Нижегородский государственный архитектурно-строительный университет

Кислицын Дмитрий Игоревич

Курс лекций по дисциплине «Проектирование программного обеспечения»

Часть 2
Структурный подход к проектированию программного обеспечения

Нижний Новгород – 2017

Раздел 5. Описание методологии DFD

Стандарт описания бизнес-процессов DFD - Data Flow Diagram переводится как диаграмма потоков данных и используется для описания процессов верхнего уровня и для описания реально существующих в организации потоков данных.

5.1. Использование и особенности DFD диаграмм

Созданные модели потоков данных организации могут быть использованы при решении таких задач, как:

1) определение существующих хранилищ данных (текстовые документы, файлы, Система управления базой данных — СУБД);
2) определение и анализ данных, необходимых для выполнения каждой функции процесса;
3) подготовка к созданию модели структуры данных организации, так называемая ERD-модель (IDEF1X);
4) выделение основных и вспомогательных бизнес-процессов организации.

Диаграммы потоков данных показывают, как каждый процесс преобразует свои входные данные в выходные, и выявляют отношения между этими процессами. DFD представляет моделируемую систему как сеть связанных работ.

При построении DFD-схемы бизнес-процесса **нужно помнить, что данная схема показывает движение материальных и информационных потоков и ни в коем случае не говорит о временной последовательности работ**, хотя в большинстве случаев временная последовательность работ и совпадает с направлением движения потоков в бизнес-процессе.

5.2. Графический язык моделирования DFD диаграмм

Существуют две нотации DFD:

1) Йордана-Де Марко,
2) Гейна-Сарсона.

Их краткое описание представлено в таблице 5.1.

Требования к оформлению функций:

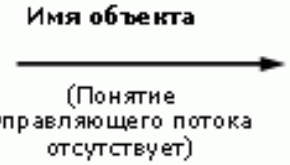
1. Каждая функция должна иметь идентификатор;
2. Названия работы нужно формулировать согласно следующее формуле:

Название работы = Действие + Объект, над которым действие осуществляется

Например, если эта работа связана с действием по продаже продукции, то ее нужно назвать «Продажа продукции».

3. Название работы должно быть по возможности кратким (не более 50 символов) и состоять из 2 - 3 слов. В сложных случаях также рекомендуется для каждого краткого названия работы сделать ее подробное описание, которое поместить в глоссарий.

Таблица 5.1.

Элемент	Описание	Нотация Йордона-Де Марко	Нотация Гейна-Сарсона
Функция	Работа.		
Поток данных	Объект, над которым выполняется работа. Может быть логическим или управляющим. (Управляющие потоки обозначаются пунктирной линией со стрелкой).		
Хранилище данных	Структура для хранения информационных объектов.		
Внешняя сущность	Внешний по отношению к системе объект, обменивающийся с нею потоками.		

Требования к оформлению потока данных:

1. Название потока нужно формулировать согласно следующей формуле:

Название потока = Объект, представляющий поток + Статус объекта

Например, если речь идет о продукции, которую отгрузили клиенту, то поток можно назвать <Продукция, отгруженная> или <Продукция, отгруженная клиенту>. В данном случае <Продукция> это объект, представляющий поток, а <отгруженная клиенту> — статус объекта.

2. Название должно быть по возможности кратким и состоять из 2 - 3 слов.

5.3. Построение DFD-модели

Построение DFD-модели базируется на принципе декомпозиции. DFD-модель включает в себя три документа, которые ссылаются друг на друга: Графические диаграммы, Миниспецификация, Словарь данных.

1. Контекстная диаграмма или иерархия контекстных диаграмм

Первым шагом является построение контекстной диаграммы (рис. 5.1). Диаграмма имеет звездообразную топологию, в центре которой находится так называемый главный процесс, соединенный с приемниками и источниками информации, посредством которых с системой взаимодействуют пользователи и другие внешние системы.

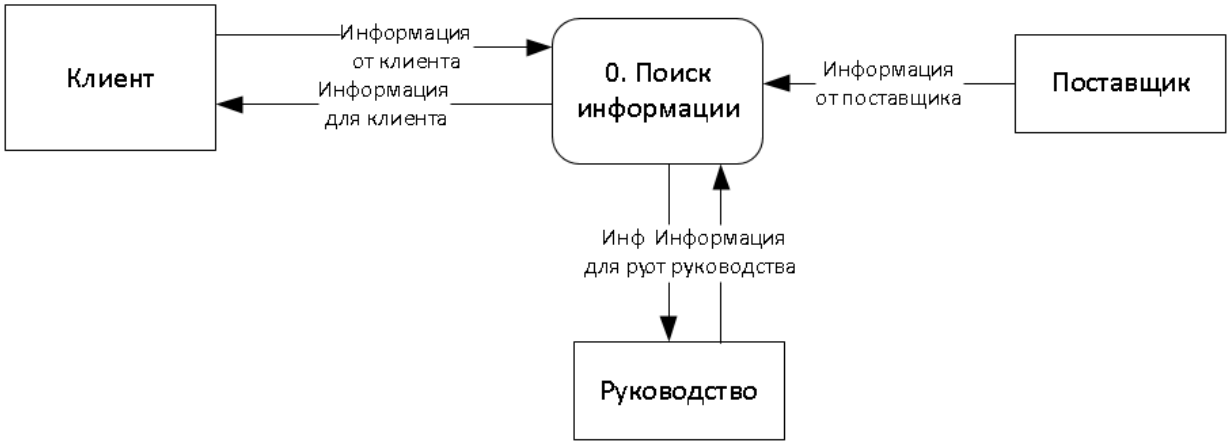


Рис. 5.1.

Однако в некоторых случаях целесообразнее и нагляднее построить несколько контекстных диаграмм с иерархией:

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

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.

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

2. Детализация контекстной диаграммы

Для каждой подсистемы, присутствующей на контекстных диаграммах, выполняется ее детализация при помощи диаграммы DFD. Каждый процесс, в свою очередь, может быть детализирован при помощи отдельной диаграммы или миниспецификации (рис. 5.2).

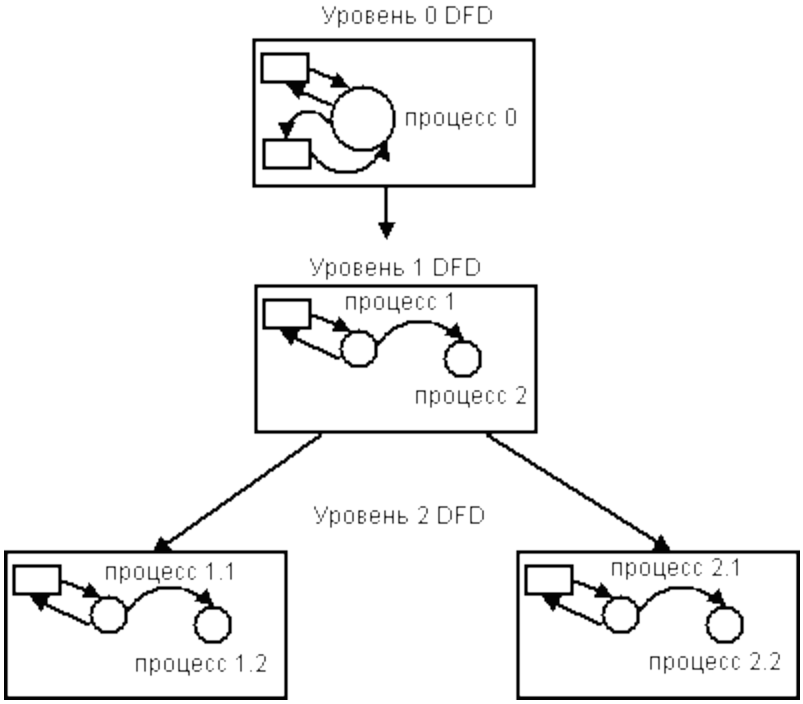


Рис. 5.2.

При детализации должны выполняться следующие правила:

- **правило балансировки** - при детализации процесса дочерняя диаграмма в качестве внешних источников/приемников данных может иметь только те компоненты (подсистемы, процессы, внешние сущности, накопители данных), с которыми имеет информационную связь соответствующий процесс на родительской диаграмме;
- **правило нумерации -** при детализации процессов должна поддерживаться их иерархическая нумерация;
- **правило семи -** для того, чтобы диаграмма легко читалась, количество функций на диаграмме не должно быть больше семи.

Например, процессы, детализирующие процесс с номером 12, получают номера 12.1, 12.2, 12.3 и т.д.

3. Миниспецификация

Миниспецификация — документ, детально описывающий логику процесса. Она содержит номер процесса, списки входных и выходных данных, тело процесса — подробный алгоритм функции, преобразующий входные потоки данных в выходные.

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

- у процесса небольшое количество входных и выходных потоков данных (2-3 потока);
- процесс можно описать в виде последовательного алгоритма;

- процесс выполняет единственную логическую функцию преобразования входной информации в выходную;
- описать логику процесса можно в виде миниспецификации небольшого объема (не более 20-30 строк).

4. Словарь данных

В словаре данных определяется структура и содержание всех потоков данных и накопителей данных, которые присутствуют на диаграммах.

Для каждого потока в словаре хранятся: имя потока, тип, атрибуты (таблица 5.2).

Таблица 5.2.

Тип	**Атрибуты**
1. Простой / групповой (объединяющий несколько потоков). 2. Внутренний/ внешний. 3. Поток данных/ поток управления. 4. Непрерывный (принимающий любые значения в рамках диапазона)/дискретный (принимающий конкретные значения).	1. Имена-синонимы потока. 2. В случае группового потока, все потоки, которые поток объединяет. 3. Единицы измерения потока. 4. Диапазон значения и типичное значение с информацией по обработке экстремальных ситуаций. 5. Список значений и их смысл для дискретного потока. 6. Список номеров диаграмм, в которых поток встречается. 7. Список потоков, в которые поток входит (если в свою очередь входит в другой групповой поток). 8. Комментарии.

5.4. Проверка DFD модели

После построения законченной модели системы ее необходимо проверить на полноту и согласованность.

Модель считается полной, если все ее объекты (подсистемы, процессы, потоки данных) подробно описаны и детализированы.

Модель считается согласованной, если для всех потоков данных и накопителей данных выполняется **правило сохранения информации**: все поступающие куда-либо данные должны быть считаны, а все считываемые данные должны быть записаны.

Ниже представлен пример DFD-диаграммы, описывающей деятельность ресторана на уровне декомпозиции контекстной диаграммы (рис. 5.3).

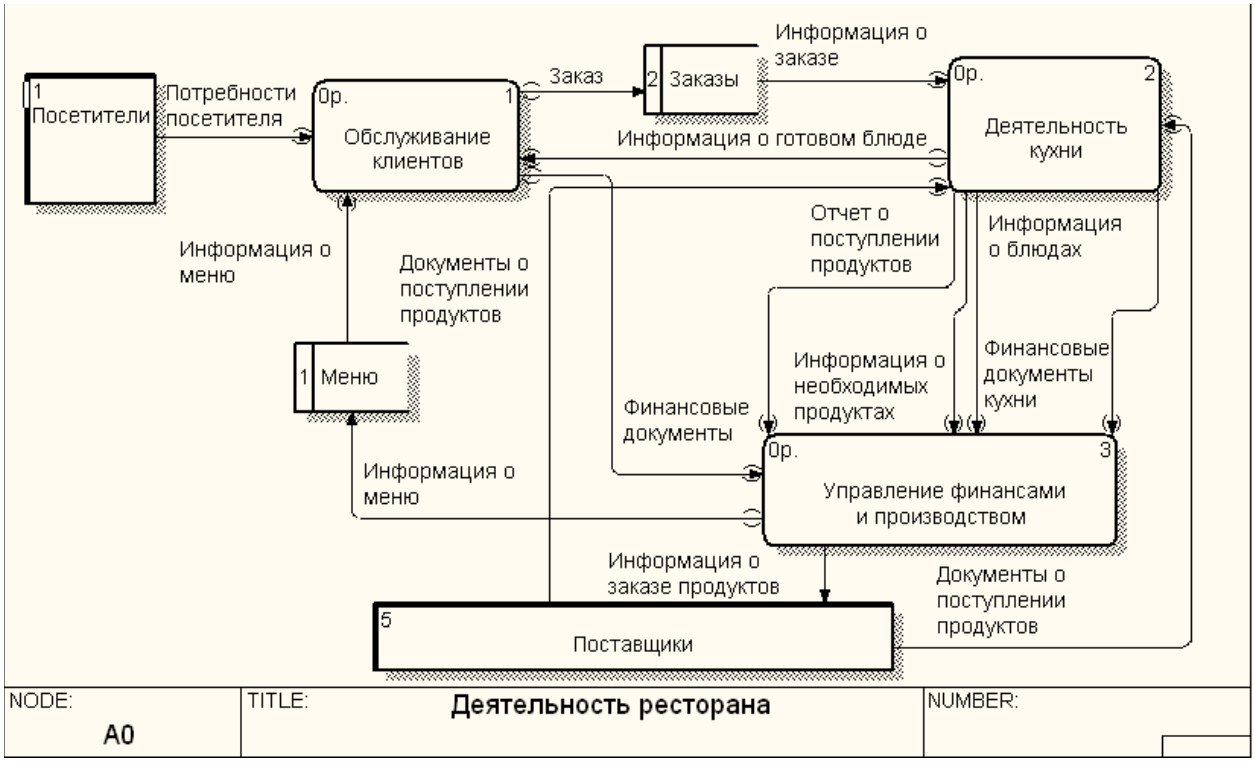


Рис. 5.3.

В деятельности ресторана выделены три функции:

1) обслуживание клиентов,
2) деятельность кухни,
3) управление финансами и производством.

Внешними сущностями по отношению к рассматриваемой (разрабатываемой) системе на данном уровне декомпозиции являются:

1) посетители,
2) поставщики.

В рамках рассматриваемой системы предусмотрены два хранилища данных, которые могут быть реализованы через таблицы базы данных:

1) меню,
2) заказы.

Стрелками указаны объекты, поступающие в функции, а также являющиеся результатом работы функций.

Раздел 6. Описание методологии ERD

Хранилище данных, указанное в методологии DFD, реализуется через базу данных (БД). Первым шагом при создании логической модели БД является построение диаграммы ERD (Entity Relationship Diagram). ERD-диаграммы состоят из трех частей: сущностей, атрибутов и взаимосвязей. Сущностями являются существительные, атрибуты - прилагательными или модификаторами, взаимосвязи - глаголами.

ERD-диаграмма позволяет рассмотреть систему целиком и выяснить требования, необходимые для ее разработки, касающиеся хранения информации.

ERD-диаграммы можно подразделить на отдельные части, соответствующие отдельным задачам, которые решает проектируемая система. Это позволяет рассматривать систему с точки зрения функциональных возможностей, делая процесс проектирования управляемым.

6.1. ERD-диаграммы

Как известно основным компонентом реляционных БД является таблица. Таблица используется для структуризации и хранения информации. В реляционных БД каждая ячейка таблицы содержит одно значение. Кроме того, внутри одной БД существуют взаимосвязи между таблицами, каждая из которых задает совместное пользование данными таблицы.

ERD-диаграмма графически представляет структуру данных проектируемой информационной системы. Сущности отображаются при помощи прямоугольников, содержащих имя. Имена принято выражать существительными в единственном числе, взаимосвязи - при помощи линий, соединяющих отдельные сущности. Взаимосвязь показывает, что данные одной сущности ссылаются или связаны с данными другой (рис. 6.1).

Рис. 6.1.

6.2. Определение сущностей и атрибутов

Сущность - это субъект, место, вещь, событие или понятие, содержащие информацию. Точнее, сущность - это набор (объединение) объектов, называемых экземплярами. В приведенном примере сущность Преподаватель представляет всех возможных преподавателей. Каждый экземпляр сущности обладает набором характеристик. Так, каждый

преподаватель может иметь имя, должность, принадлежность к кафедре т. д. В логической модели все эти характеристики называются атрибутами сущности.

На рисунке 6.2 показана ERD-диаграмма, включающая в себя атрибуты сущностей.

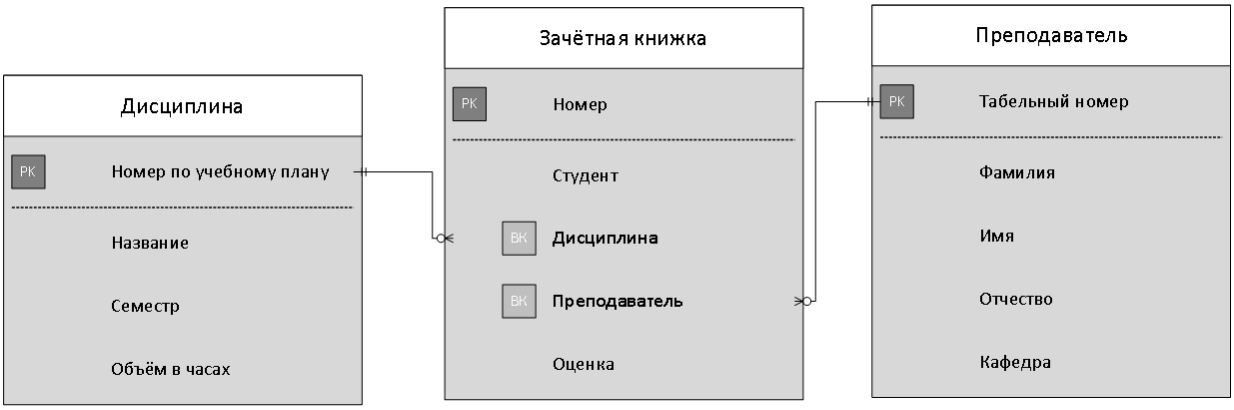


Рис. 6.2.

6.3. Логические взаимосвязи

Логические взаимосвязи представляют собой связи между сущностями. Они определяются глаголами, показывающими, как одна сущность относится к другой.

Некоторые примеры взаимосвязей:

- команда включает много игроков,
- самолет перевозит много пассажиров,
- продавец продает много продуктов.

Во всех этих случаях взаимосвязи отражают взаимодействие между двумя сущностями, называемое «один-ко-многим». Это означает, что один экземпляр первой сущности взаимодействует с несколькими экземплярами другой сущности. Взаимосвязи отображаются линиями, соединяющими две сущности с точкой на одном конце и глаголом, располагаемым над линией.

Кроме взаимосвязи «один-ко-многим» существует еще один тип - это «многие-ко-многим». Этот тип связи описывает ситуацию, при которой экземпляры сущностей могут взаимодействовать с несколькими экземплярами других сущностей. Связь «многие-ко-многим» используют на первоначальных стадиях проектирования. Этот тип взаимосвязи отображается сплошной линией с точками на обоих концах.

Связь «многие-ко-многим» может не учитывать определенные ограничения системы, поэтому может быть заменена на «один-ко-многим» при последующем пересмотре проекта.

6.4. Проверка адекватности логической модели

Если взаимосвязи между сущностями были правильно установлены, то можно составить предложения, их описывающие. Например, по модели, показанной на рисунке 6.3, можно составить следующие предложения:

«Студент получает оценки за экзамены. Много оценок (по разным предметам) выставляются одному студенту».

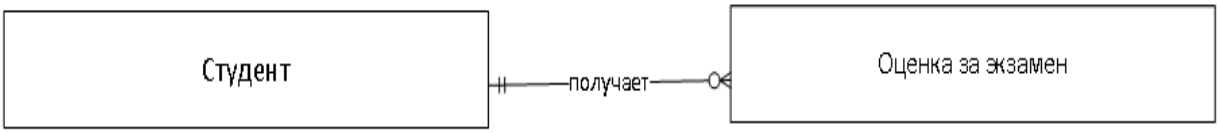


Рис. 6.3.

Составление таких предложений позволяет проверить соответствие полученной модели требованиям и ограничениям создаваемой системы.

6.5. Выбор первичного ключа

При создании сущности необходимо выделить группу атрибутов, которые потенциально могут стать первичным ключом (**primary key**), предназначеным для уникальной идентификации экземпляра сущности, затем произвести отбор атрибутов для включения в состав первичного ключа, следуя следующим рекомендациям (атрибуты первичного ключа обозначаются символами (PK) после своего имени).

Первичный ключ должен быть подобран таким образом, чтобы по значениям атрибутов, в него включенных, можно было точно идентифицировать экземпляр сущности. Никакой из атрибутов первичного ключа не должен иметь нулевое значение. Значения атрибутов первичного ключа не должны меняться. Если значение изменилось, значит, это уже другой экземпляр сущности.

При выборе первичного ключа можно внести в сущность дополнительный атрибут и сделать его ключом. Так, для определения первичного ключа часто используют уникальные номера, которые могут автоматически генерироваться системой при добавлении экземпляра сущности в БД. Применение уникальных номеров облегчает процесс индексации и поиска в БД.

Первичный ключ, выбранный при создании логической модели, может быть неудачным для осуществления эффективного доступа к БД и должен быть изменен при проектировании физической модели.

При проведении связи между двумя сущностями в дочерней сущности автоматически образуются внешние ключи (**foreign key**). Связь образует ссылку на атрибуты первичного ключа в дочерней сущности, и эти атрибуты образуют внешний ключ в дочерней сущности. Атрибуты внешнего ключа обозначаются символами (FK) после своего имени.

Рассмотрим процесс построения логической модели на примере БД студентов системы «Служба занятости в рамках вуза». Первым этапом является определение сущностей и атрибутов. В БД будут храниться записи о студентах, следовательно, сущностью будет студент (таблица 6.1).

Таблица 6.1. Атрибуты сущности «Студент»

Атрибут	Описание
Номер	Уникальный номер для идентификации пользователя
Ф.И.О.	Фамилия, имя и отчество пользователя
Пароль	Пароль для доступа в систему
Возраст	Возраст студента
Пол	Пол студента
Характеристика	Memo-поле с общей характеристикой пользователя
E-mail	Адреса электронной почты
Телефон	Номера телефонов студента (домашний, рабочий)
Опыт работы	Специальности и опыт работы студента по каждой из них
Специальность	Специальность, получаемая студентом при окончании учебного заведения
Специализация	Направление специальности, по которому обучается студент
Иностранный язык	Список иностранных языков и уровень владения ими
Тестирование	Список тестов и отметки о их прохождении
Экспертная оценка	Список предметов с экспертными оценками по каждому из них
Оценки по экзаменам	Список сданных предметов с оценками

В полученном списке существуют атрибуты, которые нельзя определить в виде одного поля БД. Такие атрибуты требуют дополнительных определений и должны рассматриваться как сущности, состоящие, в свою очередь, из атрибутов. К таковым относятся: опыт работы, иностранный язык, тестирование, экспертная оценка, оценки по экзаменам. Определим их атрибуты (таблицы 6.2 – 6.5).

Таблица 6.2. Атрибуты сущности «Опыт работы»

Атрибут	Описание

Специальность	Название специальности, по которой у студента есть опыт работы
Опыт	Опыт работы по данной специальности в годах
Место работы	Наименование предприятия, где приобретался опыт

Таблица 6.3. Атрибуты сущности «Иностранный язык»

Атрибут	**Описание**
Язык	Название иностранного языка, которым владеет студент
Уровень владения	Численная оценка уровня владения иностранным языком

Таблица 6.4. Атрибуты сущности «Тестирование»

Атрибут	**Описание**
Название	Название теста, который прошел студент
Описание	Содержит краткое описание теста
Оценка	Оценка, которую получил студент в результате прохождения теста

Таблица 6.5. Атрибуты сущности «Экспертная оценка»

Атрибут	**Описание**
Дисциплина	Наименование дисциплины, по которой оценивался студент
Ф.И.О. преподавателя	Ф.И.О. преподавателя, который оценивал студента
Оценка	Экспертная оценку преподавателя
Атрибут	Описание
Предмет	Название предмета, экзамен по которому сдавался
Оценка	Полученная оценка

Составим ERD-диаграмму, определяя типы атрибутов и проставляя связи между сущностями (рис. 6.4). Все сущности будут зависимыми от сущности «Студент». Связи будут типа «один-ко-многим».

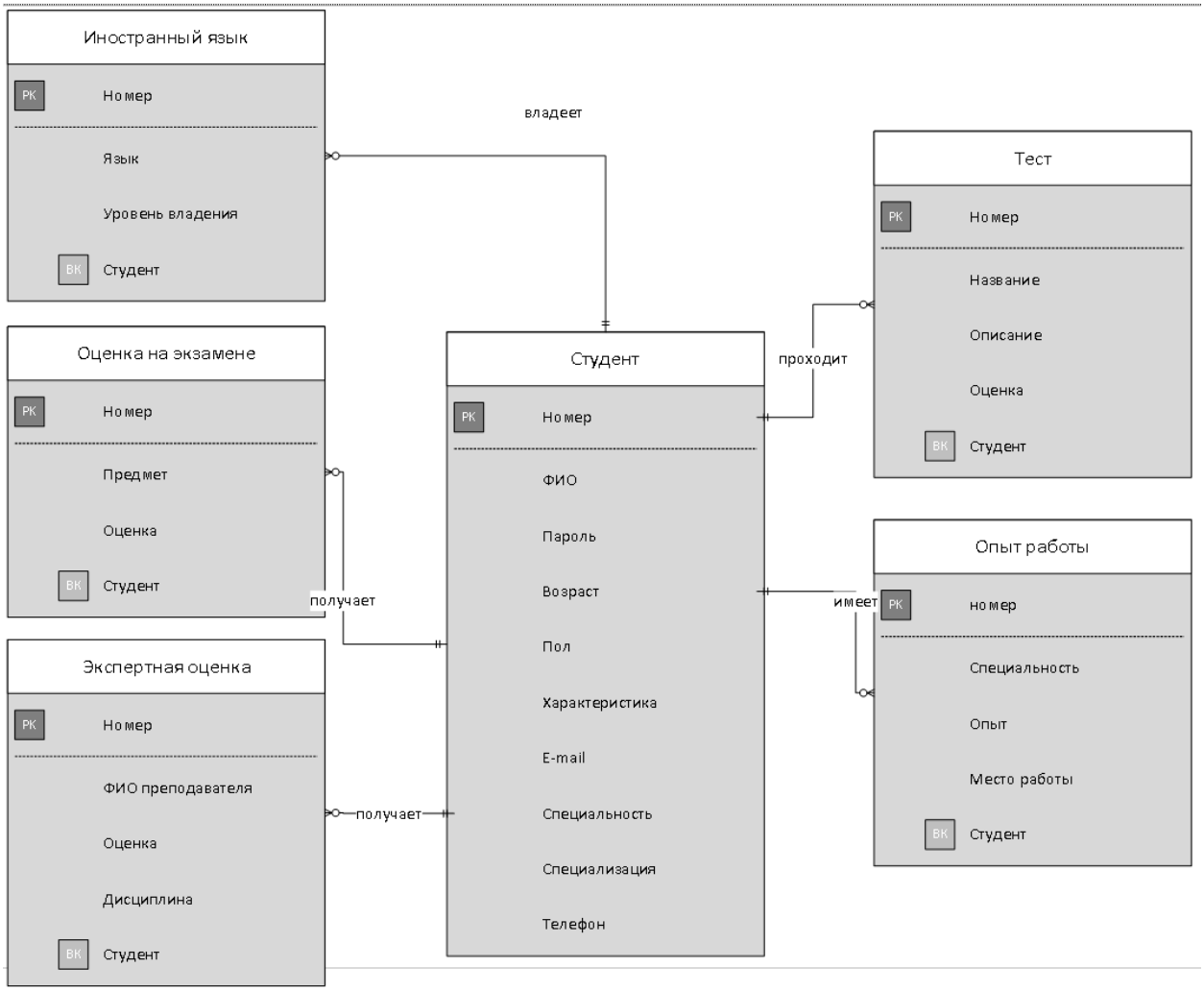


Рис. 6.4.

На полученной диаграмме рядом со связью отражается ее имя, показывающее соотношение между сущностями. При проведении связи между сущностями первичный ключ мигрирует в дочернюю сущность в качестве внешнего ключа (foreign key).

Следующим этапом при построении логической модели является определение типов атрибутов (табл. 6.6).

Таблица 6.6. Типы атрибутов

Атрибут	**Тип**
Номер	Number
Ф.И.О.	String
Пароль	String
Возраст	Number
Атрибут	Тип
Пол	String

Характеристика	String
E-mail	String
Специальность	String
Специализация	String
Опыт	Number
Место работы	String
Язык	String
Уровень владения	Number
Название	String
Описание	String
Оценка	Number
Дисциплина	String
Ф.И.О. преподавателя	String
Предмет	String

Раздел 7. Описание методологии STD

Диаграмма переходов состояний STD (State Transition Diagrams) 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 для управляющего процесса-предка.

С помощью STD-диаграмм 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-диаграмма состоит из следующих объектов:

Состояние – набор характеристик, описывающих условия устойчивости системы. Находясь в определенном состоянии и имея информацию о прошлой истории системы, можно определить очередное состояние в зависимости от текущих входных событий (потоков). Имя состояния должно отражать реальную ситуацию, в которой находится система, например, нагревание, охлаждение и т.п.

Начальное состояние – узел STD-диаграммы, являющийся стартовой точкой для начального системного перехода. STD-диаграмма имеет ровно одно начальное состояние, соответствующее состоянию системы после ее инсталляции (внедрения), но перед началом реальной работы, 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 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»=999, где пароль - входной поток управляющего процесса-предка.

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

управляющего процесса-предка, то имя этого потока должно заключаться в кавычки, например:

«Введенная карта» =true,

где Введенная карта – выходной поток управляющего процесса-предка.

Фактически условие есть некоторое внешнее или внутреннее событие, которое система способна обнаружить и на которое она должна отреагировать определенным образом, изменяя свое состояние. При изменении состояния система обычно выполняет одно или более действий (например, для информационной системы: производит вывод, выдает сообщение на терминал, выполняет вычисления). Таким образом, действие представляет собой отклик, посылаемый во внешнее окружение, или вычисление, результаты которого запоминаются в системе (обычно в хранилищах данных, представленных на DFD-диаграмме, структура и содержание которых представлены на ERD-диаграмме), для того, чтобы обеспечить реакцию на некоторые из планируемых в будущем событий.

На STD-диаграмме состояния представляются узлами, а переходы – дугами (рис. 7.1). Условия (по-другому называемые стимулирующими событиями) идентифицируются именем перехода и возбуждают его выполнение. 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.

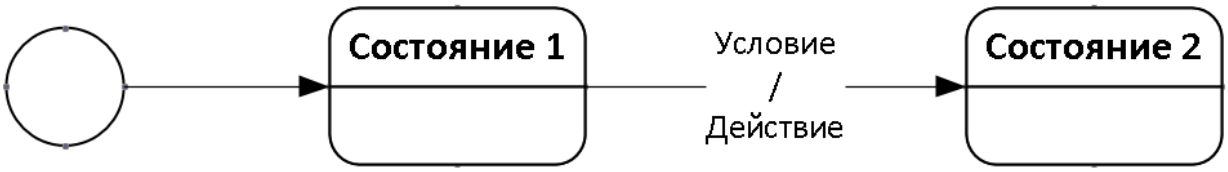


Рис. 7.1.

При построении STD-диаграммы рекомендуется следовать перечисленным ниже правилам:

- строить STD-диаграмму на как можно более высоком уровне детализации DFD-диаграммы;
- строить как можно более простые STD-диаграммы;
- по возможности детализировать STD-диаграммы;
- использовать те же принципы именований состояний, событий и действий, что и при именовании процессов и потоков.

Применяются два способа построения STD-диаграмм. Первый способ заключается в идентификации всех возможных состояний и дальнейшем исследовании всех не бессмысленных связей (переходов) между ними. По второму способу сначала строится начальное состояние, затем следующие за ним и т.д. Результат (в обоих случаях) – предварительная STD-диаграмма, для которой затем осуществляется контроль состоятельности, заключающийся в ответе на следующие вопросы.

1. Все ли состояния определены и имеют уникальное имя?

2. Все ли состояния достижимы?
3. Все ли состояния имеют выход?
4. Реагирует ли система (для каждого состояния) соответствующим образом на все возможные условия (особенно на ненормальные)?
5. Все ли входные (выходные) потоки управляющего процесса отражены в условиях (действиях) на STD-диаграмме?

В качестве примера рассмотрим диаграмму переходов состояний для системы управления лифтом (рис. 7.2). Для этого используем DFD-диаграммы этой системы.

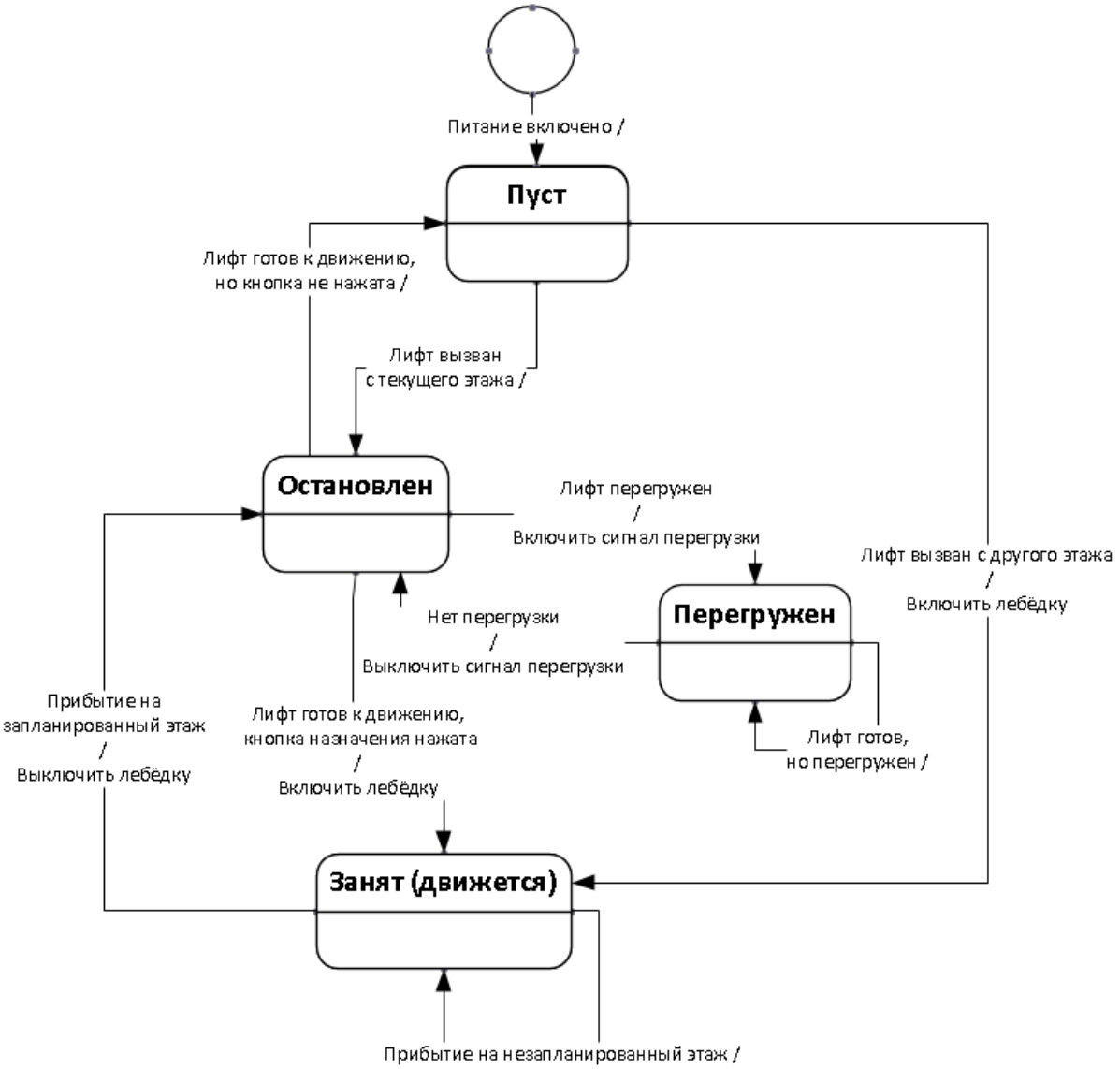


Рис. 7.2.

В ситуации, когда число состояний и/или переходов велико, для проектирования STD-диаграммы могут использоваться таблицы или матрицы переходов состояний (табл. 7.1). Обе эти нотации позволяют зафиксировать ту же самую информацию, что и диаграммы переходов состояний, но в другом формате.

Матрица переходов состояний содержит по вертикали перечень состояний системы, из которых осуществляется переход, а по горизонтали – состояния, в которые осуществляется переход. При этом каждый элемент матрицы содержит соответствующие условия и действия, обеспечивающие переход из «вертикального» состояния в «горизонтальное». В качестве примера данного варианта матрицы переходов состояний приведена таблица, соответствующая диаграмме переходов состояний.

Таблица 7.1. Матрица переходов состояний

	Занят (движется)	**Остановлен**	**Пуст (стоит)**	**Перегружен (стоит)**
Занят (движется)	прибытие на незапланированный этаж /	прибытие на запланированный этаж / выключить лебёдку		
Остановлен	лифт готов к движению, кнопка назначения нажата / включить лебёдку		лифт готов к движению, но кнопка не нажата	лифт перегружен / включить сигнал перегрузки
Пуст (стоит)	лифт вызван с другого этажа / включить лебёдку	лифт вызван с текущего этажа /		
Перегружен (стоит)		нет перегрузки / выключить сигнал перегрузки		лифт готов, но перегружен

Раздел 8. Описание методологии SADT

Методология SADT (Structured Analisys and Design Technique - технология структурного анализа и проектирования) разработана Дугласом Т. Россом в 1969 - 1973 годах. Технология изначально создавалась для проектирования систем более общего назначения по сравнению с другими структурными методами, выросшими из проектирования программного обеспечения. SADT - одна из самых известных и широко используемых методик проектирования. Новое название методики, принятое в качестве стандарта - IDEF0 (Icam DEFinition) - часть программы ICAM (Integrated Computer-Aided Manufacturing - интегрированная компьютеризация производства), проводимой по инициативе ВВС США.

IDEF0 используется для создания функциональной модели**,** отображающей структуру и функции системы, а также потоки информации и материальных объектов, преобразуемые этими функциями.

Методология IDEF0 незначительно отличается от классической схемы описания бизнес-процессов DFD. Основным отличием является классификация входов работы.

8.1. IDEF0 — модель

Модель включает следующие документы, которые ссылаются друг на друга:

- **Графические диаграммы** - главный компонент IDEF0-модели, который графически, с помощью блоков и стрелок и их соединений, отображает информацию о моделируемой системе. Блоки представляют основные функции. Эти функции могут быть разбиты (декомпозированы) на составные части и представлены в виде более подробных диаграмм. Процесс декомпозиции продолжается до тех пор, пока объект не будет описан на уровне детализации, необходимом для достижения целей конкретного проекта.
- **Текст.**
- **Глоссарий** - Для каждого элемента диаграммы создается и поддерживается набор определений, ключевых слов, пояснений, характеризующих объект, который представляет данный элемент. Этот набор называется глоссарием и является описанием сущности данного элемента. Глоссарий гармонично дополняет наглядный графический язык, снабжая диаграммы необходимой дополнительной информацией (например, для управляющей интерфейсной дуги «распоряжение об оплате» глоссарий может содержать перечень полей соответствующего дуге документа, необходимый набор виз и т.д.).

Основу графического языка IDEF0, синтаксис и семантика которого определены с абсолютной строгостью, составляют блоки и соединяющие их

стрелки, которые формируют иерархию детализируемых диаграмм (таблица 8.1).

Таблица 8.1.

Элемент	Графическое отображение и значение	Требования к оформлению
Функциональный блок	Изображается в виде прямоугольника. Представляют функции, определяемые как деятельность, процесс, операция, действие или преобразование.	1. Должен иметь уникальный идентификационный номер в правом нижнем углу. 2. Название должно быть в отглагольном наклонении.
Интерфейсная дуга (стрелка, дуга)	Изображается в виде однонаправленной стрелки. Представляют данные или материальные объекты, связанные с функциями.	1.Должна иметь уникальное наименование. 2.Наименование должно быть оборотом существительного. 3.Началом и концом дуги могут быть только функциональные блоки. 4.Источником может быть только выходная сторона блока, а приемником любая из трех оставшихся.

Стандарт предлагает следующую типизацию входов работ (рис. 8.1.):

- **Вход**. Входит в работу слева и показывает информационные и материальные потоки, которые преобразуются в бизнес процессе.
- **Управление.** Входит в работу сверху и показывает материальные и информационные потоки, которые не преобразуются в процессе, но нужны для его выполнения.
- **Механизм.** Входит в работу снизу и показывает людей, технические средства, информационные системы и т.п., при помощи которых бизнес процесс реализуется.
- **Результаты.** Выходят из блока справа.

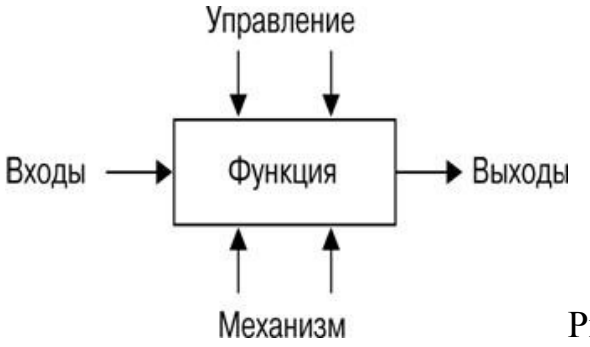


Рис. 8.1.

8.2. Принцип декомпозиции при построении модели бизнес процессов

1. Контекстная диаграмма: цель и точка зрения.

Моделирования бизнес процесса начинается с контекстной диаграммы. Эта диаграмма называется А–0 (А минус ноль). На ней система представляется в виде одного блока и дуг, изображающих окружение системы (рис. 8.2). С помощью диаграммы можно увидеть взаимодействие моделируемой системы с внешней средой, все ее входы и выходы. Диаграмма А–0 устанавливает область моделирования и границы.

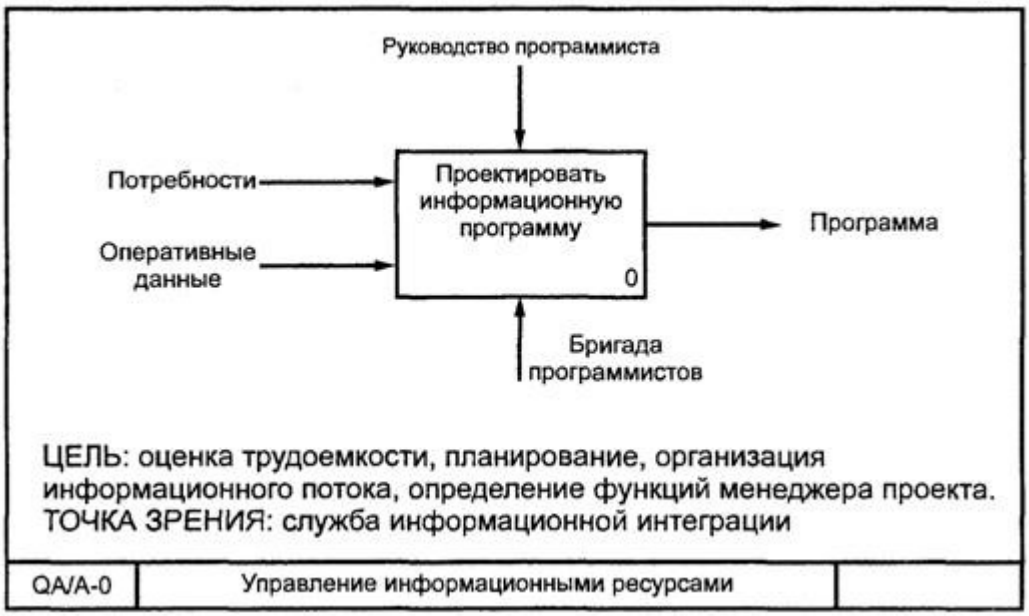


Рис. 8.2.

В пояснительном тексте к контекстной диаграмме должна быть указана **цель** построения диаграммы и зафиксирована **точка зрения**. Точка зрения определяет уровень детализации, направление развития модели и позволяет разгрузить модель. Так при моделировании можно отказаться от детализации и исследования отдельных элементов, не являющихся необходимыми, исходя из выбранной точки зрения на систему.

2. Детализация.

Затем блок, который отображает всю систему, детализируется на другой диаграмме (рис.8.3).

Далее каждая функция диаграммы может быть детализирована на дочерней. Каждая функция моделируется отдельным блоком. Каждый родительский блок подробно описывается дочерней диаграммой на более нижнем уровне. Так происходит до тех пор, пока не будет получена структура, позволяющая ответить на вопросы, сформулированные в цели моделирования.

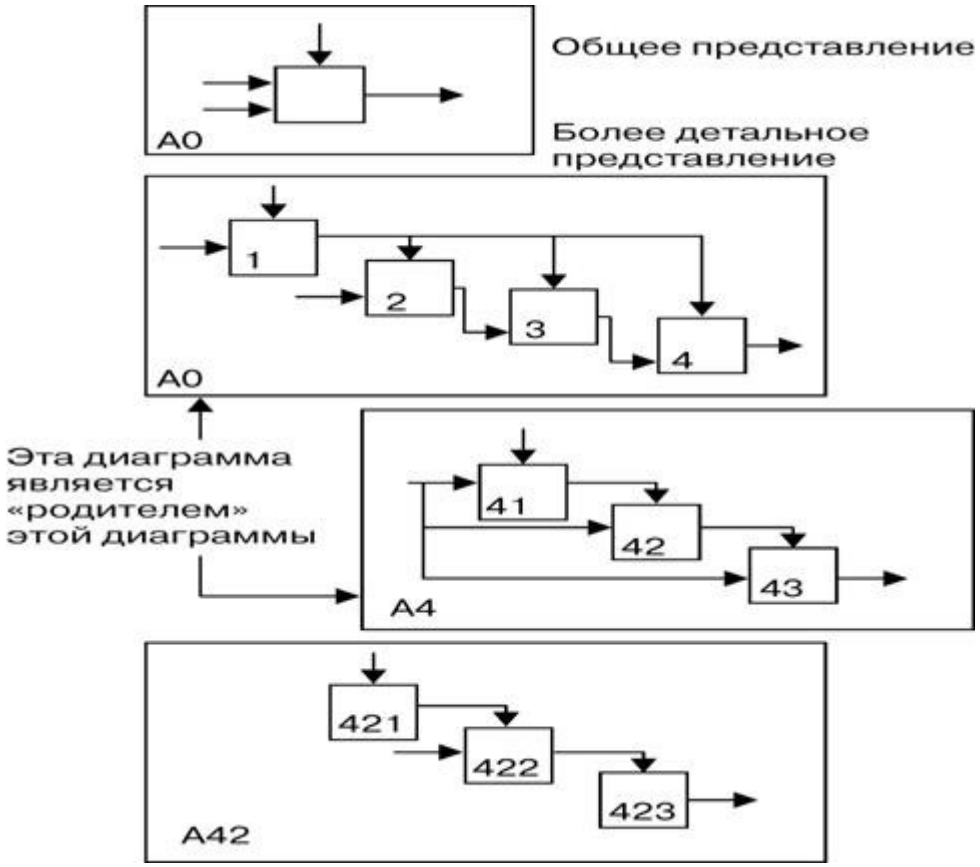


Рис. 8.3.

Для достижения структурной целостности модели, используются следующие правила:

- Все интерфейсные дуги, входящие в данный блок, или исходящие из него фиксируются на дочерней диаграмме.
- При нумерации блоков, цифра в правом нижнем углу прямоугольника указывает на уникальный порядковый номер самого блока на диаграмме, а обозначение под правым углом указывает на номер дочерней для этого блока диаграммы.

Часто бывают случаи, когда отдельные стрелки не имеет смысла продолжать рассматривать в дочерних диаграммах ниже какого-то определенного уровня в иерархии, или наоборот - отдельные блоки не имеют практического смысла выше какого-то уровня. С другой стороны, иногда возникает необходимость избавиться от отдельных «концептуальных» стрелок и не детализировать их глубже некоторого уровня.

Для решения подобных задач в стандарте IDEF0 предусмотрено понятие **туннелирования**. Обозначение «туннеля» в виде двух круглых скобок вокруг начала стрелки обозначает, что эта стрелка не была унаследована от функционального родительского блока и появилась (из «туннеля») только на этой диаграмме. В свою очередь, такое же обозначение вокруг конца стрелки в непосредственной близи от блока – приёмника означает тот факт, что в дочерней по отношению к этому блоку диаграмме эта стрелка отображаться и рассматриваться не будет.

8.3. Принцип ограничения сложности

Для того, чтобы ограничить перегруженность моделей и сделать их удобными для восприятия, в стандарте приняты соответствующие ограничения сложности:

- ограничение количества функциональных блоков на диаграмме тремя-шестью. Верхний предел (шесть) заставляет разработчика использовать иерархии при описании сложных предметов, а нижний предел (три) гарантирует, что на соответствующей диаграмме достаточно деталей, чтобы оправдать ее создание;
- ограничение количества подходящих к одному функциональному блоку (выходящих из одного функционального блока) интерфейсных дуг четырьмя.

Разумеется, строго следовать этим ограничениям вовсе необязательно, однако, как показывает опыт, они являются весьма практичными в реальной работе.

Для анализа диаграммы с точки зрения ее перегруженности и сложности для восприятия, используется количественный анализ. Для анализа используются следующие показатели:

1) количество блоков на диаграмме - *N;*
2) уровень декомпозиции диаграммы - *L*;
3) сбалансированность диаграммы - *B;*
4) число стрелок, соединяющихся с блоком – *A,*
5) число совпадений имен блоков со словами из словаря - *C.*

Необходимо стремиться к тому, чтобы количество блоков на диаграммах нижних уровней было бы ниже количества блоков на родительских диаграммах.

Так же диаграммы должны быть сбалансированы. Это означает, что в рамках одной диаграммы не должно происходить ситуации, когда входящих стрелок и стрелок управления значительно больше, чем выходящих.

Следует отметить, что данная рекомендация может не выполняться в моделях, описывающих производственные процессы. Например, при описании процедуры сборки в блок может входить множество стрелок, описывающих компоненты изделия, а выходить одна стрелка — готовое изделие. Введем коэффициент сбалансированности диаграммы:

$$K_b = \left| \frac{\sum_{i=1}^{N} A_i}{N} - max_{i=1}^{N}(A_i) \right|.$$

Необходимо стремиться, чтобы *Kb* был минимален для диаграммы и убывал с увеличение уровня декомпозиции.

Помимо анализа графических элементов диаграммы необходимо рассматривать наименования блоков. Для оценки имен составляется словарь элементарных (тривиальных) функций моделируемой системы. Фактически в данный словарь должны попасть функции нижнего, уровня декомпозиции диаграмм.

Например, для модели БД элементарными могут являться функции «найти запись», «добавить запись в БД», в то время как функция «регистрация пользователя» требует дальнейшего описания.

После формирования словаря и составления пакета диаграмм системы необходимо рассмотреть нижний уровень модели. Если на нем обнаружатся совпадения названий блоков диаграмм и слов из словаря, то это говорит, что достаточный уровень декомпозиции достигнут.

Коэффициент, количественно отражающий данный критерий, можно записать как: ***L*C***.

Чем ниже уровень модели (больше L), тем ценнее совпадения.

На рис. 8.4 представлен пример диаграммы, описывающий процесс преобразования сырья в готовые изделия, который выполняется персоналом предприятия на основании задания и чертежей.

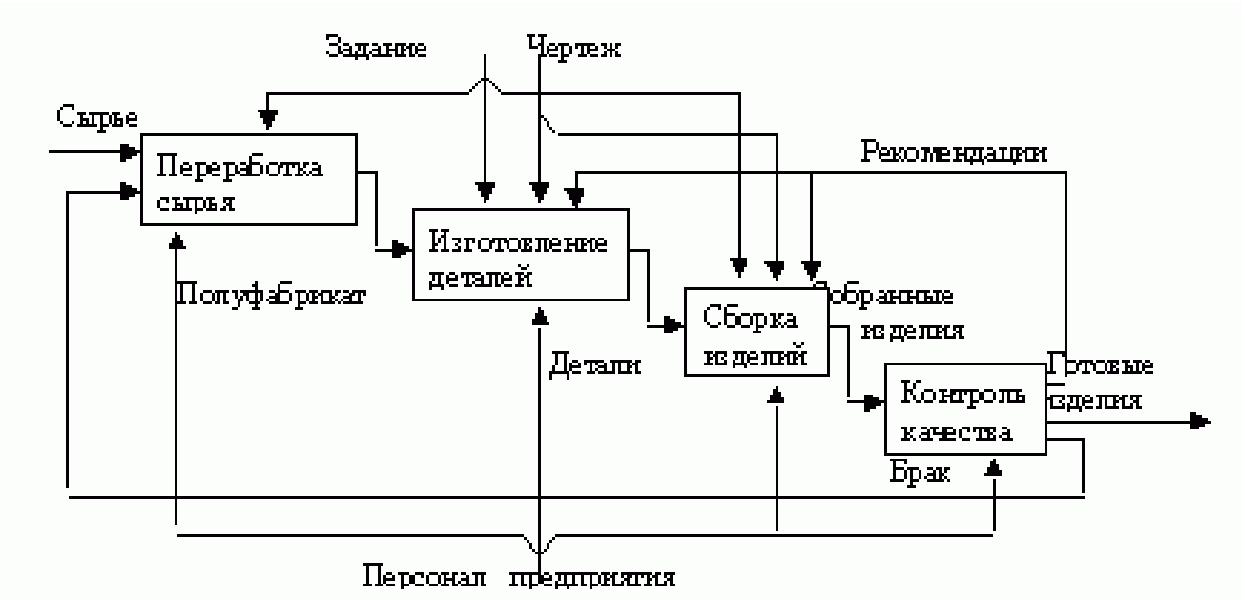


Рис. 8.4.