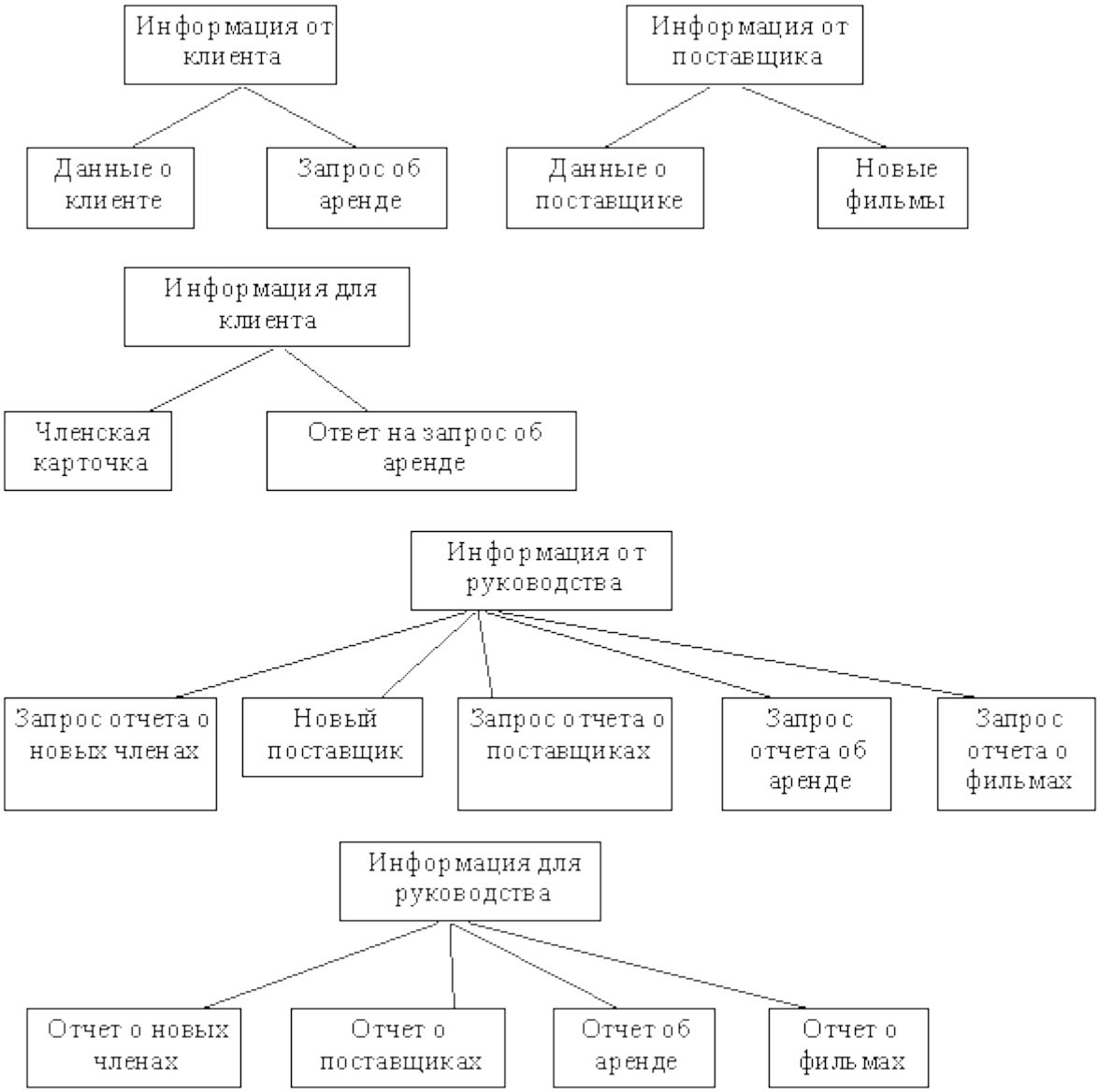

Рис. 2.44. Диаграмма структур данных

Результатами проектирования архитектуры являются:

- модель процессов (диаграммы архитектуры системы (SAD) и миниспецификации на структурированном языке);
- модель данных (ERD и подсхемы ERD);
- модель пользовательского интерфейса (классификация процессов на интерактивные и неинтерактивные функции, диаграмма последовательности форм (FSD - Form Sequence Diagram), показывающая, какие формы появляются в приложении и в каком порядке. На FSD фиксируется набор и структура вызовов экранных форм. Диаграммы FSD образуют иерархию, на вершине которой находится главная форма приложения, реализующего подсистему. На втором уровне находятся формы, реализующие процессы нижнего

уровня функциональной структуры, зафиксированной на диаграммах SAD.

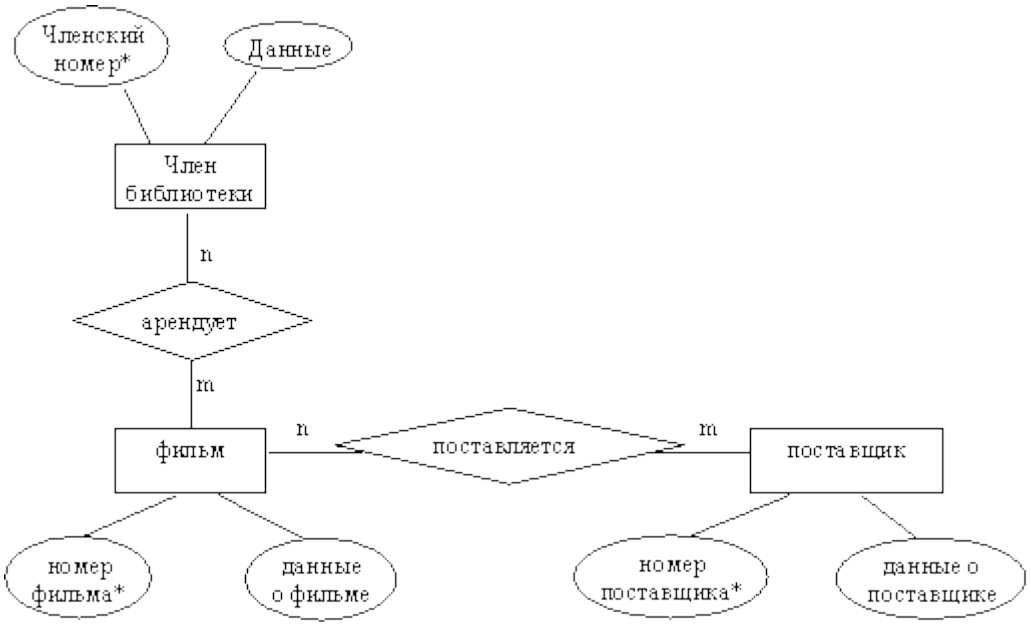

Рис. 2.45. Диаграмма "сущность-связь"

На фазе детального проектирования строится модульная модель. Под модульной моделью понимается реальная модель проектируемой прикладной системы. Процесс ее построения включает в себя:

- уточнение модели базы данных для последующей генерации SQL-предложений;
- уточнение структуры пользовательского интерфейса;
- построение структурных схем, отражающих логику работы пользовательского интерфейса и модель бизнес-логики (Structure Charts Diagram - SCD) и привязка их к формам.

Результатами детального проектирования являются:

- модель процессов (структурные схемы интерактивных и неинтерактивных функций);
- модель данных (определение в ERD всех необходимых параметров для приложений);
- модель пользовательского интерфейса (диаграмма последовательности форм (FSD), показывающая, какие формы появляются в приложении и в каком порядке, взаимосвязь между каждой формой и определенной структурной схемой, взаимосвязь между каждой формой и одной или более сущностями в ERD).

На фазе реализации строится реализационная модель. Процесс ее построения включает в себя:

- генерацию SQL-предложений, определяющих структуру целевой БД (таблицы, индексы, ограничения целостности);
- уточнение структурных схем (SCD) и диаграмм последовательности форм (FSD) с последующей генерацией кода приложений.

На основе анализа потоков данных и взаимодействия процессов с хранилищами данных осуществляется окончательное выделение подсистем (предварительное должно было быть сделано и зафиксировано на этапе формулировки требований в техническом задании). При выделении подсистем необходимо руководствоваться принципом функциональной связанности и принципом минимизации информационной зависимости. Необходимо учитывать, что на основании таких элементов подсистемы как процессы и данные на этапе разработки должно быть создано приложение, способное функционировать самостоятельно. С другой стороны при группировке процессов и данных в подсистемы необходимо учитывать требования к конфигурированию продукта, если они были сформулированы на этапе анализа.

Список литературы

1. Вендров А.М. CASE-технологии. Современные методы и средства проектирования информационных систем. – М.: Финансы и статистика, 2005. – 176 с.
2. Черемных С.В., Ручкин В.С., Семенов И.О. Структурный анализ систем. IDEF-технологии. - М.: Финансы и статисткика, 2001.
3. Похилько, А.Ф. CASE-технология моделирования процессов с использованием средств BPWin и ERWin: учебное пособие / А.Ф. Похилько, И.В. Горбачев. - Ульяновск: УлГТУ, 2008. - 120 с.