Инструментарий ARIS

Методы

Версия 4.0

Апрель 2000г.

IDS SCHEER

Содержание

1. Введение

Усиление тенденции стандартизации и значительное падение цен на аппаратное обеспечение обусловили существенное обеспечение подходов к разработке информационных систем.

В прошлом промышленные разработки ограничивались в основном вопросами интегрирования систем и их оптимизации. Однако в последние годы центральным аспектом разработок все чаще становится создание решений по специальным требованиям заказчиков. Распределенные информационные системы, которые могут объединяться в интегрированные информационные сети, приобретают все большую популярность благодаря их доступности и сыкономить время и деньги при создании информационной инфраструктуры для поддержки бизнес-процессов.

Жесткие организационные структуры, построенные по функциональному признаку, имеют центральную ориентацию и часто зависят от ограниченных возможностей централизованной архитектуры «хост-терминал». Это приводит к потере гибкости в деятельности компании. Для отдельных специалистов были очевидны возможности, открывающиеся в результате начавшейся децентрализации компьютеров и компьютерных служб и появления в связи с этим новейших концепций создания информационных систем (например, архитектура клиент/сервер, автоматизация рабочих процессов). В условиях постоянно усиливающейся конкуренции эти идеи стали актуальными практически для любой компании.

Гибкие информационные системы, сфокусированные на автоматизации внутренних бизнес-процессов, - один из решающих факторов в конкурентной борьбе компании. Выделить бизнес-процессы, проанализировать их взаимосвязи и предложить оптимизированную инфраструктуру информационной поддержки позволяет только комплексный анализ деятельности компании. Однако в отличие от традиционной централизованно-функциональной среды управление в этих новых структурах значительно сложнее. Для успеха здесь в первуб очередь необходимы ясно определенное и унифицированно описанное распределение ответственности, максимальная прозрачность структур, гомогенная коммуникационная база, интегрирующая все уровни компании, а также простое управление проектом. Все это должно быть ориентировано на достижение целей, которые ставит перед собой компания.

Методы моделирования деятельности компании являются одним из инструментов для управления этими сложными задачами. Модели деятельности – это решающая предпосылка для анализа бизнес-процессов, выстраивания проектов в одну линию с целями компании и, наконец, для построения эффективных информационных структур в виде сложных распределенных интегрированных систем, поддерживающих все основополагающие организационные структуры.

Моделирование реальных ситуаций в работе компании и отработка комплексных бизнес-процессов стали теммой все более широких обсуждений. Появление совершенно различных методов моделирования усиливает эту тенденцию, а их огромное множество приводит к еще большим усложнениям и путанице. В следствие этого предпринимаются попытки создать стандартизованные концепции (архитектуры) для процесса разработки информационных систем и методов моделирования.

Одной из таких концепций является – Архитектура Интегрированных Информационных Систем – ARIS (Architecture of Integrated Information Systems), разработанная проф. Шеером. Эта концепция имеет два основных преимущества:

- позволяет выбрать методы и интегрировать их, опираясь на основные особенности моделируемого объекта;
- служит базой для управления сложными проектами, поскольку благодаря структурным элементам содержит встроенные модели процедур для разработки интегрированных информационных систем.

Такая архитектура дает возможность вводить в применяемые методы элементы стандартизации. Новые методы моделирования, а также те, в основе которых лежит концепция ARIS, были интегрированы в рамках архитектуры, что позволило создать комплексный метод моделирования бизнес-процессов.

Более того, архитектура ARIS явилась основой **ARIS Toolset** – инструментальной среды, разработанной компанией IDS Scheer AG. Инструментарий ARIS позволяет проводить построение, анализ и оценку рабочих процессов компании в терминах методологии организации бизнес-процессов. Кроме того, ARIS предоставляет достаточно простые средства для документирования и моделирования процессов.

Данное руководство содержит краткий обзор методов моделирования, входящих в состав архитектуры ARIS. В сочетании с руководствами пользователя оно позволяет успешно применять ARIS Toolset. Его можно также рассматривать как пособие для тех, кто использует методы моделирования независимо от среды ARIS Toolset.

В гл. 2 дается краткое введение в структуру, и обсуждаются структурные элементы архитектуры ARIS.

В гл. 3 и 4 описываются методы моделирования. Структура гл. 4 соответствует архитектуре ARIS. Это позволяет проиллюстрировать трансформирование отдельных методов моделирования в структурные элементы архитектуры ARIS. Разделы главы посвящены различным типам моделей. Каждый тип в свою очередь делиться на отдельные концептуальные уровни (см. гл. 2).

В гл. 5, 6 и 7 описываются методы моделирования UML, OMT и управления знаниями.

2. Архитектура интегрированных информационных систем

2.1. Концепция архитектуры ARIS

Концепция Архитектуры Интегрированных Информационных Систем (ARIS) основана на идее интеграции, которая является составной частью комплексного анализа бизнес-процессов. Первый шаг при создании архитектуры состоит в разработке модели бизнес-процесса, описывающей все его основные функции. Полученная таким образом чрезвычайно сложная модель разделится на подмодели, или типы ммоделей, в соответствии с типами представлений. Это позволяет существенно снизить степень ее сложности. Содержимое типов моделей может быть описано методами, предназначеными для конкретного типа представления. Многочисленные взаимосвязи между типами моделей при этом не учитываются. Впоследствии эти взаимосвязи инкорпорируются в общую модель для анализа всего процесса без какой-либо избыточности.

Вторым подходом, уменьшающим сложность модели бизнес-процесса, является анализ каждого типа моделей на различных уровнях. В соответствии с концепцией модели жизненного цикла различные методы описания информационных систем дифференцируются по степени их близости к информационным технологиям. Это гарантирует целостность описания на всех этапах, начиная от проблем управления бизнесом до технической реализации информационной системы.

Архитектура ARIS создает основу для разработки и оптимизации интегрированных информационных систем, а также для описания их реализации. Выбор уровней и типов описаний формирует архитектуру ARIS, которая используется в качестве модели для построения процессов, связанных с управлением бизнесом, их анализа и оценки.

2.2. Типы моделей

Начнем с анализа бизнес-процесса, представленного на рис. 2.2-1.

Процесс запускается по событию *Заказ клиента поступил*. Это событие инициирует выполнение функции *Получить заказ*. Для того чтобы выполнить эту операцию, необходимы описания состояний ее инфраструктуры. В данном примере это состояние определяется данными о клиенте и перечнем изделий, включенных в заказ.

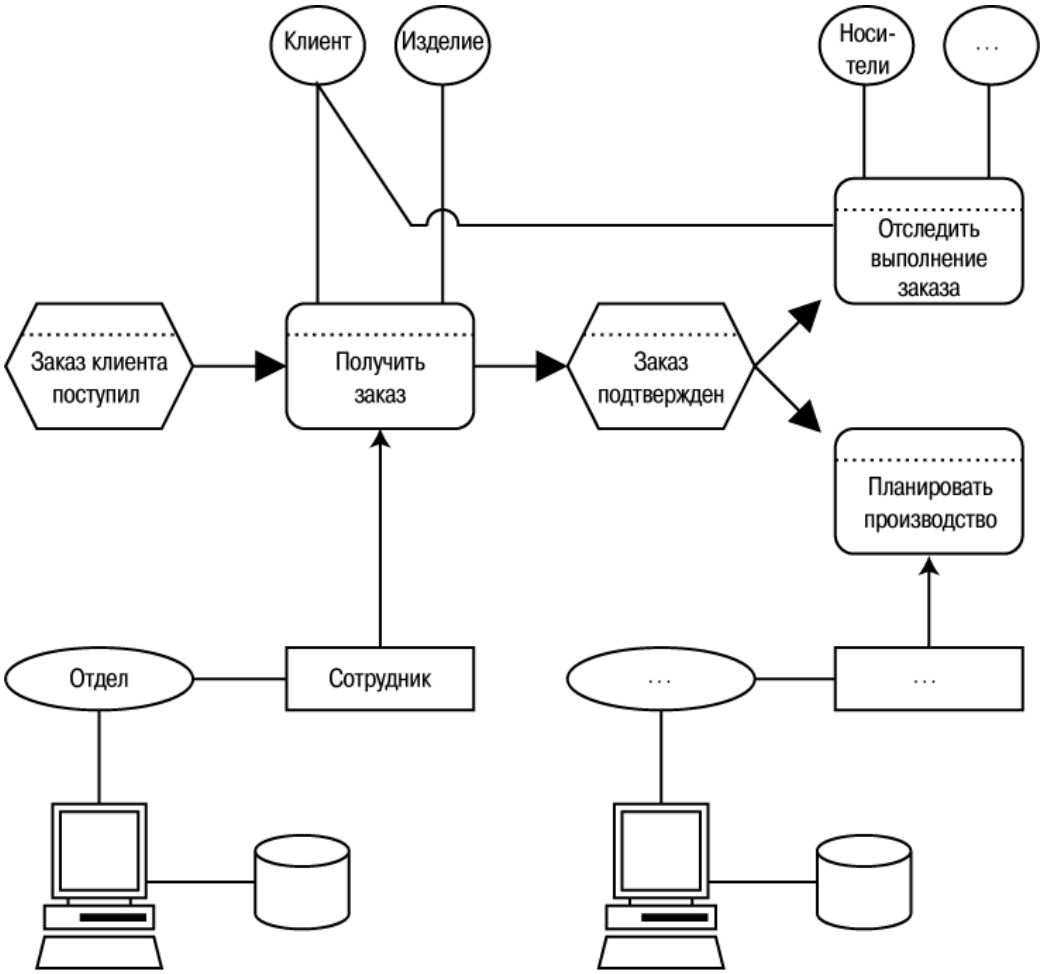


Рис. 2.2-1. Модель бизнес-процесса

Состояния данных, связанных с инфраструктурой, могут быть изменены как во время выполнения функции, так и при выполнении всего процесса. Например, данные о товарах на складе постоянно корректируются в соответствии с новыми данными о резервировании товара.

Функции выполняют сотрудники отдела. Для решения своих задач они используют различные ресурсы и оборудование: персональные компьютеры, принтеры и т.д.

Событие *Заказ подтвержден* является результатом выполнения функции *Получить заказ*. Событие *Заказ подтвержден* инициирует выполнение дополнительных операций, таких, как *Отследить выполнение заказа*, *Планировать производство*. Для выполнения этих операций необходимы описания множества

параметров состояний (данных), а также людские и технические ресурсы. Эти ресурсы могут быть связаны с отдельными компонентами других процессов. Таким образом, возможна ситуация, при которой требуются одни и те же описания состояний или одни и те же ресурсы.

Компоненты и их взаимосвязи, которые должны быть описаны в бизнес-процессе, включают процессы, события, состояния, пользователей, организационные единицы и информационные ресурсы. Учет абсолютно всех воздействий на элементы процесса для каждого события значительно усложняет модель и приводит к избыточности в ее описании.

Для упрощения модель делится на отдельные типы моделей (см. также рис. 2.2-2), в описании которых используются элементы дискретного моделирования и различные методики. С каждым типом моделей можно работать независимо от других типов. Типы моделей формируются таким образом, чтобы компоненты внутри каждого из них были тесно взаимосвязаны, в то время как отдельные типы в достаточной степени независимы.

Такие события, как *Заказ клиента поступил* или *Счет выписан,* определяют изменения в состояниях информационных объектов (данных). Состояния справочных полей, например *Состояние клиента* или *Состояние изделия,* также представляются данными. Таким образом, состояния и события формируют **представление данных,** или **информационную модель,** в архитектуре ARIS.

Функции, которые должны быть выполнены, и их взаимосвязи образуют **функциональное представление,** или **функциональную модель**. Эта модель содержит описание самих функций, а также перечень отдельных подфункций (операций) и их связей, как с основной функцией, так и между собой.

Организационное представление, или **организационная модель,** – это совокупность организационных единиц, их взаимосвязей и соответствующих структур.

Используемые ресурсы (оборудование и программы) составляют четвертый тип моделей - **модель ресурсов**. Однако, эта модель важна для анализируемых бизнес-процессов только в той степени, в какой она необходима для описания компонент, более тесно связанных с самим бизнес-процессом. По этой причине компоненты других типов моделей (данные, функции и организационные структуры) рассматриваются с точки зрения их привязанности к ресурсам. Таким образом, ресурсы связаны с моделями различных типов на уровне спецификации проекта и описания его реализации (см. также раздел 2.3).

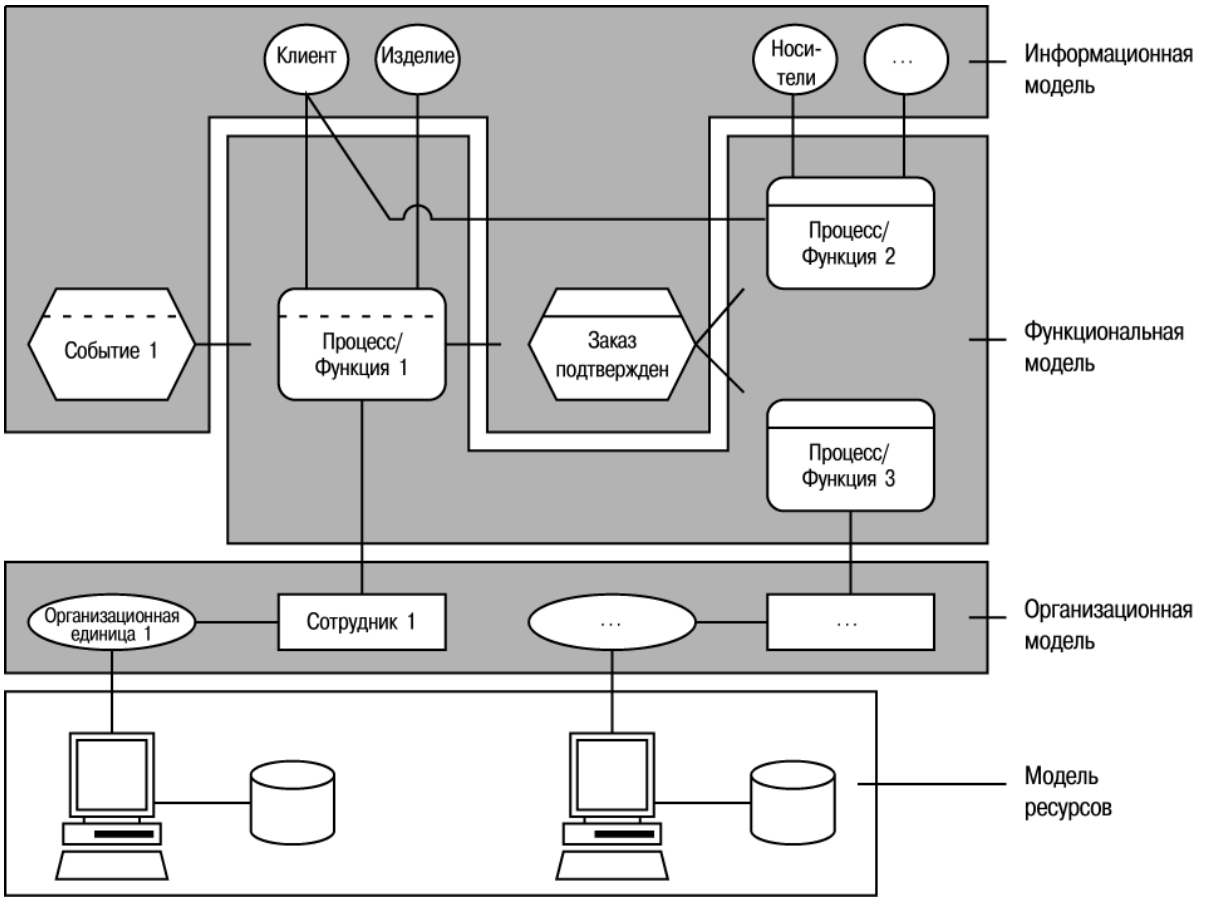


Рис. 2.2-2. Типы моделей, составляющих модель процесса

В модели всего жизненного цикла, об этом в 2.3, описание ресурсов заменяется независимым объектом.

Декомпозиция бизнес-процесса на отдельные типы моделей уменьшает степень его сложности за счет исключения из рассмотрения взаимосвязей между компонентами процесса, относящихся к другому типу моделей. В связи с этим вводится дополнительный тип модели – **управляющая модель**, в которой описываются взаимосвязи между моделями различных типов. Интеграция этих взаимосвязей с помощью модели специального типа позволяет вводить дополнительные взаимосвязи без какой-либо избыточности.

Управляющая модель – важнейшая компонента архитектуры ARIS, отличающая ее от архитектур, предлагаемых другими авторами.

Таким образом, мы приходим к четырем типам моделей, используемых в архитектуре ARIS (см. рис. 2.2-3), которые подробно обсуждаются в последующих разделах.

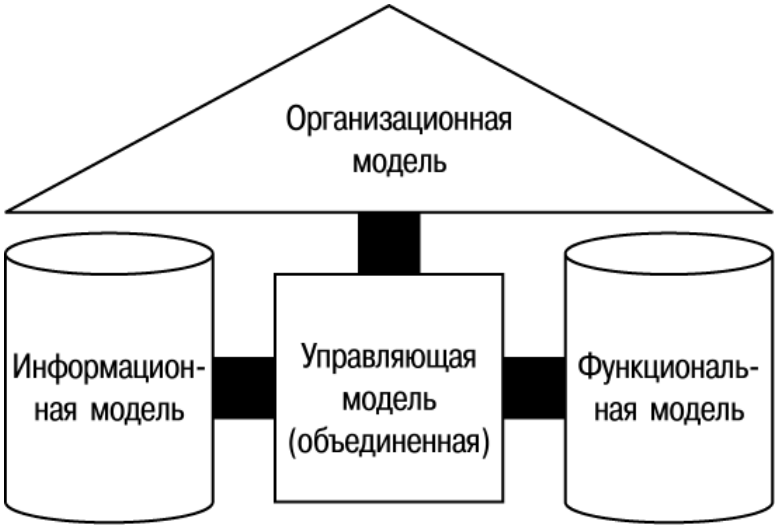


Рис. 2.2-3. Модели архитектуры ARIS при описании бизнес-процесса

2.3. Уровни представления моделей

Модель ресурсов в ARIS структурируется в соответствии с концепцией жизненного цикла на уровне представления моделей информационных систем.

Модель жизненного цикла, представляемая в виде последовательности уровней или этапов, предназначена для описания жизненного цикла информационной системы (ИС). Однако модель жизненного цикла ARIS не может рассматриваться как процедурная модель для разработки некоторого независимого объекта на каждом уровне представления. Различные уровни представления выделены в модели в зависимости от степени их близости к информационным технологиям (ИТ).

Это различие выражено в трехярусной модели ARIS (рис. 2.3-1).

Рис. 2.3-1. Уровни представления информационной системы

Анализ **проблем бизнеса** – начальная точка в разработке информационной системы. Модели на этом уровне – это не очень детальные описания бизнес-процессов, однако они достаточно точно отражают цели, которые стоят перед пользователем информационной системы, и его язык. На этом этапе в описание включаются некоторые сведения по характеристикам будущей информационной системы, связанным с характеристиками бизнес-процессов. Для описания проблем бизнеса

используются только полуформальные методы. Однако полученные модели еще не содержат достаточно детальной информации и однозначных технических формулировок, чтобы служить исходным материалом для автоматической передачи их непосредственно на этап реализации ИС.

На уровне **формулировки требований** необходимо описать программное решение (прикладную информационную систему) для рассматриваемой проблемы бизнеса. Оно должно поддерживаться формализованным описанием требований с целью последующего использования в качестве стартовой точки для трансляции сформулированных требований в программную систему. Этот процесс также очень близок к семантическому (смысловому) моделированию. Формулировка требований тесно связана с описанием проблем бизнеса. Эта связь показана широкой стрелкой на рис. 2.3-1.

Уровень **спецификации проекта** достигается, как только концептуальные понятия проблем бизнеса, сформулированные на уровне формулировки требований, трансформируются в категории, связанные с информационными технологиями. На данном уровне описываются уже не функции, а пользовательские или модульные транзакции, которые выполняют функции, как это было определено ранее. Это может рассматриваться как отображение сформулированных требований в категории и методы описания, связанные непосредственно с ИС и выраженные в терминах информационных технологий. Таким образом, уровни формулировки требований и спецификации проекта связаны достаточно тесно.

Спецификация проекта может изменяться, не оказывая влияния на результаты предыдущего уровня формулировки требований. Однако это не означает, что формулировка требований и спецификация проекта могут прорабатываться независимо друг от друга. После того, как завершен этап формулировки требований, его наиболее важная содержательная часть, отражающая категории управления бизнесом, должна быть определена таким образом, чтобы все, относящиеся к области информационных технологий и программных решений (например, производительность информационной системы) не влияло на предметное содержание.

На уровне **описания реализации** спецификация проекта трансформируется в конкретные аппаратные и программные компоненты. Таким образом, осуществляется физическая связь с информационной системой.

Отдельные уровни описания имеют различные циклы корректировки. Частота корректировок выше всего на уровне описания реалиации и ниже всего на уровне формулировки требований.

Уровень описания реализации очень тесно связан с разработкой информационной системы: на этом уровне производится многократная корректировка функционирования системы по результатам коротких циклов (тестов) ее работы.

Уровень формулировки требований особенно важен, поскольку его можно рассматривать как репозитарий для прикладных программных систем, используемых в течение длительного времени, и как стартовую точку при описании реализации. Документы, созданные на уровне формулировки требований, имеют наиболее продолжительный жизненный цикл, и в силу их близости к описанию проблем бизнеса, которое также является документом, они чрезвычайно полезны для разработки информационных систем. По этой причине уровень формулировки требований, или семантическая модель, имеет наивысший приоритет. Семантические модели образуют связь между пользователями и первоначальным описанием их проблем на языке, ориентированном на категории информационных систем.

Создание различных типов моделей и проработка каждой из них по уровням описания в сочетании с формулировкой проблем бизнеса и составляет процесс работы в архитектуре ARIS. Как показано на рис. 2.3-2, каждый тип модели подвергается разложению на три уровня: формулировку требований, спецификацию проекта и описание реализации.

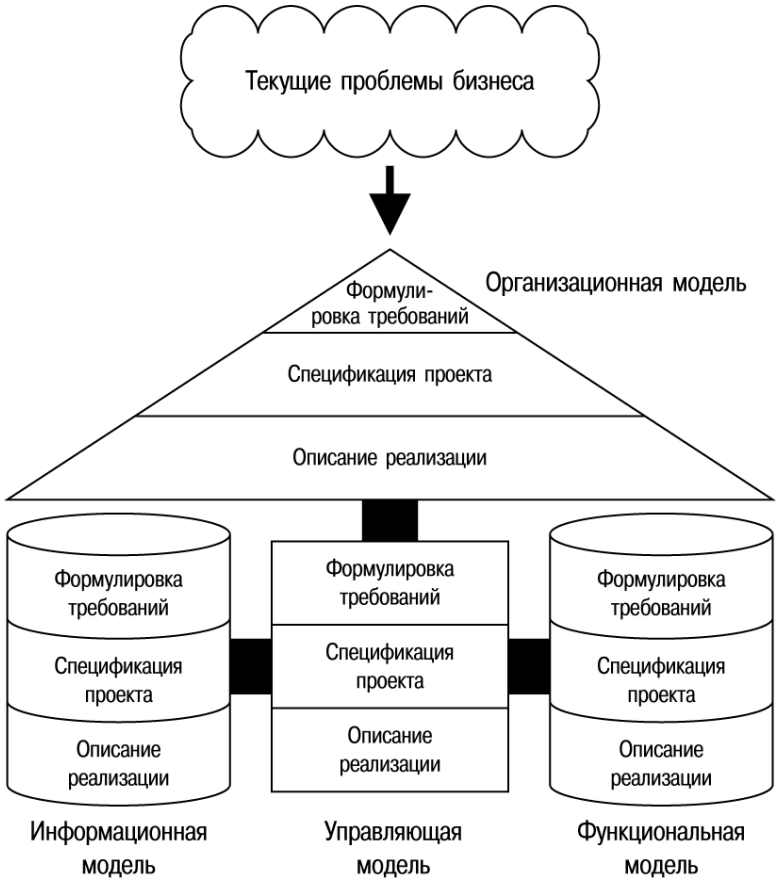


Рис. 2.3-2. Архитектура ARIS

Таким образом, мы зафиксировали в архитектуре ARIS набор типов моделей, каждая из которых «расписывается» по уровням. Вместе с описанием проблем бизнеса, которое служит стартовой точкой для анализа, они составляют тринадцать компонент архитектуры ARIS. Теперь необходимо выбрать и представить методы описания каждой компоненты архитектуры.

Критерии для выбора этих методов следующие:

- простота и выразительность средств изображения,
- поддержка смыслового содержания, для отображения специфики предмета,
- возможность использования полного набора методов для различных типов приложений,
- степень знакомства с методами и наличие необходимой литературы,
- определенная степень независимости методов от технической реализации в информационных и коммуникационных системах.

Методы, применяемые при построении моделей различных типов, описаны в следующих главах.

3. Анализ процессов

3.1. Описание проблем бизнеса

Перед тем, как моделировать отдельные компоненты архитектуры ARIS (различные типы моделей и уровни их представлений), необходимо осмыслить семантику (содержательную часть) бизнес-процесса, т.е. понять проблемы бизнеса. Недостатки используемых информационных систем должны быть описаны так, чтобы концепция проектируемой системы была ориентирована на поддержку бизнес-процессов и целевые установки бизнеса. Выявленные недостатки более четко определяют цели, которые должны быть достигнуты в результате вновь создаваемых информационных систем.

Следовательно, модель, отражающая эти проблемы бизнеса, должна описывать широкий спектр бизнес-процессов с точки зрения данных, функций и организационных структур, включая взаимосвязи, существующие между ними. Более того, модель должна описывать целевые установки таким образом, чтобы это описание служило основой для остальной части процесса моделирования.

Требования к полноте описания текущего состояния бизнеса, а также к компактности отображения недостатков имеющихся информационных систем обусловливают ограниченность возможностей общеизвестных методов моделирования. Основные возможности этих методов рассчитаны на анализ различных аспектов бизнеса, поэтому они могут применяться только для создания отдельных типов моделей.

Взаимосвязи процессов представляются в компактном виде с помощью диаграмм (PCDs), которые также позволяют описывать информационные системы.

3.2. Диаграммы процессов (PCDs)

Диаграмма типа PCDs (Process Chain Diagram) – это диаграмма цепочки процесса, или просто **диаграмма процесса**. В диаграмме этого типа цепочка (последовательность шагов/функций) процесса отображается в виде замкнутого цикла. Отдельные типы моделей бизнес-процессов, определенные ранее (организационная модель, модель данных, функциональная модель и модель ресурсов), а также их взаимосвязи здесь выражены более отчетливо.

На рис. 3.2-1 приведена диаграмма процесса *обработка заказа*.

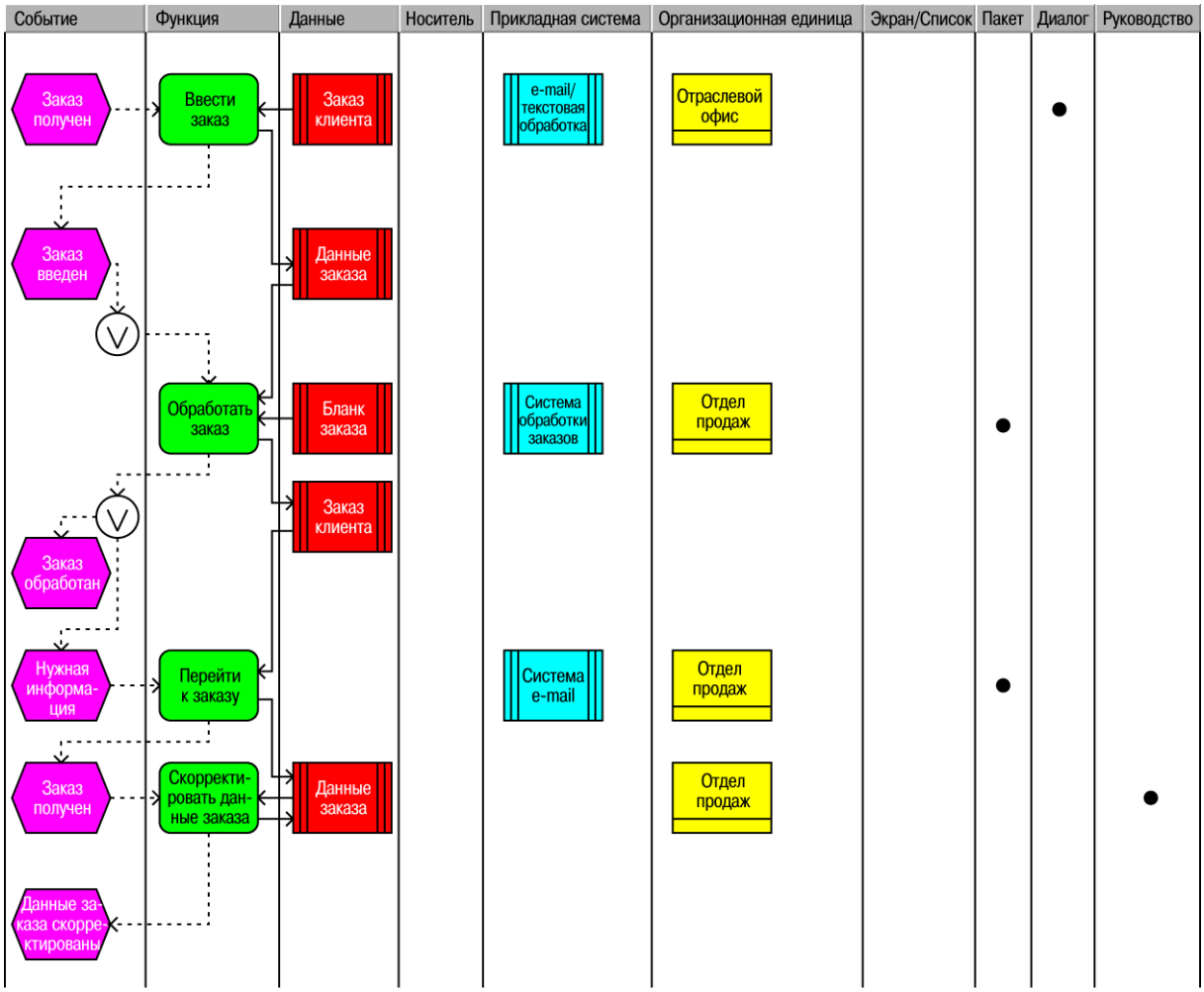


Рис. 3.2-1. Диаграммы процесса

Первый и второй столбцы отображают логическую последовательность выполнения процесса. Отдельные функции процесса представлены во втором столбце. Онии связаны с событиями, которые инициируют выполнение функций или переключение между ними. Функции и события связаны пунктирными стрелками, указывающими события, которые пререключают функции, а также события, которые сами генерируются функциями. Таким образом, эти стрелки отображают поток управления функциями.

В приведенном примере функция *Ввести заказ* инициируется событием *заказ получен*. В результате выполнения этой функции происходит событие, *заказ введен*, которое в свою очередь инициирует следующую функцию *обработать заказ*. Это отношение между событиями и функциями составляет процедурную

последовательность функций как логическую цепочку событий, которую называют цепочкой процесса. Логическая взаимозависимость возможных точек ветвления и циклов потока управления может быть выражена посредством логических операций, связывающих функции и события.

Входные и выходные данные, требующиеся функциям, показаны в следующем столбце в виде кластеров данных. Входные данные для функции *обработать заказ*, - это *данные заказа*; генерируемые выходные данные – *заказ клиента*. Отображаться могут не только информационные объекты, но и носители (среда), на которых находится информация. Это может быть документ, список, написанная вручную квитанция или устройство памяти (например, жесткий диск).

Организационные единицы (отделы), ответственные за выполнение отдельных функций, показаны в последнем столбце.

Столбцы *тип обработки* (диалог, пакет, ручная) и *прикладная система* представляют дополнительную информацию о степени использования информационных технологий (ИТ) для поддержки отдельной функции. Прикладные системы или отдельные компоненты прикладной системы могут быть указаны в столбце *прикладная система*. Столбец *тип обработки* содержит более подробную информацию о том, как выполняется функция, причем ее выполнение может быть интерактивным, пакетным или ручным.

При анализе бизнес-процесса, который отображает некоторую реальную ситуацию, недостатки его организации или причины неэффективности могут быть определены с помощью диаграммы процесса. Такими недостатками могут быть дезинтеграция между ручной обработкой и обработкой с помощью ИТ или организационная дезинтеграция (например, сотрудники отдела, выполняющие данную функцию, часто меняются). Кроме того, лишние входы (процедурная избыточность данных) и задержки в выполнении функций становятся четко видимыми. Это приводит к появлению многочисленных идей по совершентствованию рассматриваемого бизнес-процесса.

При описании начальной ситуации диаграммы процессов создаются с относительно высоким уровнем обобщения (без детализации). Поскольку эти диаграммы используются главным образом для отображения взаимосвязей всех компонент ARIS (моделей различных типов), они служат основой для создания управляющей модели ARIS (см. раздел 4.4). При создании управляющей модели используются не только диаграммы процесса, но и диаграммы цепочки процесса, управляемого событиями (EPCs – Event-driven Process Chain, – см. раздел 4.4.1.2.1). Будем их называть **событийными диаграммами процессов**. С точки зрения моделирования событийные диаграммы процессов также полезны, как и простые диаграммы процессов. Единственное отличие заключается в том, что элементы диаграмм EPCs не должны располагаться в предопределенных столбцах. Таким образом, модель небольшой процедуры может быть построена с помощью одного из

методов (PCDs или EPCs), в то время как для представления модели всего бизнес-процесса предпочтительнее EPCs.

4. Моделирование в архитектуре ARIS. Типы моделей и уровни описаний

4.1. Функциональная модель

4.1.1. Формулировка требований

Методы моделирования описывают функции, используя также объекты, относящиеся к другим типам моделей ARIS. Например, отображение взаимосвязей между данными и функциями дает возможность описать процесс преобразования, выполняемого функцией, через входные/выходные данные для этой функции.

С другой стороны, архитектура ARIS опирается на точное разделение компонент. В функциональной модели используются только такие средства из других типов моделей, которые отражают связи между функциями. Один из примеров – взаимосвязь между функциями и данными в том виде, в каком она отображается в управляющей модели ARIS.

Определение

Функция – это предметно-ориентированное задание или действие, выполняемое над объектом, в результате которых достигается одна или несколько целей, стоящих перед компанией.

Функции отображаются в виде прямоугольников с закругленными углами (см. рис. 4.1.1-1).

Рис. 4.1.1-1. Пример функции «проверить достоверность запроса клиента»

Обычно критерием для использования функции является наличие информационного объекта, такого, как, например, *запрос клиента* или *заказ на продукцию*. Эта взаимосвязь должна отображаться в имени функции, как показано на рис. 4.1.1-1. *Запрос клиента* определяет объект, *проверить достоверность* определяет задание, которое должно быть выполнено над этим объектом. Для описания функций используются глаголы (например, *произвести*, *продать*), однако в отдельных случаях

(для более правильного с точки зрения русского языка построения фразы) разрешается употреблять существительные, обозначающие действия.

4.1.1.1. Функциональное дерево

Функции могут быть описаны с различными уровнями детализации. На самом верхнем уровне описываются наиболее сложные функции, представляющие собой отдельный бизнес-процесс или последовательность процессов. Рассмотрим, например, процесс обработки запроса клиента на всем его протяжении, начиная от получения запроса клиента до отгрузки товара. Такой бизнес-процесс состоит из сложной функции, которая может быть разделена на подфункции. Следовательно, термин *функция* может быть использован на всех уровнях детализации. Последовательная детализация функций образует иерархическую структуру их описаний. Для более содержательного описания отдельного уровня иерархии могут быть использованы также другие термины: транзакция, процесс, подфункция, базовая функция (операция).

Разделение функций на элементы может происходить на нескольких иерархических уровнях. Базовые функции представляют самый нижний уровень в семантическом дереве функций.

Определение

Базовая функция – это функция, которая уже не может быть разделена на составные элементы с целью анализа бизнес-процесса.

Для представления иерархической структуры функций служит диаграмма дерева функции, или иерархическая диаграмма (см. рис. 4.1.1-2).

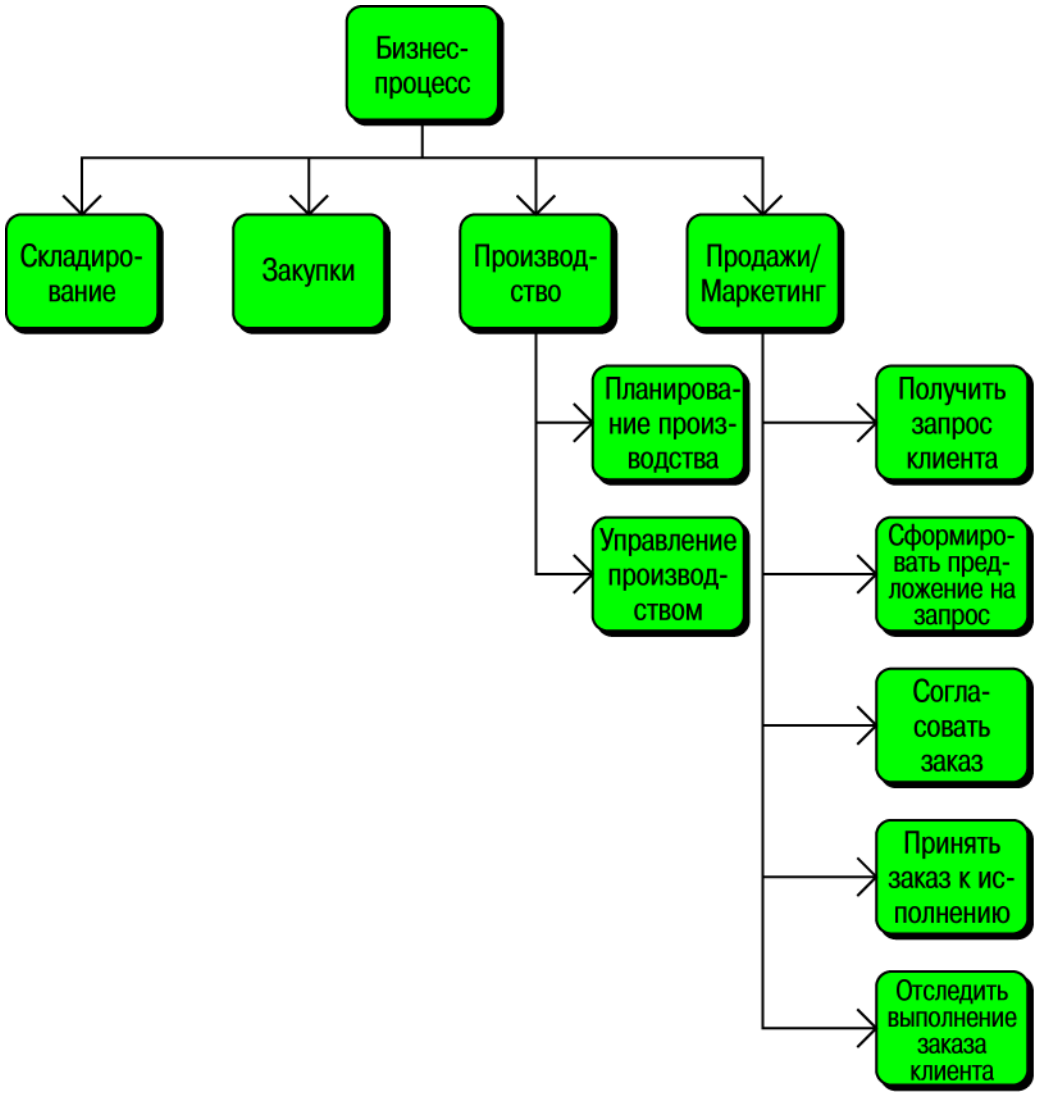


Рис. 4.1.1-2. Дерево функций (частичная модель)

Функции объединяются в функциональное дерево в соответствии с различными критериями. Наиболее часто для этих целей используются такие критерии, как обработка одного и того же объекта (объектно-ориентированный), принадлежность одному и тому же процессу (процессо-ориентированный), выполнение одинаковых операций (операционно-ориентированный).

Объектно-ориентированный подход при детализации функции иллюстрирует рис. 4.1.1-3.

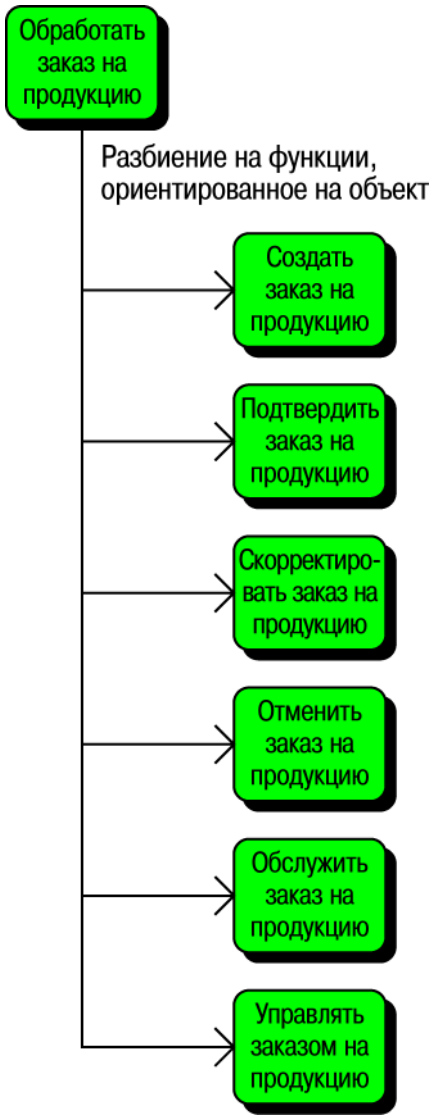


Рис. 4.1.1-3. Объектно-ориентированное функциональное дерево

Функция *обработать заказ на продукцию* детализируется на функции *создать заказ на продукцию, скорректировать заказ на продукцию, отменить заказ на продукцию, обслужить заказ на продукцию, подтвердить заказ на продукцию* и *управлять заказом на продукцию*. Эти функции описывают различные операции (*создать*, *скорректировать*, *отменить* и т.д), которые выполняются над одним и тем же объектом, в данном случае - над объектом *заказ на продукцию*.

Если функциональное дерево используется в рамках моделирования бизнес-процесса, предпочтительнее применять метод, позволяющий построить процессо-

ориентированное функциональное дерево. На рис. 4.1.1-4 представлена процессо-ориентированная детализация функции (последовательность функций, составляющих процесс).

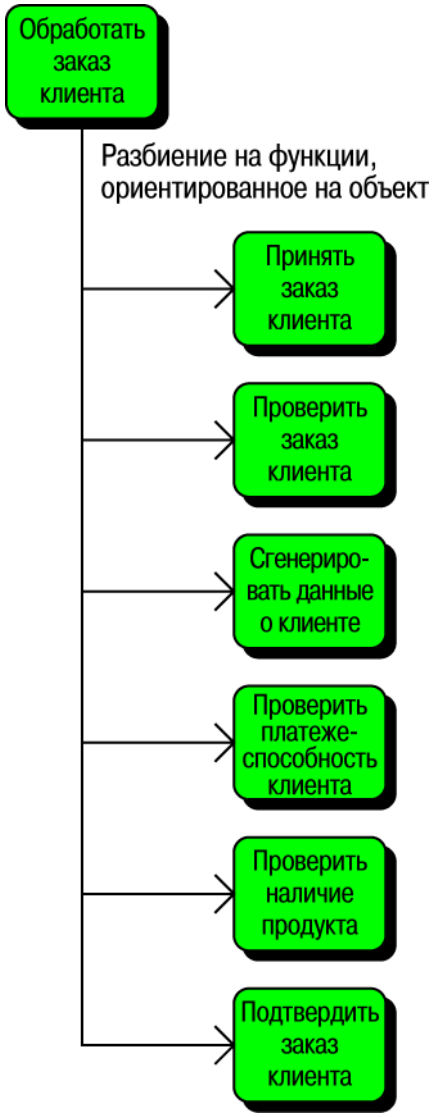


Рис. 4.1.1-4. Процессо-ориентированное функциональное дерево

Функции *принять заказ клиента*, *проверить заказ клиента*, *сгенерировать данные о клиенте*, *проверить платежеспособность клиента*, *проверить наличие продукта* и *подтвердить заказ клиента* составляют бизнес-процесс *обработать заказ клиента*. В отличие от объектно-ориентированного разбиения критерием при процессо-

ориентированной детализации служат операции, которые выполняются над различными объектами (заказ клиента, наличие продукта) в рамках бизнес-процесса.

При операционно-ориентированном подходе функция верхнего уровня декомпозируется на подфункции, каждая из которых выполняет ту же операцию, но с различными объектами. На рис. 4.1.1-5 приведена функция, выполняющая операцию *скорректировать*. Функции могут принадлежать различным процессам и привлекаться к обработке различных объектов. Однако выполняемый ими тип операции над отдельными объектами всегда один и тот же.

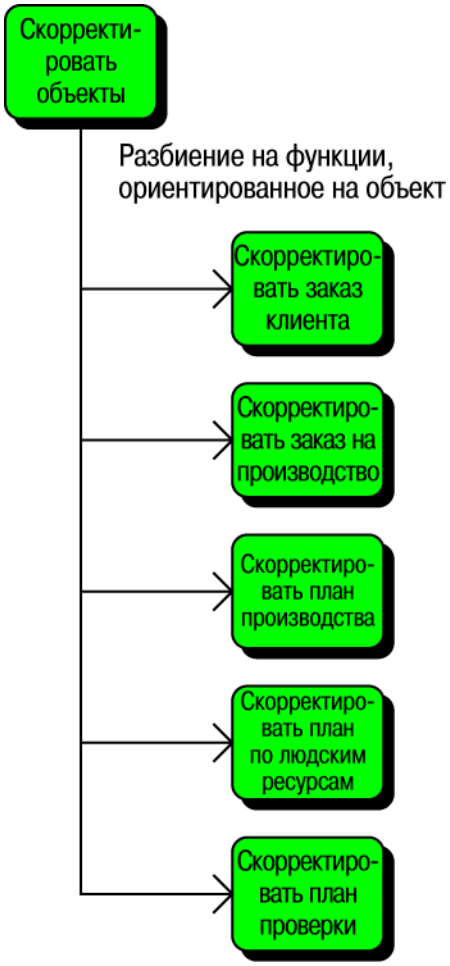


Рис. 4.1.1-5. Операционно-ориентированное функциональное дерево

Способ представления в виде функционального дерева позволяет уменьшить степень сложности и является статичным описанием функции. Динамические описания могут потребоваться при анализе последовательности функций, выполняемых в

хронологическом порядке в рамках некоторой процедуры. Событийные диаграммы процессов (EPCs) - эффективное средство для представления хронологической последовательности функций как логической цепочки событий. Эти диаграммы состоят из событий и функций. События формируют связи между функциями и описываются в модели данных ARIS. Событийные диаграммы процессов обсуждаются при рассмотрении управляющих моделей в соответствии с принципами разделения типов моделей в архитектуре ARIS (см. раздел 4.4.1).

При описании функции с объектно-ориентированной точки зрения используется не только такое ее свойство, как возможность декомпозиции на элементы, но и другие свойства функции, представляющие интерес. В особенности это относится к свойствам, которые учитываются при проектировании бизнес-процессов.

Таким образом, каждое описание функции должно включать информацию, будет ли эта функция инициирована пользователем или она может работать автоматически. Это позволяет объединять все аналогичные функции, не требующие вмешательства пользователя, в один пакет (пакетное задание).

При реорганизации бизнес-процессов анализируются количественные характеристики выполняемых функций, например, число запросов, обрабатываемых за день, или совокупное время работы функции, которое формируется из отдельных временных элементов (время настройки, время обработки и время ожидания). ARIS сохраняет эту информацию как атрибуты объекта типа *функция*.

4.1.1.2. Y - диаграмма

Y-диаграмма представляет функции (задания) компании на самом верхнем уровне агрегации. Здесь мы имеем дело с основными макрофункциями: *прототипированием изделия*, *управлением материалами*, *обслуживанием*. Структурное представление в виде модели Y-CIM классифицирует отдельные функции. Левая ветвь Y содержит основные управленческо-административные функции, связанные с планированием и управлением производством, а правая - технико-ориентированные функции планирования производства и реализации продукции. Функции планирования расположены в верхних частях Y, в то время как функции управления и реализации находятся в нижней части.

Модель Y-CIM представляет основу для классификации всех функций компании, участвующих в производстве.

В рамках ARIS этот тип диаграмм может применяться при функционально-ориентированном анализе сложных моделей. Отображаемые объекты являются объектами типа *функция*. С точки зрения формирования иерархической структуры этот тип объекта может быть связан, например, с диаграммами типа *функциональное дерево* и *EPCs* (событийные диаграммы процесса).

Y-диаграмма представлена на рис. 4.1.1-6.

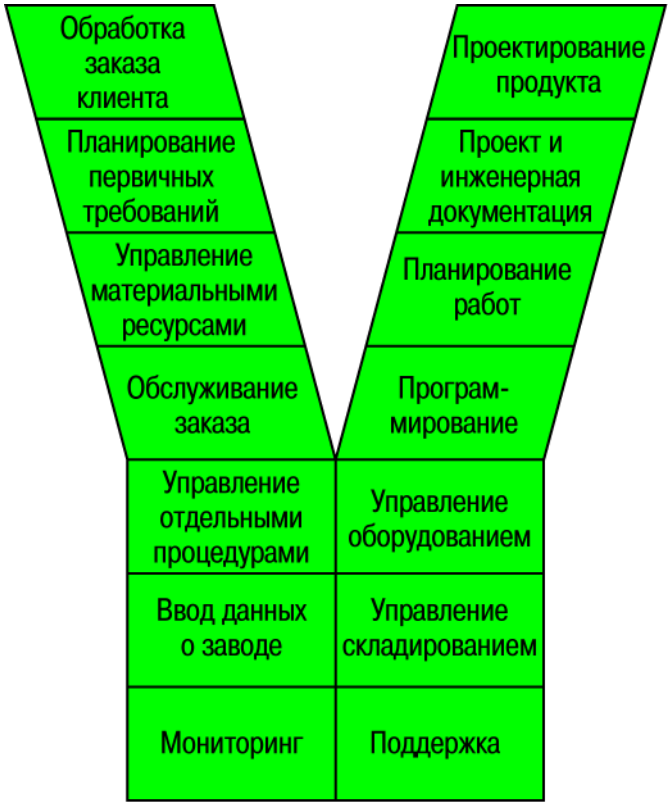


Рис. 4.1.1-6. Y-диаграмма

4.1.1.3. Диаграмма SAP-приложений

Диаграмма SAP-приложений позволяет представлять модели-прототипы SAP R/3, ориентированные на модули системы управления предприятием SAP R/3. В модели-прототипе R/3 матрица выбора процессов связана с каждым объектом диаграммы данного типа. Она отображает основные процессы, доступные в отдельных модулях R/3, и сценарии процессов, которые могут быть затем проиллюстрированы.

Диаграмма SAP-приложений в системе SAP R/3 представлена на рис. 4.1.1-7.

Рис. 4.1.1-7. Диаграмма SAP-приложений

4.1.1.4. Диаграмма целей

Прежде чем начать моделирование, анализ или оптимизацию рабочего процесса, необходимо определить цели компании в области совершенствования бизнес-процессов.

С помощью диаграммы целей можно описать цели компании и построить их иерархию.

Определение

Цель – это определение будущих задач компании, которые предполагается выполнить, при поддержке способствующих успеху факторов и реализации новых бизнес-процессов.

Факторы успеха при достижении сформулированной цели можно определить, выстроить иерархию этих факторов и установить их соответствие целям.

Определение

Факторы успеха определяют аспекты деятельности, которые необходимо рассмотреть, чтобы достичь отдельной цели компании. В диаграмме целей фактор успеха должен быть связан с соответствующей целью компании.

Данный тип диаграмм связывается с другими диаграммами на уровне формулировки требований с помощью объекта типа *функция*. Для каждой цели можно отобразить функцию (бизнес-процесс), которая ведет к достижению цели. При моделировании и оптимизации бизнес-процессов необходимо указать приоритеты объектов и сответствующих функций.

Диаграммы целей показана на рис. 4.1.1-8.

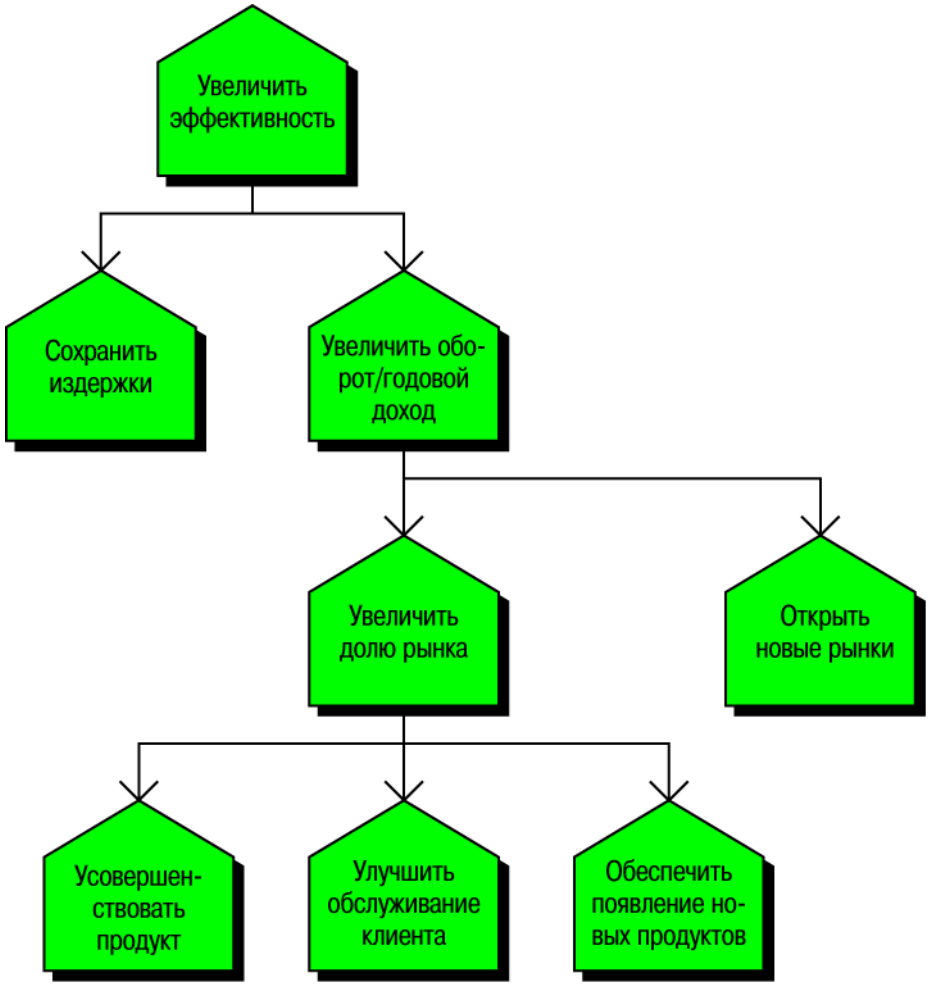


Рис. 4.1.1-8. Диаграмма целей

4.1.2. Спецификация проекта. Диаграмма типа прикладной системы

Уровень спецификации проекта для функциональной модели включает спецификацию прикладной системы (ПС) и типов модулей, модульную структуру ПС,

прорисовку отдельных шагов-транзакций, а также определение входных и выходных графических интерфейсов. Эта информация предоставляется в виде списков и экранов.

На уровне спецификации проекта в рамках функциональной модели необходимо ответить на следующие ключевые вопросы:

- Какова может быть поддержка функций, определенных с помощью типов ПС, типов модулей или проектов этих функций?
- Можем ли мы что-либо сказать о модульной структуре ПС или типах модулей?
- Какие списки и экраны потребуются для выполнения функции?
- Какие списки могут быть созданы с помощью прикладной системы данного типа или модуля данного типа и какие экраны поддерживают прикладную систему и модули данных типов?
- Какая технологическая база имеется в распоряжении для реализации прикладной системы данного типа (операционная система, интерфейс пользователя или система управления базами данных)?
- Как соотносится с целями компании прикладная система определенного типа?

Тип ПС может рассматриваться как тип основного объекта в рамках функциональной модели на этапе спецификации проекта.

В отличие от конкретной прикладной системы, которая рассматривается только на уровне описания реализации функциональной модели (например, прикладная система компании будет идентифицирована номером лицензии), тип ПС формируется посредством объединения всех прикладных систем, базирующихся на одной и той же технологии.

Определение

Тип прикладной системы представляет отдельные прикладные системы, базирующиеся на той же технологии.

Пример. Инструментарий ARIS, версия 4.1, представляет тип ПС. Вы можете приобрести несколько лицензий этого типа ПС и таким образом стать собственником нескольких конкретных прикладных систем.

На рис. 4.2.1.-1 приведен тип прикладной системы.

Рис. 4.1.2-1. Типа прикладной системы

Тип прикладной системы – это некоторая обобщенная ПС, состоящая, как правило, из отдельных модулей. Диаграмма типа прикладной системы – это представление ее модульной структуры. Типы модулей образуют отдельные части типа ПС, как показано на рис. 4.1.2-2.

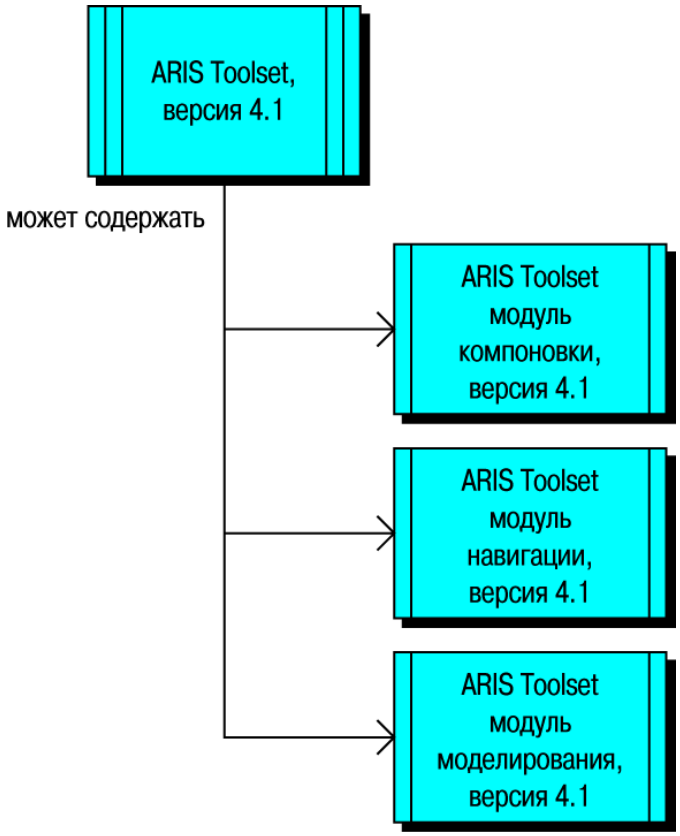


Рис. 4.1.2-2. Модульная структура типа ПС

Инструментарий ARIS Toolset, версия 4.1, состоит из модулей следующих типов: модуль компоновки, версия 4.1, модуль навигации, версия 4.1, и модуль моделирования, версия 4.1. Так же, как типы прикладной системы, типы модулей

объединяют отдельные модули, базирующиеся на одной и той же технологии. Типы модулей содержатся в типах прикладной системы и представляют отдельные выполняемые компоненты.

Определение

Типы модулей – это отдельно выполняемые компоненты типа ПС. Типы модулей представляют отдельные модули, которые базируются строго на одной и той же технологической базе.

Типы прикладных систем и типы модулей могут быть иерархически упорядочены любым способом. На самом нижнем уровне типы модулей можно декомпозировать на типы функций.

Определение

Типы функций – это наименьшие элементы модуля, имеющие смысл транзакции. Они реализуются с помощью отдельных программных элементов и должны быть полностью выполняемыми, причем так, чтобы их выполнение происходило за единственный шаг.

Рис. 4.1.2-3 иллюстрирует тип функции.

Рис. 4.1.2-3. Графическое представление типа функций

С помощью диаграммы типа ПС в рамках формулировки требований отдельные функции могут быть соотнесены и поддержаны определенными типами ПС и типами модулей. Это позволяет отразить связь между сформулированными требованиями и спецификацией проекта в рамках функциональной модели. Пример, иллюстрирующий сказанное, приведен на рис. 4.1.2-4.

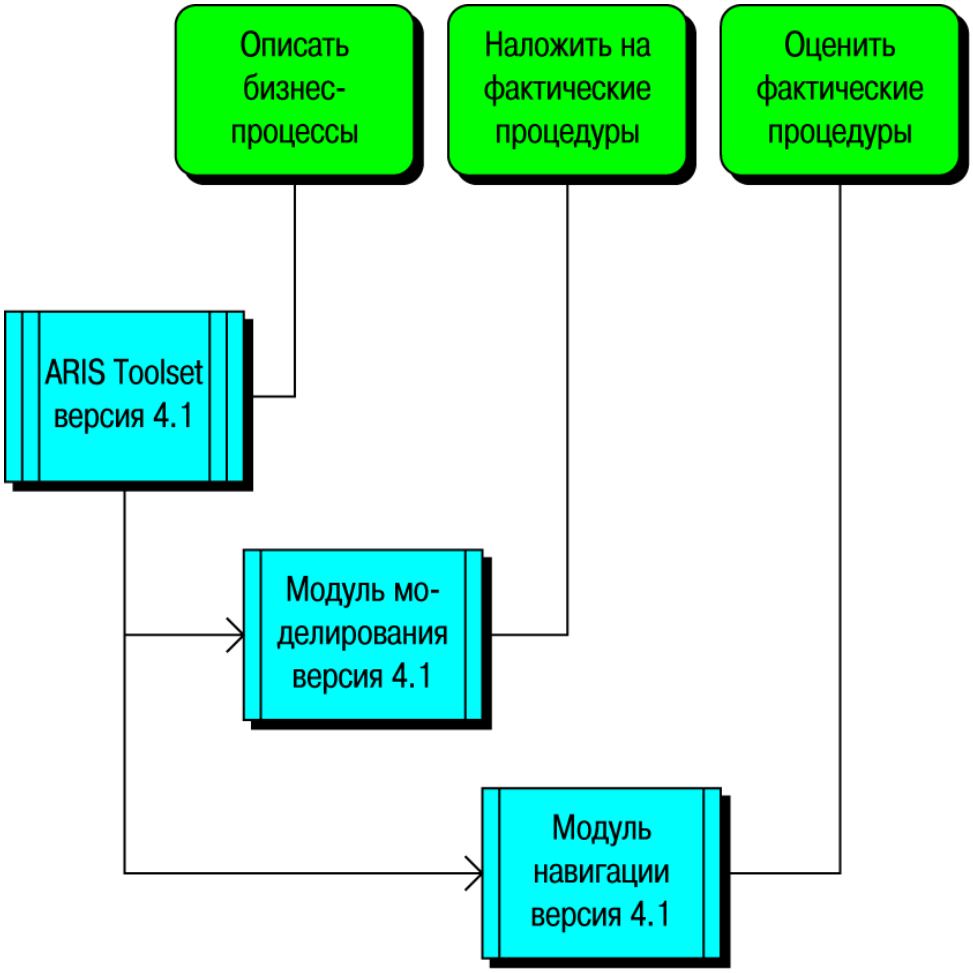


Рис. 4.1.2-4. Соотнесение функций типам ПС

Для того чтобы дальше специфицировать технологическую базу для типов ПС и типов модулей, им могут быть приписаны соответствующие типы пользовательских интерфейсов, СУБД и операционных систем, а также языков программирования, которые использовались для их реализации. При этом допускаются кратные отношения.

Таким образом, тип ПС может соответствовать пользовательскому интерфейсу Windows 98 и Windows NT. Это означает, что данная версия типа ПС способна работать с пользовательскими интерфейсами обоих типов. Только в том случае, когда пользовательский интерфейс соответсвует конкретной прикладной системе на уровне описания реализации в рамках функциональной модели, становится необходимой уникальная взаимосвязь. Эта взаимосвязь описывает конфигурацию отдельной лицензии на типы ПС, которые были приобретены компанией.

Пример возможной привязки в рамках диаграммы типа ПС представлен на рис. 4.1.2-5.

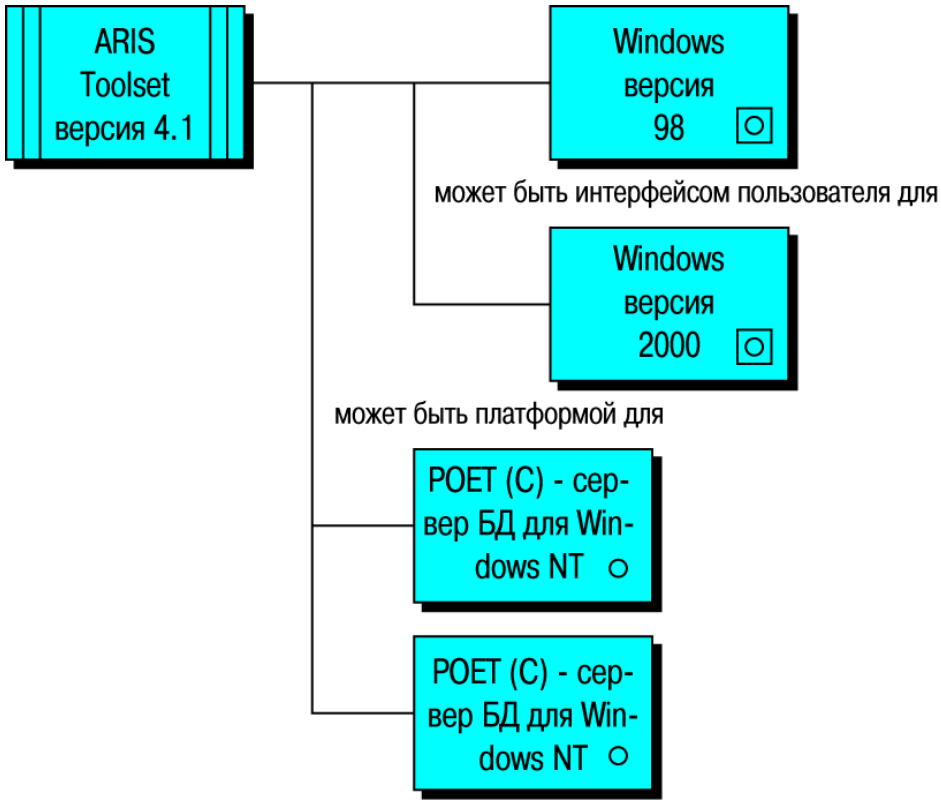


Рис. 4.1.2-5. Конфигурация типа ПС

Выполнение объектно-ориентированной функции с помощью прикладной системы подразумевает возможность использования различных экранов, создание различных списков, которые предоставляются прикладной системой, и работу с ними. Поэтому можно ввести такие объекты, как *список* и *экран*, которые связываются либо с объектно-ориентированной функцией, либо с типами ПС и типами модулей.

При определении общих процедурных последовательностей без ссылки на конкретные типы ПС спецификация необходимых экранов и списков может производиться с помощью объектов *список-прототип* и *экран-прототип*. Объекты обоих типов в общем случае определяют, какой тип списка или экрана должен использоваться (например, ввод данных клиента), без ссылки на конкретные списки или экраны типа ПС. Далее *списки-прототипы* и *экраны-прототипы* могут быть привязаны к конкретным спискам и экранам. Таким образом, привязка определяет возможности реализации, что проиллюстрировано на рис. 4.1.2-6.

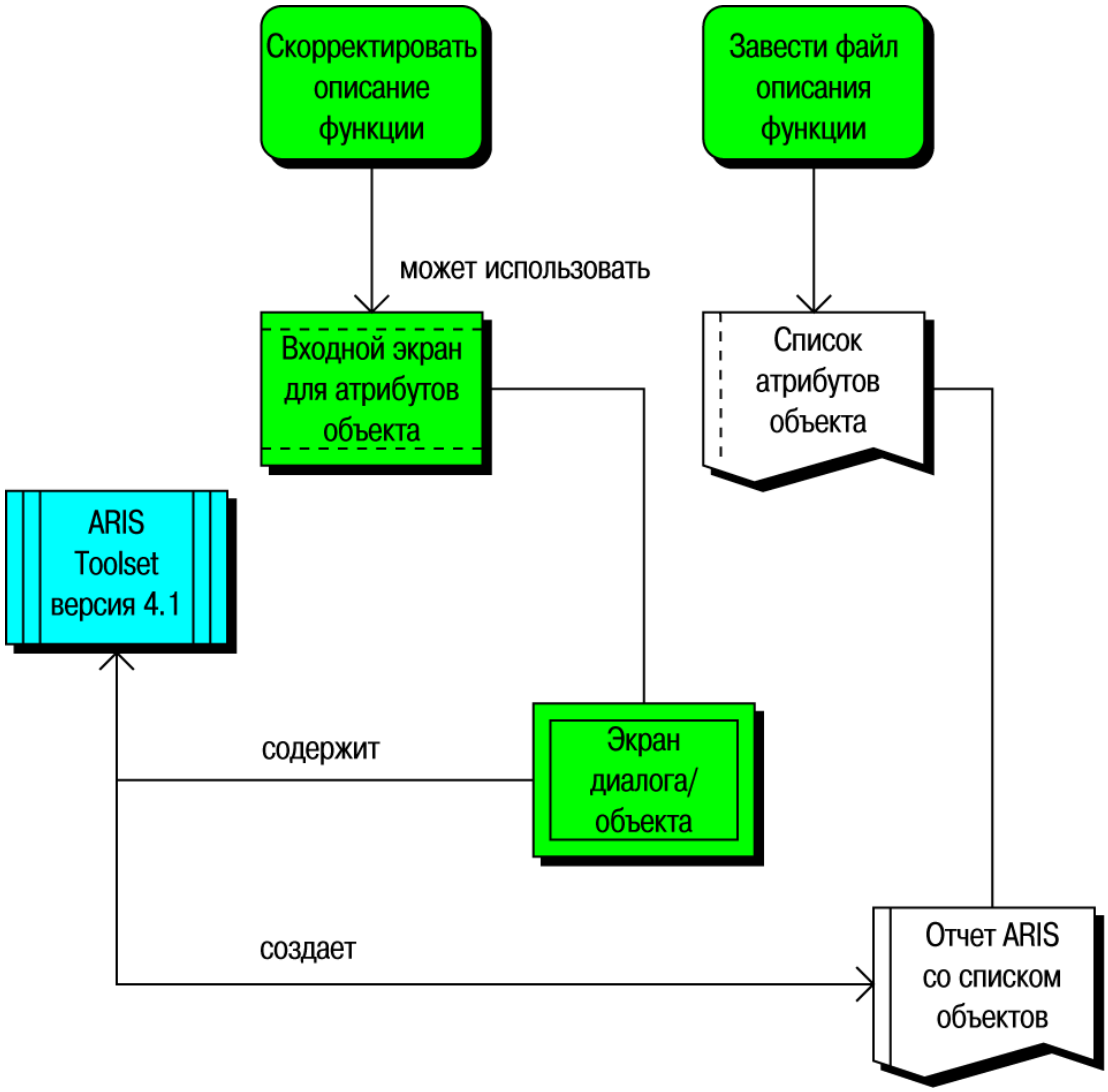


Рис. 4.1.2-6. Привязка экранов и списков

В гл. 9 английской версии руководства «Методы ARIS» содержится список всех типов объектов диаграммы типа ПС и возможные отношения.

4.1.3. Описание реализации. Диаграмма прикладной системы

С помощью диаграммы прикладной системы можно связать конкретные прикладные системы и модули с типами ПС и типами модулей, как это описано в разделе, посвященном спецификации проекта. Имеются в виду все копии одного типа ПС, которые могут быть, например, идентифицированы уникальным номером лицензии.

Определение

Прикладная система (модуль) – это единственный экземпляр типа ПС (типа модуля), который может быть уникально идентифицирован.

Графическое представление прикладных систем и модулей приведено на рис. 4.1.3-1.

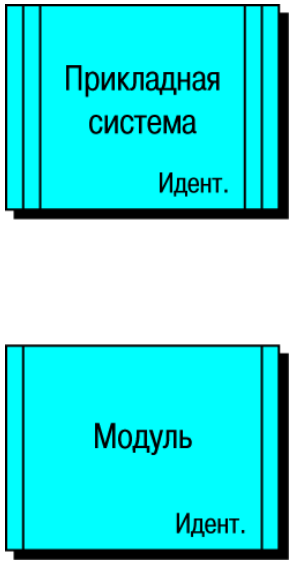


Рис. 4.1.3-1. Графическое представление прикладной системы и модуля

Если компания имеет несколько лицензий на один тип ПС (тип модуля), то на диаграмме прикладной системы несколько прикладных систем (модулей) могут соответствовать одному типу ПС (типу модуля). Это показано на рис. 4.1.3-2.

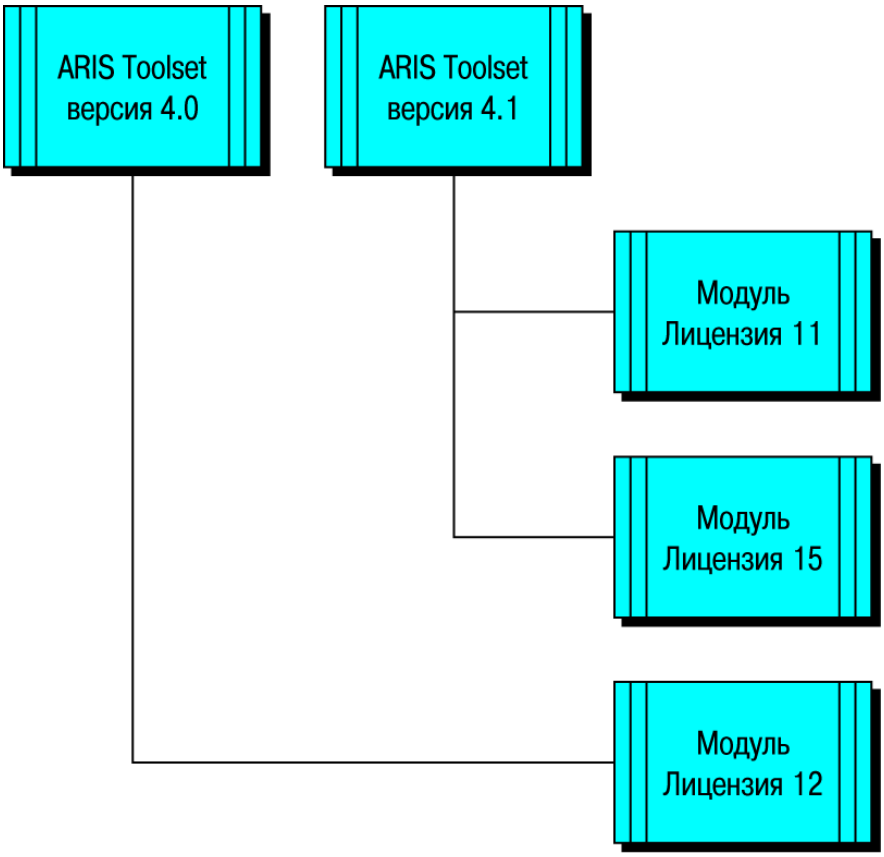


Рис. 4.1.3-2. Привязка прикладных систем к типам ПС

Диаграмма прикладной системы отражает фактическую модульную структуру прикладной системы. Хотя в спецификации проекта определяются все возможные модульные компоненты одного типа ПС, мы рассмотрим только одну лицензию прикладной системы с целью единообразно определить модульные элементы каждой лицензии. Следовательно, компания может использовать несколько прикладных систем одного типа ПС, с совершенно различной модульной структурой (см. рис. 4.1.3-3).

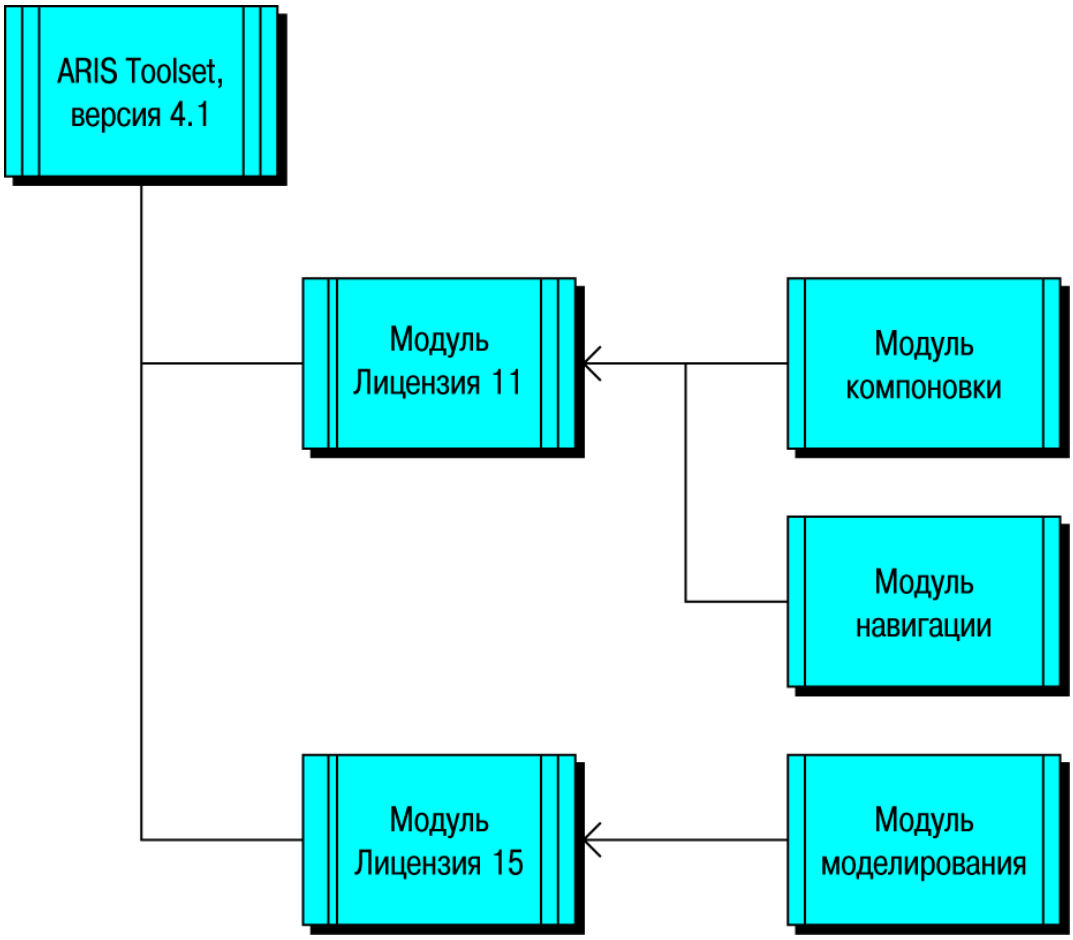


Рис. 4.1.3-3. Различная модульная структура двух прикладных систем одного типа

Кроме представления фактически имеющихся прикладных систем и модулей, на уровне описания реализации имеется возможность определить техническую (физическую) реализацию ПС как отдельных программных файлов.

Диаграммы ПС наглядно показывают какие типы программных элементов необходимы для реализации данного типа ПС и типа модуля.

Определение

Программный элемент – это отдельный программный файл, расположенный на носителе данных (например, файл EXE или COM), и приобретаемый вместе с лицензией. Типы программных элементов – это совокупность типов программных элементов, базирующихся на одной и той же технологии.

Рис. 4.1.3-4 иллюстрирует привязку типов программных элементов к типам ПС и отдельных программных элементов к типам программных элементов.

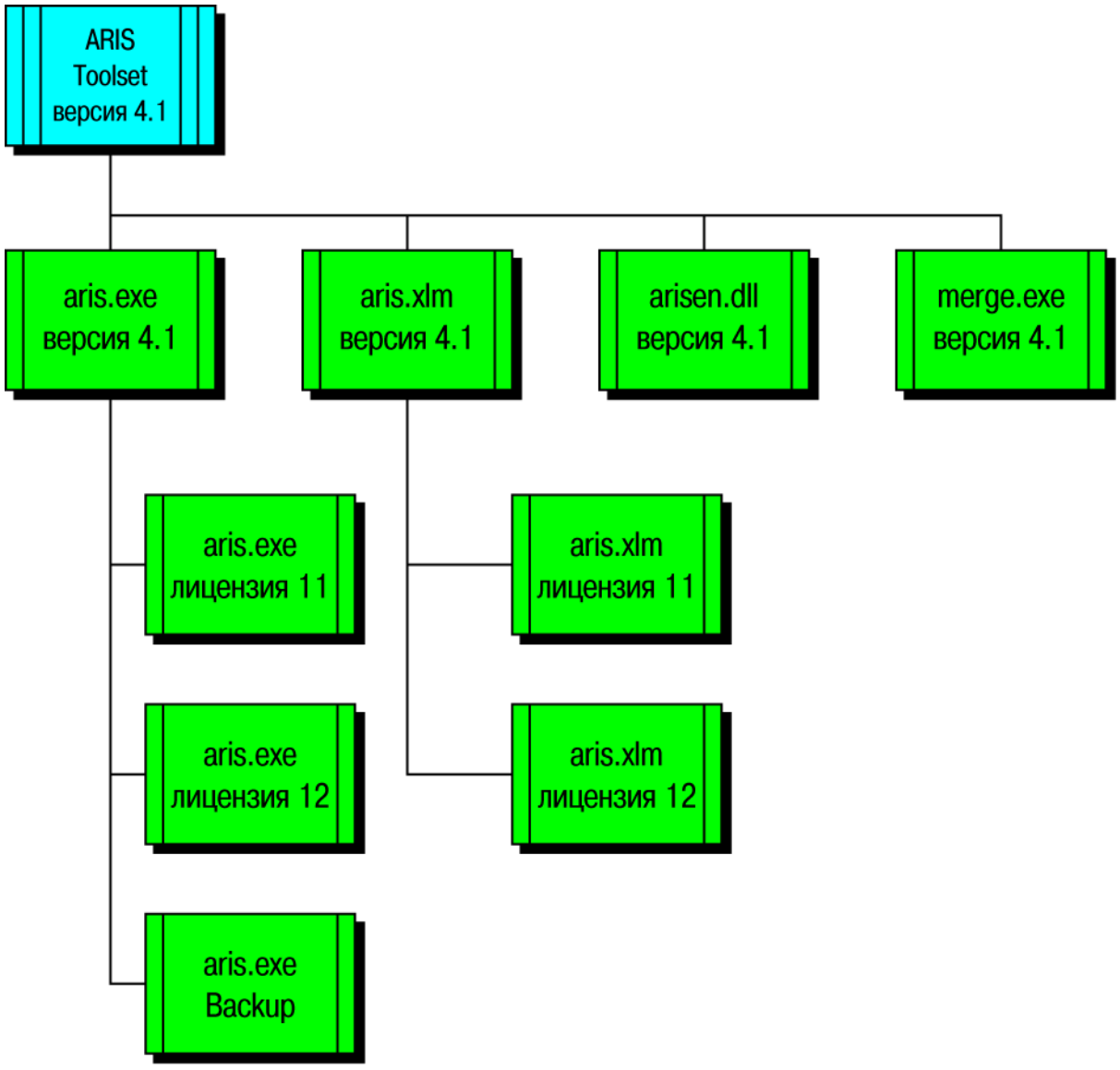


Рис. 4.1.3-4. Привязка типов ПС, типов программных элементов и программных элементов

Для типа ПС *ARIS Toolset, версия 4.1,* типами программных элементов являются *aris.exe*, *aris.xlm*, *arisen.dll* и *merge.exe*. Если в компании куплены многопользовательские лицензии или созданы резервные копии, то для каждого типа программного элемента могут быть доступны несколько программных элементов.

Типы программных элементов и программные элементы могут быть иерархически выстроены любым способом. В диаграмме ПС для более точной специализации конкретной программы можно использовать описания доступа типов программных элементов к библиотекам программ.

В гл. 9 английской версии руководства «Методы ARIS» содержится список всех типов объектов и типов отношений для диаграмм прикладных систем.

4.2. Модель данных

4.2.1. Формулировка требований

Формулировка требований в рамках модели данных включает описание семантической модели данных в рассматриваемой предметной области. В соответствии с принципом разделения ARIS это описание содержит как объекты, специфицирующие начальное и конечное события в цепочке процесса, так и описание состояний инфраструктуры, связанной с процессом.

При сравнении методов моделирования функций и данных к последним предъявляются особые требования с точки зрения применяемого метода. В функциональной модели единственный рассматриваемый объект – это функция. В терминах взаимосвязей между функциями описываются только отношения старшинства и подчиненности.

Модель сущность-отношение (ERM) Чена является наиболее широко распространненым методом создания семантических моделей. Этот метод моделирования оперирует такими терминами, как *тип сущности*, *атрибут* и т.д. Многочисленные взаимосвязи между этими объектами значительно сложнее классификации по сравнению с функциональным моделированием.

В последующих разделах дается введение в метод моделирования, базирующийся на модели сущность-отношение (ERM). В первую очередь рассматриваются объекты и взаимосвязи исходной модели Чена. Далее к исходной модели добавляются некоторые операции.

4.2.1.1. Базовая модель ERM

В исходной модели Чена различаются такие понятия, как сущность, атрибут и отношение. В общем случае уровень типа может отличаться от уровня экземпляра.

Определение

Сущность – это реальный или абстрактный объект, представляющий интерес для задач в конкретной области деятельности.

Рассмотрим в качестве примера бизнес-процесс в аспекте архитектуры ARIS. В соответствии со структурной моделью ARIS интересующие нас объекты данных – это

объекты инфраструктуры и объекты, специфицирующие события. В процессе *обработка заказа клиента* мы могли бы выделить следующие сущности:

- клиент 1235,
- изделие 4711,
- заказ 11.

Сущности более точно могут быть описаны с помощью определенных атрибутов (свойств). В данном случае, это означает, что клиент может быть более точно определен именем, фамилией и адресом.

Определение

Сущности одного и того же типа, сгруппированные в наборы, определяют тип сущностей. Экземпляром типа сущности является сущность.

Сущности одного и того же типа могут быть описаны одними и теми же атрибутами. Таким образом, клиент *Иванов* и клиент *Петров* вместе образуют сущность типа *клиент*, а изделие *4711* вместе с изделием *4712* образуют сущность *изделие*. Типы сущностей отображаются в модели ERM в виде прямоугольников (см. рис. 4.2.1-1). В последующем изложении типы сущностей обозначены прописными буквами.

Рис. 4.2.1-1. Типов сущностей

Определение

Атрибуты – это свойства, описывающие типы сущностей.

Экземпляры атрибутов – это фактические значения атрибутов, присвоенные отдельным сущностям. Например, клиент *1235* может быть описан посредством экземпляров атрибутов *Петров*, *Иван*, *Москва* и т.д. Соответствующими атрибутами здесь являются *фамилия*, *имя* и *город*.

Атрибуты обычно изображаются эллипсами или окружностями. Далее на рисунках атрибуты представлены эллипсами. На рис. 4.2.1-2 приведены атрибуты для типа сущности *клиент*.

Рис. 4.2.1-2. Атрибутов типа сущности «клиент»

Часто достаточно сложно определить различие между типами сущностей и атрибутами - это удается сделать, только исходя из контекста моделируемой процедуры. Например, адрес клиента может пониматься как сущность, а не как атрибут *клиента*. В этом случае можно было бы ввести новый тип сущности – *адрес*, который имел бы собственное отношение к сущности *клиент*.

Запомните, что сущности могут обладать атрибутами вне зависимости от того, имеете ли вы дело с типом сущности или атрибутом. С другой стороны, атрибуты не могут обладать атрибутами. Таким образом, если в модели ERM создается атрибут, и предполагается, что он будет в дальнейшем описан другими атрибутами, то атрибут становится типом сущности. Другой важный критерий при определении сущности – будет ли рассматриваемый объект взаимосвязан с другими типами сущностей или нет. Если такая связь предполагается, то этот объект также является типом сущности.

Определение

Отношение – это логическая связь между сущностями.

Следовательно, существование отношений прямо зависит от существования сущностей.

Определение

Отношения одного и того же типа, сгруппированные в наборы, определяют тип отношений.

Тип отношения между *поставщик* и *деталь* может быть *снабжает*. Здесь и в последующем тексте типы отношений обозначаются прописными буквами. В модели ERM типы отношений изображаются ромбами, которые соединены с типами сущностей (см. рис. 4.2.1-3).

Рис. 4.2.1-3. Тип отношений

Типы отношений часто могут читаться только в одном направлении, при котором связи имеют смысл. В приведенном выше примере предполагается, что отображено отношение, *поставщик снабжает деталями*. Справа налево можно было бы прочесть: *детали снабжают поставщика*, что не имеет смысла. Если верное направление нельзя определить однозначно, то преодолеть затруднение можно посредством выбора подходящих терминов, возможно - на более абстрактном уровне.

Теперь следует дифференцировать типы отношений. В этом контексте критериями отличия являются: число типов сущностей, которые соединены типом отношений, и степень сложности отношений.

Типы отношений различаются по числу типов сущностей, связанных с ними. Будем различать унарные, бинарные и n-арные отношения.

Определение

Степень сложности, или мощность отношения, указывает, сколько сущностей одного типа связаны с сущностью другого типа.

Отношения, образованные указанным образом, представлены на рис. 4.2.1-4.

Различаются четыре типа отношений:

- отношение 1:1,
- отношение 1:n,
- отношение n:1,
- отношение n:m.

При отношении 1:1 только одна сущность из второго набора соответсвует каждой сущности из первого набора. При отношении 1:n только одна сущность из второго набора связана с каждой сущностью из первого набора, но n сущностей из первого набора связаны с каждой сущностью из второго набора. Отношение n:1 отображает ту же ситуацию, но в обратном порядке. При отношении n:m несколько сущностей из второго набора связаны с одной сущностью из первого набора и наоборот.

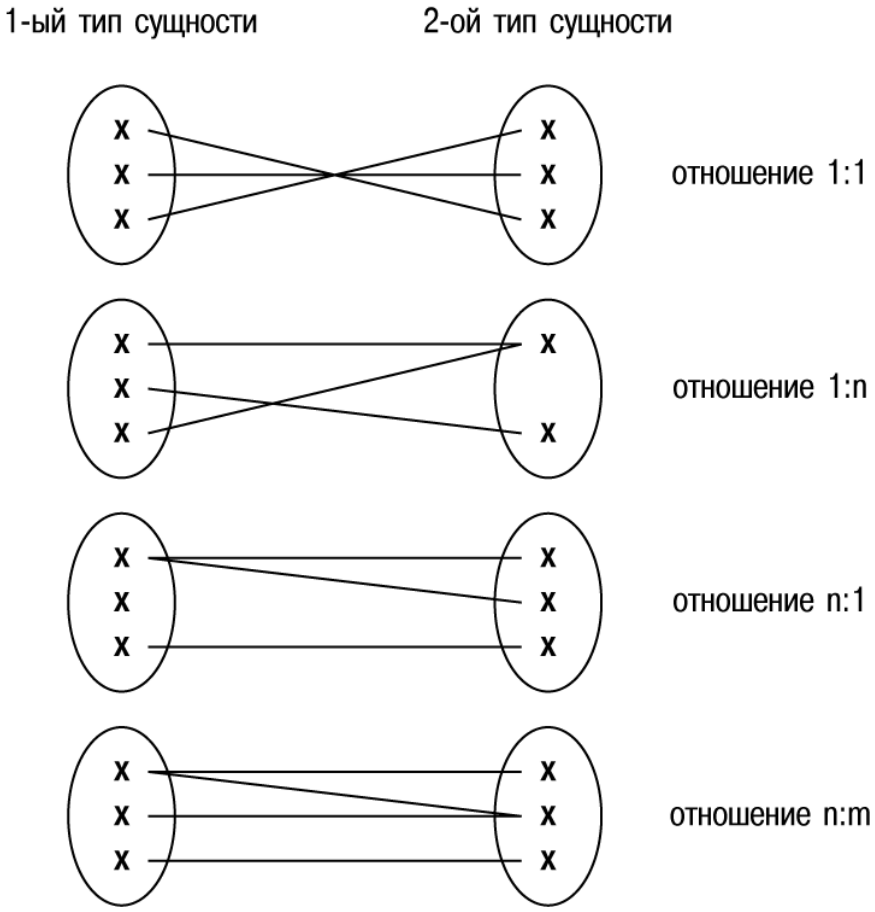


Рис. 4.2.1-4. Мощность отношений между двумя типами сущностей

Мощность типов отношений (тип атрибута *степень сложности*) представлена соединениями на диаграмме сущность-отношение (см. рис. 4.2.1-5).

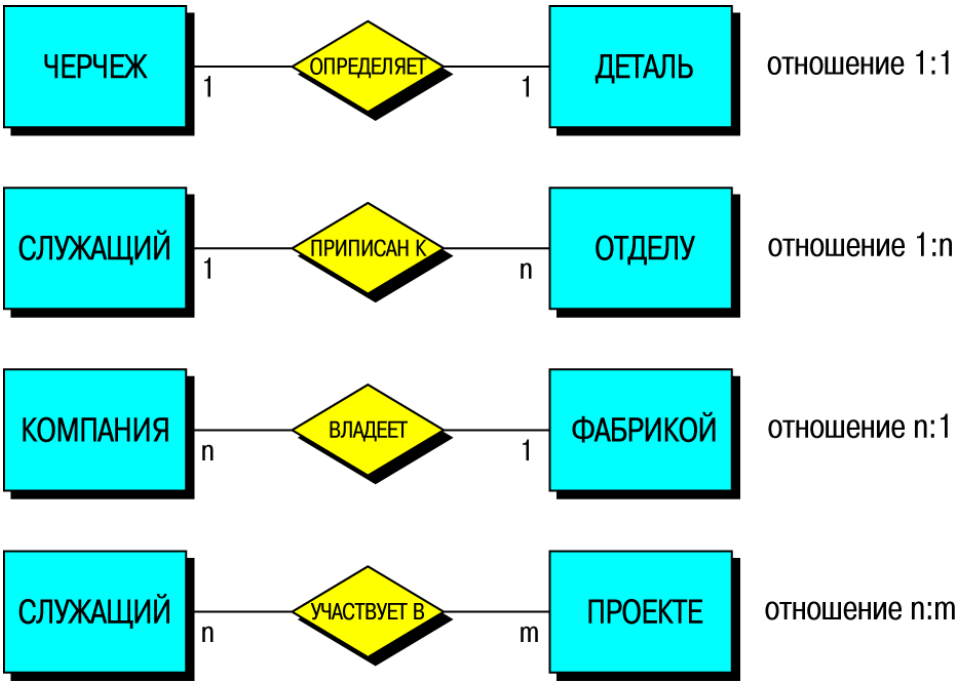


Рис. 4.2.1-5. Представление мощности типов отношений в модели ERM

Мощность определяет максимальное число принадлежащих некоторому типу отношений, в которых может участвовать одна сущность, принадлежащая к некоторому типу сущностей. Это означает, что для отношения n:1 на рис. 4.2.1-5 компания, имеющая тип сущности КОМПАНИЯ, может многократно находиться в различных отношениях ВЛАДЕЕТ, если эта компания имеет дело с различными фабриками. Однако каждая отдельная фабрика имеет только одно отношение типа ВЛАДЕЕТ: она может принадлежать только одной компании.

Следует отметить, что оригинальная работа Чена иначе интерпретирует мощность отношений. Нотация, принятая в этом руководстве, предоставляет более четкие формулировки, в частности в том случае, когда описываются отношения между более чем двумя типами сущностей. Для того чтобы избежать ненужной путаницы, в дальнейшем мы не будем обсуждать детали работы Чена.

Благодаря тому факту, что отношения могут существовать также между сущностями одного типа, тип сущности и тип отношения могут быть связаны двумя параллельными соединениями. Для того чтобы дифференцировать эти соединения, их можно связать отношением старшинства и подчиненности. Пример рекурсивных отношений приведен на рис. 4.2.1-6. *Главная деталь* состоит из различных *подчиненных деталей*. С другой стороны, подчиненная деталь может быть компонентой в различных главных деталях.

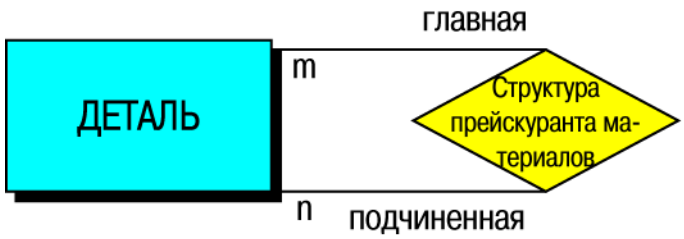


Рис. 4.2.1-6. Модель ERM для списка материалов

Посредством атрибутов описываются не только типы сущностей. Они же используются при описании типов отношений (см. рис. 4.2.1-7).

Определение

Диапазон значений атрибутов называется областью.

Соотнесение элементов области элементам сущности или типам отношений также описывается отношениями. Они могут быть представлены линиями, которые помечаются именами атрибутов.

Определение

Отношение типа 1:1 должно существовать между одним типом сущности и, по крайней мере, одной областью. Значения этой области могут однозначно идентифицировать отдельные сущности. Они называются ключевыми атрибутами данного типа сущности.

В примере, приведенном на рис. 4.2.1-7, отдельные сущности типа КЛИЕНТ однозначно идентифицированы ключевым атрибутом *номер клиента*.

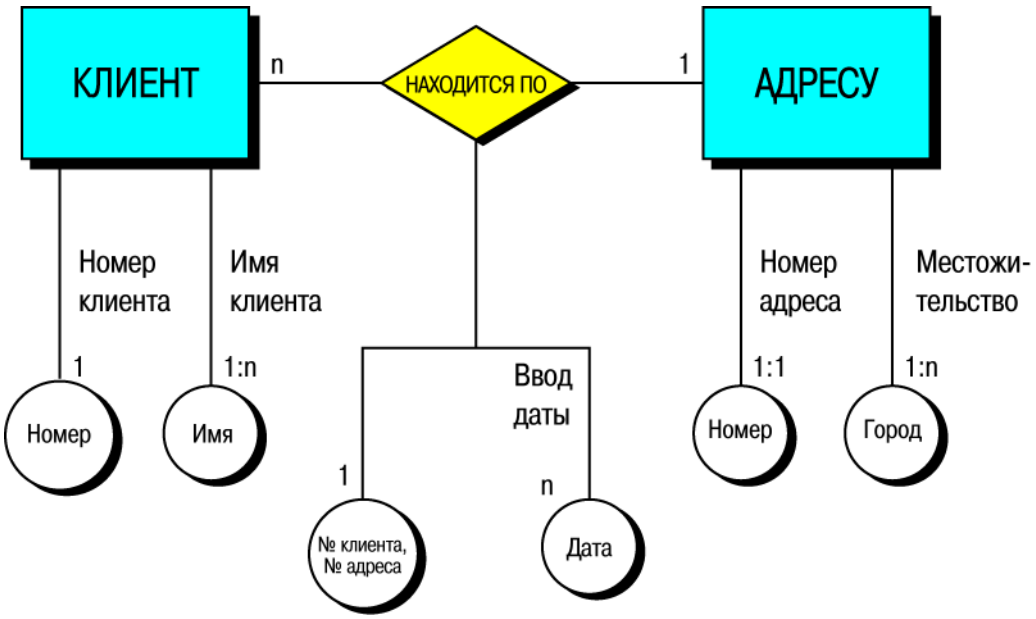


Рис. 4.2.1-7. Определение атрибутов в модели ERM

Отношения идентифицируются слиянием ключевых атрибутов связанных сущностей. Таким образом, ключевые атрибуты отношения типа *Находится по* – это номер клиента и номер адреса.

Описательные атрибуты соответствующих объектов данных определяются значениями, взятыми из областей, которые имеют мощность отношения 1:n с сущностью или типом отношения.

4.2.1.2. Расширенная модель «Сущность-отношение» (eERM)

В течение последних нескольких лет оригинальная модель Чена была значительно расширена. В данном руководстве будут рассматриваться такие расширенные модели, которые существенны для модели данных в архитектуре ARIS.

4.2.1.2.1. Расширение модели с помощью операторов проектирования

Операторы проектирования обеспечивают формальную поддержку процесса создания модели данных. Их применение гарантирует систематический подход и дает возможность тому, кто знает существующую структуру данных, понять суть процесса проектирования. С помощью операторов проектирования разработаны новые концепции, основанные на существующих. Процесс проектирования является интеллектуальной процедурой, в большей степени выполняемой на уровне управления

корпоративными знаниями. Анализ условий выполнения бизнес-процессов с точки зрения их структур данных помогает разработчикам, как структурировать известные условия, базируясь на новом представлении, так и создавать новые отношения, не рассмотренные до сих пор.

Опираясь на ряд различных подходов к расширению модели «сущность-отношение», можно выделить четыре основных оператора проектирования:

- классификацию,
- обобщение,
- агрегацию,
- группировку.

Классификация

Определение

При помощи классификации объекты (сущности) одного и того же типа идентифицируются и ассоциируются в соответствии с некоторым признаком (типом сущностей). Один объект идентичен другому, если он описан теми же свойствами (атрибутами).

Таким образом, классификация ведет к ранее описанной идентификации типов сущностей (см. рис. 4.2.1-8).

Рис. 4.2.1-8. Классификация клиентов

Обобщение/Специализация

Определение

При обобщении аналогичные типы объектов группируются под одним, более старшим типом объекта.

Как показано на рис. 4.2.1-9, тип сущности *Клиент* и тип сущности *Поставщик* объединены под общей концепцией *Участник бизнес-процесса*. Свойства (описанные атрибутами), которые являются общими для обоих исходных объектов, присваиваются обобщенному объекту. Таким образом, остается описать только такие атрибуты,

которые отличны от атрибутов исходных типов объектов. Образование нового типа сущности *Участник бизнес-процесса* представляется графически в виде треугольника, который также является отношением и означает *есть*.

Определение

Под специализацией понимается разделение некоторого общего множества (например, объектов) на подмножества.

Оператор специализации является инверсным по отношению к оператору обобщения (пример: *Участник бизнес-процесса* разделяется на *Клиент* и *Поставщик*). Специализированные объекты наследуют свойства обобщенных объектов. Кроме наследования, специализированные типы объектов могут обладать также собственными атрибутами. Графически специализация и обобщение представляются одинаково.

По указанной причине соединения на рисунке не отображаются стрелками, указывающими направление.

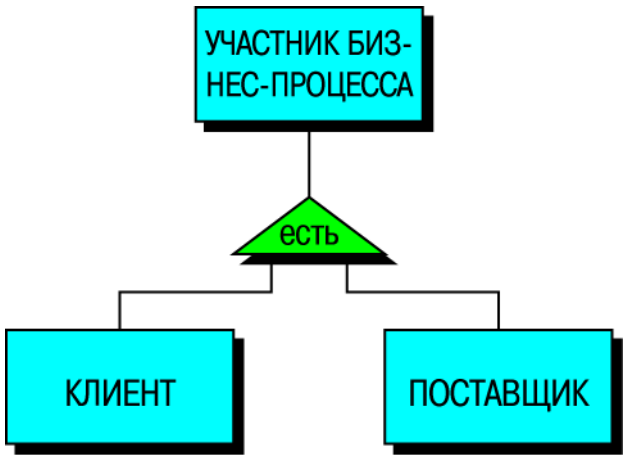


Рис. 4.2.1-9. Обобщение/Специализация

Специализация в первую очередь поддерживает поход к структуре данных «сверху-вниз», обобщение же используется при подходе «снизу-вверх».

В рамках специализации полнота и степень разделяемости на поднаборы могут определяться в процессе их создания.

О неразделяемых подмножествах будем говорить в том случае, когда экземпляр одного объекта может быть частью обоих подподмножеств. Для приведенного выше примера это означает, что некоторый клиент в то же самое время может быть и

поставщиком. Если экземпляр ассоциируется только с одним подмножеством, то множества являются разделяемыми.

Когда все специализированные типы объектов, возможные для одного критерия специализации, входят в состав одного обобщенного типа объекта, мы говорим о полной специализации. Возьмем, например, тип сущности *Человек*: он может быть разделен на типы сущностей *Мужчина* и *Женщина* (см. рис. 4.2.1-10). Специализация полностью определена, если мы в качестве критерия специализации выбираем *пол*.

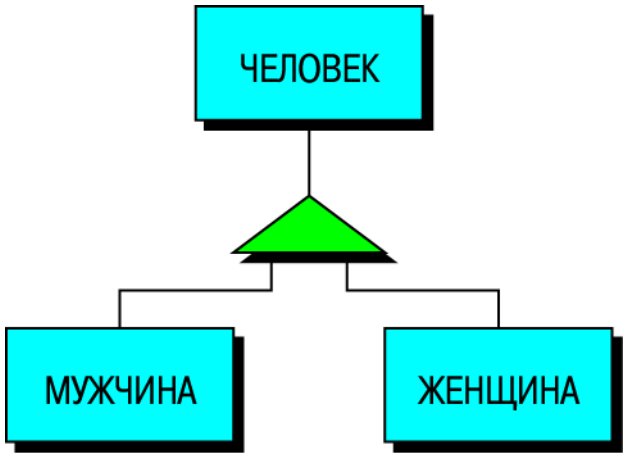


Рис. 4.2.1-10. Полная специализация

Комбинация этих критериев приводит в результате к следующим четырем случаям, которые выделяются, чтобы более точно определить оператор обобщение/специализация:

- раздельная/полная,
- раздельная/неполная,
- не раздельная/полная,
- не раздельная/неполная.

Агрегация

Определение

Агрегация описывает формирование нового типа объекта с помощью комбинации существующих типов объектов. В этом контексте новый тип объекта может нести новые свойства.

В eER-модели агрегация представляется как формирование типов отношений (см. рис. 4.2.1-11). Агрегация типов сущностей *Заказ на производство* и *Маршрутизация* создает новый объект *Маршрутизация заказа*.

Рис. 4.2.1-11. Пример агрегации

Оператор агрегации также применяется к отношениям. В этом случае существующий тип отношения трактуется, как тип сущности и таким образом может стать отправной точкой для создания другого, нового отношения (см. на рис. 4.2.1-12).

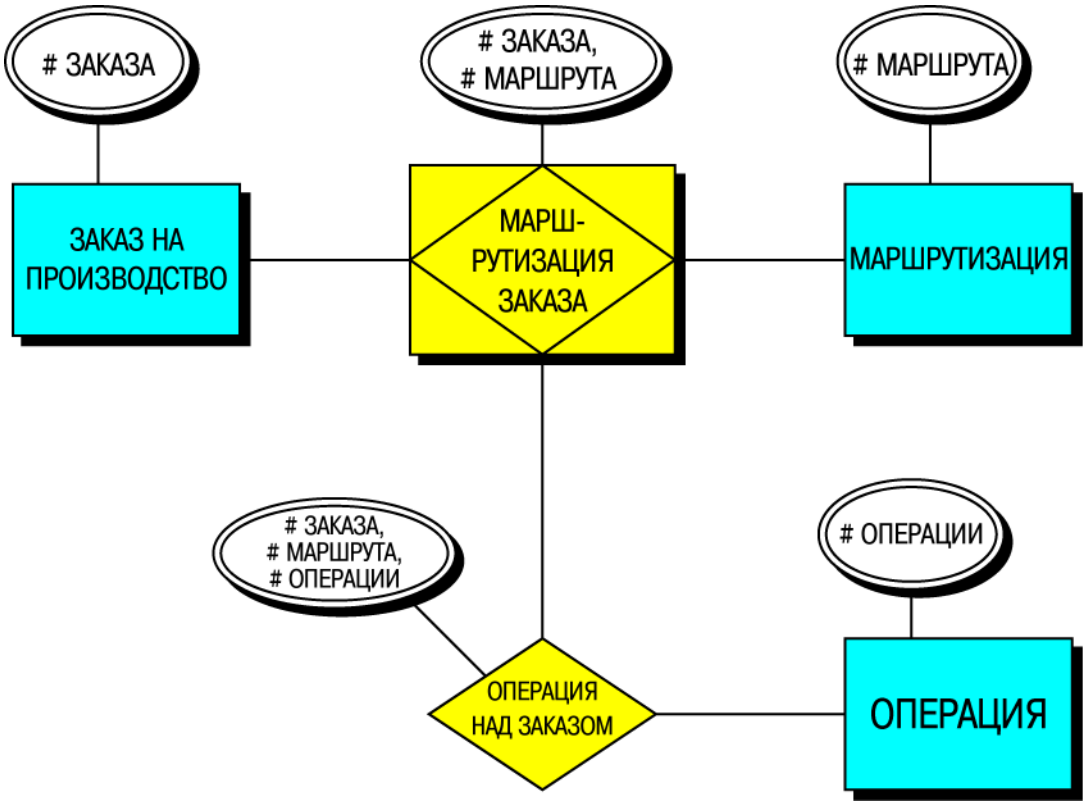


Рис. 4.2.1-12. Агрегация с реинтерпретацией типа отношений

Первая агрегация используется для формирования типа отношения *Маршрутизация заказа* из типов сущностей *Заказ на производство* и *Маршрутизация*. Ключевые атрибуты *номер заказа на производство* и *номер маршрута* образуют сложный ключевой атрибут для отношения к *Маршрутизации заказа*. Теперь многочисленные операции могут быть связаны с *Маршрутизацией заказа*. Следовательно, отношение *Операция над заказом* формируется между типами отношений *Маршрутизация заказа* и типом сущности *Операция*. Поскольку отношения могут создаваться только между типами сущностей, необходимо, чтобы первоначальный тип отношения *Маршрутизация заказа* был интерпретирован как тип сущности. На рис. 4.2.1-12 это иллюстрирует ромб, заключенный в прямоугольник.

В конце концов, этот реинтерпретированный тип отношения трактуется как *нормальный* тип сущности. Процедура создания типа отношения из нескольких существующих типов сущностей отображается с помощью линий, направленных к вершинам ромба. Однако линии, означающие новые отношения по ре-интерпретированным типам отношений, подходят к краям прямоугольника, окружающим ромб, а не к самому ромбу.

Несмотря на то, сложный ключ можно заменить более простым, мы будем использовать сложные ключи, поскольку они позволяют лучше отслеживать процесс создания модели данных.

В ER-модели сложный структурный элемент разделяется таким образом, чтобы образовать прозрачную структуру. Чтобы облегчить понимание и не нарушить стройность концепции вводят сложные объекты в виде кластеров данных.

Определение

Кластер данных описывает логическую модель некоторых типов сущностей и отношений в модели данных. Это необходимо для описания сложных объектов.

Кластеры данных состоят не только из типов сущностей и отношений, но и из других кластеров данных. В отличие от типов сущностей и отношений кластеры данных могут быть легко встроены в иерархию, что позволяет в процессе создания модели данных поддерживать подход «сверху-вниз». Использование кластеров данных может быть полезно и при объединении подмоделей в ходе проектирования «снизу-вверх».

На рис. 4.2.1-13 представлен графический элемент кластера данных. Как показано на рис. 4.2.1-14, кластер данных – это логическая модель некоторых типов сущностей и отношений. Для описания сложного объекта *Заказ клиента* необходимы типы сущностей и отношений *Клиент*, *Время*, *Предмет заказа*, *Изделие* и *Позиция заказа*.

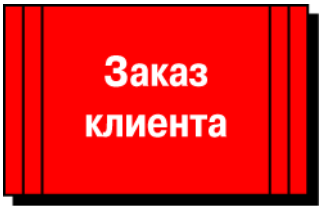


Рис. 4.2.1-13. Кластер данных (графическое представление)

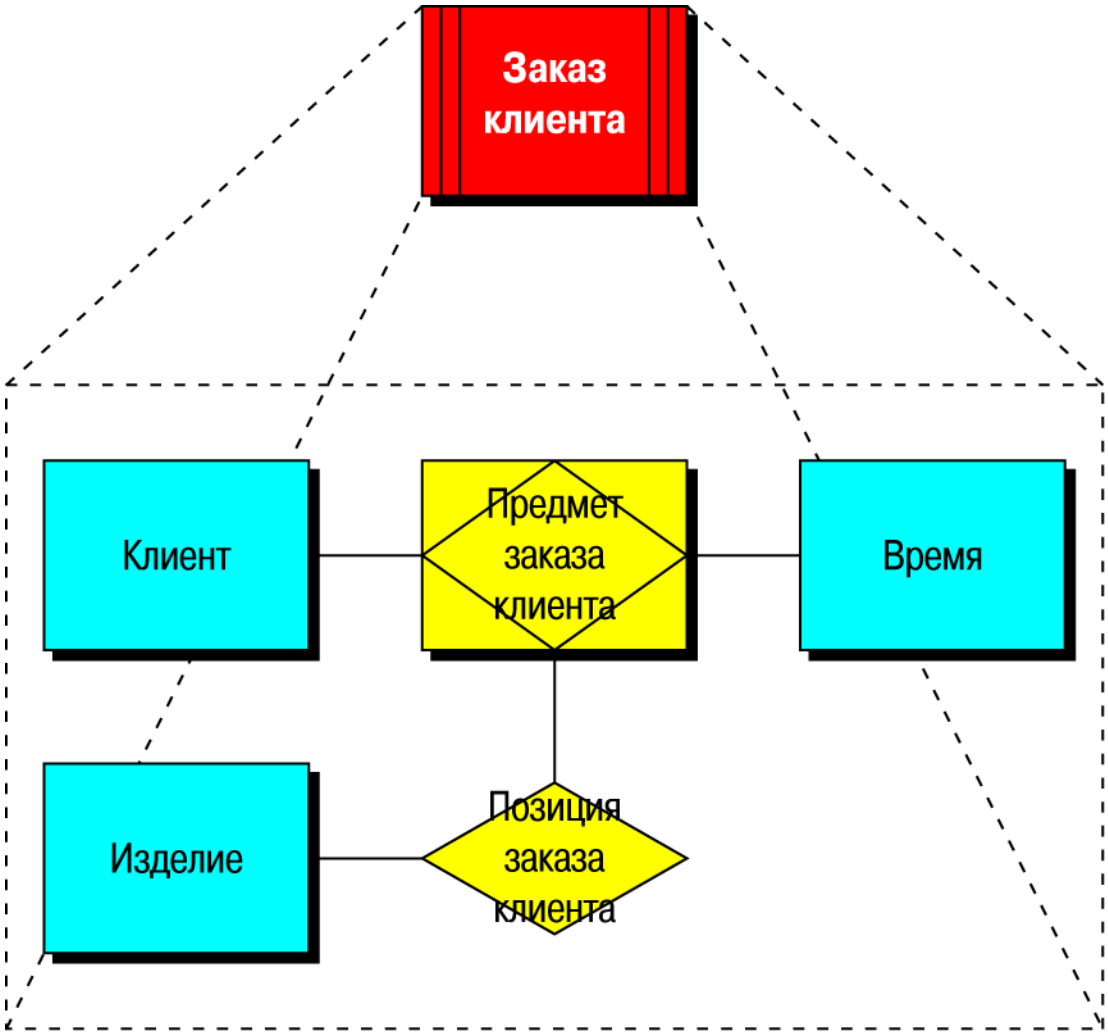


Рис. 4.2.1-14. Кластера данных для нескольких объектов

Группировка

Определение

В процессе группировки формируются группы, составленные из элементов некоторого множества сущностей.

В примере, на рис. 4.2.1-15 все составные части *Оборудования* объединяются в *Группу оборудования*. *Группа оборудования* – это независимый объект, который может быть описан более точно с помощью дополнительных атрибутов (наименование группы оборудования, число деталей оборудования), которые не характеризуют отдельные части оборудования. Другим примером модет служить группировка рабочих мест в отделы или объединение элементов, связанных с обработкой заказа, в группу *Заказ*.

Рис. 4.2.1-15. Группировка

4.2.1.2.2. Расширенное понятие мощности

При ссылке на число, характеризующее мощность отношений, мы упоминали только про верхние пределы допустимого числа экземпляров отношений. Мощности на рис. 4.2.1-16 указывают, что проект может быть связан с максимальным числом (m) служащих и один служащий может участвовать в максимальном числе (n) проектов.

Рис. 4.2.1-16. Верхнее/нижнее ограничение (1)

Кроме верхнего ограничения, может также представлять интерес нижнее ограничение в том случае, когда нужно специфицировать минимальное число экземпляров отношений. Для этих целей мощность отношений может быть выражена, например, двумя буквами (a,b). Две буквы (a1, b1) на рис. 4.2.1-17 указывают, что для каждого проекта можно определить, **по крайней мере,** a1 и **не более чем** b1 экземпляров отношений типа *Работать над*. Это означает, что каждый проект может

выполняться, **по крайней мере,** a1 и **не более чем** b1 сотрудниками. Две буквы (a2, b2) указывают, что один сотрудник может участвовать, **по крайней мере,** в a2 и **не более чем** в b2 проектах.

Рис. 4.2.1-17. Верхнее/нижнее ограничение (2)

Таким образом, каждое отношение выражается двумя степенями сложности (min - минимальной и max - максимальной). Нижним границам часто присваиваются значения 0 или 1. Диапазон значений верхней границы определяется как 1 ≤ max ≤ * (где * - универсальный знак).

Нижняя граница min = 0 указывает, что сущность может участвовать в одном отношении, но это необязательно. Нижняя граница min = 1 указывает, что сущность должна участвовать, по крайней мере, в одном отношении.

Нижние границы на рис. 4.2.1-18 указывают, что сотрудник может участвовать в проекте, но это необязательно (min = 0), в то время как проект должен выполняться, по крайней мере, одним сотрудником (min = 1). Этим здесь подчеркивается, что могут быть сотрудники, которые не участвуют ни в одном проекте. И наоборот, по крайней мере, по одному сотруднику должно быть прикреплено к каждому проекту.

Рис. 4.2.1-18. Верхнее/нижнее ограничение (3)

Если разрешены минимальное значение 0 или 1 и максимальные значения 1 или *, то для пары (min,max) возможны следующие четыре сочетания: (1,1), (1,m), (0,1) и (0,m).

Приведенные ниже сокращения являются обычными для указанных сочетаний:

- 1 (соответствует (1,1)),
- c (соответствует (0,1)),
- m (соответствует (1,m)),

- cm (соответствует (0,m)), где c = *choice* (выбор) и m = *multiple* (многократный).

На рис. 4.2.1-19 представлен пример рис. 4.2.1-18 с использованием указанной нотации.

Рис. 4.2.1-19. Верхнее/нижнее ограничение (4)

4.2.1.2.3. Идентификация и существенная зависимость

Благодаря расширению понятия мощности посредством определения нижней и верхней границ, как это обсуждалось в разделе 4.2.1.2.2, может быть рассмотрена зависимость между объектами данных.

По определению, типы отношений и типы реинтерпретированных отношений существуют в силу существования типов сущностей, связанных с ними; таким образом, они не существуют в изоляции. Это означает, что они зависимы от других типов сущностей, как с точки зрения существования, так и в плане идентификации.

Кроме того, существуют типы сущностей, которые в действительности обладают единственным ключевым атрибутом, но при этом зависят от существования других сущностей. Эти типы зависимостей могут появляться, например, при групповых операциях. Как показано на рис. 4.2.1-20, отдел имеет смысл только в том случае, если он содержит, по крайней мере, одно рабочее место. В свою очередь рабочее место только тогда имеет смысл, когда оно входит в отдел. Эти зависимости существования выражаются степенью сложности, или мощностью. В нотации (min,max) они определяются как (1,1) и (1,*).

Рис. 4.2.1-20. Зависимость существования

Определение зависимости существования в модели данных обеспечивает целостность основных данных, что является важным условием при реализации. Это означает, что выполнение этого условия гарантирует, что целостность содержимого

базы данных обеспечивается даже после выполнения некоторых транзакций. В приведенном выше примере удалить отдел можно только в том случае, если будут удалены все рабочие места, входящие в данный отдел.

4.2.1.2.4. Моделирование технических терминов, используемых в компании. Модели технических терминов (Technical Term Models)

В моделировании, особенно в моделировании данных, мы вынуждены иметь дело с часто встречающимся затруднением - многочисленностью терминов, определяющих информационные объекты в больших компаниях. Например, то, что понимается под термином *заказ* в отделе закупок, полностью отличается от того, что под этим подразумевают сотрудники производственного отдела. При введении соответствующей терминологии для компании и ее отделов определяемая информация становится более понятной.

По этой причине набор методов ARIS содержит так называемые модели технических терминов, которые не только позволяют манипулировать различными терминами как синонимами, но и дают возможность поддерживать отношения между объектами в моделях данных (тип сущности, тип отношения и т.д.) так же, как технические термины, применяемые в компании.

Для представления этих отношений вводится тип объекта *технический термин*. Теперь с каждым информационным объектом модели данных могут быть связаны разные технические термины (см. рис. 4.2.1-21).

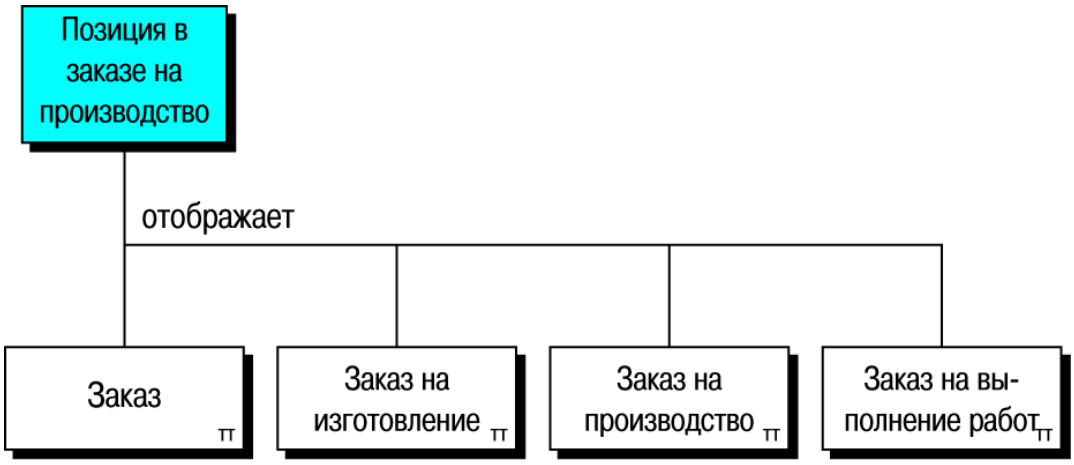


Рис. 4.2.1-21. Технические термины

Технические термины могут быть взаимосвязаны и иерархически упорядочены.

Термины, определяемые моделью технических терминов, могут использоваться и в других диаграммах, которые содержат информационные объекты, например, в диаграммах процессов для представления входа/выхода данных для функции.

4.2.1.2.5. Диаграмма атрибутов eERM (eERM attribute allocation diagram)

Модели данных типа eERM (расширенные модели «сущность-отношение»), которые отображают только типы сущностей и типы отношений, очень часто имеют довольно сложную структуру. Диаграммы, включающие атрибуты сущностей и отношений, теряют свою наглядность.

С помощью диаграмм атрибутов eERM можно описать атрибуты для каждого типа сущности и отношения на отдельной диаграмме. В эту диаграмму можно включить тип объекта из диаграммы eERM (тип сущности или тип отношения) в виде копии экземпляра. Таким образом, может быть смоделировано распределение атрибутов по объектам eERM. В этом контексте можно различать, является ли атрибут, связанный с объектом eERM, ключевым атрибутом, внешним ключом или описательным атрибутом. На рис. 4.2.1-22 приведен соответствующий пример.

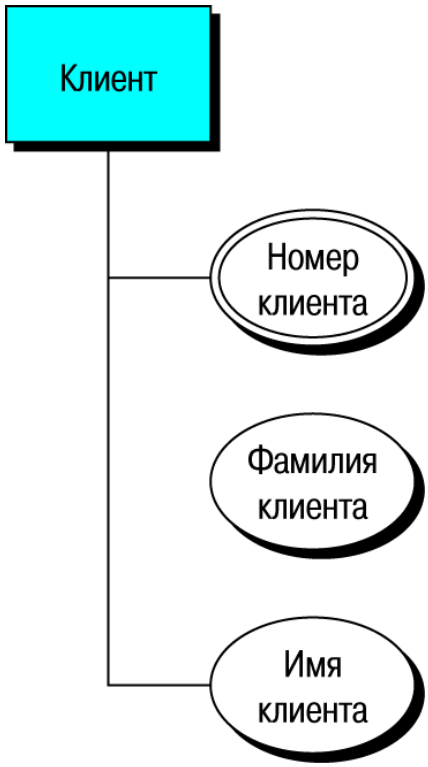


Рис. 4.2.1-22. Атрибуты eERM для типа сущности

Кроме представления и распределения отдельных атрибутов eERM, на этом типе диаграммы можно также отобразить группу типов атрибутов и их распределение.

Определение

Группа типов атрибутов представляет группу атрибутов одного типа сущности из диаграммы eERM, которые семантически близко связаны. Это позволяет создать группу атрибутов, содержащую все атрибуты eERM, которые вместе образуют, например, вторичный ключ.

Группы типов атрибутов могут быть представлены, так как показано на рис. 4.2.1.23.

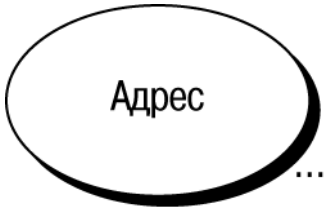


Рис. 4.2.1-23. Группа типов атрибутов

В гл. 9 английской версии руководства «Методы ARIS» содержится полный набор всевозможных отношений для диаграммы атрибутов eERM.

4.2.1.3. Альтернативные формы представления

4.2.1.3.1. SAP — SERM

Кроме рассмотренного представления, обычно используются и другие представления модели ERM. Типы отношений часто не представляются в виде отдельных объектов; они скорее указываются как соединения между типами сущностей. Затем вводятся типы отношений, обладающие собственными атрибутами, которые называют слабыми типами сущностей.

Одним из примеров таких форм представления является подход SERM (структурная модель «сущность-отношение»), разработанная Синцом. С помощью визуализации первичных и зависимых объектов данных направление развития структур данных становится прозрачным, причем для визуализации используются направленные графы, а объекты модели упорядочиваются слева (сильные типы сущностей) направо (слабые типы сущностей и типы отношений). Таким образом, основное различие между моделями SERM и моделями, базирующимися на eERM-методе, описанном в разделе 4.2.1.2., состоит в графическом представлении.

Метод моделирования, разработанный SAP AG, и информационная модель, базирующаяся на этом методе, связывает концепцию модели SERM с подходом eERM.

В этом контексте при создании объекта не существует какого-либо графического различия между типами сущностей и типами отношений. Зависимости между информационными объектами отражаются с помощью сложных ссылок, представленных стрелками. Однако между иерархическим, агрегатным и референтным отношениями графическое различие существует (см. рис. 4.2.1-24).

Иерархические отношения отображают односторонние существенные зависимости между информационными объектами.

Агрегатные отношения соответствуют образованию типов отношений, базирующихся на подходе eERM.

Референтные (ссылочные) отношения описывают логические зависимости между реинтерпретированными типами сущностей и первичными типами сущностей, аналогично примеру, рассмотренному нами при описании моделей.

Специализация обозначается треугольником - по аналогии с моделью eERM. Для того чтобы передать идею метода моделирования, на рис. 4.2.1-24 приведен пример для модели eERM вместе с обозначениями для модели SAP-ER. Это ясно показывает, что фундаментальное представление о предмете может быть передано из одной формы представления в другую без какой-либо потери информации.

После того как процедура создания модели данных завершена, модель SAP-ER является одной из представляющих ее форм. Благодаря графически-ориентированному упорядочиванию в модели SAP-ER информационных объектов, обеспечивается более быстрая навигация и ориентация, особенно необходимая при сложных моделях данных.

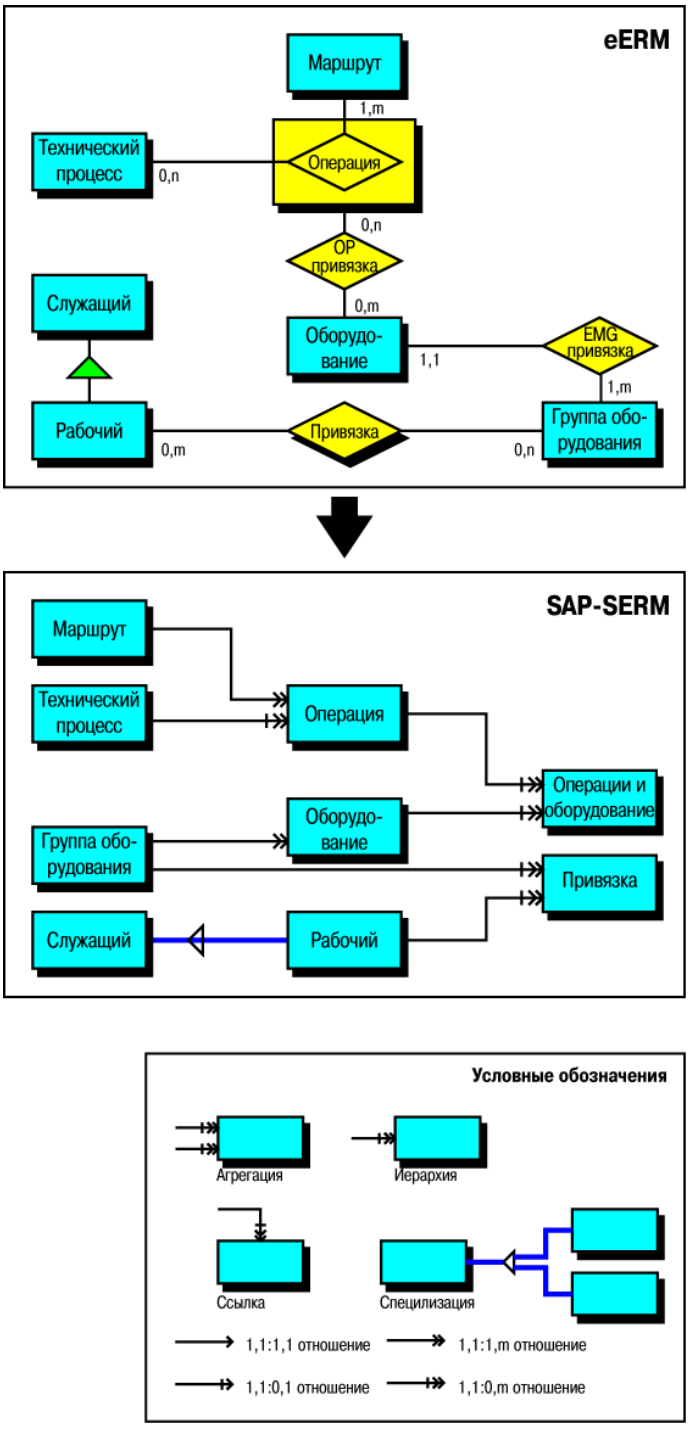


Рис. 4.2.1-24. Представление моделей eERM и SAP-ERM

4.2.1.3.2. Модель данных IEF

Модель данных IEF соответствует обозначениям, принятым при моделировании данных с помощью инструментария CASE Information Engineering Facility™ (IEF) фирмы Texas Instruments Inc.

Так же как и нотация SAP-SERM, нотация IEF не поддерживает типы объектов при представлении отношений между типами сущностей.

На рис. 4.2.1-25 представлена модель данных в нотации IEF.

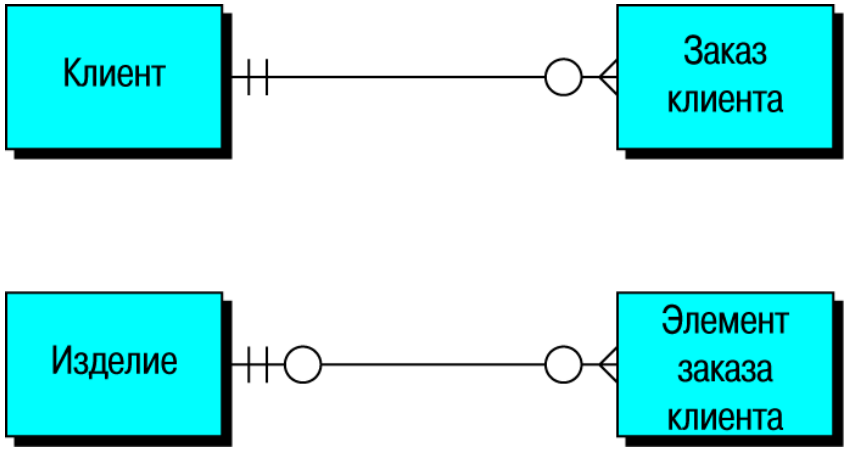


Рис. 4.2.1-25. Модель данных в нотации IEF

4.2.1.3.3. Модель SeDaM

Нотация семантической модели данных SeDaM (Semantic Data Model) – это нотация BASF AG. Она также не поддерживает типы объектов для представления отношений между типами сущностей. Типы сущностей последовательно не упорядочиваются слева направо (см. нотацию SAP-SERM). Типы объектов *кластер данных* и *обобщенный тип* также доступны.

На рис. 4.2.1-26 представлена модель данных в нотации SeDaM.

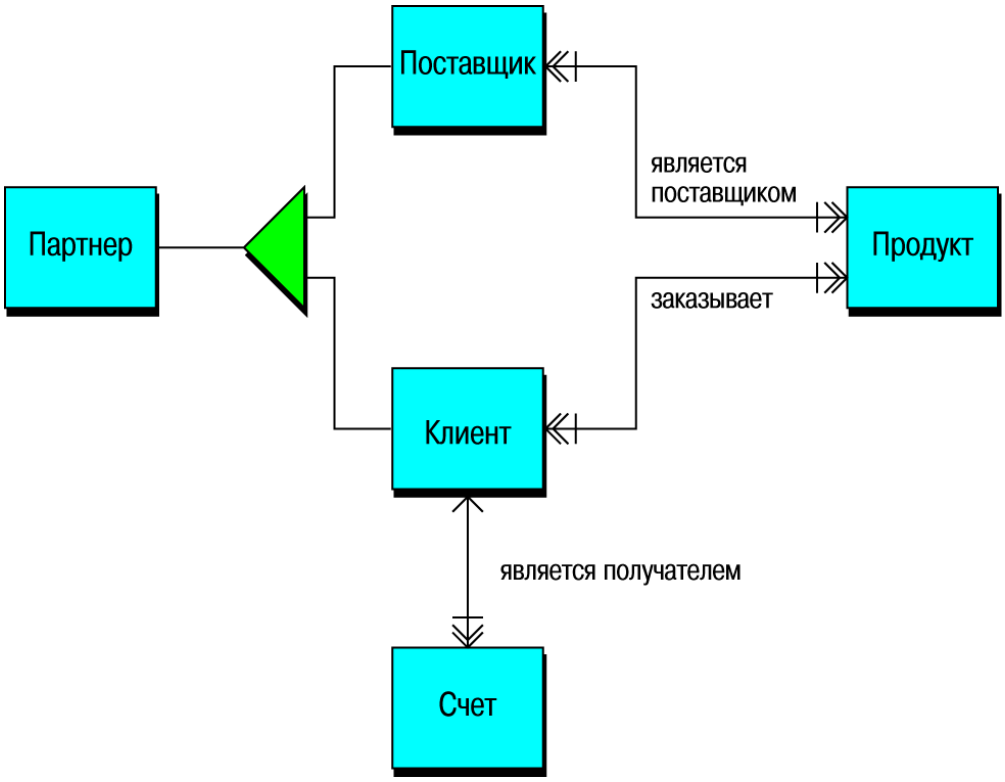


Рис. 4.2.1-26. Модель данных в нотации SeDaM

В гл. 9 английской версии руководства «Методы ARIS» содержится суммарный перечень всевозможных отношений для модели SeDaM.

4.2.1.4. Резюме по наиболее важным концепциям и формам представления модели eERM

На рис. 4.2.1-27 суммированы концепции и формы представления, характерные для наиболее важных структурных элементов и операций при проектировании расширенной модели «сущность-отношение» (eERM).

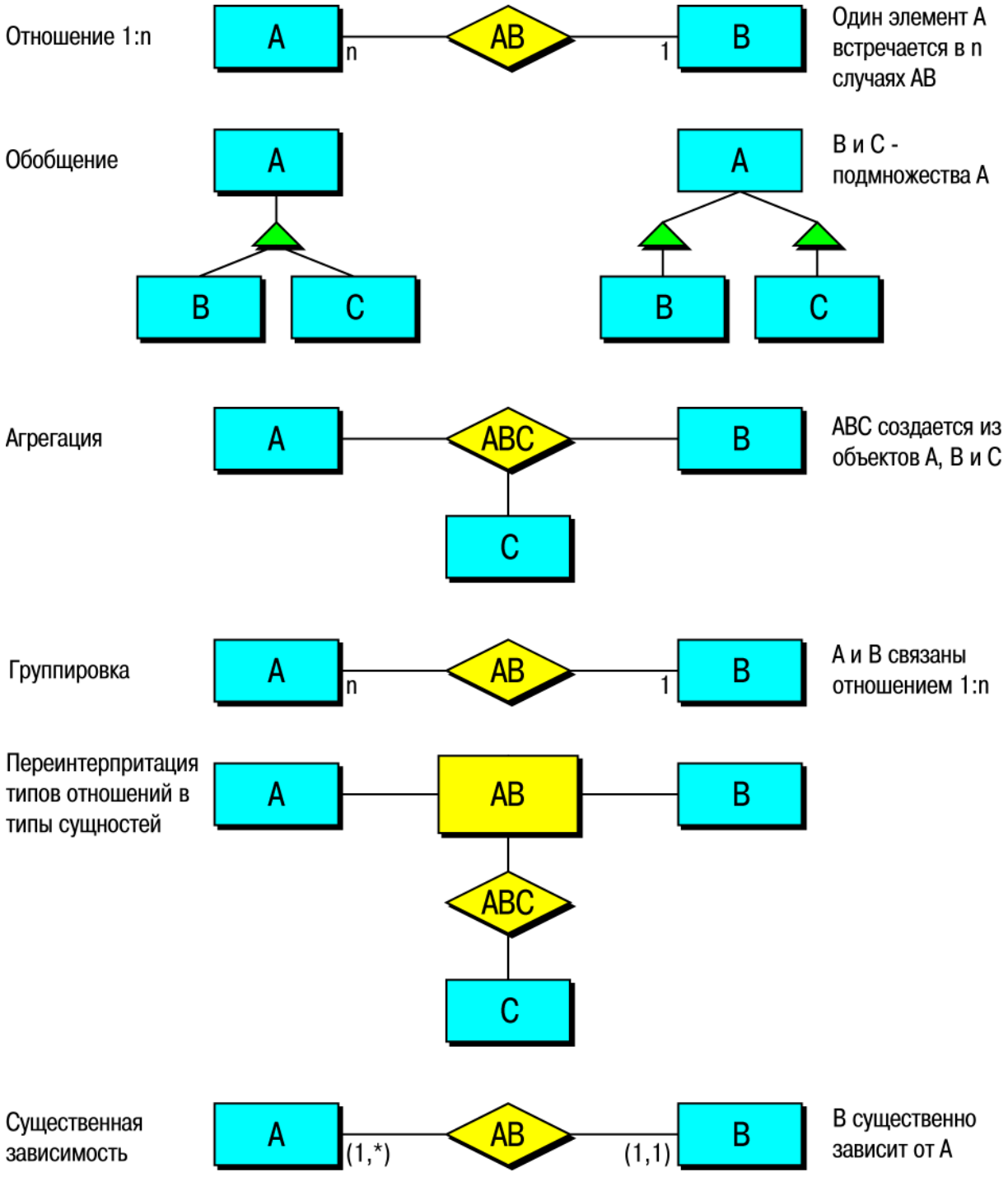


Рис. 4.2.1-27. Концепции и формы представления в модели eERM

4.2.1.5. Моделирование потоков материалов. Диаграмма материалов

Поток материалов отображается в моделях процессов (модели eEPC и PCD с потоками материалов) с помощью отдельных функций бизнес-процессов в виде входа/ выхода. Аналогично связи между информационными объектами и функциями (преобразование информации описывается функциями) связь материального потока и функций отражает преобразование типов материалов на входе в типы материалов на выходе.

С помощью диаграммы материалов можно определять типы материалов, упорядочивать их в соответствии с иерархией и классифицировать по классам материалов.

Определение

Тип материала – это совокупность материалов, имеютщих одинаковые характеристики.

Определение

Типы материалов могут объединяться, образуя класс материалов. При таком объединении проблема схожести (т.е. отнесение материала к некоторому классу) решается с помощью различных критериев классификации. Другими словами, один тип материала может соответствовать нескольким классам материалов.

Типы материалов могут быть связаны с типами упаковочных материалов. Это означает, что определенные типы материалов могут перемещаться, только если используется конкретный тип упаковочных материалов.

Упаковочный материал также может быть определен, классифицирован и иерархически упорядочен. Это позволяет, например, описать структуру и ограничения сложных упаковочных товарных комплексов.

Определение

Тип упаковочного материала - это совокупность отдельных упаковочных материалов, имеющих одинаковые характеристики.

Определение

Типы материалов могут быть объединены в классы упаковочных материалов. При таком объединении используются различные критерии классификации. Другими словами, один тип упаковочного материала может соответствовать нескольким классам материалов.

На рис. 4.2.1-28 приведена диаграмма материалов, включающая иерархию уровней и классификацию.

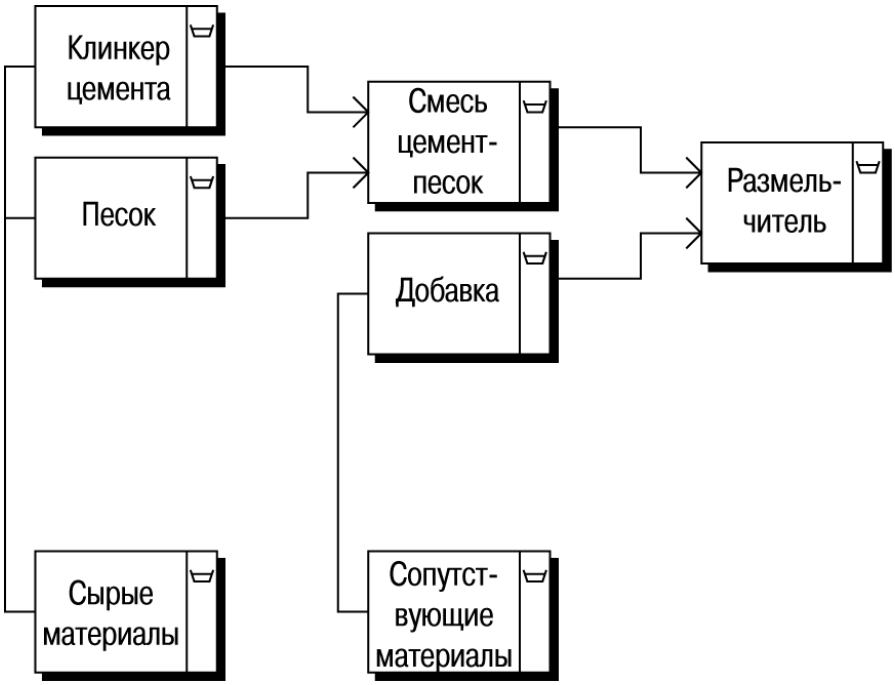


Рис. 4.2.1-28. Диаграмма материалов

4.2.1.6. Модель данных стоимостей, вычисленных по методу ABC

4.2.1.6.1. Диаграмма стоимостных драйверов (СД)

Область применения диаграмм СД связана с определением стоимости по методу ABC – операционное исчисление стоимости (см. ARIS ABC). Диаграмма СД описывает иерархию стоимостных драйверов.

Определение

Стоимостной драйвер – это имеющий определенный смысл элемент некоторой измеренной/референсной величины, предназначенный для оценки стоимости отдельных операций (функций) в рамках бизнес-процесса. Референсная величина представляет собой значение, которое получено из доступных источников информации и находится в постоянной зависимости (пропорции) от стоимостных оценок.

Стоимостные драйверы определяются только для процессов, производительность которых зависит от объемов, или индуктированных процессов. Стоимостные драйверы

не могут быть определены для таких процессов, как «Управления департаментом». Примером стоимостного драйвера может служить «Длина улицы» в процессе «Покрытие улицы асфальтом».

Иерархия стоимостных драйверов представляется в диаграмме СД с помощью направленных соединительных линий, имеющих тип «Определяет объем». Атрибуты *Числитель отношения СД* и *Знаменатель отношения СД* должны сопровождать эту связь. Если не определен *Знаменатель отношения СД*, то его значение принимается за 1. Частное от деления указанных атрибутов определяет количество связей между обоими стоимостными драйверами для вычисления стоимостных характеристик процесса.

Диаграммы СД представлена на рис. 4.2.1-29. В приведенном примере имеются два стоимостных драйвера *Количество автомобилей (лимузинов)* и *Количество дверей.* Чтобы показать, что каждый лимузин имеет 4 двери, атрибут *Числитель отношения СД* должен установить значение 4 на линии, соединяющей стоимостной драйвер *Количество автомобилей (лимузинов)* со стоимостным драйвером *Количество дверей*.

Рис. 4.2.1-29. Диаграмма стоимостных драверов

Стоимостные драйверы связаны с конкретным процессом через управляющую модель. Множитель для каждой функции в рамках процесса определяется автоматически в соответствии с иерархией стоимостных драйверов.

4.2.1.6.2. Диаграмма стоимостных категорий

Область применения диаграммы стоимостных категорий – вычисление стоимости по методу ABC (см. ARIS ABC). Иерархия стоимостных категорий представляется с помощью диаграмм стоимостных категорий.

Определение

Стоимостные категории служат для систематического структурирования всей стоимостной информации, которая появляется в результате создания или оценки стоимостных драйверов (результатов). Вопрос заключается в следующем: откуда берутся стоимости?

Например, стоимость материалов – это стоимостные категории для использования материалов и их уценки, что определяет величину сокращения активов.

Все стоимости могут быть структурированы в соответствии с различными критериями. Если стоимости разбиваются по производственным факторам, то это приводит к структурированию стоимости персонала (зарплата, комиссионные и т.п.), стоимости материалов (например, стоимость заготовок, стоимость электроэнергии) и издержек, связанных с налогами и другими выплатами. Стоимостные категории могут также определяться в соответствии с наиболее важными операциями, такими, как стоимость закупок, стоимость хранения на складе, стоимость производства, административные издержки, стоимости продаж. Обе описанные структуры могут быть еще более детализированы.

Иерархия стоимостных категорий представляется направленными линиями, имеющими тип *является вышестоящим*.

Наиболее важным атрибутом для стоимостных категорий является шкала результата. Он описывает единицы, в которых измеряется результат стоимостной категории, например, стоимость оплачиваемых часов или квадратных метров в комнате.

На рис. 4.2.1-30 для типа сущности, приведенного в качестве примера на рис. 4.2.1.-1, показана диаграмма стоимостной категории производственных факторов с подструктурой, описывающей стоимость персонала.

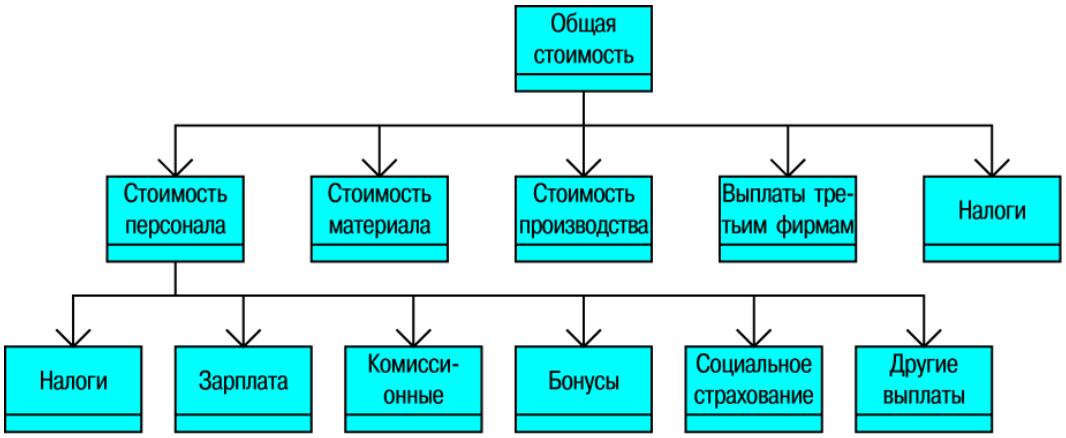


Рис. 4.2.1-30. Диаграмма стоимостной категории

В ARIS ABC диаграмма стоимостной категории может быть связана с экземпляром вычислений стоимости по методу ABC. Модель стоимостных центров перечисляет все стоимостные центры, которые будут проанализированы в экземпляре вычислений стоимости по методу ABC. Некоторые стоимостные категории и необходимый набор значений могут определяться для каждого стоимостного центра (каждый - соответственно методу вычислений, тарифам, объемам и/или результатам стоимостных центров). Другие факты могут быть описаны для каждого стоимостного центра в таблице стоимостных категорий в рамках (с учетом) модели стоимостных центров.

4.2.1.7. Модель данных в управлении проектами

4.2.1.7.1. Диаграмма носителей информации

Диаграмма носителей информации – это опциональная (необязательная) компонента в реализации прцедуры управления пректами в ARIS Toolset. Эта диаграмма связана с моделью данных на уровне формулировки требований. Она позволяет описывать входные и выходные данные в форме документов, журналов и диаграмм ARIS.

Модели ARIS могут быть описаны с помощью диаграмм PPC (цепочка процесса проекта, см. управленческую модель на уровне описания требований) в качестве «привязки» экземпляра «кластера», т. е. данные могут быть специфицированы в общем, виде в «привязанном» кластере. Необходимые рабочие документы, например, файлы, подготовленные с помощью текстовых редакторов, могут быть представлены и активизированы (вызваны) с помощью ARIS Toolset через атрибуты *Соединитель 1*, *Соединитель 2, Соединитель 3.*

Рис. 4.2.1-31. Диаграмма носителей информации

4.2.2. Спецификация проекта

4.2.2.1. Реляционная диаграмма. Диаграмма атрибутов

При спецификации проекта логические структуры данных, разработанные с учетом сформулированных требований, трансформируются в описание, на основе которого могут быть построены конкретные системы баз данных. В этом контексте ARIS использует концепцию реляционной модели.

Реляционная диаграмма и диаграмма атрибутов служат для определения существующих отношений, атрибутов и их взаимосвязей с информационными объектами, вводимыми при формулировке требований.

При формировании реляционной диаграммы, прежде всего, могут быть определены необходимые отношения.

Определение

Отношение описывает тип сущности по ее атрибутам. Это подмножество всех возможных комбинаций диапазонов значений отдельных атрибутов.

На рис. 4.2.2.1 приведено приводится графическое представление отношения.

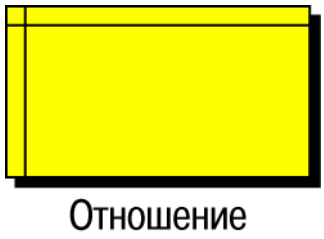


Рис. 4.2.2-1. Представление отношения

Каждая сущность, входящая в модель ER, представляет собой отношение в реляционной модели. При преобразовании типов отношений модели eER очень важным аспектом в решении вопроса, должно ли создаваться соответствующее отношение для каждого отдельного типа отношений, является мощность отношений. Отношения n:m, отличные от отношения 1:n, должны быть представлены соответствующими отношениями.

Для каждого отдельного отношения реляционная диаграмма указывает, какой тип сущности или отношения модели eER представлен.

Отношение может быть в дальнейшем специфицировано перечнем его атрибутов. В зависимости от того, является ли определенный атрибут ключевым, внешним ключевым или описательным, он может быть обозначен с помощью соединения, связывающего отношение и его атрибут. С другой стороны, на основе сформулированных требований можно установить отношение каждого отдельного атрибута к атрибуту модели ER, которое их отражает.

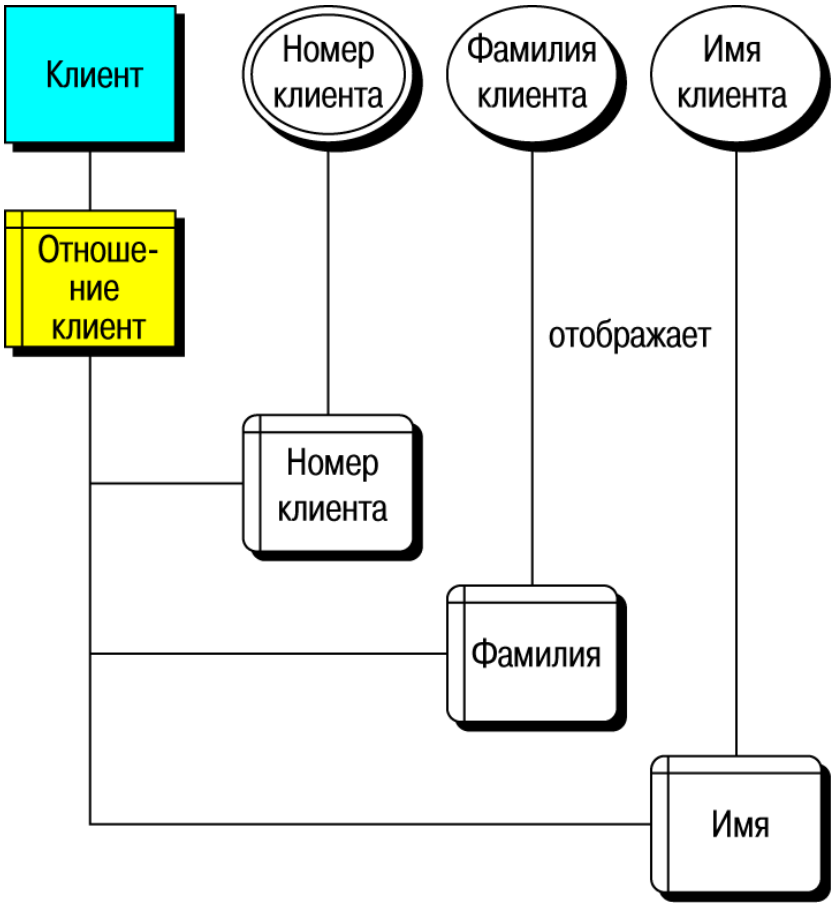


Рис. 4.2.2-2. Представление атрибутов и объектов данных на уровне формулировки требований

Для уменьшения степени сложности представления атрибуты каждого отношения могут быть описаны с помощью связанной с данным отношением диаграммы атрибутов, которая показана на рис. 4.2.2-3.

Рис. 4.2.2-3. Диаграмма атрибутов

На уровне спецификации проекта кластеры данных, содержащие сформулированные требования, описываются с помощью типа объекта – *Представление*. *Представление* определяется на основе определения кластера данных.

Определение

Представление понимается как логический взгляд на некоторую совокупность отношений.

Отношения, связанные с *Представлением*, также могут быть описаны реляционной диаграммой (см. на рис. 4.2.2-4).

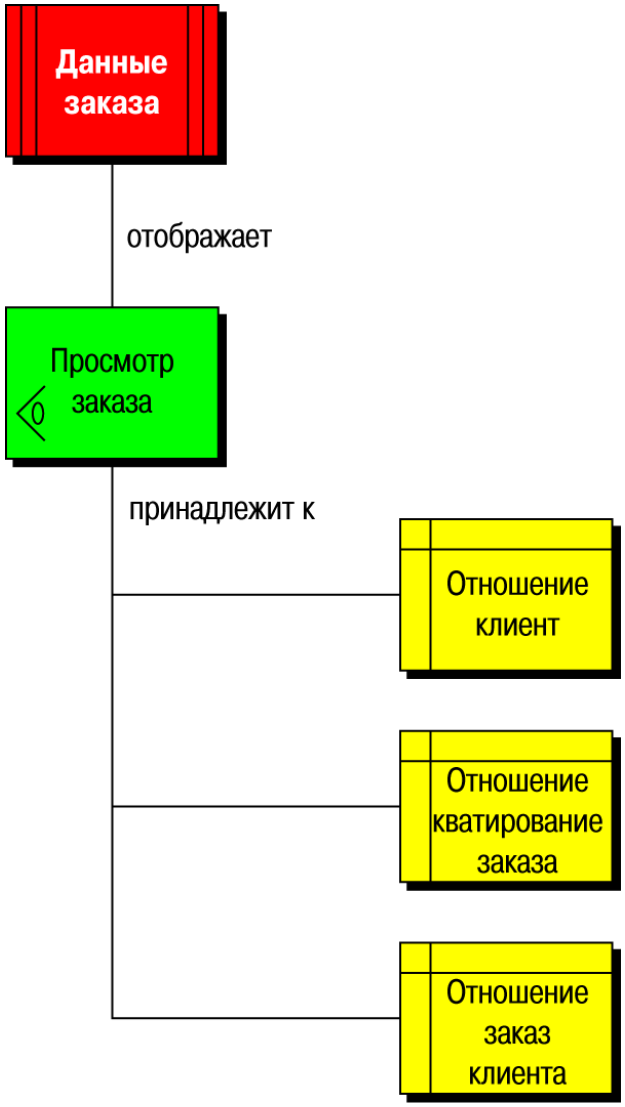


Рис. 4.2.2-4. Определение «представления»

Отношения мощностью 1:n невозможно отразить в реляционной модели ER с помощью соответствующих отношений. Отношение отражается интеграцией ключевых атрибутов более старших типов сущностей с отношением подчиненных типов сущностей. При этом первоначальный ключевой атрибут становится внешим ключом отношения.

Атрибут в реляционной модели, отражающий тип отношения в модели ER, может быть представлен в реляционной диаграмме обычным соединением (см. рис. 4.2.2-5).

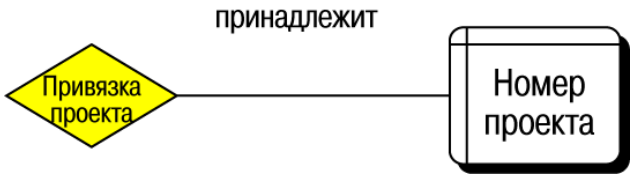


Рис. 4.2.2-5. Связь атрибута и типа отношения в модели ERM

В гл. 9 английской версии руководства «Методы ARIS» содержится перечень всех объектов и типов отношений реляционной модели.

4.2.2.2. Моделирование системного интерфейса. Системные атрибуты. Области системных атрибутов

Диаграмма типа *системные атрибуты* разработана главным образом задач, ориентированых на экспорт данных из ARIS Toolset. Эта диаграмма дает возможность иерархически упорядочить типы сущностей, события, технические термины, функции, информационные носители, организационные единицы и персонал, а также однозначно и в полном соответствии с требованиями к обработке данных специфицировать их. При обычных требованиях к базам данных можно различать следующие типы атрибутов: первичные и внешние ключи, описательные и обязательные поля. Для того чтобы определить типы областей этих объектов данных, вводится тип диаграммы *область системного атрибута* (см. рис. 4.2.2-7).

В отличие от ERM-атрибутов основное свойство системных атрибутов состоит в представлении и управлении данными, ориентированными на интерфейс. Системные атрибуты объектов имеют два поля значений, которые заполняться соответствующей информацией. Это гарантирует большую гибкость в экспортируемом содержимом.

В приведенном ниже примере показан фрагмент определения в ARIS Toolset заголовка проекта для передачи в систему управления проектом.

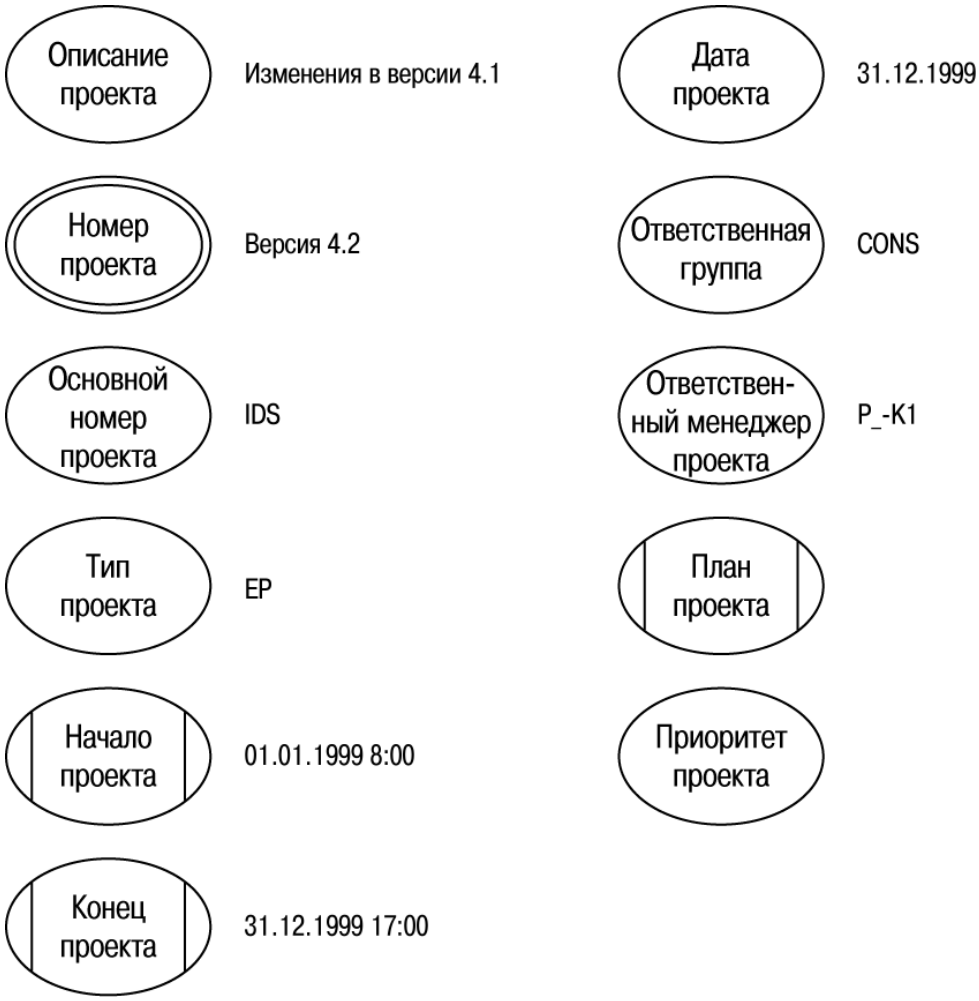


Рис. 4.2.2-6. Модель системных атрибутов

Диаграмма типа *область системного атрибута* определяет объекты системных атрибутов в соответствии с типом данных; например, она специфицирует тип области (char, int, date, и т.д.) и длину поля. Эта диаграмма используется главным образом для обеспечения информацией процесса экспорта данных во внешние системы.

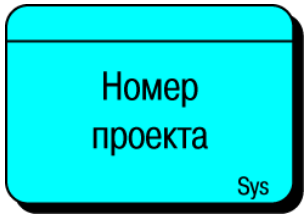


Рис. 4.2.2-7. Область системного атрибута

4.2.3. Описание реализации. Табличная диаграмма

Таблицы и поля систем управления базами данных могут быть описаны табличной диаграммой. На рис. 4.2.3-1 графически оизображены таблицы и поля.

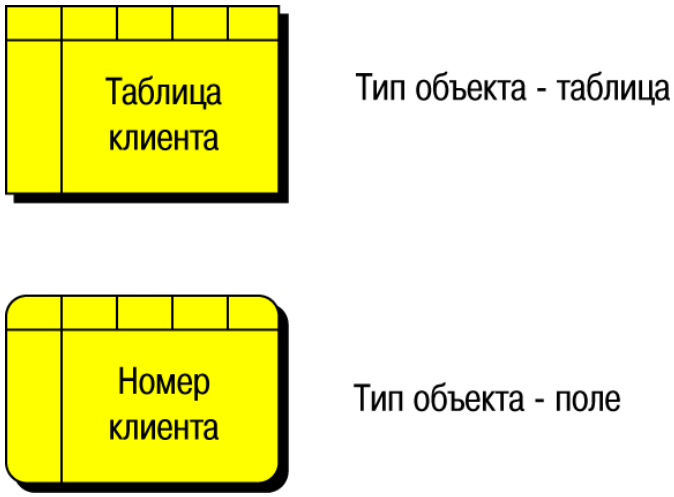


Рис. 4.2.3-1. Таблица и поля

К каждой таблице могут быть «привязаны» поля. При дальнейшей спецификации каждому отдельному полю присваиваются индексы сортировки и области их изменений (см. на рис. 4.2.3-2).

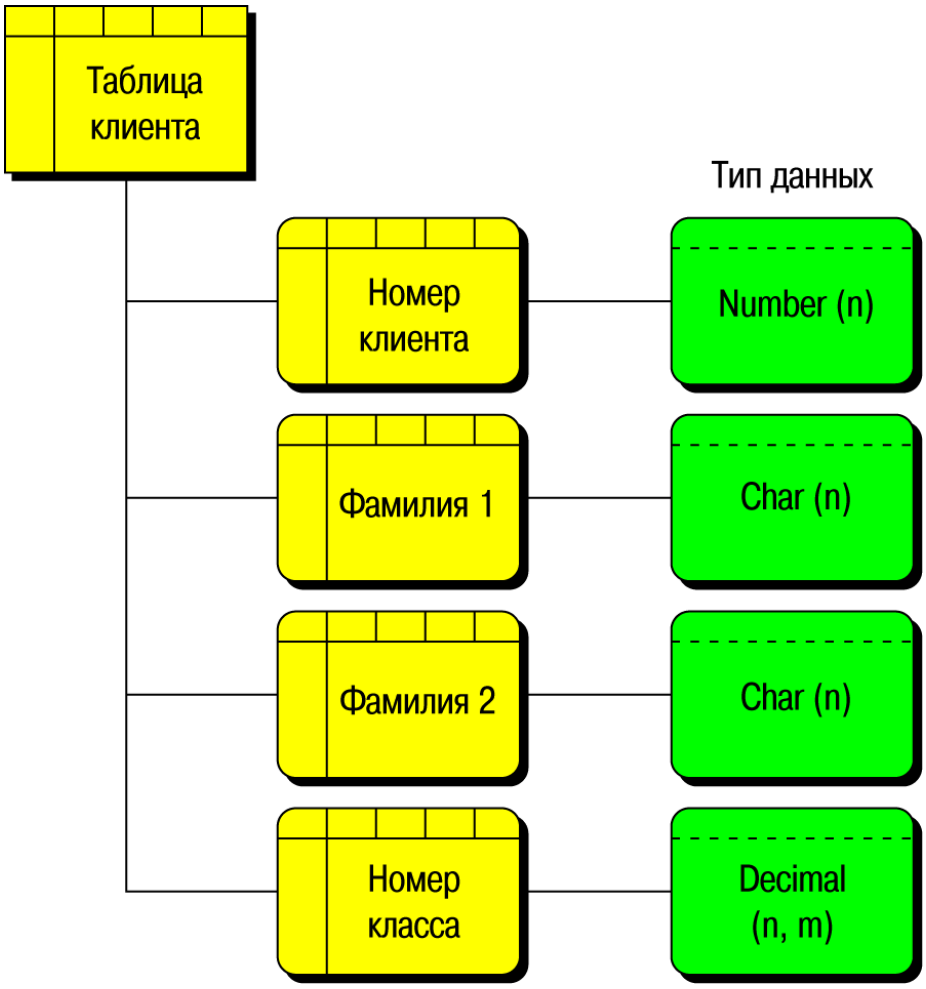


Рис. 4.2.3-2. Размещение полей

Отношения реляционной модели не всегда преобразуются в таблицы и поля в соответствии с отношением 1:1 (например, по причинам, относящимся к функционированию базы данных), поэтому между таблицами и отношениями или типами сущностей могут быть многосторонние отношения. При выборе соответствующих соединений эти отношения можно отобразить в виде табличных диаграмм. Кластеры данных, определенные при формулировке требований, или модели, определенные в реляционной модели, представляются табличной диаграммой объекта *Модель (физическая)*.

Преобразование и документирование таблиц и полей баз данных, используемых в компании, иногда производится без учета определений реляционной схемы. Именно поэтому отношения реализации могут быть отображены не только между отношениями

(или атрибутами) и таблицами (или полями), но и между типами сущностей (или атрибутами модели ER) и таблицами (или полями).

Можно также показать, какие отношения и атрибуты выполняются или (оставляя в стороне реляционные определения) какие типы сущностей, типы отношений и ERM-атрибуты отображаются таблицами и полями. На рис. 4.2.3-3 приведен пример этих двух форм представления.

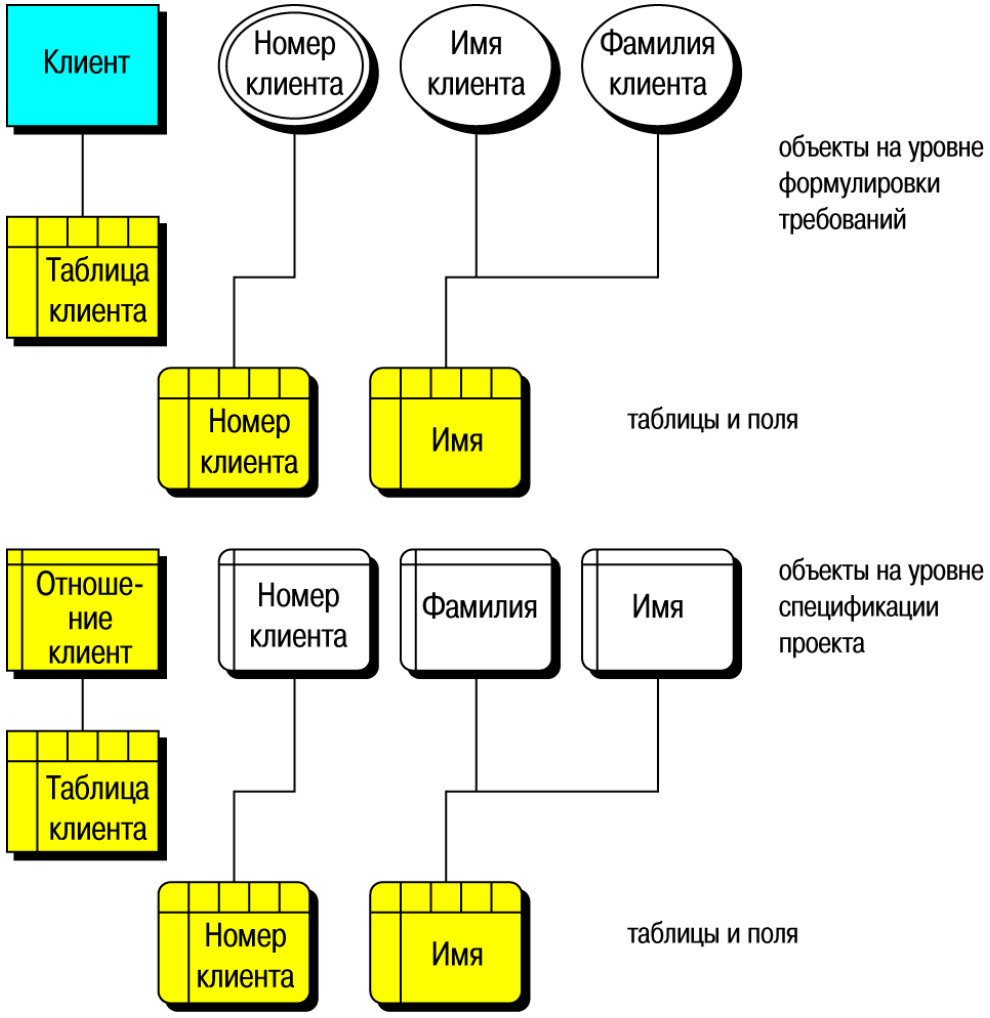


Рис. 4.2.3-3. Описание объектов на уровне формулировки требований и спецификации проекта

Для того чтобы определить точное местоположение некоторых таблиц и полей в компании, необходимо определить каждый отдельный экземпляр таблицы. То же самое относится к случаю, когда предполагается, что организационные единицы имеют

авторизованный доступ к определенным таблицам и полям. Тип объекта *таблица*, введенный ранее, определяет логическую структуру физической таблицы и ее поля на *уровне типа*. Определенные таким образом многочисленные экземпляры каждой таблицы могут быть доступны (сохранены в другой среде) в различных местах компании. Для иллюстрации этого факта вводятся типы объектов *таблица (образец)* и *поле (образец)*.

Благодаря этим объектам можно быть точно, определить количество экземпляров таблиц и полей. Эти связи представлены на рис. 4.2.3-4.

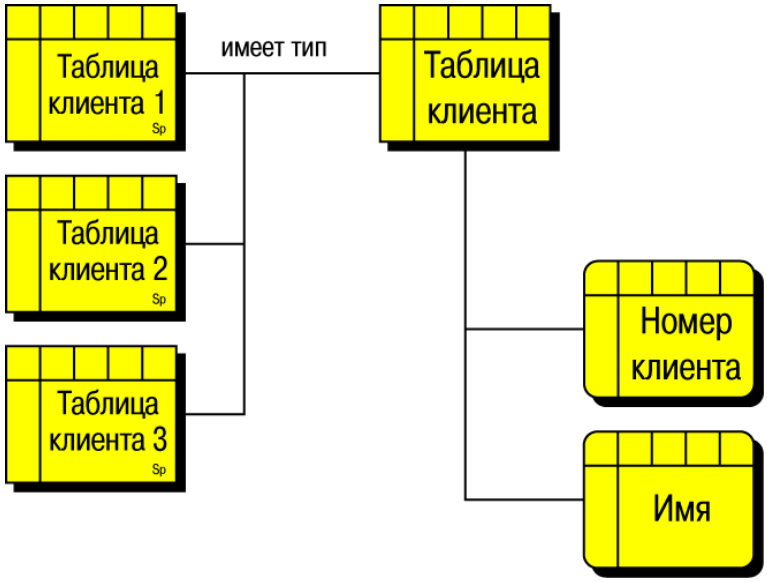


Рис. 4.2.3-4. Экземпляр таблицы

В гл. 9 английской версии руководства «Методы ARIS» содержится перечень всех объектов и типов отношений для табличной диаграммы.

4.3. Организационная модель

4.3.1. Формулировка требований

4.3.1.1. Организационная инфраструктура бизнеса

Компания – сложная социальная структура, которая может быть разделена на легко управляемые элементы. Для того чтобы справиться с ее сложностью, определяются структурные схемы ее организации и устанавливаются правила поведения. Результат этого процесса – «правильная» организация. До недавнего времени роль организационного анализа в аспекте развития информационных систем редко служила объектом иссследований. Новейшие концепции управления бизнесом (например, CIM – Computer Integrated manufacturing) были связаны с их непосредственной областью применения до тех пор, пока это касалось организационных аспектов. По этой причине архитектура ARIS поддерживает независимую для описания модель организационных структур.

Рассматривая организационную структуру компании, будем различать организацию структуры и организацию процедур.

Организационная структура включает правила, позволяющие произвести статичное структурирование компании. Организация процедур основана на правилах, отвечающих задачам, которые должны выполняться. Ориентированная на задачи структура (в смысле распределения функций между исполнителями) в архитектуре ARIS представляется с помощью *управляющей* модели. Таким образом, организационная модель является компонентой, которая позволяет анализировать организационную структуру компании.

Разработка «совершенной» организационной структуры для сведения координирующих усилий к минимуму зависит от инфраструктуры и целей компании. Следовательно, нельзя определить некую универсальную «совершенную» организационную структуру как структуру-прототип. Точное определение того, как необходимо структурировать организационные единицы, зависит от различных факторов.

При функционально-ориентированной структуре организации одно функциональное подразделение (закупки, призводство, финансы и бухгалтерия) несет ответственность за все продукты и территории. Преимущество узкой специализации служащих «компенсируется» непомерными требованиями к коммуникации и координации функциональных подразделений.

В течение длительного времени разработка, и использование информационных систем были ориентированы на функциональное деление компаний. Однако при такой структуре анализ сложных процессов с помощью обработки взаимосвязанных объектов

данных одной категории не позволяет установить взаимосвязи между отдельными функциями. По этой причине проблема интегрированной обработки данных сведена к требованию целостности базы данных, которая могла бы поддерживать различные функции. Однако цель функциональной интеграции находится в противоречии с задачей уменьшения сложности функциональной структуры.

При стремлении к функциональной интеграции часто применяются другие критерии организационного деления, которые могут быть ориентированы на территории или виды продукции. На рис. 4.3.1-1 приведена диаграмма, иллюстрирующая принципы структурирования и интеграции.

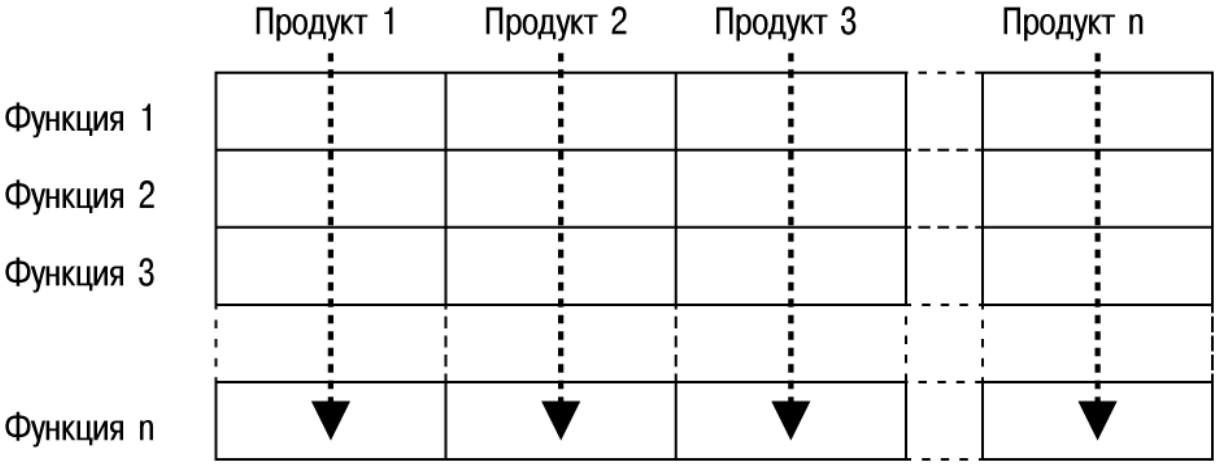


Рис. 4.3.1-1. Организационное деление в терминах продукции

Организационные единицы в составе территориально-структурированной организации определяются в соответствии с локальной политикой дистрибьюции как всей компании, так и ее части. Этот тип структуры удобен для функций, связанных с продажами, поскольку могут быть отражены и более точно отработаны специфичные для данной территории аспекты, например, изменяющееся законодательство и налоговый кодекс.

В компаниях, организационная структура которых ориентирована на производимые продукты, организационные единицы выделяются в соответствии с каждым продуктом и/или группой продуктов. По возможности интегрируются все функции, связанные с производством данной группы продуктов. Цель такой интеграции – сократить потребность в коммуникациях, что обычно свойственно функциональной структуре. Однако это приводит к необходимости иметь промежуточные звенья между подсистемами, связанными с той или иной группой продуктов.

Для нейтрализации негативных эффектов часто используются гибридные организационные формы. На рис. 4.3.1-2 приведен пример гибридной формы организации (рассматривается функция закупок).

	Группа продуктов 1	Группа продуктов 2	Группа продуктов 3
Централизованная закупка	Выбор поставщика		
	Термины и условия		
Планирование			
Оформление заказов			
Контроль и аудит накладных			

Рис. 4.3.1-2. Гибридные формы организации

При чисто функциональной структуре предполагается, что по всем группам продуктов делаются централизованные закупки. Такая организационная форма обеспечивает синергетический эффект при работе с группами продуктов, однако общая процедура закупки для всех подразделений привела бы к значительным проблемам координации. При разбиении функции закупки на несколько подфункций в соответствии с различными группами продуктов должны быть образованы отделы по закупкам для каждой группы продуктов. Но в этом случае при выборе продавцов или при переговорах по оформлению контрактов синергетические эффекты могут быть получены только ценой значительных усилий по координации действий отдельных подразделений.

Как показано на рис. 4.3.1-2, при «гибридной» организации те функции закупки, для которых ожидаются значительные эффекты, разбиваются на функциональные уровни - подфункции, часть которых выполняется централизованным органом. Функции (в данном контексте - подфункции), для которых должны быть учтены отдельные требования и/или ограничения по отдельным группам продуктов, разделяются на основе объектно-ориентированного принципа в соответствии с группами продуктов. Таким образом, эти функции сразу же могут быть интегрированы в процедуры процесса, выполняемые отдельными группами. Это означает, что некоторые процедуры процессов оперативно выполняются децентрализованно, однако взаимосвязь между ними реализуется на более высоком и централизованном уровне координации.

Этим гибким организационным формам в архитектуре ARIS придается особое значение, так как они отражают процессо-ориентированный подход. Методы, ориентированные на финансы, например концепция центров прибыли, требуют создания организационных структур, где одновременно анализируются различные критерии организационного деления.

4.3.1.2. Организационная схема

Организационная схема – типичная форма представления организационных структур, которая описывает организационные единицы (такие, как исполнители заданий) и их взаимосвязи в зависимости от критериев, в соответствии, с которыми организована структура.

Определение

Организационные единицы – это исполнители заданий, которые реализуются для достижения целей деятельности компании.

Отношения – это связи между организационными единицами. Соответствующий пример приведен на рис. 4.3.1-3.

Для того чтобы специфицировать отношения руководства/подчинения, будем более точно различать типы соединений, связывающие организационные единицы. В этом контексте соединения имеют одно из следующих значений:

- техническое руководство,
- дисциплинарное руководство,
- является компонентой для

Распределение задач иллюстрирует организационная схема на рис. 4.3.1-3, где функциональная ответственность изображена эллипсами.

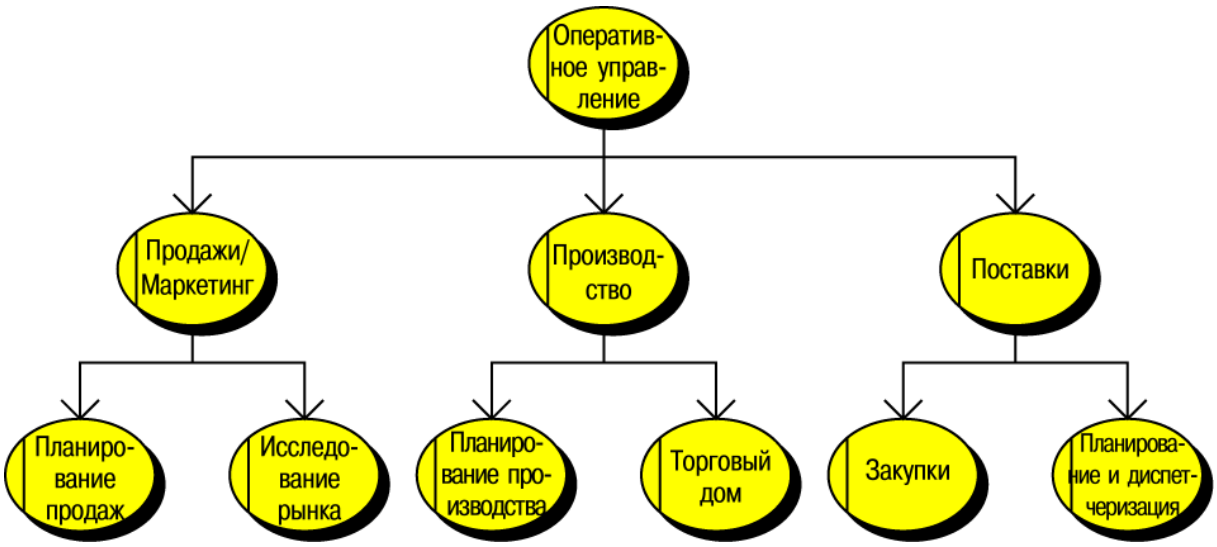


Рис. 4.3.1-3. Организационная схема

Для представления описаний отдельных должностей в компании необходим независимый тип объекта - *Должность*. Этот тип объекта отображается, как это показано на рис. 4.3.1-4.

Одна организационная единица может быть связана с несколькими должностями. Смысл соединений соответствует связям между организационными единицами.

Должности и организационные единицы присваиваются лицами, которые занимают рассматриваемые должности. ARIS содержит специальные объекты для представления отдельных сотрудников, что также отражено на рис. 4.3.1-4. Взаимосвязь сотрудников с организационной единицей указывает, что данное лицо является служащим, приписанным к данной организационной единице. С другой стороны, взаимосвязь с отдельной должностью определяет настоящую должность, занимаемую сотрудником в компании.

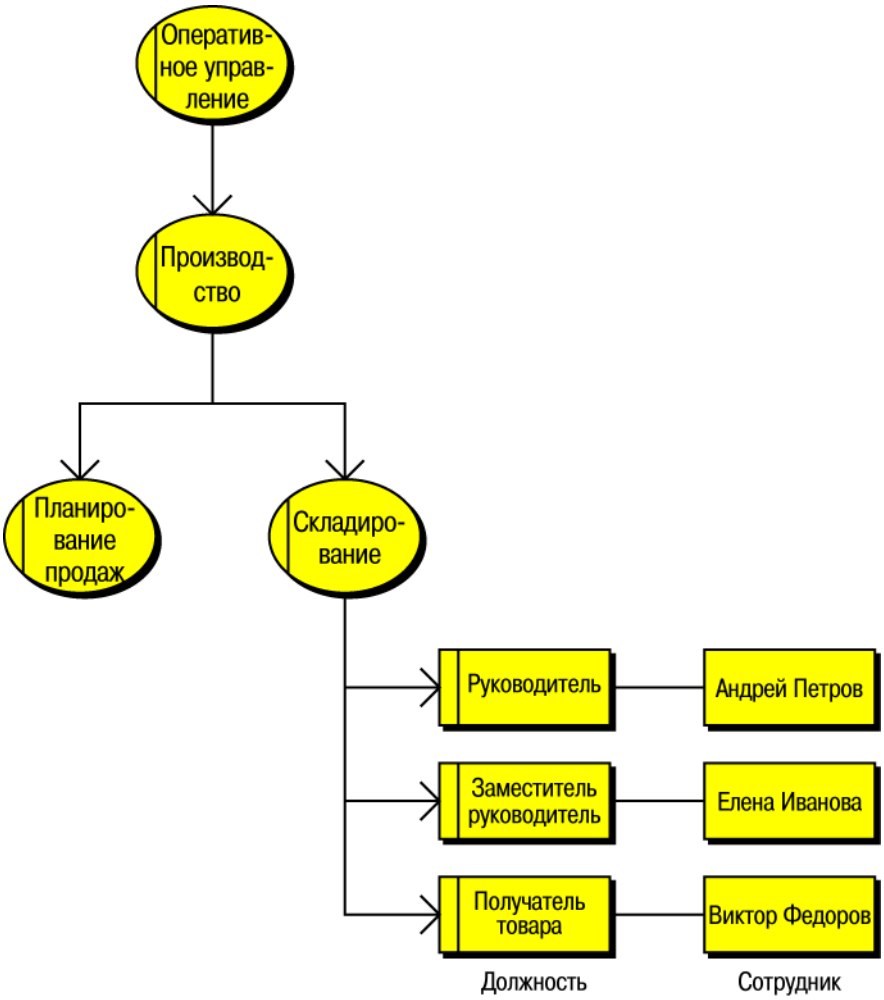


Рис. 4.3.1-4. Организационная схема должностей и сотрудников

Организационным единицам и сотрудникам может быть присвоен тип, т.е. можно определить, является ли организационная единица отделом, главным отделом или группой. Например, сотрудники могут соответствовать таким типам, как руководитель отдела, руководитель группы или менеджер проекта.

Для представления этих типов используются объекты *Тип организационной единицы* и *Тип сотрудника*. На рис. 4.3.1-5 приведена привязка типов организационным единицам и сотрудникам.

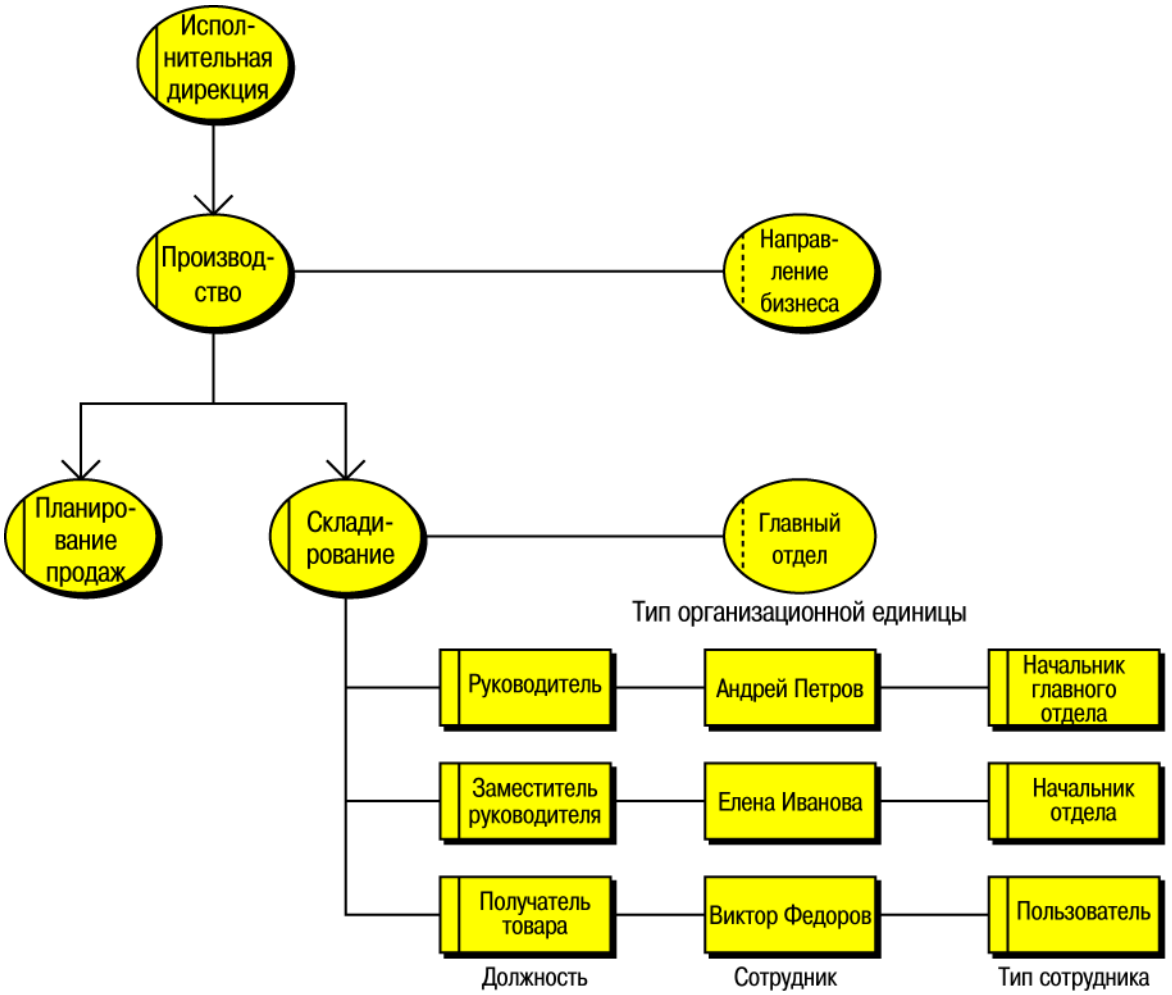


Рис. 4.3.1-5. Типы сотрудников

С помощью типов объектов можно представить основные правила работы, которые являются абстрактным описанием конкретных организационных единиц или сотрудников компании. Например, в рамках некоторого процесса можно специфицировать, что только определенному типу сотрудников разрешается выполнить одну конкретную функцию или получить доступ к отдельному информационному объекту.

Моделирование организационной структуры компании – стартовая точка в создании топологии компьютерной сети, которая должна быть определена на уровне спецификации проекта и которая, как предполагается, будет поддерживать организационную структуру наиболее оптимальным образом. Соединения сети и сетевые узлы, расположенные в определенных местах компании, являются главными

элементами топологии сети. Таким образом, местоположение организационной единицы – это наиболее важная связь между сформулированными требованиями и спецификацией проекта. Следовательно, для каждой организационной единицы можно определить место, причем это должно быть сделано как можно раньше – на уровне формулировки требований. Пример, иллюстрирующий сказанное приведен на рис. 4.3.1-6.

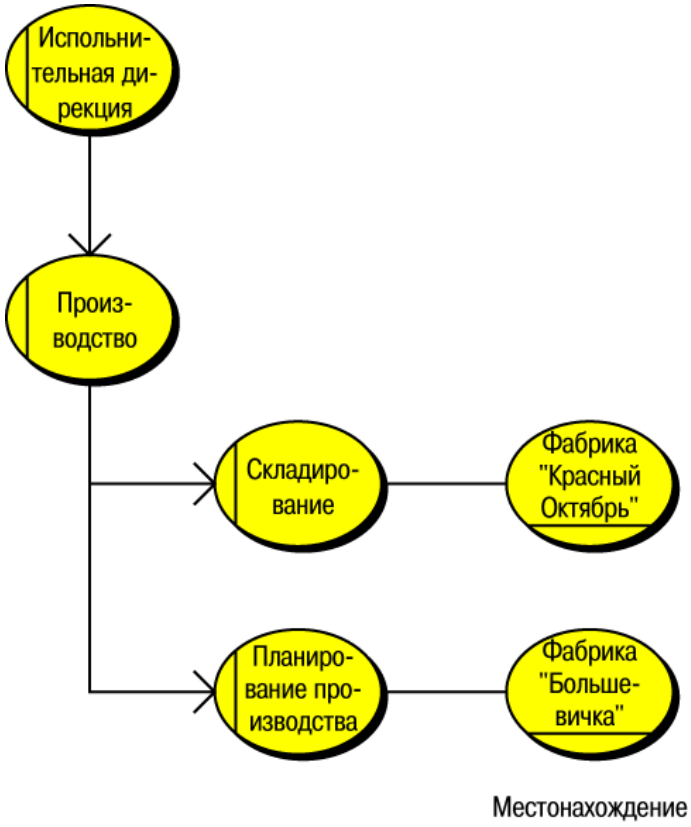


Рис. 4.3.1-6. Описание местоположений

Местоположение может занимать любую позицию в иерархической структуре организации. Оно может определять как отдельное здание, так и при более детальном анализе отдельный офис или даже единственное рабочее место в большой комнате. Таким образом, на этапе спецификации проекта, возможно, соотнести узлы сети с отдельным рабочим местом в организационной единице. Например, можно определить, что из офиса 102 должны быть доступны три узла сети.

На рис. 4.3.1-7 представлена иерархия местоположений.

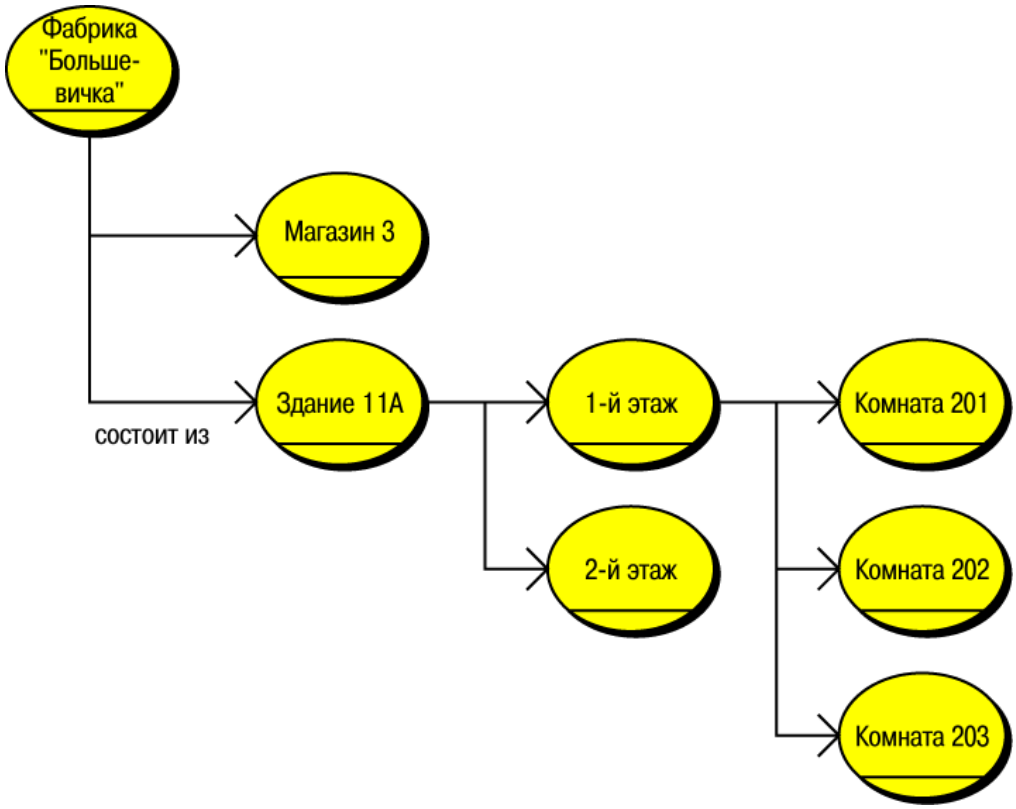


Рис. 4.3.1-7. Иерархия местоположений

4.3.1.3. Календарь смен

Календарь смен связан с человеческими и материальными ресурсами и составляется, когда ресурсы доступны. Календарь смен «приписывается» ресурсам в организационной схеме или диаграмме eEPC и в общем случае может быть связан с любым человеческим или материальным ресурсом. Если существует иерархия человеческих ресурсов, календарь смен всегда располагается на самом нижнем уровне иерархии.

Календарь смен – многоуровневая объектная модель. На самом нижнем уровне находятся объекты типа *Перерыв*. *Перерыв* – это ежедневный интервал времени в рамках смены, в течение которого не выполняется никакая работа. *Перерыв* определяется относительным временем его начала и продолжительностью, причем относительное время начала всегда ссылается на смену, с которой данный перерыв связан. Например, если смена начинается в 8.00 утра и перерыв имеет относительное время начала 2 часа, то перерыв начинается в 10.00.

Следующий уровень в иерархии включает объекты типа *Смена*. *Смена* – это ежедневный интервал времени, в течение которого выполняется работа. Смена определяется относительным временем начала и продолжительностью. Смена может иметь один и более перерывов. Относительное время начала всех перерывов должно находиться во временных рамках смены.

Типичный пример смен – утренняя, дневная, вечерняя и ночная смены. Каждая из них повторяется каждые 24 часа. Цикл смен – это еженедельный интервал времени или интервал, длящийся несколько дней, в течение которого выполняется работа. Цикл смен определяет дни, когда некоторая смена работает, а когда нет. Сам цикл смен определяется относительным временем начала цикла и его продолжительностью. Если цикл смен должен повторяться непрерывно, для индентификации этого вводится специальный атрибут *Период*, который определяет, как часто данный цикл должен повторяться.

Циклы смен часто охватывают период от одной до двух недель. Сотрудник может работать в утреннюю смену в течение одной недели и в дневную в течение другой. Эта последовательность может повторяться непрерывно через циклы смен.

Далее в примере рассматриваются два цикла смен, которые определяются следующим образом:

1. Цикл смен (утренняя смена)	Относительное начало цикла	0
	Продолжительность цикла	5 дней
	Повторяемость цикла	да
	Период	14 дней
2. Цикл смен (дневная смена)	Относительное начало цикла	7 дней
	Продолжительность цикла	5 дней
	Повторяемость цикла	да

	Период	14 дней

Отдельные смены повторяются с 14-дневным ритмом, с периодичностью две недели. Если один и тот же сотрудник работает в утреннюю смену по субботам каждые 4 недели, то третий цикл смен должен быть определен следующим образом:

3. Цикл смен (утренняя смена)	Относительное начало цикла	20 дней
	Продолжительность цикла	1 день
	Повторяемость цикла	да
	Период	14 дней

На рис. 4.3.1-8 представлена модель с описанием смен и их циклов.

Рис. 4.3.1-8. Календарь смен

План смен – это совокупность всех циклов и их смен, которая описывает рабочие часы некоторого ресурса. Это описание содержит только то, что периодически повторяется. Специальные правила, определяющие отпуска болезни, праздники и другие дни, в течение которых работа не выполняется, в это описание не включаются.

Тип объекта, *План смен* включает атрибуты *Начало плана* и *Продолжительность плана*. Эти атрибуты определяют временные рамки, в течение которых этот план имеет

силу. Атрибуты *Повторяемость цикла* и *Период* также существуют для объекта *План смен*.

4.3.2. Спецификация проекта. Топология сети

Организационная структура компании, представленная ее организационной схемой, может быть поддержана коммуникационной и информационной инфраструктурами системы. Структурные требования к таким информационным системам в основном могут быть описаны на этапе спецификации проекта с использованием топологии сети.

В начале диаграмма топологии сети может содержать различные типы сетей.

Определение

Тип сети представляет отдельные экземпляры сети, которые базируются на одной и той же технологии.

Пример типа сети приведен на рис. 4.3.2-1:

Рис. 4.3.2-1. Графическое представление типа сети

Типы сетей могут быть взаимосвязаны, и поскольку они являются логическими конструкциями, их также можно разместить в соответствии с иерархией.

Более того, каждый тип сети может быть связан с возможными типами узлов сети и типами соединений сети. Таким образом, сразу можно сформулировать технологические ограничения, определяющиеся выбором конкретной сети. Каждый тип соединения сети может включать информацию о типах узлов сети, в которых она оканчивается.

Типы компонент аппаратного обеспечения (АО) относятся в этом случае или к оборудованию сети, реализующему сетевые структуры, или к типам компонент оборудования, которые могут быть подсоединены к типам узлов сети.

Аналогично операционным системам или типам сетей типы компонент АО не являются отдельными частями компонент оборудования, которые могут быть идентифицированы (например, инвентарные номера, присвоенные компанией). Напротив, они представляют все компоненты оборудования, базирующиеся на одной и той же технологии. Типы компонент АО могут располагаться в любом месте иерархии.

Определение

Типы компонент АО – это отдельные компоненты оборудования, которые базируются на одной и той же технологии.

Вместе с типами узлов и соединений сети может быть создана модель-прототип топологии сети. Она определяет, какие типы компонент аппаратуры могут использоваться для реализации сетевого соединения или типа узла. Примером типа соединения может быть отдельный тип сетевого кабеля. Кроме того, можно показать, какие типы компонент АО к какому типу узла сети могут быть подсоединены. Более того, типы компонент апаратуры могут иметь взаимосвязь с другими типами компонент, которые необходимы для создания типов узлов (см. рис. 4.3.2-2).

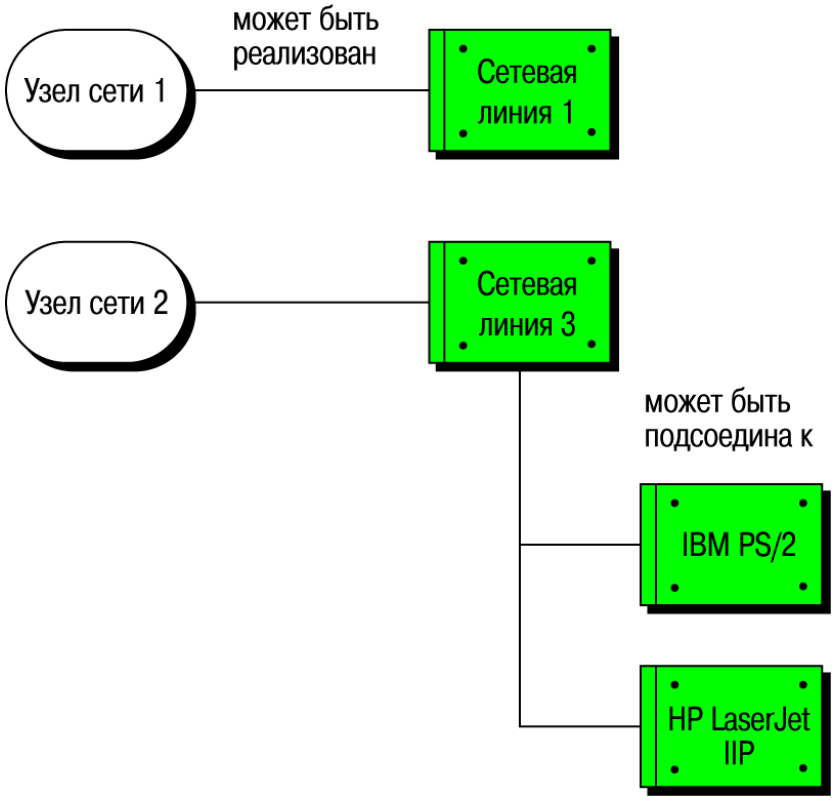


Рис. 4.3.2-2. Топология сети

Связь между топологией сети и объектами, определяемыми на уровне формулировки требований, устанавливается по двум направлениям. С одной стороны, может быть специфицирована организационная единица или должность, которые ответственны за каждый тип компонент АО; с другой стороны, можно определить, в каком месте компании находится данный тип сети, тип узла сети, тип соединения и тип компонент аппаратуры. Таким образом, местоположение – это центральное звено между сформулированными для организационной модели требованиями и спецификацией проекта.

В гл. 9 английской версии руководства «Методы ARIS» содержится перечень всех объектов и типов отношений диаграммы топологии сети.

4.3.3. Описание реализации

4.3.3.1. Диаграмма сети

Диаграмма сети используется для представления реализации топологии сети, определяемой на уровне спецификации проекта.

Посредством объекта *Сеть* регистрируются конкретные сети компании. Для каждой сети должны быть определены узлы сети и соединения.

С помощью моделирования можно описать точное местоположение каждой сети, ее узла и соединения. В этом контексте местоположение может определять предприятие, отдельное здание, индивидуальный офис или единственное рабочее место.

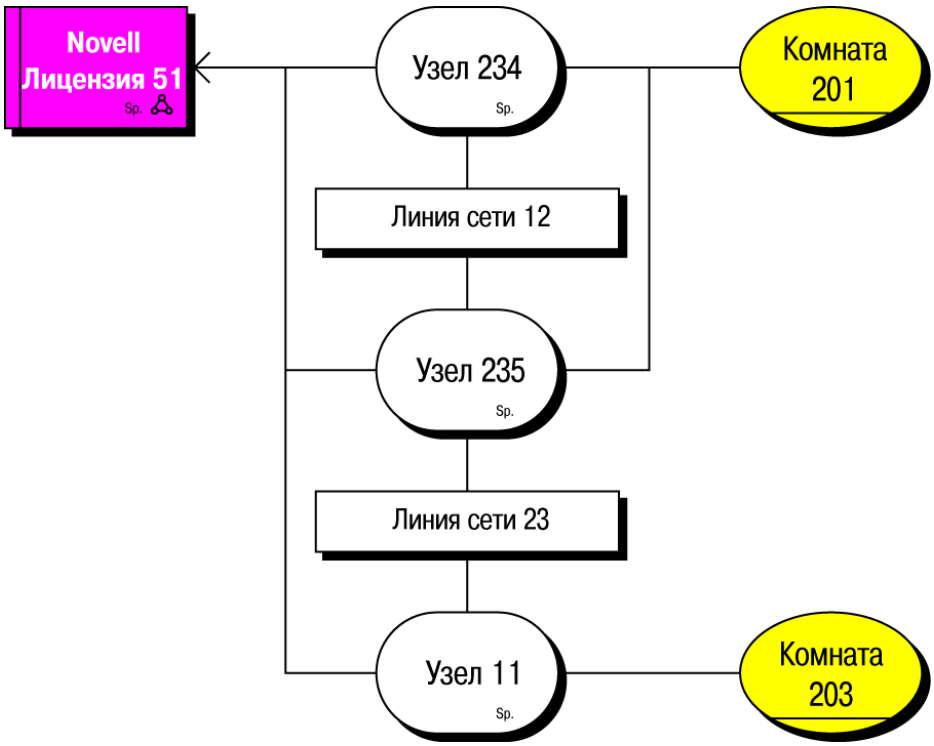


Рис. 4.3.3-1. Диаграмма сети с учетом местоположения

Можно зарегистрировать компонент у АО, представленную в каждом соединении и узле сети. Кроме того, можно также представить структуру каждой компоненты аппаратуры. С одной стороны, компоненты АО служат узлами и соединениями реализуемой сети, с другой стороны, они могут быть связаны с узлами сети.

Эта взаимосвязь также описывается с помощью диаграммы сети. Для каждого отдельного объекта моделируется взаимосвязь с соответствующим объектом на уровне спецификации проекта. Таким образом, может быть показано, что на фабрике «Красный Октябрь» сеть работает под управлением Novell, версия 4.11.

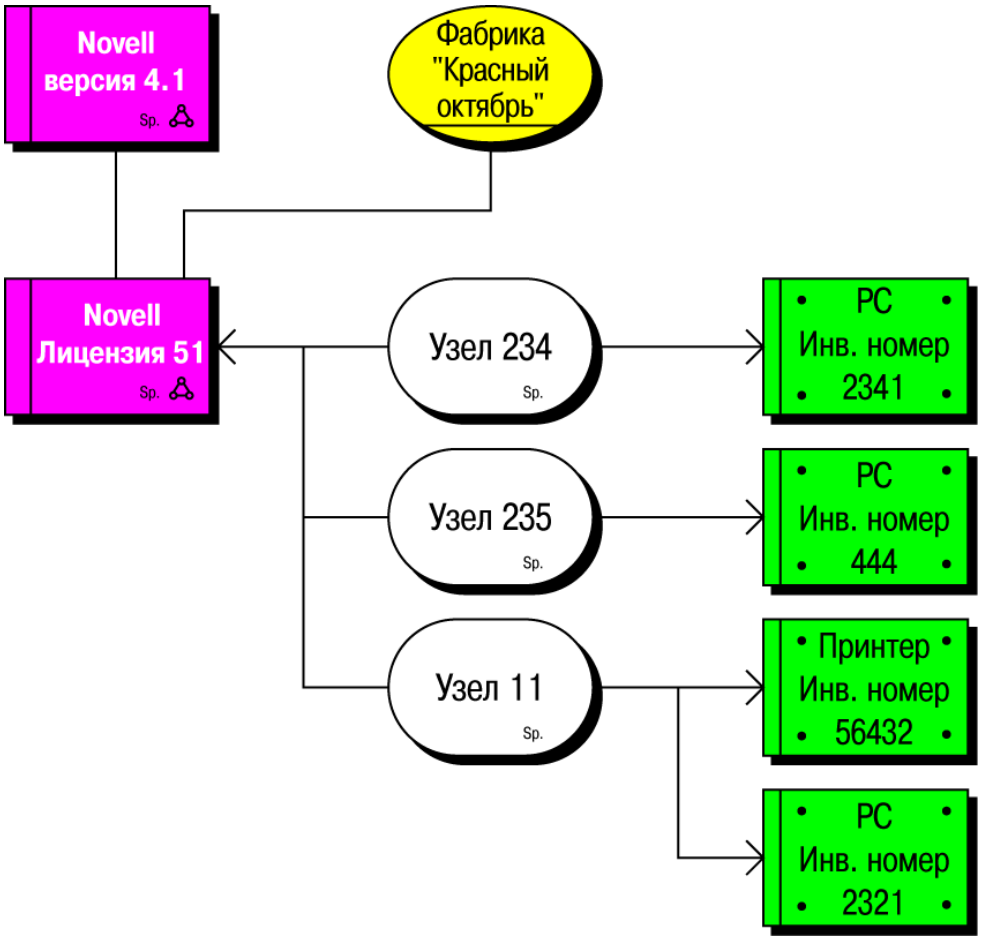


Рис. 4.3.3-2. Диаграмма сети с компонентами АО и указанием местоположения

Таким образом, на диаграмме сети устанавливаются два типа связи: связь описания типа сети со спецификацией проекта и связь описания компонент сети и конкретных местоположений с формулировкой требований.

В гл. 9 английской версии руководства «Методы ARIS» содержится список всех объектов и типов отношений для диаграммы сети.

4.3.3.2. Моделирование потока материалов. Технические ресурсы

Для представления потоков материалов при моделировании бизнес-процесса (диаграммы eEPC и PCD с потоком материалов) потоки материалов связываются с отдельными функциями бизнес-процесса в виде входа и выхода функций. Аналогично связи информационных объектов с функциями (преобразование информации представляется посредством функций) эта связь описывает преобразование типов материалов, поступающих на вход функции, в типы материалов на выходе функции. Кроме того, последовательности процессов дают возможность включить информацию по техническим ресурсам, которая необходима для преобразования материалов. В этом контексте будем различать операционные ресурсы, складское оборудование, транспортные системы и операционное обеспечение.

При помощи диаграммы типа *Технические ресурсы* эти ресурсы можно иерархически упорядочить, присвоить им тип и классифицировать. Существуют следующие типы объектов:

Операционные ресурсы

Определение

Операционные ресурсы – это экземпляры различных типов операционных ресурсов, которые доступны для выполнения задач, стоящих перед компанией. Операционные ресурсы часто идентифицируются с помощью различных инвентарных номеров (например, номер завода).

Тип операционных ресурсов

Определение

Тип операционных ресурсов представляет совокупность различных видов операционных ресурсов, которые имеют одинаковую технологическую базу.

Класс операционных ресурсов

Определение

Типы операционных ресурсов могут быть объединены в классы операционных ресурсов. Их схожесть определяется в соответствии с различными критериями классификации. Следовательно, один тип операционных ресурсов может соответствовать нескольким классам операционных ресурсов.

Складское оборудование

Определение

Складское оборудование – это экземпляры различных типов складского оборудования, которое доступно и предназначено для выполнения стоящих перед компанией задач. Складское оборудование часто идентифицируется присвоенными ему номерами складов.

Тип складского оборудования

Определение

Тип складского оборудования представляет совокупность нескольких видов складского оборудования, которые имеют одинаковую технологическую базу.

Класс складского оборудования

Определение

Типы складского оборудования могут быть объединены, образуя классы складского оборудования. Их схожесть определяется в соответствии с различными критериями. Следовательно, один тип складского оборудования может соответствовать нескольким классам складского оборудования.

Техническое оперционное обеспечение

Определение

Техническое операционное обеспечение – это экземпляр типа технического операционного обеспечения. В общем случае он может быть идентифицирован посредством инвентарного номера.

Тип технического операционного обеспечения

Определение

Тип технического операционного обеспечения представляет совокупность видов технического операционного обеспечения, которые имеют одинаковую технологическую базу.

Класс технического операционного обеспечения

Определение

Типы технического операционного обеспечения могут быть объединены в классы технического операционного обеспечения. Разделение на классы происходит в соответствии с различными критериями классификации. Следовательно, один тип технического операционного обеспечения может соответствовать нескольким классам технического операционного обеспечения.

Транспортная система

Определение

Транспортная система – это экземпляр типа транспортной системы. В общем случае она может быть идентифицирована инвентарным номером или номером предприятия.

Тип транспортной системы

Определение

Тип транспортной системы представляет совокупность некоторых видов транспортных систем, которые имеют одинаковую технологическую базу.

Класс транспортной системы

Определение

Типы транспортных систем могут быть объединены в классы транспортных систем. Объединение в классы осуществляется в соответствии с различными критериями классификации. Следовательно, один тип транспортной системы может соответствовать нескольким классам транспортных систем.

Возможности иерархической организации диаграммы типа *Технические ресурсы* позволяют описывать структуру технически сложных объектов (заводов, фабрик и т.д.),

а также отображать компоненты сложных производственных объектов и взаимосвязь между ними.

Кроме описания возможностей в приведенных выше терминах моделирования можно также описать размещение рабочих мест и организационную ответственность за технические ресурсы. Для этого используются типы объектов *Местоположение*, *Организационная единица*, *Должность* и *Сотрудник*, которые уже известны. Они приводились при описании диаграммы типа *Организационная схема*. Эти типы объектов могут быть связаны с типами объектов *Операционные ресурсы*, *Складское оборудование*, *Техническое операционное обеспечение* и *Транспортная система*.

На рис. 4.3.3-3 приведена диаграмма типа *Технические ресурсы*.

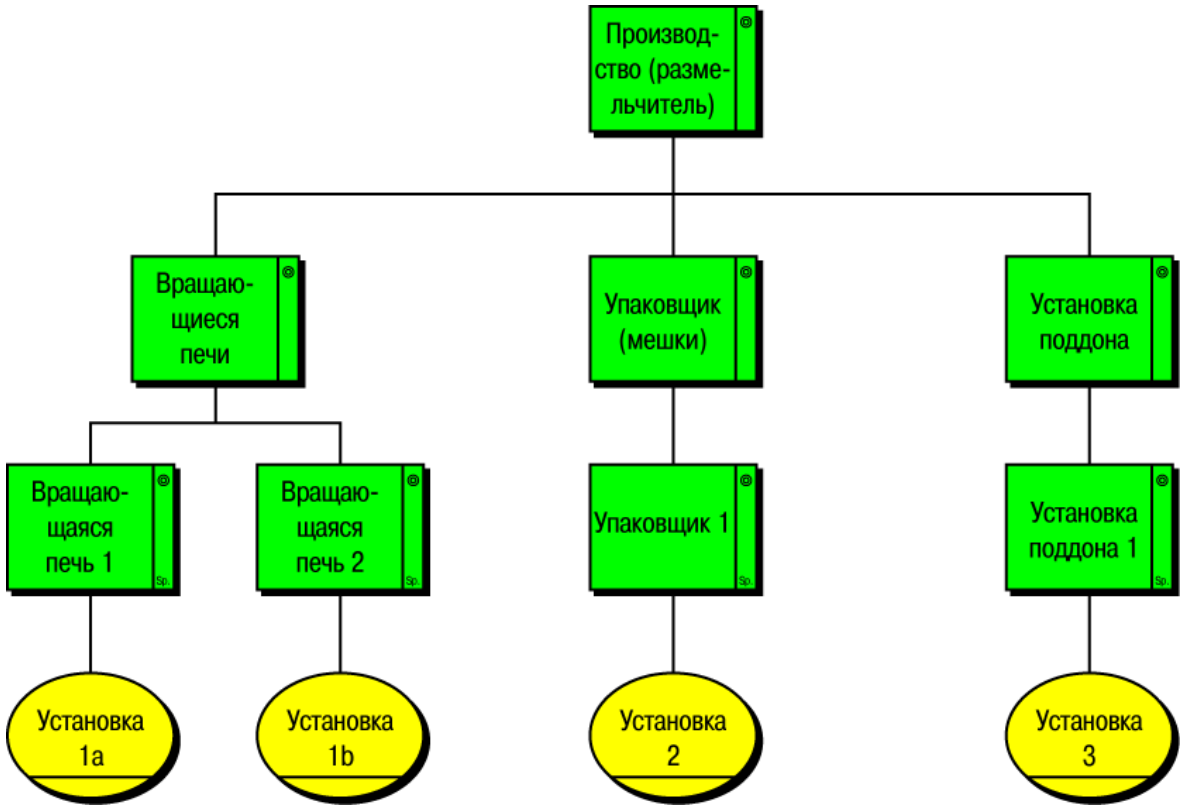


Рис. 4.3.3-3. Диаграммы «Технические ресурсы»

4.4. Модель процесса/Управляющая модель

4.4.1. Формулировка требований

Взаимосвязь между объектами модели данных, организационной модели и функциональной модели осуществляется с помощью управляющей модели. Отношения, которые мы будем рассматривать, являются результатом установления связей между элементами этих трех моделей (см. рис. 2.2 –2).

Прежде всего, рассмотрим связи между двумя моделями с помощью диаграммы, описывающей взаимосвязь между указанными тремя моделями.

4.4.1.1. Объединение функций со структурой организации. eEPC, диаграмма уровня функция/организация

Связь между функциональной моделью и организационной моделью служит для установки соответствия между функциями, определенными функциональным деревом, и исполнителями заданий (организационные единицы) в организационной схеме. Это соответствие определяет ответственность организационных единиц и уровень принятия решений для всех описанных функций. Основываясь на организационном соответствии в рамках диаграммы цепочки процесса, определяется степень функциональной интеграции, т.е. функциональные шаги в рамках бизнес-процесса, которые должны быть, выполнены организационными единицами.

На рис. 4.4.1-1 показано соответствие организационных единиц и функций. Отдельные функции расположены слева и соотнесены с организационными единицами, ответственными за выполнение отдельной функции. Местоположение функции в иерархии описывается с помощью функциональной модели (функционального дерева), причем взаимосвязи между организационными единицами представлены в организационной модели (организационная схема), поэтому определять их здесь нет необходимости.

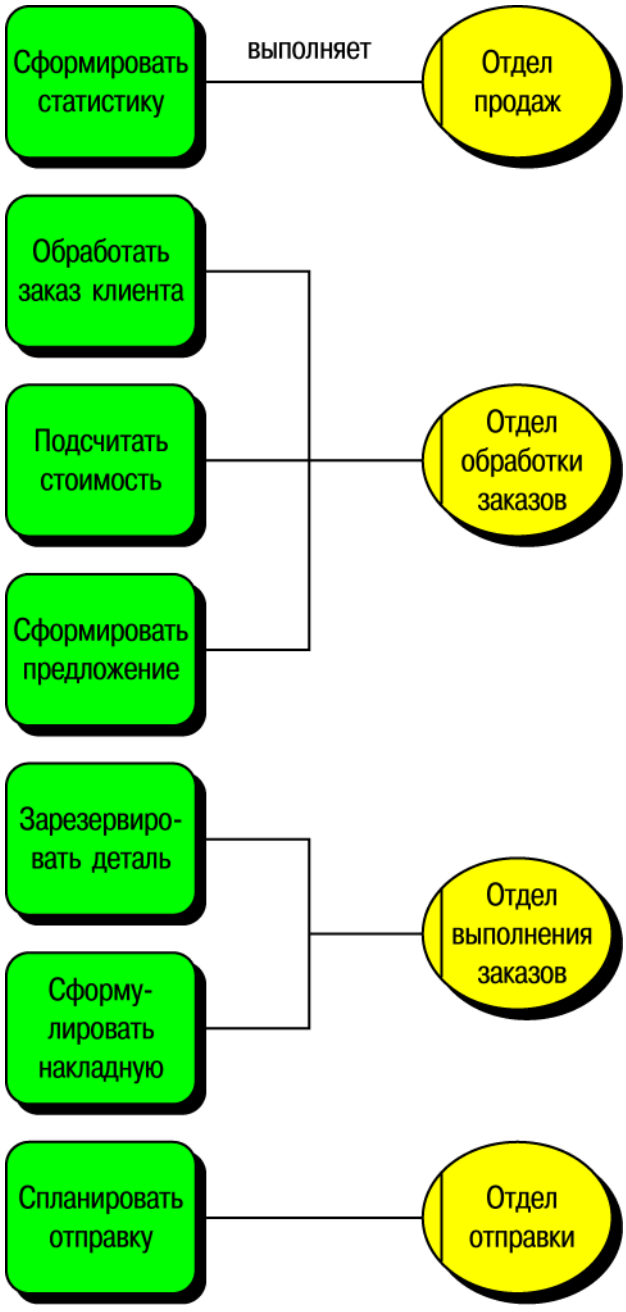


Рис. 4.4.1-1. Связь организационных единиц с функциями

4.4.1.2. Объединение функций и данных

4.4.1.2.1. Управление событиями. Событийная цепочка процесса (EPCs)

Процедурная последовательность функций в рамках бизнес-процессов отображается в виде цепочки процесса, где для каждой функции могут быть определены начальное и конечное событие. События не только переключают функции (передают управление от одной функции к другой), но и являются их результатом.

Определение

Под событием будем понимать тот факт, что информационный объект получает связанный с бизнес-процессом статус, который управляет или воздействует на дальнейшее выполнение бизнес-процесса. События переключают функции, т.е. передают управление от одной функции к другой; они могут быть также результатом выполнения функций. В отличие от функций, которые имеют некоторую продолжительность, события, происходят моментально.

Измененный статус информационного объекта может относиться или к первому появлению этого объекта (например, *запрос клиента получен*), или к модифицированному состоянию, что выражается использованием различных атрибутов (например, *предложение отклонено*). Поскольку информационные объекты и атрибуты описаны в модели данных ARIS, событийная цепочка процессов является мостиком между моделью данных и функциональной моделью. Следовательно, она является управляющей моделью ARIS.

События графически представляются шестиугольниками. Описание события должно содержать не только информационный объект (*заказ*), но и описание изменения состояния (*получен*). События представлены на рис. 4.4.1-2.

Рис. 4.4.1-2. Графическое представление событий

Определение

События переключают функции и могут быть результатом выполнения функции. Упорядочивание комбинации событий и функций в последовательность позволяет создать событийные диаграммы поцессов EPCs (Event - Driven Process Chain - цепочка процесса, управляемая событиями). С помощью этих диаграмм процедуры бизнес-процесса представляются как логические последовательности событий.

Событийная диаграмма процесса приведена на рис. 4.4.1-3. Поскольку события определяют, какое состояние или отношение будет переключать функцию и какое состояние будет определять конец ее выполнения, начальные и конечные узлы EPC всегда являются событиями.

Одно событие может инициировать выполнение одновременно нескольких функций, и наоборот, функция может быть результатом наступления нескольких событий. Эти ветвления и циклы обработки отображаются на диаграмме EPC с помощью соединителей в виде небольшого кружка (см. рис. 4.4.1-3). Однако эти соединители не только отображают графические связи между элементами модели, но и определяют логические связи между объектами.

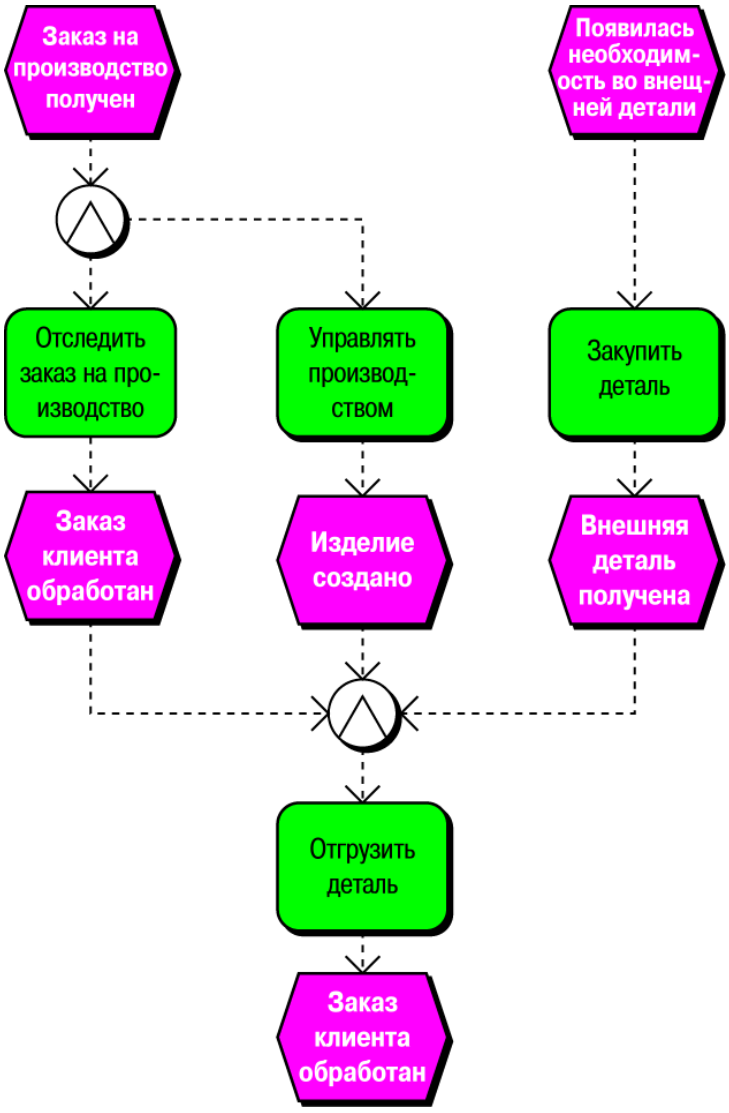


Рис. 4.4.1-3. Диаграммы EPC

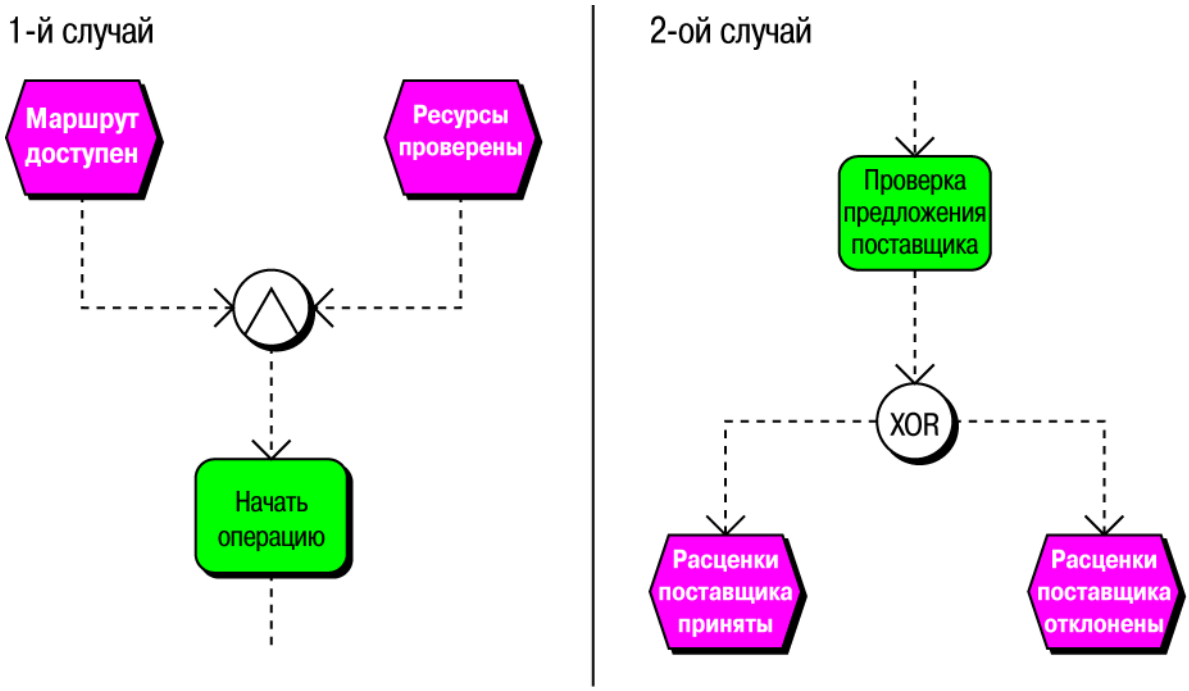


Рис. 4.4.1-4. Примеры правил

В первом примере на рис. 4.4.1-4 начальные события связываются с помощью оператора AND (И). Это означает, что процедура *Начать операцию* запускается только тогда, когда создан маршрут и проверено наличие необходимых ресурсов. Другими словами, для начала выполнения процедуры должны произойти оба события.

Во втором примере показан исключающий оператор OR (ИЛИ) или XOR. Результатом выполнения функции *Проверка предложения поставщика* может быть принятие или отклонение его расценок. Однако оба события не могут произойти одновременно. Кроме указанных двух случаев и обычного OR, можно представить и более сложные отношения. В этом контексте можно ввести в EPC общее правило, которое впоследствии будет описано более подробно в виде диаграммы правил.

Будем различать два различных типа операторов:

1) операторы событий,

2) операторы функций.

На рис. 4.4.1-5 представлены все возможные операторы событий и функций.

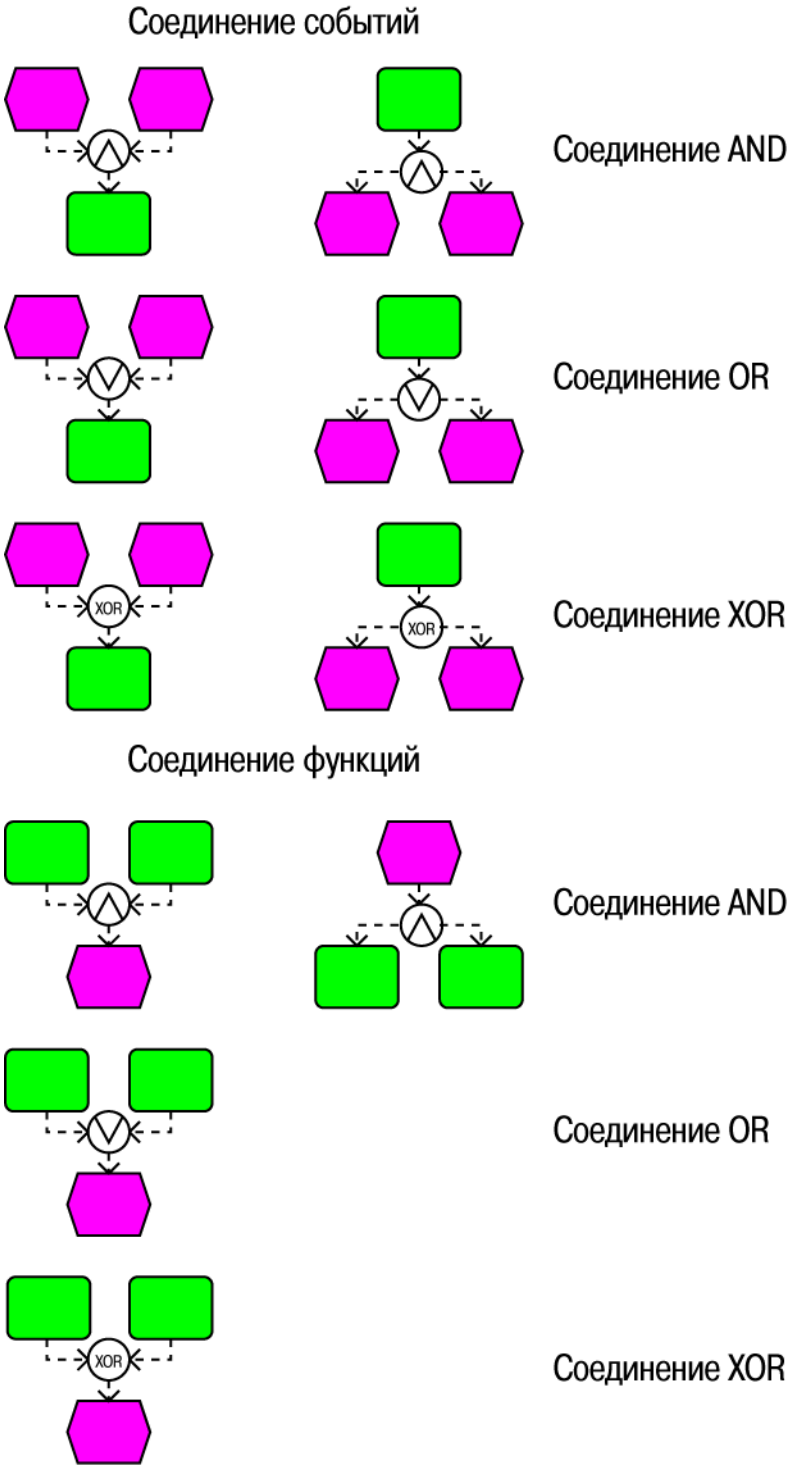


Рис. 4.4.1-5. Операторы соединений

Особое внимание необходимо уделить ограничениям, которые существуют для операторов функций. Посколько события не могут принимать решения (в то время как функции могут), переключающееся событие не должно быть связано операторами OR или XOR!

В дальнейшем будет показано на примерах, какие операторы допустимы.

1. Соединение переключающих событий

a) Оператор AND

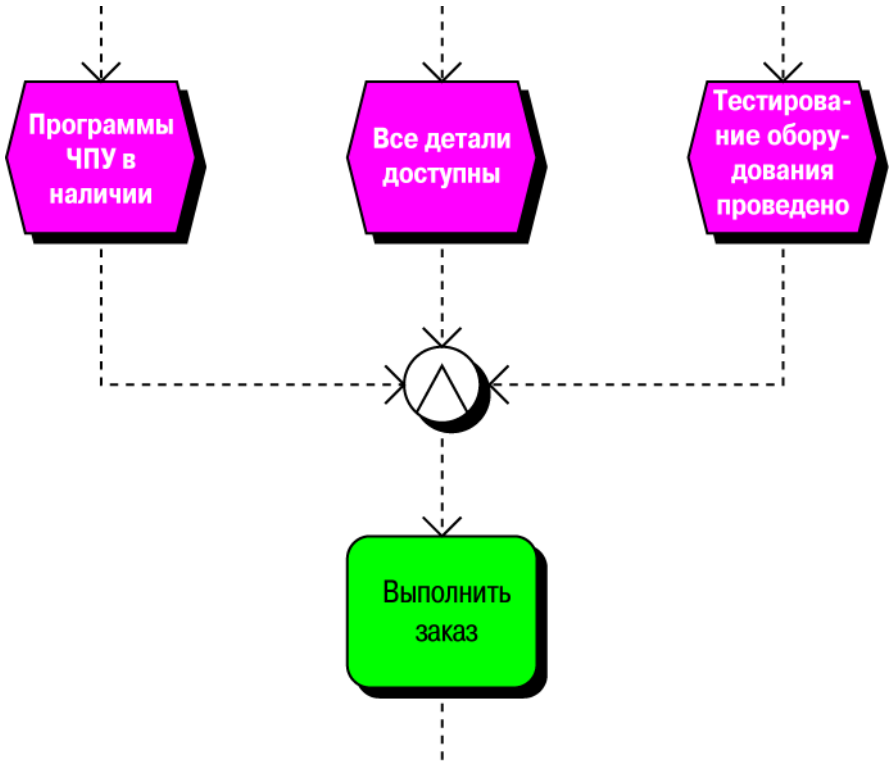


Рис. 4.4.1-6. Оператор AND для переключающих событий

Выполнение функции может быть начато после того, как произойдут все события.

б) Оператор OR

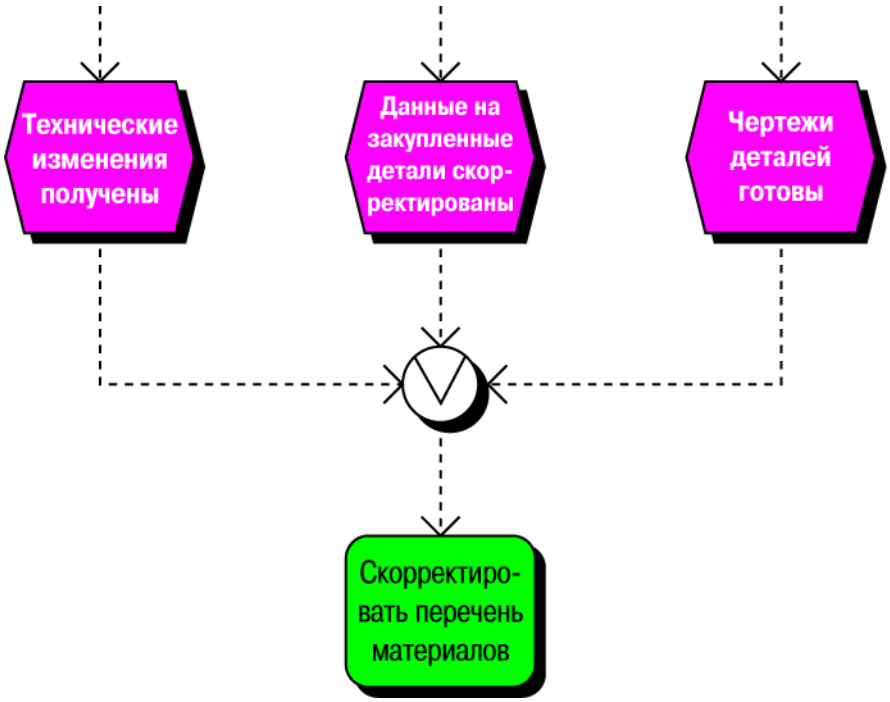


Рис. 4.4.1-7. Оператор OR для переключающих событий

Эта функция выполняется, если произойдет, по крайней мере, одно событие.

в) Оператор XOR

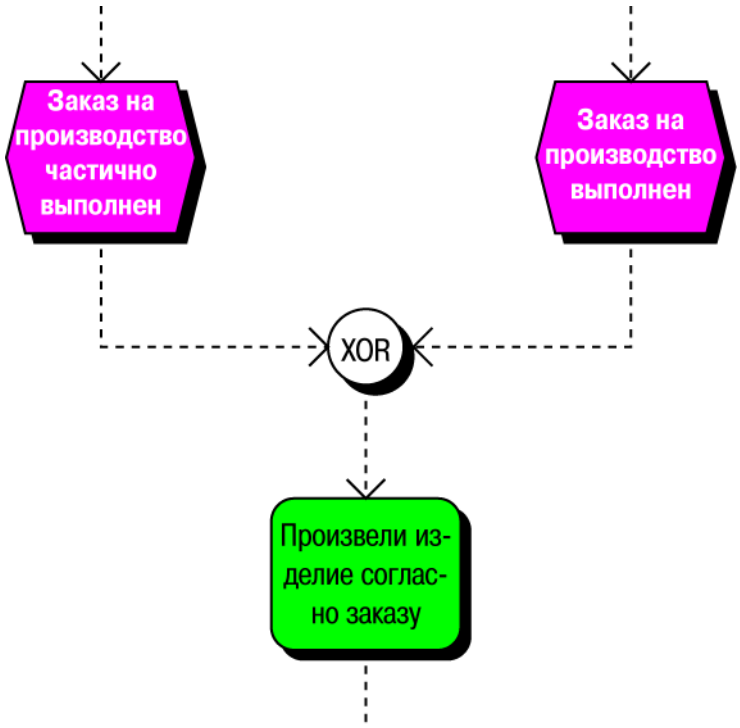


Рис. 4.4.1-8. Оператор XOR для переключающих событий

Функция начинает выполняться после того, как произойдет одно (и только одно) событие.

2. Соединение сгенерированных событий

а) Оператор AND

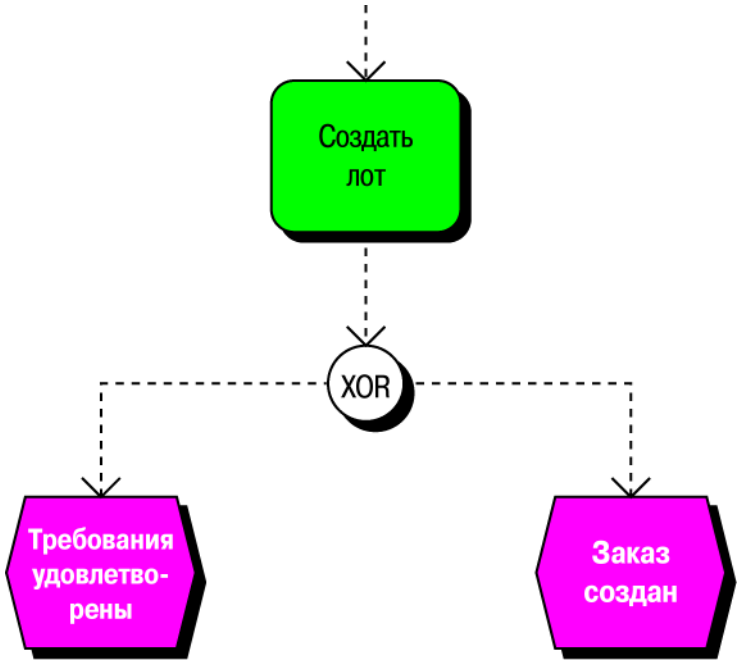


Рис. 4.4.1-9. Оператор AND для сгенерированных событий

В результате выполнения функции происходят все события.

б) Оператор OR

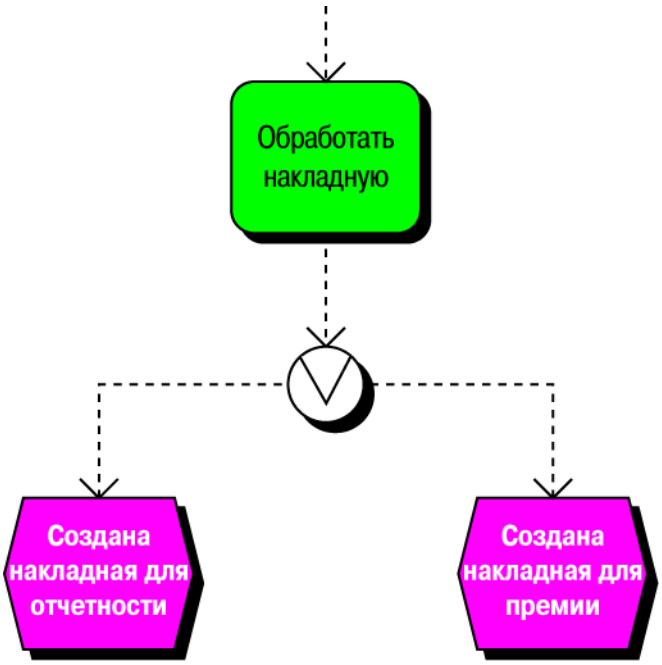


Рис. 4.4.1-10. Оператор OR для сгенерированных событий

В результате выполнения функции происходит, по крайней мере, одно событие.

в) Оператор XOR

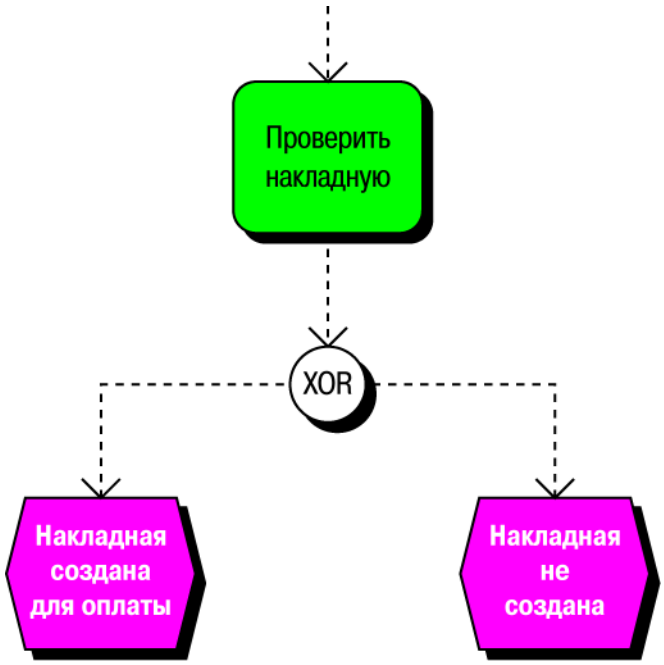


Рис. 4.4.1-11. Оператор XOR для сгенерированных событий

В результате выполнения функции происходит максимум одно событие.

3. Соединение функций со сгенерированными событиями

а) Оператор AND

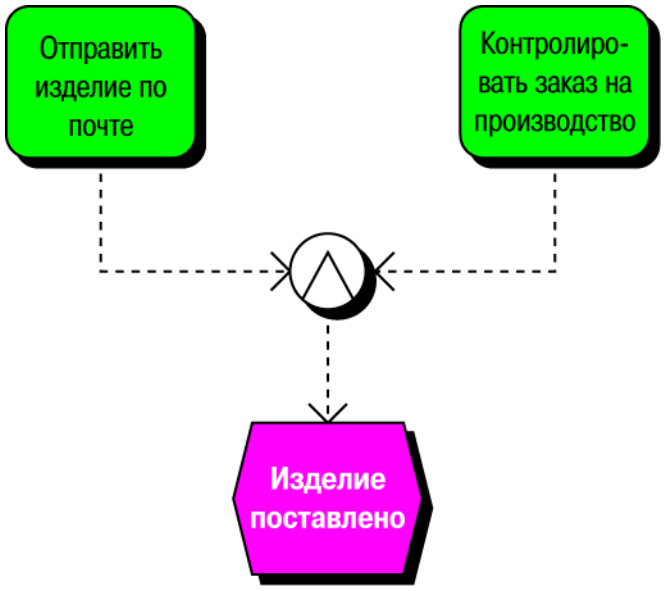


Рис. 4.4.1-12. Оператор AND для связи функций и сгенерированных событий

События происходят только после того, как все функции выполнены.

б) Оператор OR

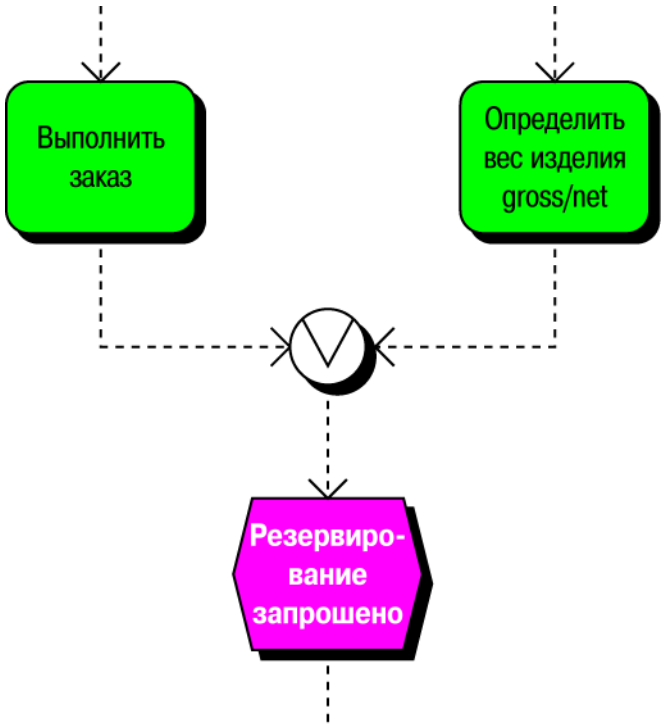


Рис. 4.4.1-13. Оператор OR для связи функций и сгенерированных событий

Событие произойдет после того, как будет выполнена, по крайней мере, одна функция.

в) Оператор XOR

Рис. 4.4.1-14. Оператор XOR для связи функций и сгенерированных событий

Событие произойдет после того, как будет выполнена одна (и только одна) функция.

4. Соединение функций с переключающими событиями

а) Оператор AND

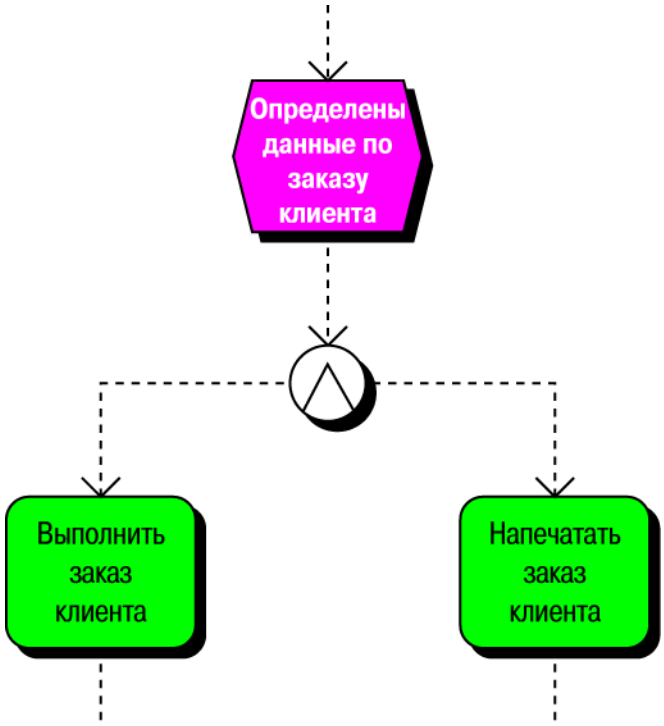


Рис. 4.4.1-15. Оператор AND для связи функций и переключающих событий

Событие переключает несколько функций.

б) Оператор OR

События не могут принимать решения! Эта связь не может быть установлена.

в) Оператор XOR

События не могут принимать решения! Эта связь не может быть установлена.

Кроме событийных диаграмм процессов, эти связи могут быть также представлены на диаграмме процесса в виде таблицы со столбцами событий и функций (см. гл. 3). Поскольку функции вызываются последовательно, в диаграмме цепочек процесса ветвления и циклы обработки могут оказаться представленными достаточно запутанным способом.

4.4.1.2.2. Диаграмма описания функций (вход/выход)

Представление управления событиями, рассмотренное в разделе 4.4.1.2.1, - только один возможный тип связи между моделью данных и функциональной моделью в архитектуре ARIS. Другой тип связи – преобразование входных данных в выходные данные и представление потока данных между отдельными функциями. Преобразование входных данных в выходные данные может быть отражено диаграммами описания функции (вход/выход), которые в основном соответствуют обычным диаграммам входа/выхода, используемым в других методах. Диаграмма описания функции представлена на рис. 4.4.1-16.

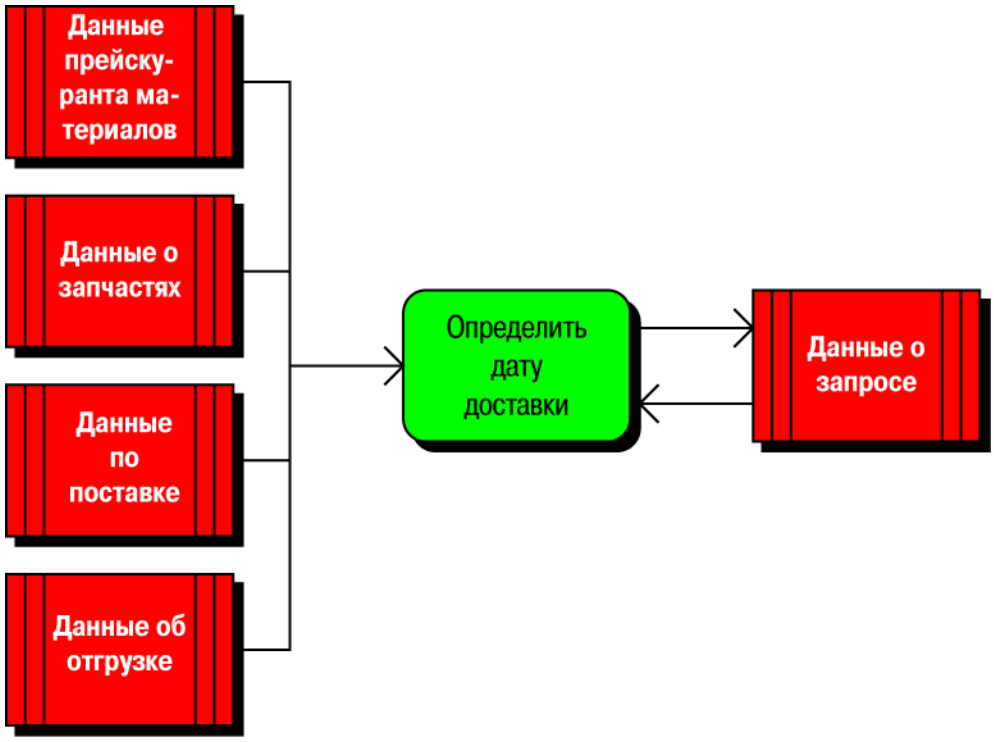


Рис. 4.4.1-16. Диаграмма описания функции (вход/выход)

Входными данными функции. О*пределить дату доставки* являются *Данные о запчастях*, *Данные прейскуранта материалов*, *Данные по поставке* и *Данные об отгрузке*. *Данные о запросе* служат как входными данными, так и выходными данными. Таким образом, диаграммы описания функций (вход/выход) содержат функции из функциональной модели и информационные объекты из модели данных. Стрелки определяют, используются ли информационные объекты в качестве входных данных, выходных данных или входных/выходных данных. Можно провести более детальную спецификацию, указывающую, создает или удаляет отдельная функция

информационный объект. В зависимости от степени детализации информационные объекты могут представлять кластеры данных (см. рис. 4.4.1-16), сущности или типы отношений, а также атрибуты модели данных.

Приведенный выше пример поясняет цель диаграммы описания функции (вход/выход): она должна представлять вход/выход данных для каждой функции.

Кроме входных/выходных данных функции и событий, доступны и могут быть использованы все другие объекты, связанные с отдельной функцией в диаграмме eEPC. Таким образом, пользователь при моделировании цепочки процесса с помощью диаграмм eEPC может ограничиться описанием событий и функций и связать с функцией диаграмму описания функций (вход/выход), в которой отражаются дополнительные связи и отношения, детализирующие эту функцию.

Это позволяет отображать бизнес-процессы более четко, а также поясняет названия нового типа модели. Пример диаграммы, более детально описывающей функцию, представлен на рис. 4.4.1-17.

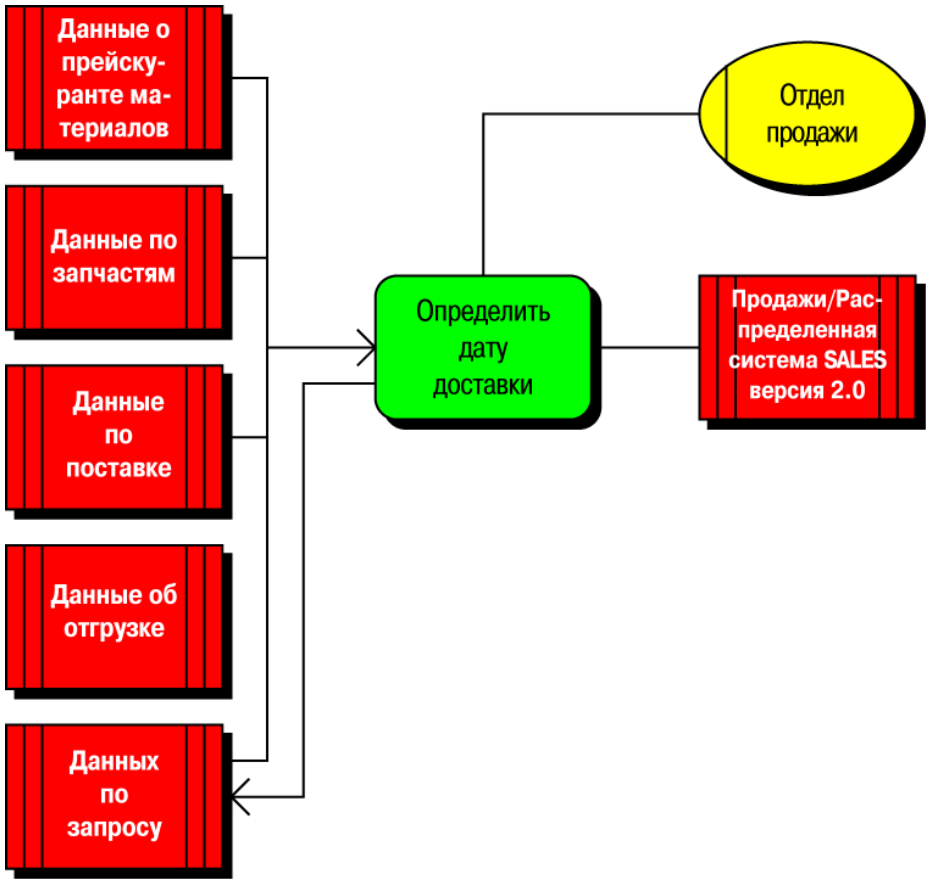


Рис. 4.4.1-17. Диаграмма описания функции

Кроме диаграммы описания функции (вход/выход), где представляется преобразование данных функцией, эта информация также может быть включена в диаграмму EPC (см. на рис. 4.4.1-18).

В этом случае значения операторов, связывающих функции и информационные объекты, аналогичны приведенным в диаграмме описания функции (вход/выход). Однако если они включены в цепочку процесса с многочисленными разветвлениями, то картина может оказаться довольно запутанной.

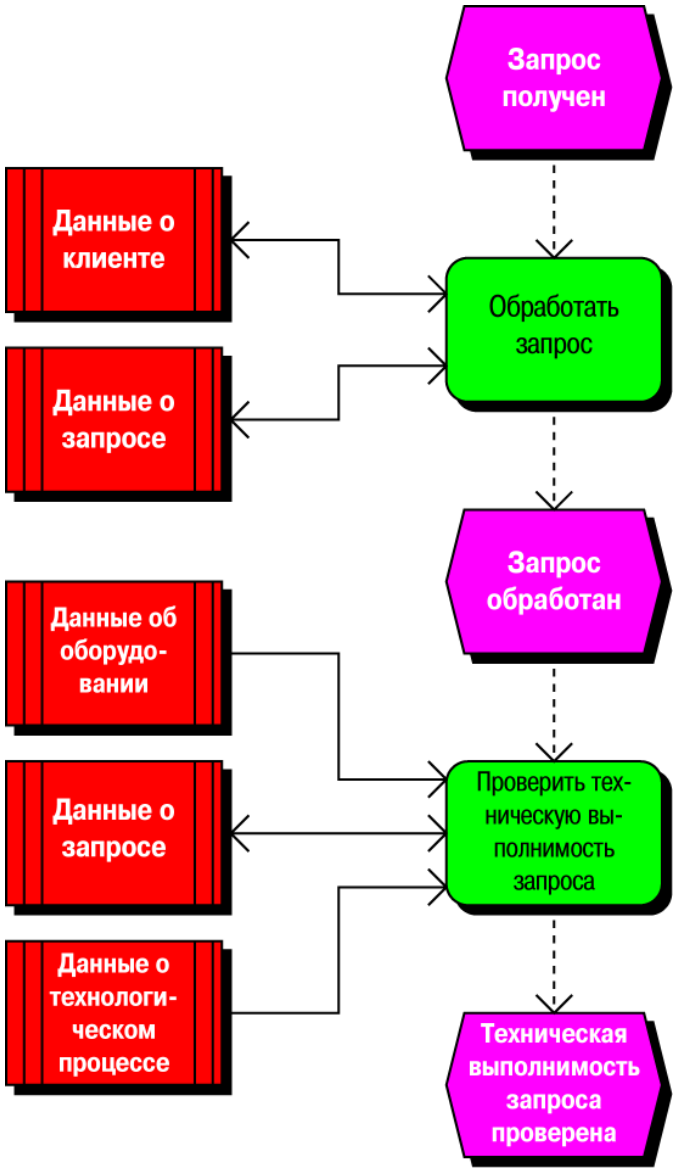


Рис. 4.4.1-18. Диаграмма EPC с данными входа/выхода

В диаграммах процесса (PCD) объекты должны быть упорядочены в колонки. В диаграмме EPC допустима свободная организация объектов. Однако добавление входных/выходных данных может привести к путанице в моделях, поэтому в диаграммах РСД рекомендуется придерживаться последовательности в представлении, соответствующей последовательности выполнения функций в бизнес-процессе. На рис. 4.4.1-18 диаграмма EPC с входными/выходными данными представлена как диаграмма РСД (см. также раздел 4.4.1.3.1).

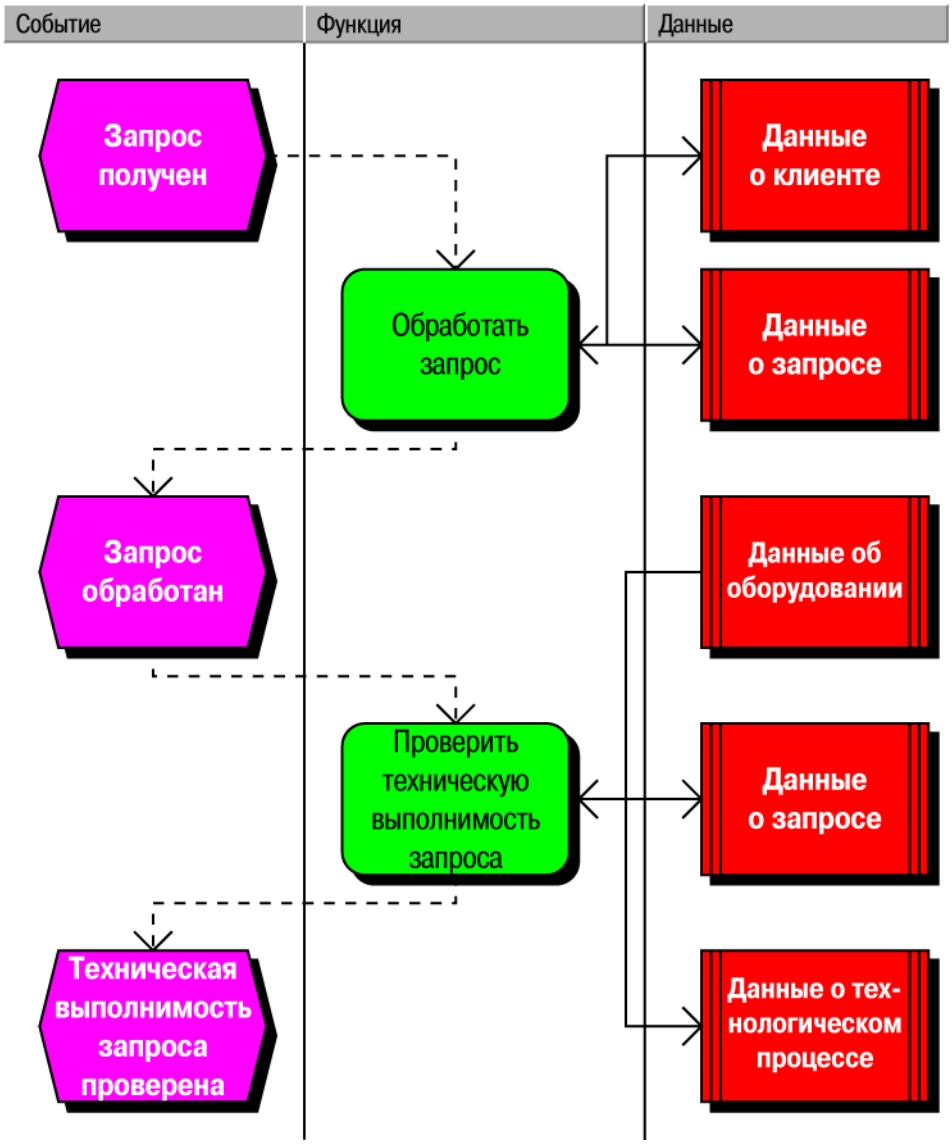


Рис. 4.4.1-19. Диаграмма PCD с данными входа/выхода

4.4.1.2.3. Диаграмма информационных потоков

Диаграммы информационных потоков (ИП) удобны для отображения потока данных между функциями, как это обсуждалось выше. В диаграмме ИП две функции могут быть взаимосвязаны с помощью объекта *поток данных*. Этот объект отражает тот факт, что существует поток данных от исходной функции к результирующей функции. Для более точного определения объекта потока данных между двумя описываемыми функциями его можно представить в виде иерархической структуры, что в свою очередь позволяет связать модель данных с этим объектом.

Эта модель данных представляет информационные объекты, которыми обмениваются функции между собой. В зависимости от степени детализации рассматриваемых функций информационные объекты могут представлять кластеры данных, типы сущностей или ERM-атрибуты. Пример такого типа представления приведен на рис. 4.4.1-20.

Рис. 4.4.1-20. Диаграмма информационных потоков

4.4.1.2.4. Диаграмма событий

События отражают тот факт, что состояние информационных объектов изменилось. Таким образом, с каждым событием связываются конкретные информационные объекты модели данных, причем событие определяет состояние (статус) этих информационных объектов на заданный момент времени.

Вначале события описываются в общем, виде на самом верхнем уровне выполняемого процесса по методу сверху-вниз (например, *Заказ клиента обработан*). Следующий шаг при моделировании процесса заключается в детальной спецификации событий. Если они упорядочены каким-либо способом, наступление событий фиксируется на самом верхнем уровне описания. Если, например, произошли события *Данные клиента проверены*, З*аказ зарегистрирован*, *Позиции заказа обработаны* и *Техническая выполнимость проверена*, то суммарно они определяют состоявшееся событие *Заказ клиента обработан*.

С помощью диаграммы событий можно представить указанную связь событий на верхнем и более детальном уровнях моделирования. Для этой цели можно связать диаграмму событий с событием на верхнем уровне (иерархии!), что позволяет

отображать события и связи между ними на детальном уровне, с включением правил для операторов, связывающих события. Более того, можно включить в этот тип диаграмм информационные объекты из модели данных и связать их с событиями. Таким образом, будет установлено, какое событие определяет изменение состояния данного информационного объекта.

На рис. 4.4.1-21 приведен соответствующий пример.

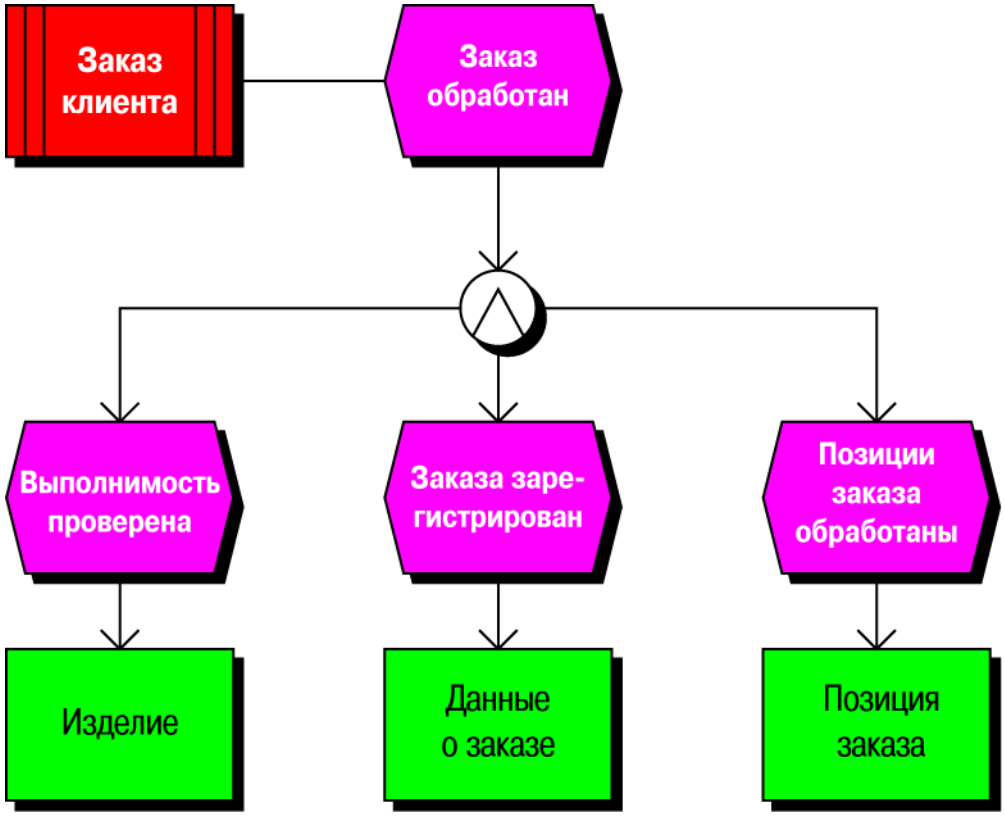


Рис. 4.4.1-21. Диаграмма событий

4.4.1.3. Функции. Организационные единицы. Данные

4.4.1.3.1. Диаграммы eEPC/PCD

Для лучшего понимания того, как моделировать функции, организационные единицы и данные, в настоящей главе обсуждаются модели типа PCD и eEPC. Диаграммы обоих типов описывают одни и те же вещи.

До сих пор мы имели дело только с двумя моделями; теперь вводится третья. Это означает, что частные модели цепочки событий вновь объединяются в обобщенную модель, и теперь мы будем исследовать взаимосвязь всех компонент архитектуры ARIS. Цепочка процесса, с которой мы начинали, вновь будет рассматриваться, но уже более подробно. Анализ детальных характеристик объектов, извлеченных из частных моделей, нам не потребуются, но будут изучаться связи между этими объектами.

На рис. 4.4.1-22 показана цепочка процесса, включающая все виды моделей. События, представляющие объекты модели данных, размещены в первом столбце. Стрелки оканчиваются во втором столбце, который содержит функции процесса. Таким образом, первый и второй столбцы определяют управление событиями. Объекты данных размещены в третьем столбце, где отображены их связи с отдельными функциями. Второй и третий столбцы PCD определяют потоки данных в цепочке процесса.

В отличие от диаграммы PCD, введенной в разделе 3.2, диаграмма цепочки процесса на этапе формулировки требований не имеет столбцов для определения типа обработки и ИТ-систем. Это необходимо для того, чтобы отобразить фактическую ситуацию в компании. Однако эти диаграммы не относятся к предметно-ориентированному описанию бизнес-процессов.

В четвертом столбце определены организационные единицы из организационной модели, которые ответственны за выполнение отдельных функций в цепочке процесса.

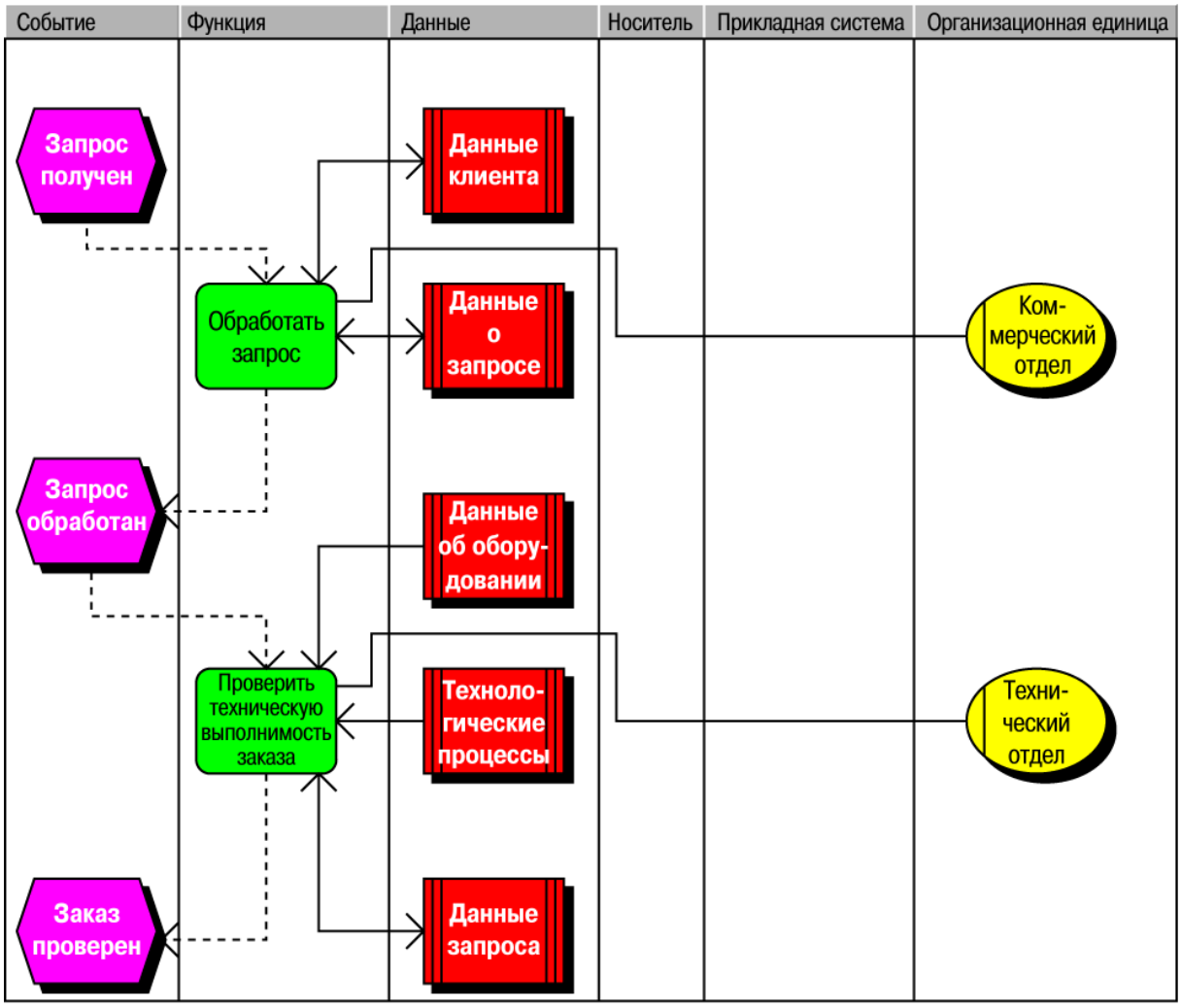


Рис. 4.4.1-22. Цепочка процесса (формулировка требований)

Цепочку процесса, приведенную на рис. 4.4.1-22, можно также представить на диаграмме EPC (см. рис. 4.4.1-23).

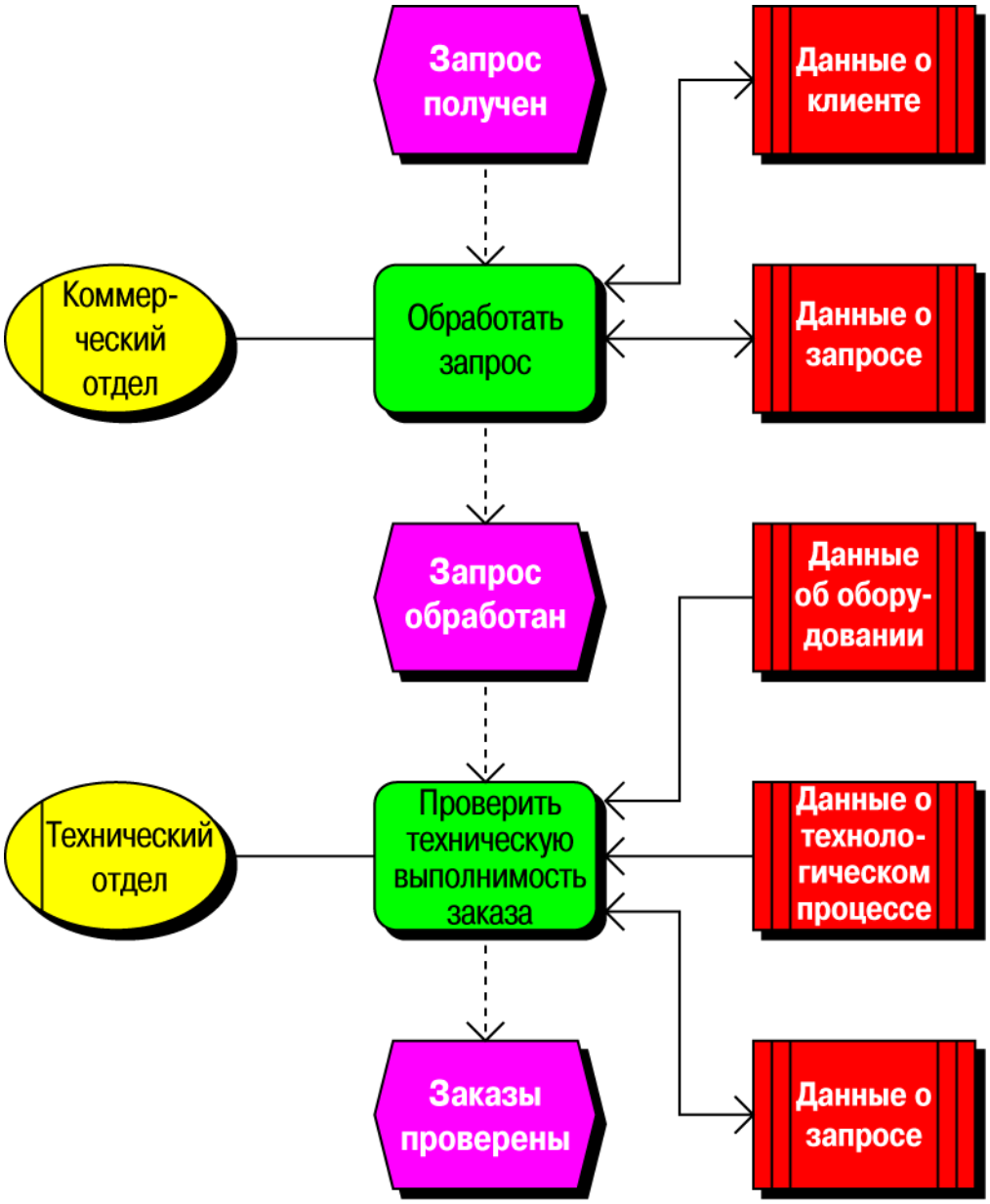


Рис. 4.4.1-23. Диаграмма EPC с функциями, данными, организационными единицами и событиями

4.4.1.3.2. Диаграмма цепочки добавленного качества

Все рассмотренные выше диаграммы цепочки добавленного качества служили для идентификации тех функций, которые непосредственно участвовали в создании

добавленного качества компании. Эти функции могут быть связаны в последовательность, которая, по существу, представляет собой цепочку добавленного качества. Пример диаграммы цепочки добавленного качества приведен на рис. 4.4.1-24.

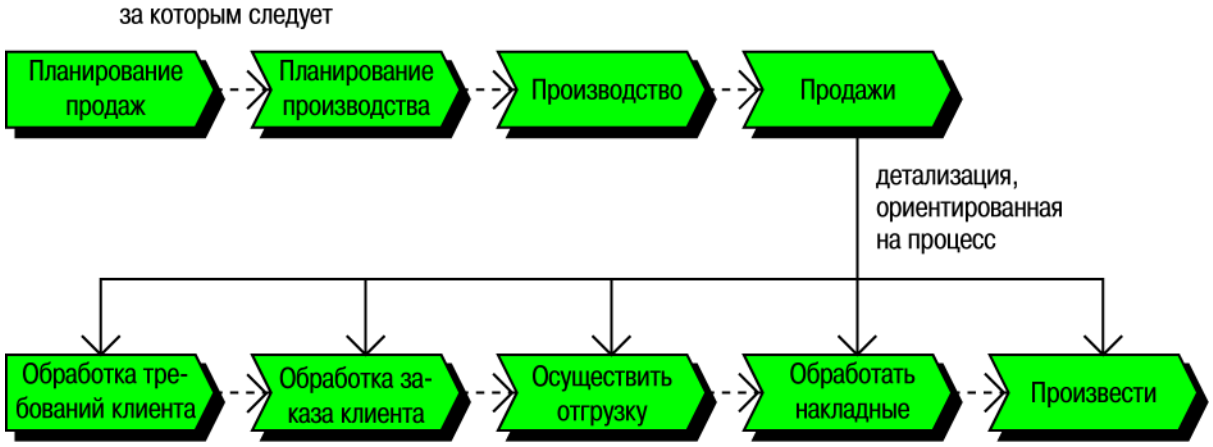


Рис. 4.4.1-24. Диаграмма цепочки добавленного качества

В диаграмме цепочки добавленного качества функции могут быть упорядочены иерархически, что аналогично функциональному дереву. Такое представление всегда включает отношения старшинства и подчиненности, ориентированные на процесс.

Диаграмма цепочки добавленного качества позволяет отобразить не только старшинство и подчиненность функций, но и их связи с организационными единицами и информационными объектами. При связывании организационных единиц с функциями мы делали различие – аналогично цепочке процесса – между предметно-ориентированным назначением функции, ее ответственностью за ИТ и фактическим выполнением.

В английской версии руководства «Методы ARIS» содержится описание дополнительных отношений в диаграмме цепочки добавленного качества.

4.4.1.3.3. Диаграмма правил

В цепочке процесса можно использовать правила в виде операторов для спецификации тех операторов, которые связывают события и функции. Представление этих правил для отображения логических операторов часто довольно сложно, особенно в тех случаях, когда операторам приписаны правила.

Избежать значительного усложнения цепочки процесса с включением правил, позволяет оператор общего правила в eEPC или PCD. Этот оператор можно связать с

диаграммой правил (иерархически!), которая описывает все детали сложного правила (см. рис. 4.4.1-25).

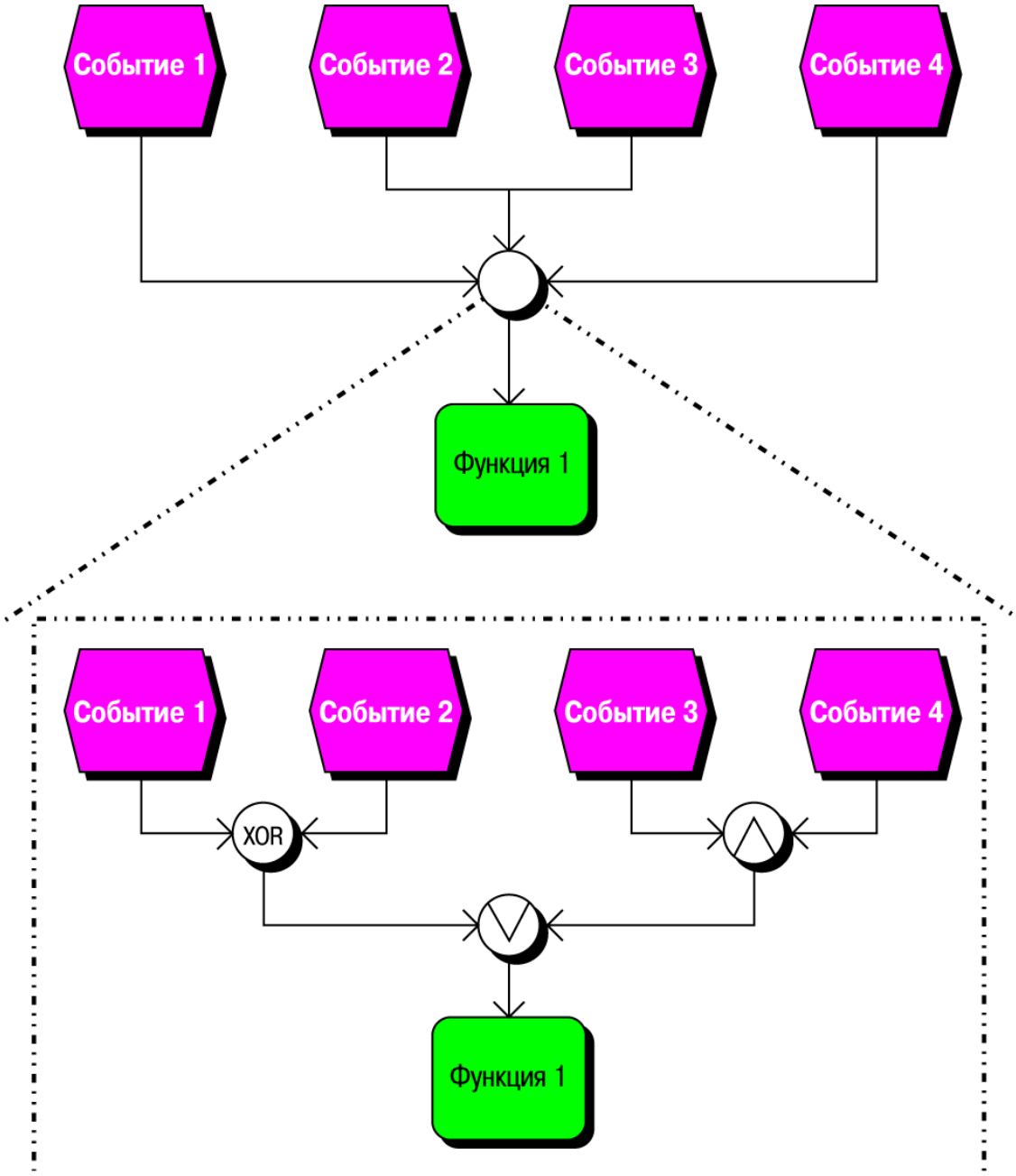


Рис. 4.4.1-25. Представление сложных операторов в диаграмме правил

4.4.1.3.4. Диаграмма коммуникаций

Большие модели-прототипы содержат огромное разнообразие моделей процессов. Включение в эти модели процессов элементов организационной модели позволяет определить, какие компоненты имеют связь с другими компонентами в рамках выполнения процесса. Диаграмма коммуникаций предоставляет возможность группировать все процессы в соответствии с взаимосвязями организационных единиц.

Следовательно, диаграмма коммуникаций отражает организационные единицы, которые взаимодействует друг с другом. Например, организационная единица *Продажи* взаимодействуют с организационной единицей *Клиент* через тип объекта *Взаимодействие*. Объекты типа *Взаимодействие* можно описать иерархически. Они могут быть связаны с диаграммой типа *Обработать матрицу выборки*. Эта диаграмма отражает все процессы, в которых отдел продаж взаимодействует с клиентом.

4.4.1.3.5. Диаграмма классификации

Диаграмма классификации позволяет классифицировать функции, привязывая их к классам типов объектов. Классификация может проводиться по различным критериям. Для спецификации критериев классификации имеется возможность связать тип объекта *Класс типов объектов* с типом объекта *Критерий классификации*.

4.4.1.3.6. Диаграмма входа-выхода

Диаграмма входа/выхода дает возможность описать входные и выходные данные, а также носители информации. Для данного типа модели в каждую ячейку может быть помещен только один графический символ, то есть каждое поле отделяется от других полей сплошной линией. Верхняя строка содержит данные или носители инфомации, которые создаются функцией (являются ее выходом). Аналогично, в левой колонке находятся данные или носители информации, которые «входят» в функцию (являются ее входом).

Если для некоторой функции необходимо представить несколько входных и-или выходных символов, они могут создаваться с помощью копий экземпляров функций.

Невидимые (подразумеваемые) отношения *представляет вход для* и *создает выход для* будут созданы автоматически в процессе создания функции и символов, представляющих данные или носители информации в диаграмме входа/выхода.

На рис. 4.4.1-26 приведен простейший пример диаграммы входа/выхода.

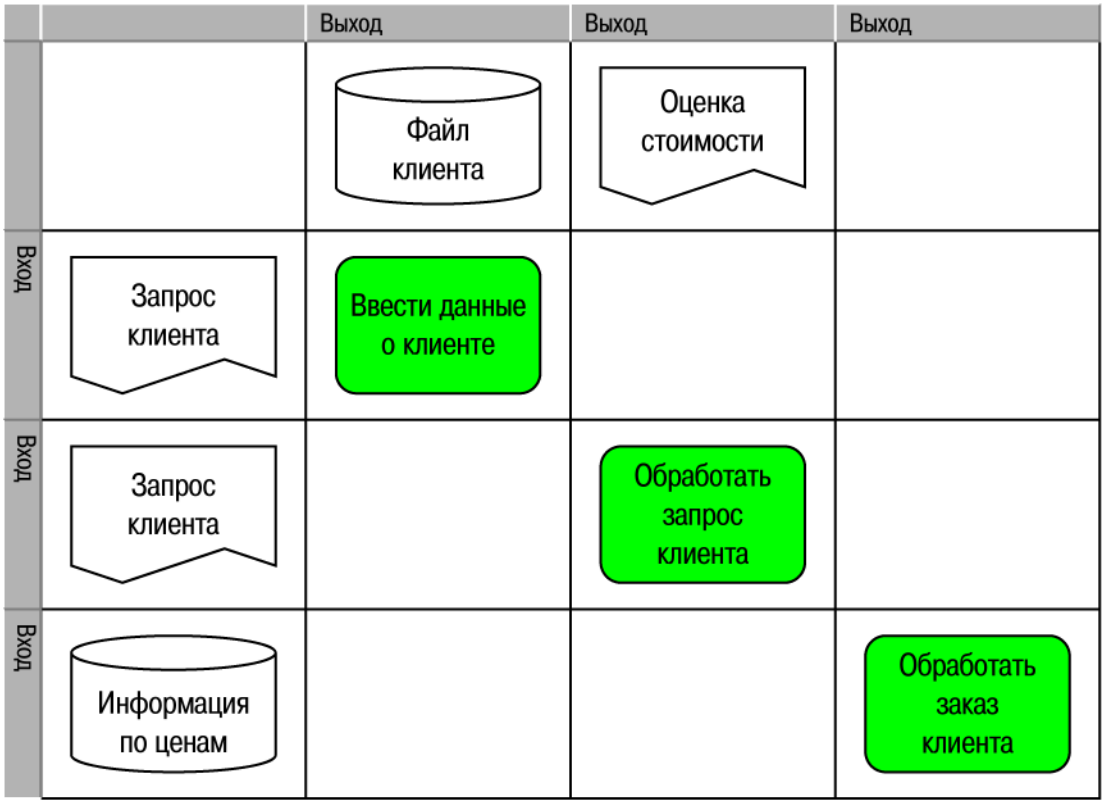


Рис. 4.4.1.-26. Диаграмма входа/выхода

4.4.1.4. Объектно-ориентированное моделирование

4.4.1.4.1. Диаграммы класса

Расширенная объектно-ориентированная концепция ARIS позволяет связать функцию класса с информационными объектами. Это означает, что все информационные объекты, которые определяются в модели данных ARIS, могут получить символ класса, что достигается назначением им функций классов (методов) и описательных атрибутов (содержимое данных). Это описание выполняется с помощью *диаграмм классов*. При привязке диаграммы класса к информационному объекту функция класса назначается этому информационному объекту.

Диаграммы классов однозначно привязываются к отдельным информационным объектам и содержат следующие элементы (графические символы для типов объектов в архитектуре ARIS):

- информационный объект, описанный как класс,
- список атрибутов, привязанный к классу,

- список событий, происшедших согласно конкретному состоянию класса,
- список функций функциональной модели, которые привязаны к классу и переключаются событиями или сами переключают события.

На рис. 4.4.1-27 представлен пример определения класса *Заказ клиента*.

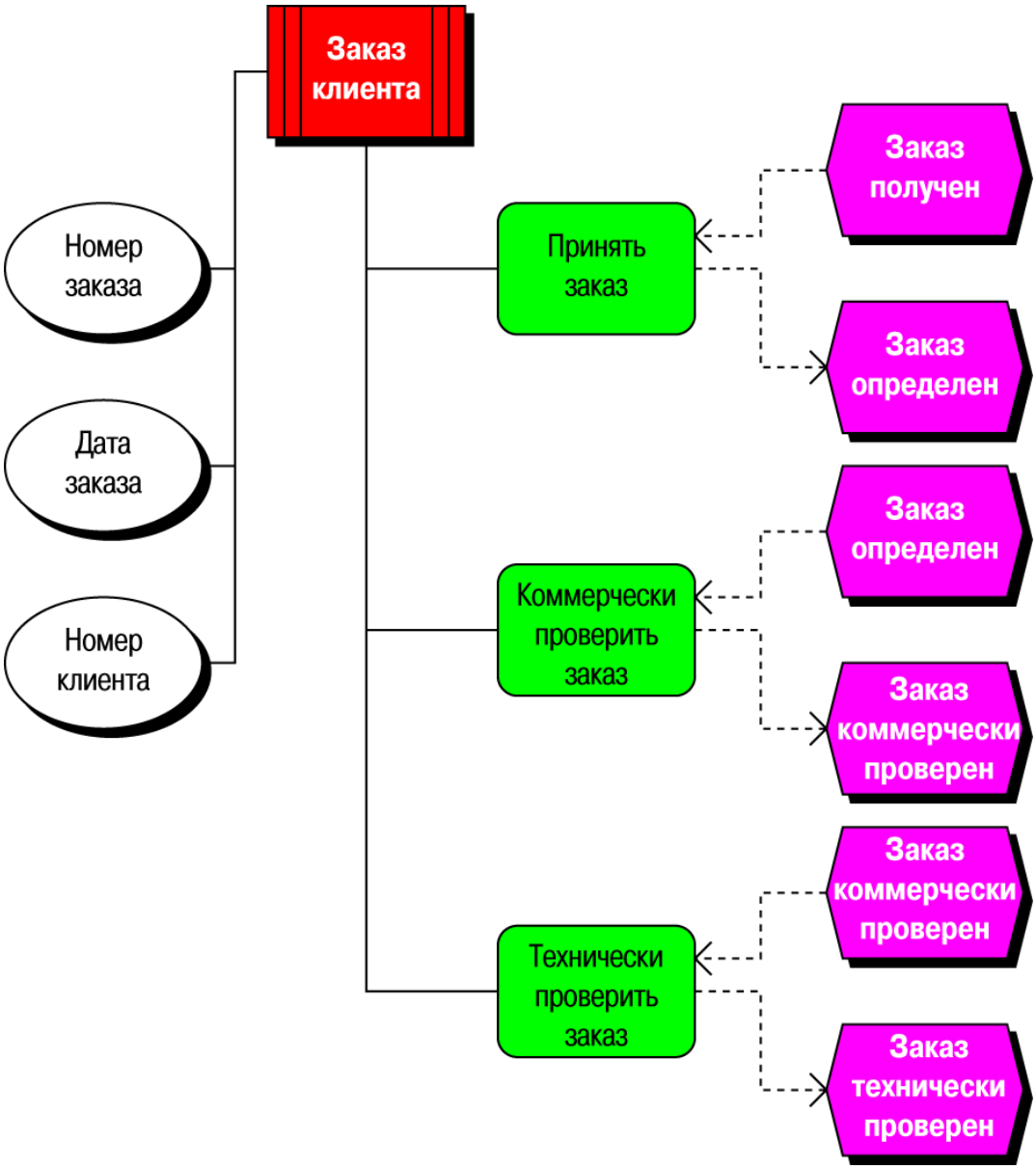


Рис. 4.4.1-27. Диаграмма класса «Заказ клиента»

4.4.1.5. Варианты процесса

4.4.1.5.1. Матрица выбора процесса

Матрица выбора процесса отображает различные сценарии выполнения процесса при помощи привязки основных процессов к отдельным сценариям.

Впоследствии можно определить, какие функции, «проигранные» в сценариях процесса, могут иметь место на предприятии. По этой причине должны быть включены все главные функции (функции сценария) прикладной системы или функции модели-прототипа производства.

При моделировании матрицы выбора процесса используются следующие типы графических символов:

- сценарий,
- процесс,
- главный процесс.

Определение

Сценарий представляет некоторый вариант выполнения процесса в матрице выбора, которая упорядочивает главные процессы в группы.

Определение

Процесс представляет функции из сценария процесса, которые более подробно описаны в модели-прототипе с помощью моделей процесса.

Определение

Главный процесс представляет главные функции в функциональном дереве, с которыми связаны процессы (функции из сценария процессов).

Матрица выбора процесса приведена на рис. 4.4.1-28.

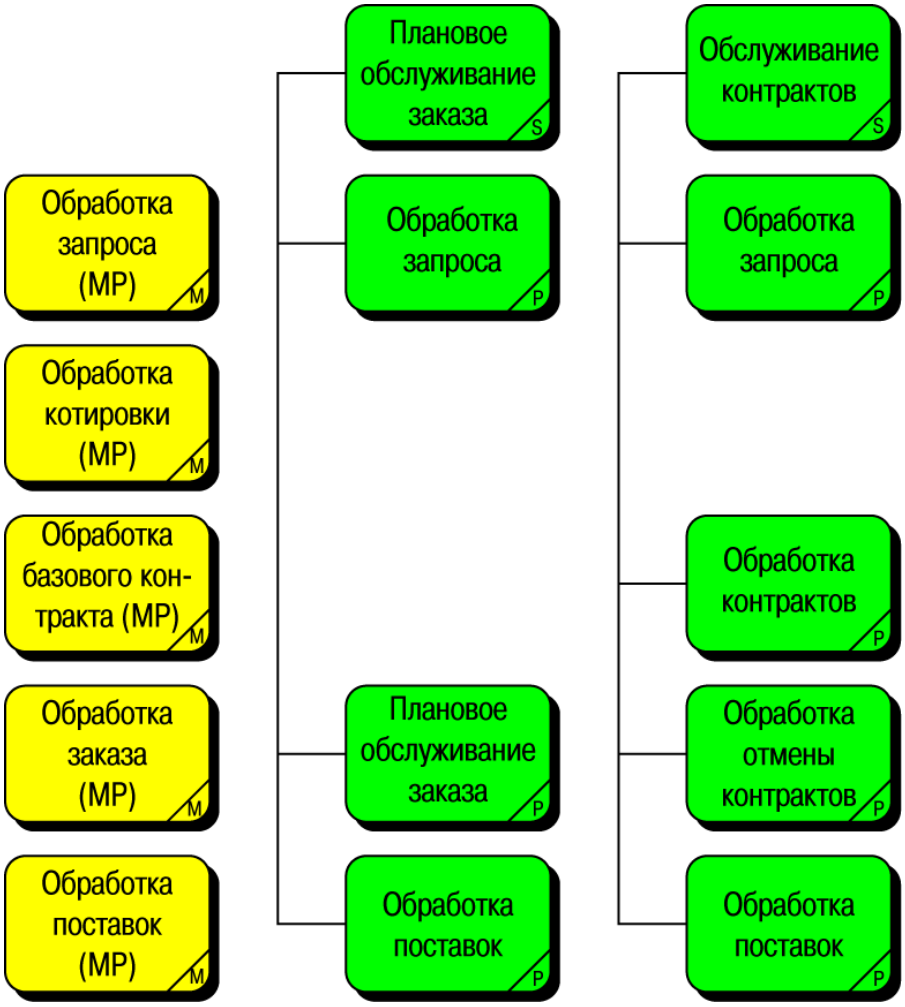


Рис. 4.4.1-28. Матрица выбора процесса (частная модель модели-прототипа SAP R/3)

4.4.1.6. Моделирование потока материалов

Модели процесса (eEPC и PCD) используются не только для отображения информационных потоков, но и для отображения преобразования материалов. В архитектуре ARIS поток материалов в бизнес-процессах представляет диаграмма специального типа - eEPC с потоком материалов, которая является расширением рассмотренной выше диаграммы eEPC.

4.4.1.6.1. eEPC с потоком материалов

Кроме типов объектов eEPC, в диаграмме eEPC с потоком материалов используются также следующие типы объектов:

- тип материала,
- тип упаковочного материала,
- тип операционных ресурсов,
- операционный ресурс,
- тип операционного технического обеспечения,
- оперативное техническое обеспечение,
- тип складского оборудования,
- складское оборудование,
- тип транспортной системы,
- транспортная система.

Объект типа *Тип материала* может быть связан с объектом типа *Функция* посредством входящего или выходящего соединения. Если связь осуществляется в виде входящего соединения, то материалы определяются как вход для функции. В этом контексте посредством выбора соответствующего типа соединения можно определить, будет ли функция использовать все материалы, их часть или не будет использовать их вовсе. Выходящее соединение определяет типы материалов, создаваемые функцией.

Для преобразования материалов требуются технические ресурсы. В цепочке процессов их можно связать с объектами типа *Функция*. Для того чтобы специфицировать возможные альтернативные ресурсы, используются два типа соединений – *требуется* и *требуется альтернативно*.

Если материалы должны быть упакованы в рамках выполнения функции, необходимы некоторые типы упаковочных материалов. Для того чтобы специфицировать типы упаковочных материалов, можно смоделировать взаимосвязь между функцией и необходимыми типами упаковочных материалов.

На рис. 4.4.1-29 показана диаграмма eEPC с потоком материалов, типами технических ресурсов и типами упаковочных материалов.

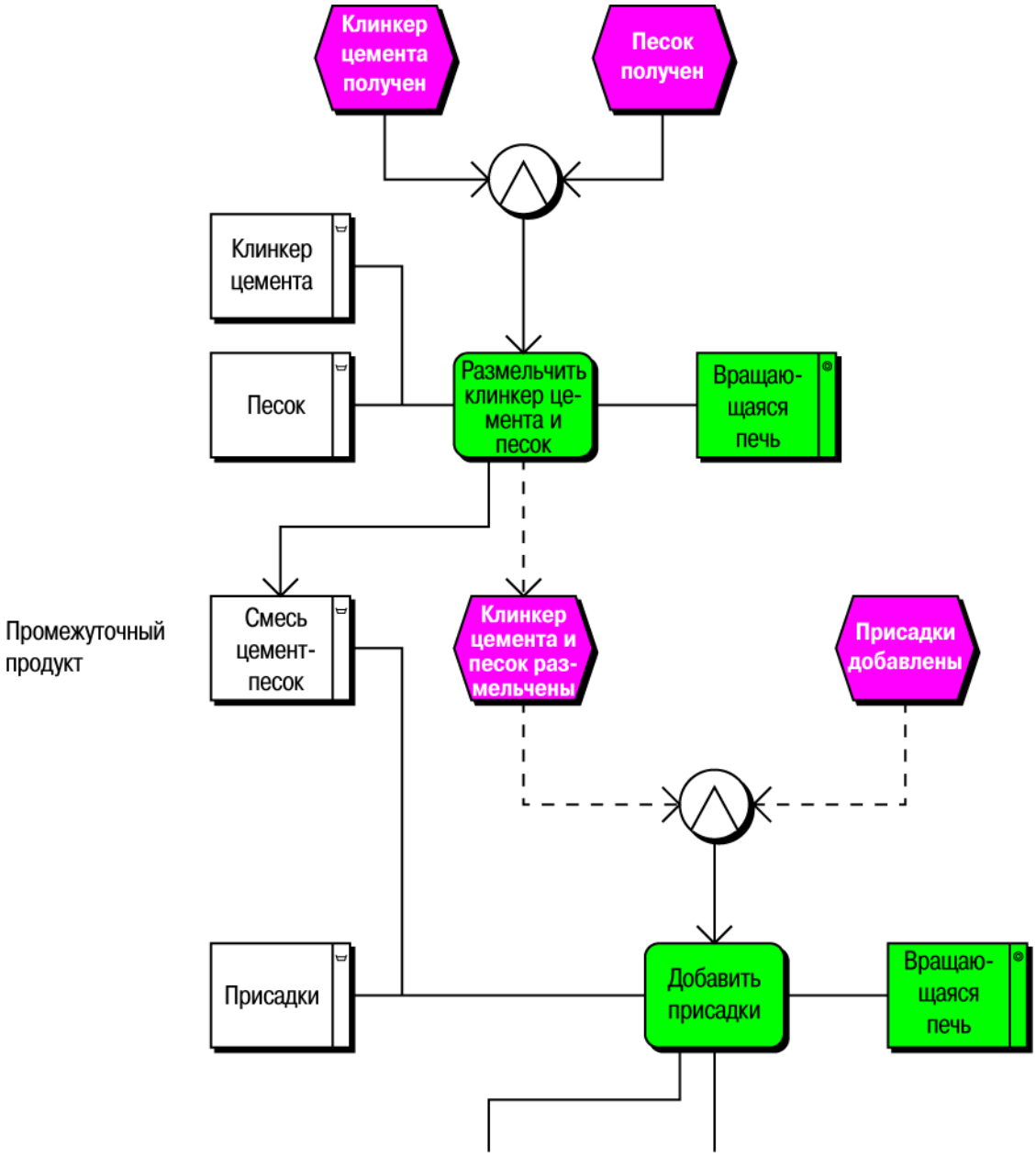


Рис. 4.4.1-29. Диаграмма eEPC с потоком материалов

4.4.1.6.2. Диаграмма потока материалов

Диаграммы потока материалов отображают поток материалов между функциями. Работа с ними во время моделирования аналогична работе с диаграммами

информационных потоков. В диаграмме потока материалов две функции связываются посредством соединения, означающего поток материалов. Это соединение указывает поток материалов от исходной функции к целевой. Если необходимо специфицировать поток материалов между рассматриваемыми функциями более подробно, можно «привязать» диаграмму потока материалов к этим соединениям, выстраивая иерархию их описаний. Данная диаграмма отображает материалы или типы материалов, которыми обмениваются функции.

4.4.1.6.3. eEPC как диаграмма типа Столбец/Строка

Приведенное выше описание аналогично представлению *eEPC-строка*.

Большинство рассммотренных описаний справедливо и для диаграмм типа *eEPC-столбец* с той лишь разницей, что все символы в этой модели располагаются в различных столбцах. Преимущество такого представления состоит в том, что eEPC-диаграмма интерпретируется гораздо проще. Элементы организационной структуры и информационной системы выносятся в заголовок диаграммы. Все другие символы размещаются во второй строке каждого столбца.

Отличительная особенность всех моделей, которые представляются с помощью строк и столбцов, заключается в том, что в них автоматически создаются невидимые (подразумеваемые) отношения. Например, при моделировании прикладных систем и функций подразумеваемое отношение *Поддерживают* автоматически создается по умолчанию в одном из столбцов диаграммы *Представление eEPC-столбец.* Отношение *Выполняет* будет создано автоматически между элементами организационной структуры и функциями. Можно ввести дополнительные столбцы, названия которых будут соответствовать подразумеваемым отношениям. Например:

- вносит вклад в ...,
- решает,
- отвечает с точки зрения ИТ за ...,
- технически отвечает за ...,
- должен быть информирован в случае отмены ...,
- должен быть информирован о результате ...,
- должен быть информирован о ...,
- принимает,
- консультирует по ...

На рис. 4.4.1-30 приведена диаграмма типа *Представление eEPC-столбец.*

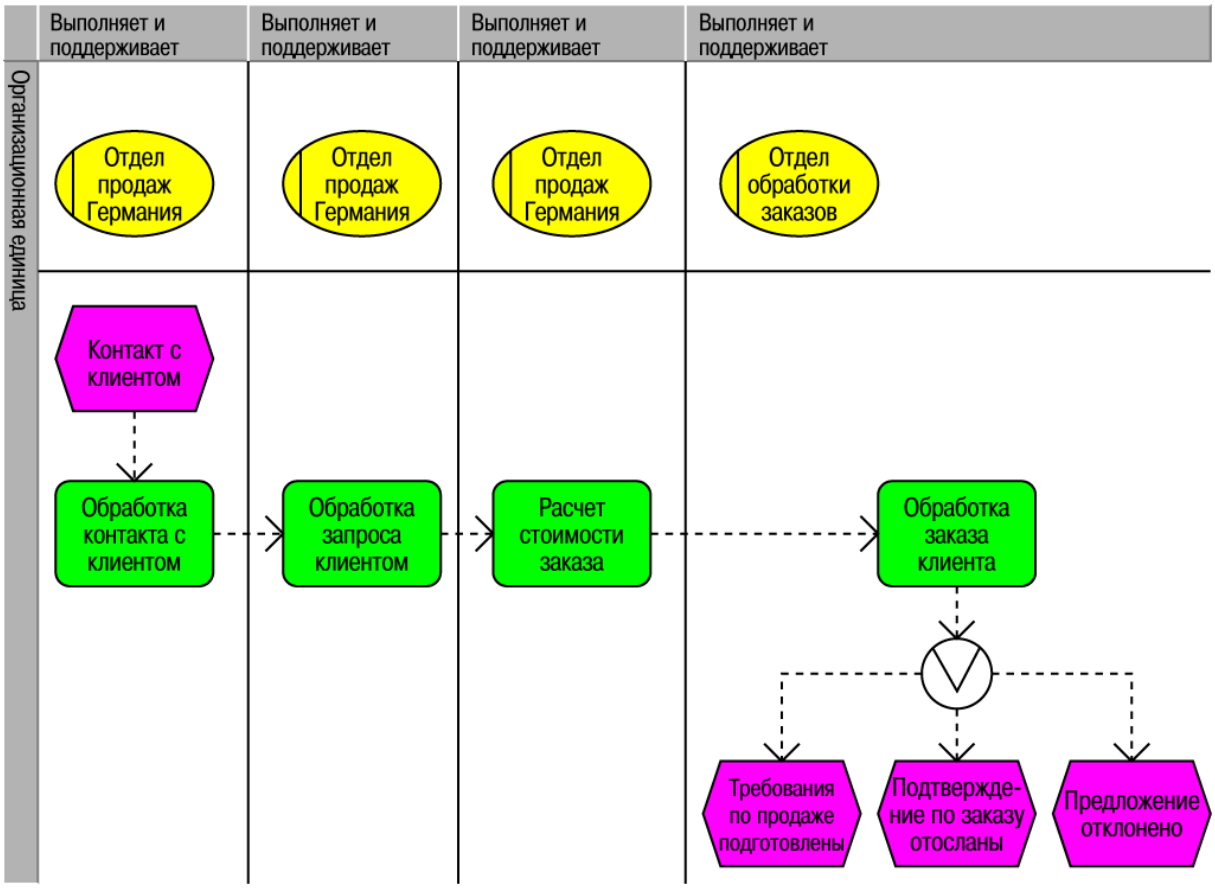


Рис. 4.4.1-30. Представление «eEPC-столбец»

Разница между представлениями eEPC-столбец и eEPC-строка заключается в различных направлениях моделирования. В диаграммах *eEPC-столбец* моделирование ведется сверху вниз, а в диаграммах *eEPC-строка* – слева направо.

4.4.1.6. Модели SAP ALE

Модели типа SAP ALE (SAP Application Linking Enabling) предназначены для моделирования связей между функциями и приложениями.

Модель SAP ALE Filter

Модель SAP ALE Filter указывает на критерий, в соответствии с которым тип функции модет быть использован в системе (в приложении) более одного раза. В этом случае объект *Тип фильтра объекта* служит критерием выбора в рамках дистрибутивной модели.

Модель потоков сообщений SAP ALE (SAP ALE Message Flow Model)

Модель потоков сообщений SAP ALE использует объекты типа *Поток сообщений* для представления потоков сообщений, включая направления потоков между типами систем и типами функций.

Модель типа сообщения SAP ALE (SAP ALE Message Type Model)

Модель типа сообщения SAP ALE использует тип объектов *Тип сообщения* для классификации сообщений, которыми обмениваются распределенные элементы системы. Классификация осуществляется в зависимости от передаваемых данных и транзакций.

4.4.1.8. Другие модели

4.4.1.8.1. Диаграмма управления бизнесом

Диаграмма управления бизнесом описывает все потенциальные риски бизнес-процесса или функции, а также управление ими.

Определение

Риск означает потенциальную опасность для процесса не достигнуть желаемой цели.

Определение

Управление риском – это путь исключения риска или уменьшения его степени.

Определение

Решение по риску означает реализацию управления отдельным риском.

Диаграммы управления бизнесом по структуре аналогична матрице или таблице. Абсцисса соответствует потенциальным рискам процесса, а ордината показывает потенциально возможные способы управления рисками. Решения по рискам в качестве операторов размещаются между риском и управлением риском. В модель дополнительно могут быть включены организационные единицы (с учетом требований пользователей) и документы, которые поддерживают реализацию управления соответствующим риском.

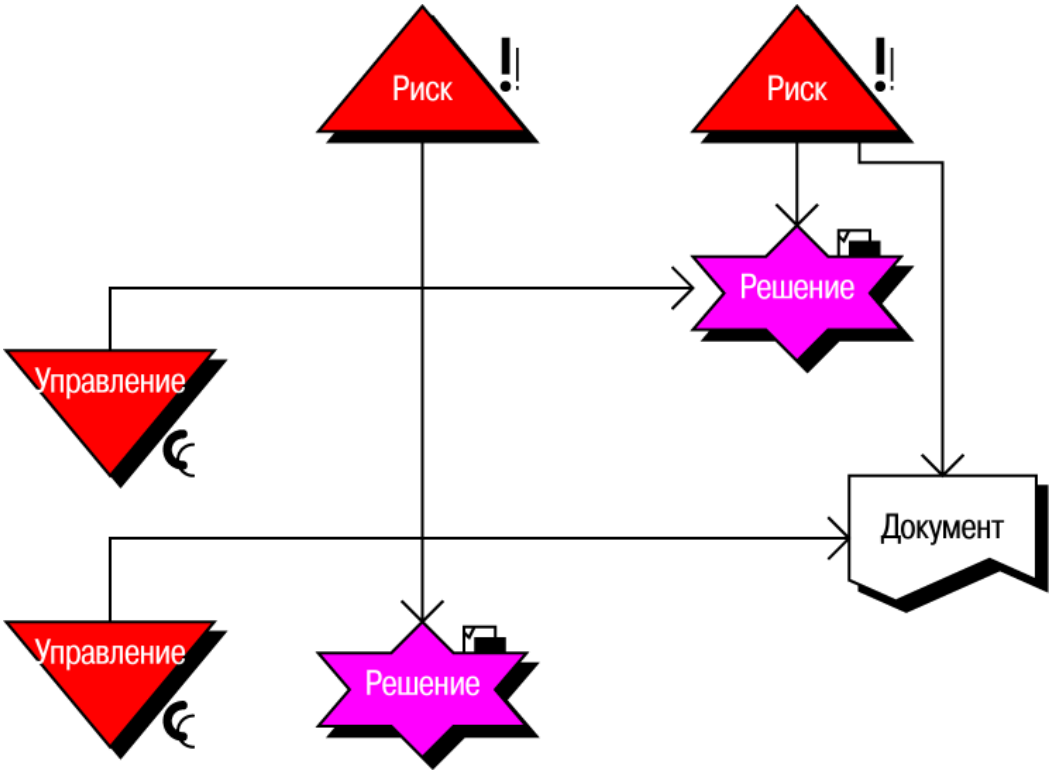


Рис. 4.4.1-31. Диаграмма управления бизнесом

Этот тип диаграммы используется, как правило, для описания стандартных процессов SAP.

4.4.1.8.2. Структурная модель

Структурная модель предназначена в основном для представления иерархиии или другого системного упорядочивания фактов (обощение или детализация фактов).

Определение

Структурный элемент представляет некоторый факт (в соответствии с предполагаемой систематизацией).

Модели, имеющие отношение к фактам, могут быть «привязаны» к структурным элементам, составляющим иерархию фактов.

Структурные модели применяются чаще всего в управлении качеством, особенно для целей сертификации. В этом случае структурная модель показывает норму для

каждой отдельной ее компоненты, а модели, которые наиболее полно удовлетворяют критериям качества, «привязываются» к отдельным структурным элементам.

Отчеты позволяют легко анализировать и оценивать эти факты, а также использовать их при документировании.

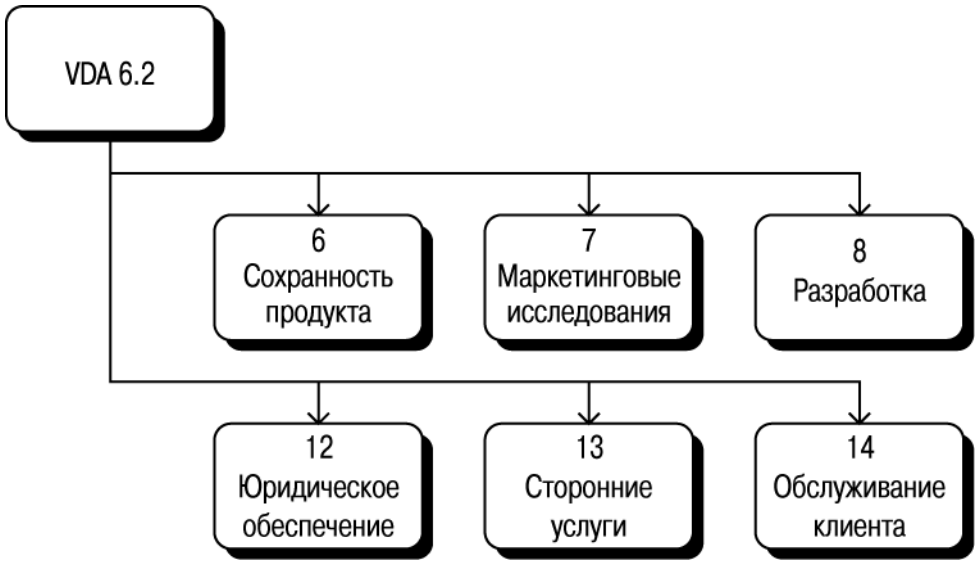


Рис. 4.4.1-32. Структурная модель

4.4.1.8.3. Промышленный и офисный процессы

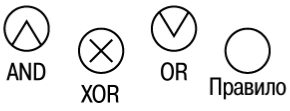
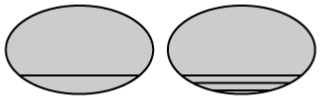
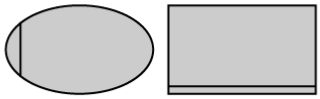
Диаграммы *промышленного процесса* и диаграммы *офисного процесса* описывают практически те же факты, что и диаграмма типа eEPC (или eEPC с потоками материалов). Однако эти две модели используют ограниченное число объектов и символов, которые представляются пиктограммами.

Преимущество пиктографического представления состоит в том, что сотрудники различных подразделений могут не только понимать эти модели без предварительной подготовки, но и самостоятельно их разрабатывать и при необходимости изменять. Например, каждому легко увидеть, что три символа, изображающие человечков, представляют группу. При абстрактной символике eEPC то же самое изображается овалами с двумя линиями и не является очевидным. Диаграммы этих двух типов позволяют проводить моделирование, анализ и оптимизацию процесса непосредственно в подразделениях.

Для максимального соответсвия символов и представляемого содержания используются два метода (типов диаграмм) описания процессов: промышленный процесс, который представляет производственные процессы (создание материальных

продуктов/товаров), и офисный процесс, отображающий процессы, протекающие в офисе (создание нематериальных продуктов/товаров).

На рис. 4.4.1-33 сравниваются символы диаграмм промышленного и офисного процессов с символикой eEPC (или eEPC с потоком материалов).

Тип объекта	Ворзможные символы типов диаграмм		
	eEPC	Промышленный процесс	Офисный процесс
Событие			
Функция		 Событие (Производство)	 Функция (Офис)
Правило	 AND XOR OR Правило		
Тип прикладной системы			
Местоположение	 Рабочее место Местоположение		
Организационная единица			

Тип объекта	Символы возможных типов диаграмм		
	eEPC	Промышленный процесс	Офисный процесс
Группа			
Должность			
Тип сотрудника	Тип сотрудника; Описание должности		
Сотрудник	Внутренний сотрудник; Внешний сотрудник (EXT)	Сотрудник (м); Сотрудник (ж)	Сотрудник (м); Сотрудник (ж)
Категория знаний			
Документированные знания			
Носитель информации		Принтер; Документ; Дискета; Шкаф; Блокнот; Телефон; Полки; Электронная почта	Принтер; Документ; Дискета; Шкаф; Блокнот; Телефон; Полки; Эл. почта; Планировать время; Корзина для мусора; Интернет; Книга; Жесткий диск; CD-ROM; Факс; Письмо; Папка; Магнитная лента
Тип материала			
Тип транспортной системы		Самолет; Грузовик; Транспортная система	
Тип операционного ресурса		Станок; Робот	

Тип объекта	Символы возможных типов диаграмм		
	eEPC	Промышленный процесс	Офисный процесс
Тип технического операционного обеспечения			—
Тип упаковочного материала			—
Складское оборудование			—

Рис. 4.4.1-33. Сравнение символики eEPC и диаграмм промышленного и офисного процессов

Вы можете переходить от одного типа представления к другому в рамках диаграмм этих трех типов, используя соответствующие объекты и копируя содержимое диаграммы одного типа в диаграмму другого типа. В процессе копирования ARIS автоматически конвертирует символы в соответствии с типом диаграммы. На рис. 4.4.1-34 представлено описание одних и тех же фактов диаграммами трех типов – eEPC, промышленного и офисного процессов.

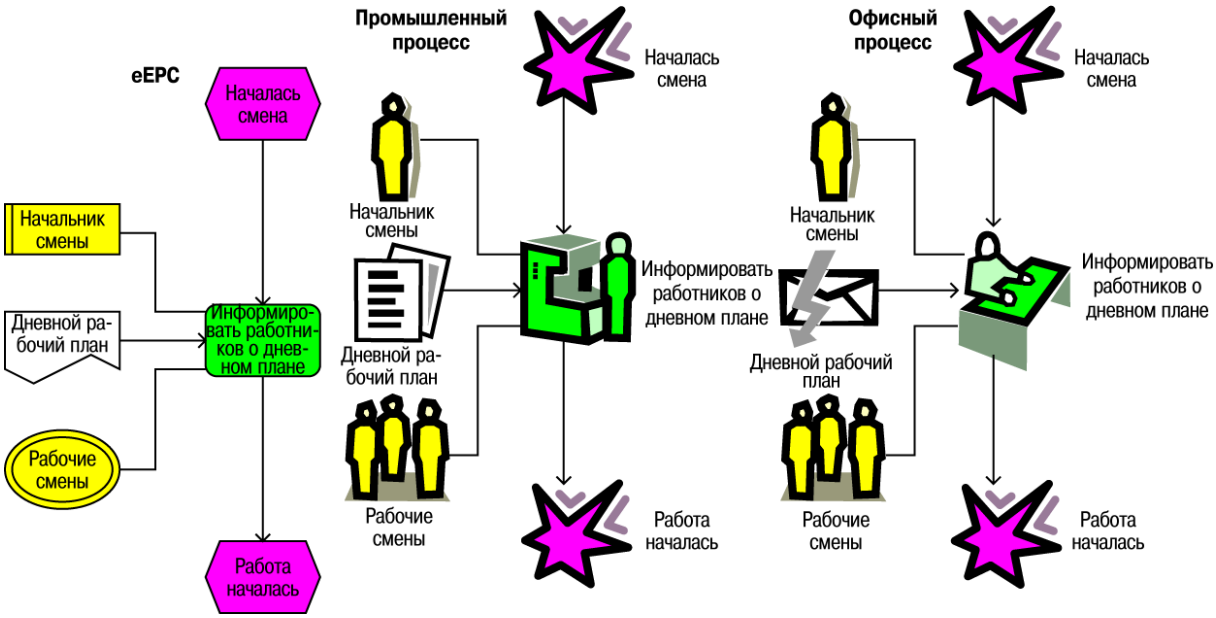


Рис. 4.4.1-34. Представление одних и тех же фактов с помощью диаграмм трех типов - eEPC, промышленного и офисного процессов

4.4.1.8.4. Цепочка процесса проектирования (PPC – Project Process Chain)

Модель типа PPC связывает **ARIS Toolset** и MS Project. В **ARIS Toolset** бизнес-процесс представляется в виде процедурной последовательности функций с помощью событийной диаграммы процесса (eEPC). Однако этот уровень абстракции недостаточен для того, чтобы планировать ресурсы и время в рамках управления проектом, так как в этом случае необходимо описать и специфицировать реальные события и конкретные значения функций. Это можно сделать с помощью диаграммы PPC, в которой используется специальный тип объекта для описания тех или иных состояний вместо типов объектов *Событие* и *Функция.*

Определение

Экземпляр события – это конкретное событие, происходящее при выполнении конкретного процесса. Можно провести оценку экземпляра события, т.е. определить, является он истиной или ложью.

Определение

Экземпляр функции – это конкретная функция, выполняемая в рамках конкретного процесса. С экземпляром функции связывается уникальное время начала и уникальное время окончания ее выполнения, а также другие необходимые атрибуты.

Структурные элементы проекта (экземпляры функций, экземпляры событий, правила и соединительные линии) служат для представления хронологической последовательности в рамках проекта. Диаграмма PPC содержит также объекты *Исполнитель* (внутренний/внешний), *Операционные ресурсы*, *Общие ресурсы*. Эти объекты необходимы для планирования загрузки и времени выполнения проекта.

Определение

Общий ресурс – это ресурс, который детально не описан, и не определено, является ли он операционным или людским ресурсом. Общие ресурсы – это те, которые необходимы для выполнения процедур.

Экземпляр функции в диаграмме PPC можно также определить, используя понятие экземпляра кластера.

Определение

Экземпляр кластера – это экземпляр объекта в модели кластер/данные. Он представляет логический взгляд на объекты или структуры кластеров данных.

В диаграмме PPC используются экземпляры кластеров для представления отношения между экземпляром функции и данными. Модель типа *Диаграмма носителя информации* (см. *модель данных* на уровне описания требований) может быть «привязана» к типу объекта *Экземпляр класса,* для того чтобы показать, на каком информационном носителе хранятся данные.

На рис. 4.4.1-35 приведена диаграмма PPC, полученная в результате конвертирования диаграммы eEPC.

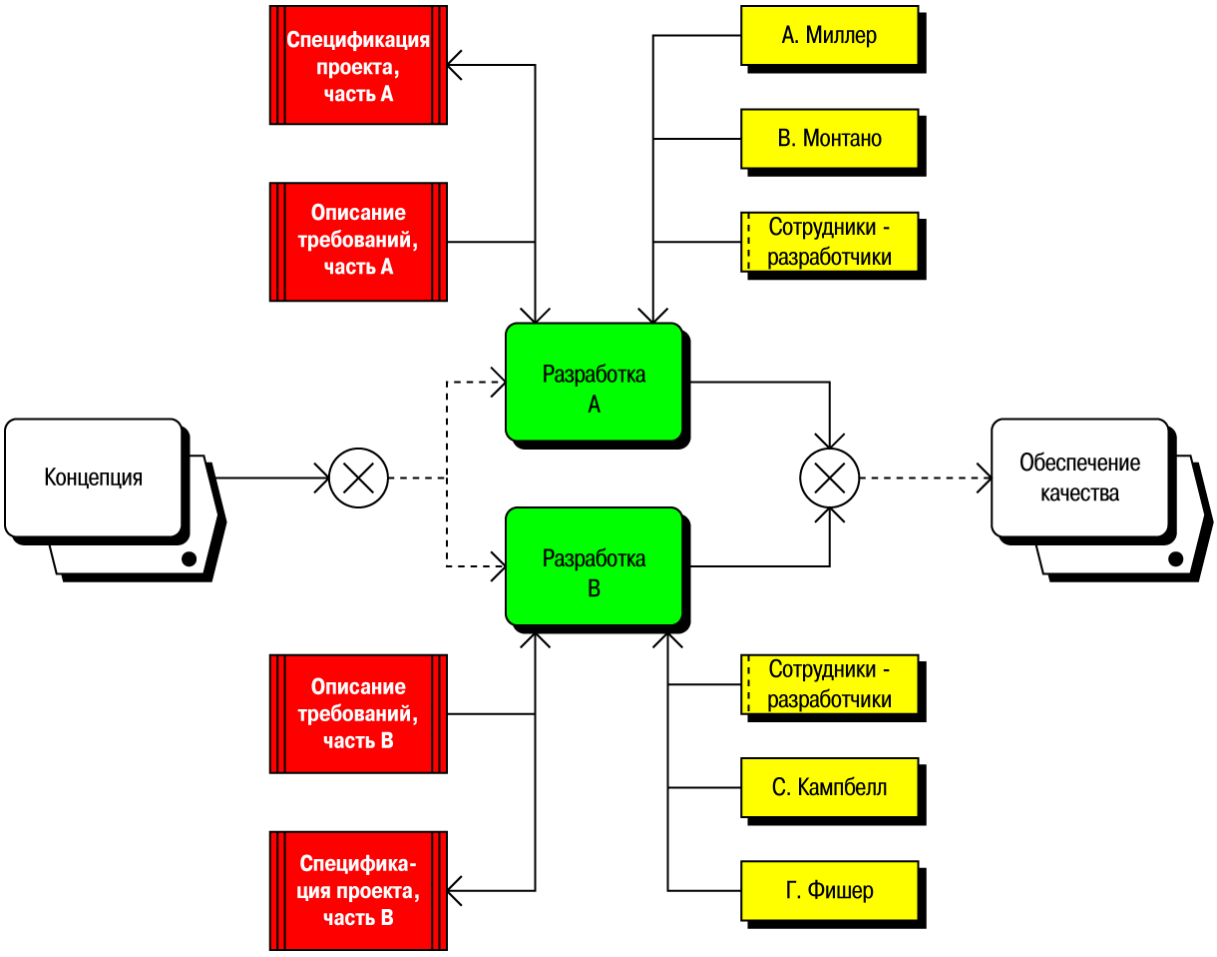


Рис. 4.4.1-35. Диаграммы PPC, полученная в результате конвертирования диаграммы eEPC

В приведенном примере оператор XOR (исключающее ИЛИ) указывает на то, что в этом месте диаграммы eEPC, которая была конвертирована, было ветвление. В диаграмме PPC это интерпретируется как альтернативные пути выполнения проекта, которые должны быть однозначно специфицированы.

Объект, с которым связана модель, идентифицируется черной точкой на нижней линии. В примере на рис. 4.4.1-35 объекты *Конценция*, *Обеспечение качества* и *Описание требований, часть B* – это объекты с привязанными к ним моделями.

Диаграмма типа PPC может быть получена не только путем конвертирования eEPC, но и создана пользователем непосредственно в ARIS Toolset.

4.4.1.8.5. Модель инициации процесса

Основная цель имитационного моделирования заключается в проведении анализа процедур процесса в динамическом режиме. Анализируемые процессы должны быть инициированы (начаты или сгенерированы) стартовым событием. В зависимости от целей и условий моделирования пользователь должен решить, каким образом и как часто необходимо производить инициацию процессов. Кроме того, должны быть средства для назначения приоритетов процессам, например, срочные работы (процессы) должны выполняться в первую очередь.

В ARIS Toolset установка приоритетов реализуется с помощью атрибута *Приоритет* в группе типов атрибутов *Имитация*, описывающих стартовое (начальное) событие. Значение атрибута *Приоритет* передается всем процессам, которые инициируются этим стартовым событием.

Описанные требования выполняются с помощью *модели инициации процесса*, которая представляет собой многоуровневую объектную модель. Объект *Интервал инициации* находится на самом нижнем уровне. Этот объект имеет следующие атрибуты: *Относительное время начала интервала, Продолжительность интервала, Количество экземпляров (прогонов) процесса, Распределение, Циклическая повторяемость, Период*. Объект *Продолжительность интервала* может принимать значение 0. Это необходимо для обозначения некоторого момента времени. Если интервалы описывают более короткие периоды времени, то циклы инициации процесса состоят из последовательности интервалов. Для моделирования дня могут использоваться четыре различных интервала, которые циклически повторяются в течение всего периода имитационного моделирования (например, в течение недели). Также возможно разделить период времени имитационного моделирования на несколько циклов (например, рабочие дни и выходные), каждый из которых будет содержать различные интервалы. План инициации процесса может иметь один или более циклов. Следующий пример иллюстрирует сказанное выше.

Модель процесса представлена диаграммой eEPC с некоторым начальным событием. Для процесса приняты правила: в рабочие дни (понедельник – пятница) в начале каждого рабочего дня 50 процессов начинаются утром в 8.00. С 8.00 до 12.00 и с 13.00 до 17.00 начинаются 20 процессов с аналогичным распределением. С 12.00 до 13.00 ни один процесс не начинается, так же как и после окончания рабочего дня (с 17.00 до 8.00 следующего дня). В субботу с 9.00 до 16.00 начинаются 60 процессов в соответствии с триангулярным распределением. Как правило, ни один процесс не

начинается в воскресенье. Такой еженедельный ритм существует в период с января по декабрь, кроме периода отпусков с июля по август. В этот период никто не работает по субботам. Для приведенного примера может быть сгенерирована модель, представленная на рис. 4.4.1-36.

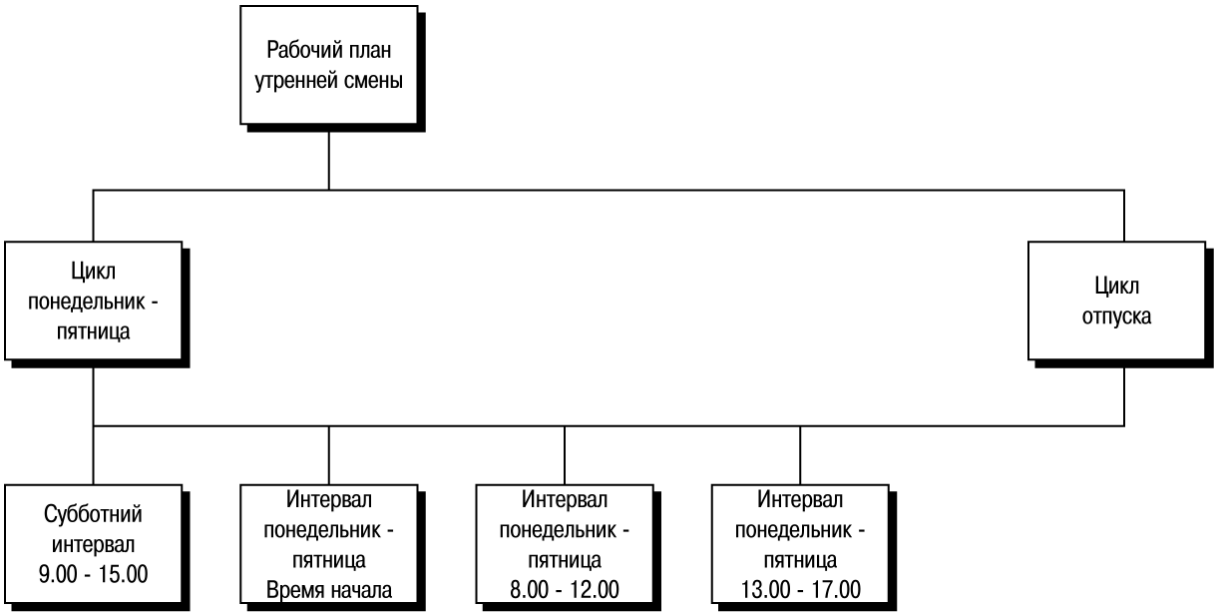


Рис. 4.4.1-36. Модель процесса инициации

4.4.1.8.6. Метод RAMS

Метод RAMS (Requirements Analysis for Manufacturing Systems), или в русском варианте АТПС (Анализ требований для производственных систем), – это метод анализа деятельности компании, предложенный фирмой Digital Equipment.

АТПС – процедура (или модель) для описания и анализа интеграционного потенциала информационных технологий и для разработки различных сценариев по различным решениям с целью облегчить понимание того, насколько информационные системы удовлетворяют требованиям пользователей. Результатом применения методологии АТПС является создание «спецификации требований», которая гарантирует координацию целей компании, бизнес-процедур, информационных потоков и информационных систем.

Модель описывает все подразделения, функции (операции) и существующие прикладные системы, располагая их по диагонали. Эта диагональ является дополнением к матричной форме представления наиболее важных информационных потоков между функциональными подразделениями. В случае необходимости в модель

могут быть введены денежные или материальные потоки, а также потоки наиболее важных товаров.

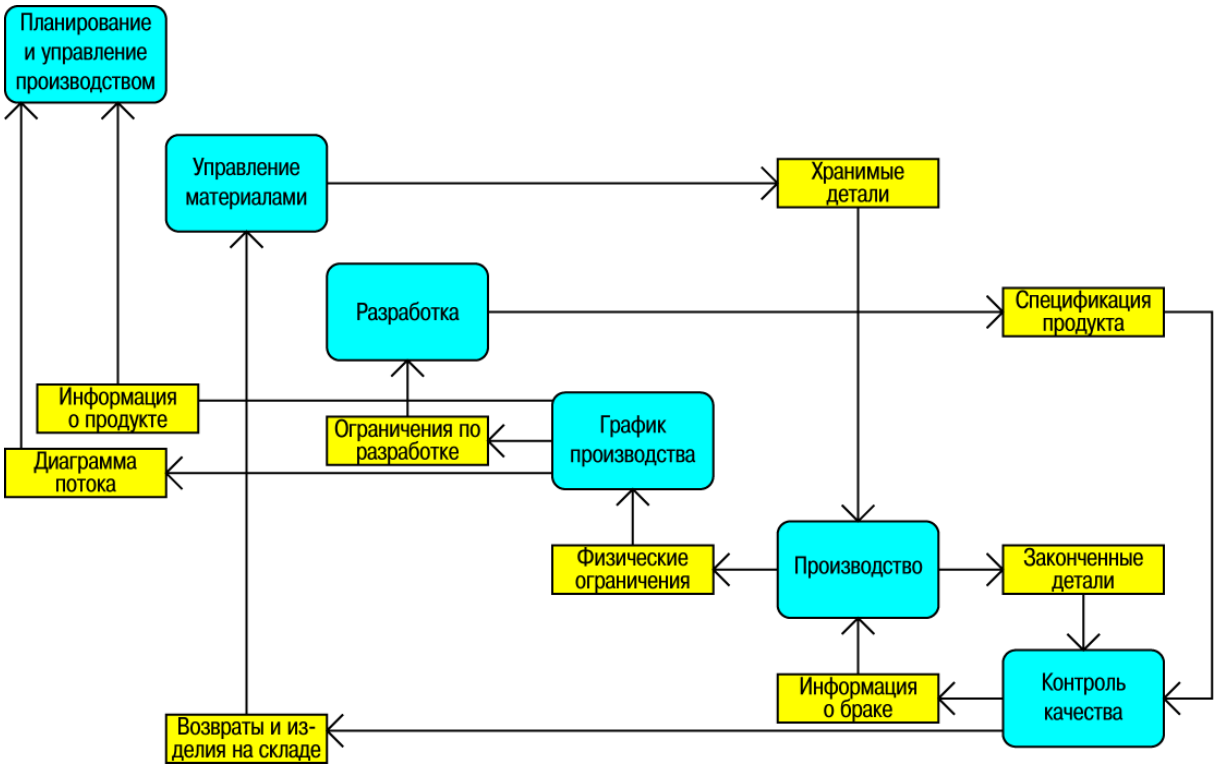


Рис. 4.4.1.-37. АТПС-диаграмма

Транзакционная модель для изучения с помощью методологии АТПС:

Шаг 1: Определяются ожидаемые результаты применения метода АТПС; перечисляются имена участников проекта и время начала проекта.

Шаг 2: Выбор отдельных подразделений, функций и существуюших прикладных систем, которые будут исследоваться в рамках проекта. Они представляются в виде матрицы, на диагонали которой располагаются наиболее важные информационные потоки между функциональными подразделениями. Если необходимо, в диаграмму могут быть добавлены наиболее важные финансовые и/или материальные потоки, а также потоки изделий. Кроме того, могут быть указаны подразделения или функции, для которых требуется более детальный анализ требований.

Шаг 3: После того, как определены рамки АТПС-проекта, проводится детальный анализ отобранных подразделений или функциональных групп в терминах их целей и бизнес-процессов, а также стоящих перед ними задач и связанных с ними требований к информационному обеспечению. Творческий подход к средствам графического представления помогает в изучении сложных процессов и процедур. Очень часто используются оригинальные формы, отчеты или экраны для более ясного представления и четкого изложения исследуемого материала. Важная задача анализа - выявление возможных искажений, разорванности информационных потоков и связей в рамках бизнес-процедур между функциями и информационными системами. Исследуется также потенциал различных видов усовершенствований (изменений) существующих форм работы и систем. В процессе анализа представляются и обсуждаются проблемы, а также возможные (наилучшие) пути их решения. Эта информация затем структурируется и детально изучается в аспекте ее влияния на деятельность компании. Если в процессе изучения делается вывод о высоком потенциале отдельных усовершенствований, они детально описываются с оценкой их полезности.

Шаг 4: Результаты детального анализа служат базисом для последующей спецификации требований. Круг проблем теперь очерчен, и могут быть предложены новые идеи и альтернативные пути их решения, спектор которых может варьироваться от сложных систем до простейших изменений процесса, причем эти решения необходимо рассматривать во взаимосвязи с предыдущим начальным состоянием. Сравнение с начальным состоянием должно быть проведено для всех функций и операций в соответствующих сферах деятельности. Результаты изучения могут иметь широчайший спектр: от оценки функциональной наполненности основных решений до формулировки требований пользователей.

Шаг 5: Результаты, полученные на каждом шаге анализа, должны учитываться при формулировке окончательного варианта спецификации требований. Вся детально проверенная информация и выработанные на основе анализа предложения должны быть суммированы в отчете, что позволит сформировать базис для будущих требований к системе. Для реализации предложенных решений на следующем шаге создается функциональная спецификация системы.

4.4.2. Спецификация проекта

4.4.2.1. Диаграмма доступа

Последующие разделы содержат описание взаимосвязей между отдельными объектами, рассматриваемых при спецификации проекта для других моделей. Эти взаимосвязи могут быть включены в диаграмму доступа управляющей модели. Для того чтобы сделать эти представления более прозрачными, двусторонние взаимосвязи будем анализировать отдельно.

4.4.2.1.1. Объединение функций и данных

В качестве первого приближения могут быть определены потоки данных между прикладными системами, типы модулей или ИТ-функции. Для этой цели поток данных в модели размещается между отдельными прикладными системами или типами модулей. Посредством привязки потока данных к eERM-диаграмме, диаграмме отношений или табличной диаграмме можно более точно определить объекты данных, передаваемые между системами различных типов.

Таким образом, объекты потока данных могут быть локализованы или на уровне формулировки требований, или на уровне спецификации проекта, или на уровне описания реализации (см. на рис. 4.4.2-1).

Рис. 4.4.2-1. Поток данных между прикладными системами различных типов

Кроме потока данных, в качестве объектов данных на уровне формулировки требований или спецификации проекта могут рассматриваться данные входа/выхода для каждого типа прикладной системы, каждого типа модулей и каждого типа ИТ-функций. Стрелок указывают направление потока данных, т.е. определяется, имеем ли мы дело с входом или выходом данных (рис. 4.4.2-2).

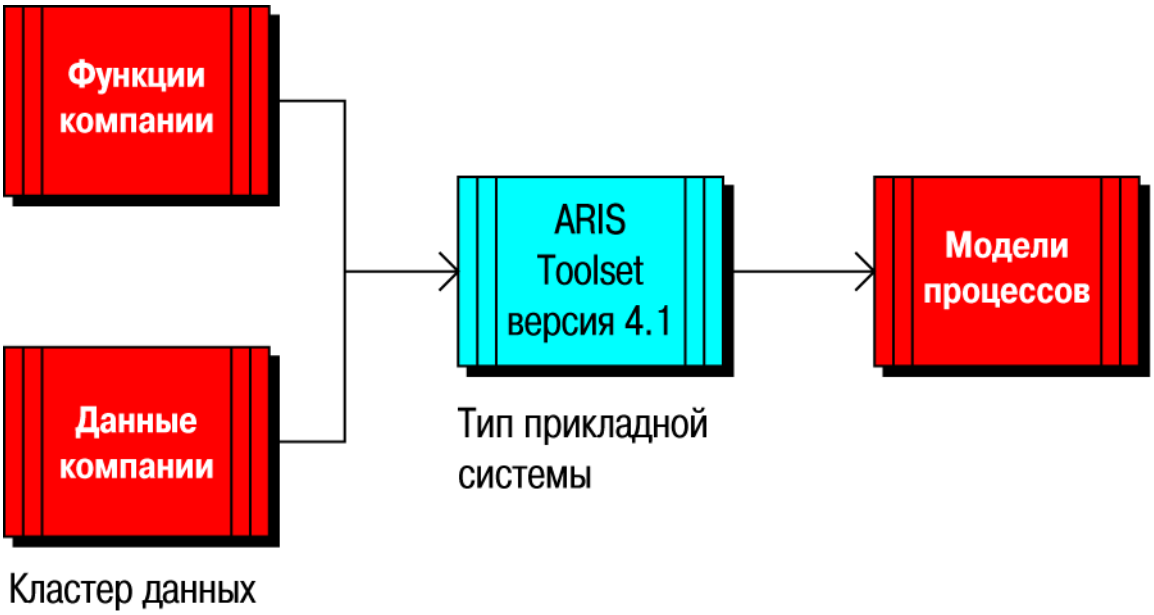


Рис. 4.4.2-2. Вход/выход данных на уровне спецификации проекта

4.4.2.1.2. Объединение организационных единиц и данных

При объединении модели данных и организационной модели на уровне спецификации проекта должны быть решены две основные задачи. С одной стороны, необходимо определить, какие организационные единицы ответственны за конкретные объекты данных в компании. С другой стороны, важно указать на более общем уровне, каким организационным единицам разрешен доступ к отдельным объектам данных.

Таким образом устанавливаются взаимосвязи между предметно-ориентированными объектами организационной модели (организационные единицы, должности, типы людских ресурсов, людские ресурсы и т.д.) и объектами данных из диаграммы отношений на уровне спецификации проекта (отношение, атрибут, модель). Следовательно, эти взаимосвязи привязаны к управляющей модели на уровне спецификации проекта.

Для авторизации доступа к отношениям или отдельным полям объектам данных могут быть присвоены либо должности, либо типы людских ресурсов. В действительности это не означает, что определенная должность позволяет иметь доступ к отдельным полям. При помощи привязки типов людских ресурсов может быть определено бизнес-правило: *к этому полю могут иметь доступ только руководители отделов*. Пример, иллюстрирующий сказанное, приведен на рис. 4.4.2-3.

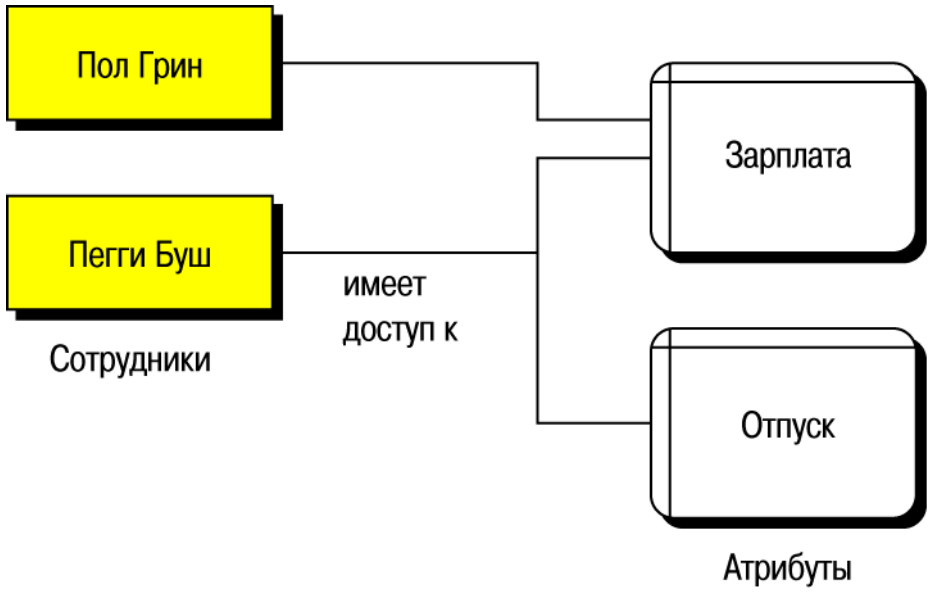


Рис. 4.4.2-3. Авторизация доступа

Кроме авторизации доступа, в той же степени важным является ясное определение ответственности за содержимое поля или всего отношения. По этой причине в диаграмму отношений может быть введено второе соединение, называемое *быть ответственным за* и расположенное между организационной единицей и объектом данных.

Немотря на то, что авторизация доступа может быть введена для нескольких должностей, ответственность за объекты данных в основном присваивается только одной должности в компании. При помощи привязки типов людских ресурсов здесь также могут быть определены бизнес-правила, относительно ответственности за объект данных (см. на рис. 4.4.2-4).

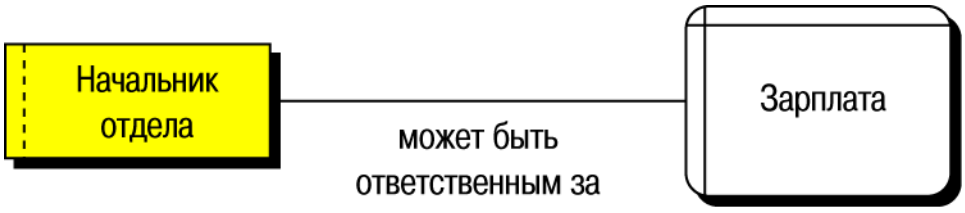


Рис. 4.4.2-4. Определение ответственности

4.4.2.1.3. Объединение организационных единиц и функций

Чтобы понять, как организационные аспекты связаны с функциональными аспектами, которые в основном определены на уровне спецификации проекта, нужно ответить на следующие вопросы:

- Кто (организационная единица, должность, людские ресурсы и т.д.) несет ответственность за типы прикладных систем и типы модулей, которые были определены в функциональной модели на уровне спецификации проекта, и кто использует эти системы?
- На каких рабочих местах компании (организационная модель) какие типы прикладных систем и типы модулей должны используются?
- Какие платформы, доступные в компании (типы компонент аппаратного обеспечения (организационная модель)), для каких типов прикладных систем являются подходящими?

Для того чтобы ответить на первый вопрос, в диаграмме доступа могут быть прорисованы соединения между элементами организационной схемы (организационные единицы, должности и людские ресурсы) и объектами диаграммы прикладной системы (тип прикладной системы, тип модуля, ИТ-функции и т.д.). Поступая таким образом, можно более точно определить смысл этого отношения. Будем различать следующие ситуации:

- организационная единица может быть *ответственна* за тип прикладной системы в той степени, в какой это касается *предметно-ориентированных* аспектов;
- организационная единица может быть *ответственна* за *разработку* типа прикладной системы;
- организационная единица может быть *пользователем* типа прикладной системы.

Вопрос местонахождения (рабочего места) может быть разрешен посредством установки связей элементов организационной модели с типами прикладных систем, типами модулей и типами ИТ-функций.

При спецификации проекта мы имели дело не с индивидуальными прикладными системами (в смысле индивидуальных лицензий на них), а с типами прикладных систем. Таким образом, эти отношения не определяют реальное местоположение прикладной системы (привязка этого типа реализуется на уровне описания реализации), но указывают возможные местоположения прикладной системы конкретного типа.

Типы компонент аппаратного обеспечения (АО), доступные в компании, определяются на уровне спецификации проекта в организационной модели. В управляющей модели устанавливаются отношения между этими типами компонент

АО и типами прикладных систем. Таким образом, определяются аппаратные платформы, на которых могут работать определенные типы прикладных систем, типы модулей или типы ИТ-функций. На этом этапе могут быть также описаны типы персональных компьютеров, операционных систем и СУБД, включаемые в функциональную модель.

В английской версии руководства «Методы ARIS» содержится список всех возможных отношений в диаграмме доступа.

На рис. 4.4.2-5 приведен пример отношений.

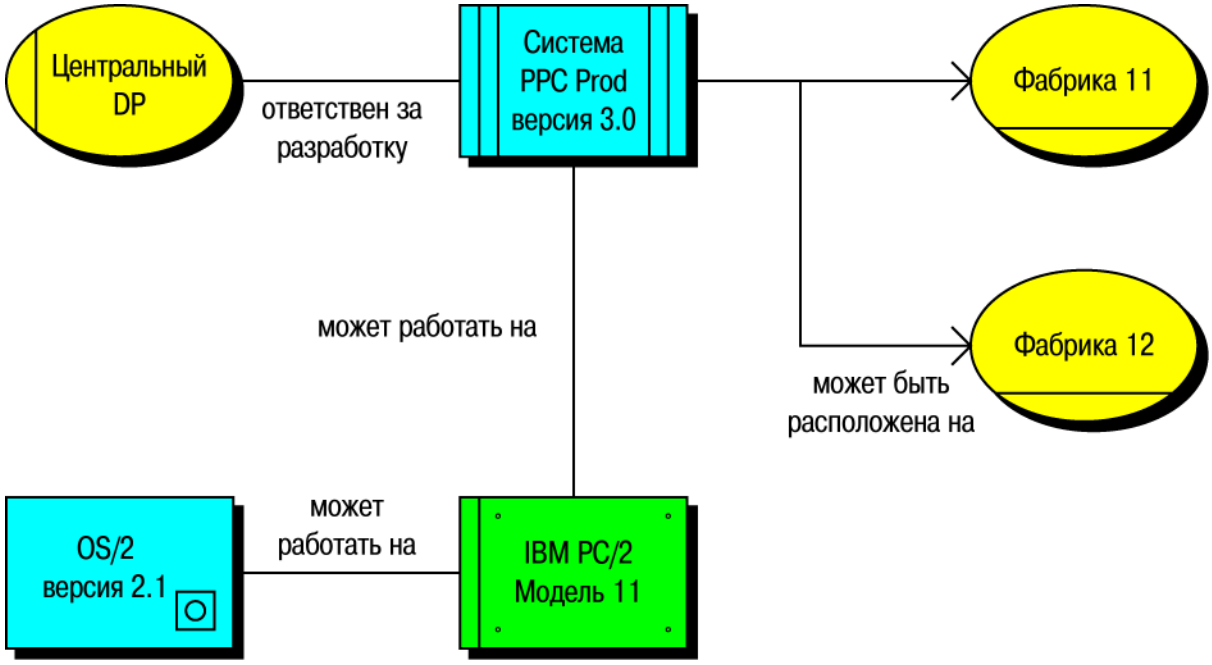


Рис. 4.4.2-5. Диаграмма доступа (фрагмент)

4.4.2.2. Структурная схема программы

В диаграмме доступа мы можем создать взаимосвязи между типами объектов организационной модели и моделями данных для типов прикладных систем, типов модулей и типов ИТ-функций, которые были определены в диаграмме типа прикладной системы (см. раздел 4.4.2.1). В этой диаграмме нельзя непосредственно отобразить привязку функций на уровне формулировки требований. Эта привязка реализуется диаграммой типа прикладной системы. Аналогично не может быть непосредственно отражена логическая последовательность событий для типов прикладных систем, типов модулей и типов ИТ-функций.

Только строго следуя архитектуре ARIS, можно проследить эти связи, просматривая ряд типов диаграмм. Определенные типы диаграмм (например, структурные схемы программ – PSP) были созданы специально для проектирования систем. Они позволяют просматривать все критерии проектирования системы в их взаимосвязи.

По этой причине в ARIS включена диаграмма типа *Структурная схема программы*. Она позволяет моделировать все взаимосвязи типов прикладных систем, типов модулей и типов ИТ-функций при помощи других типов диаграмм ARIS – без характерного для ARIS разбиения на частные модели.

Более того, имеется возможность отображать логическую последовательность событий для упомянутых выше типов объектов. По этой причине события также поддерживаются диаграммой указанного типа. Аналогично привязке функций и событий в EPC можно определить последовательность модулей в *Структурной схеме программы*. В этом контексте событие должно рассматриваться как триггер, который переключает типы модулей или типы прикладных систем. Для отображения ветвлений можно использовать соединяющие операторы, которые были введены в разделе, посвященном EPC. В отличие от EPC в структурной схеме можно определить процедурные последовательности без ввода дополнительных событий.

4.4.2.3. Блок-схема программы

Блок-схема программы описывает последовательность процедур в программе. Порядок процедур определяется связями (отношениями) между объектами. Эта блок-схема не представляет никаких данных.

На рис. 4.4.2-6 приведен простейший пример процедурной последовательности автоматического кассира (банкомата). Блок-схема ориентирована на реализацию и дает ясное представление о том, как это делать.

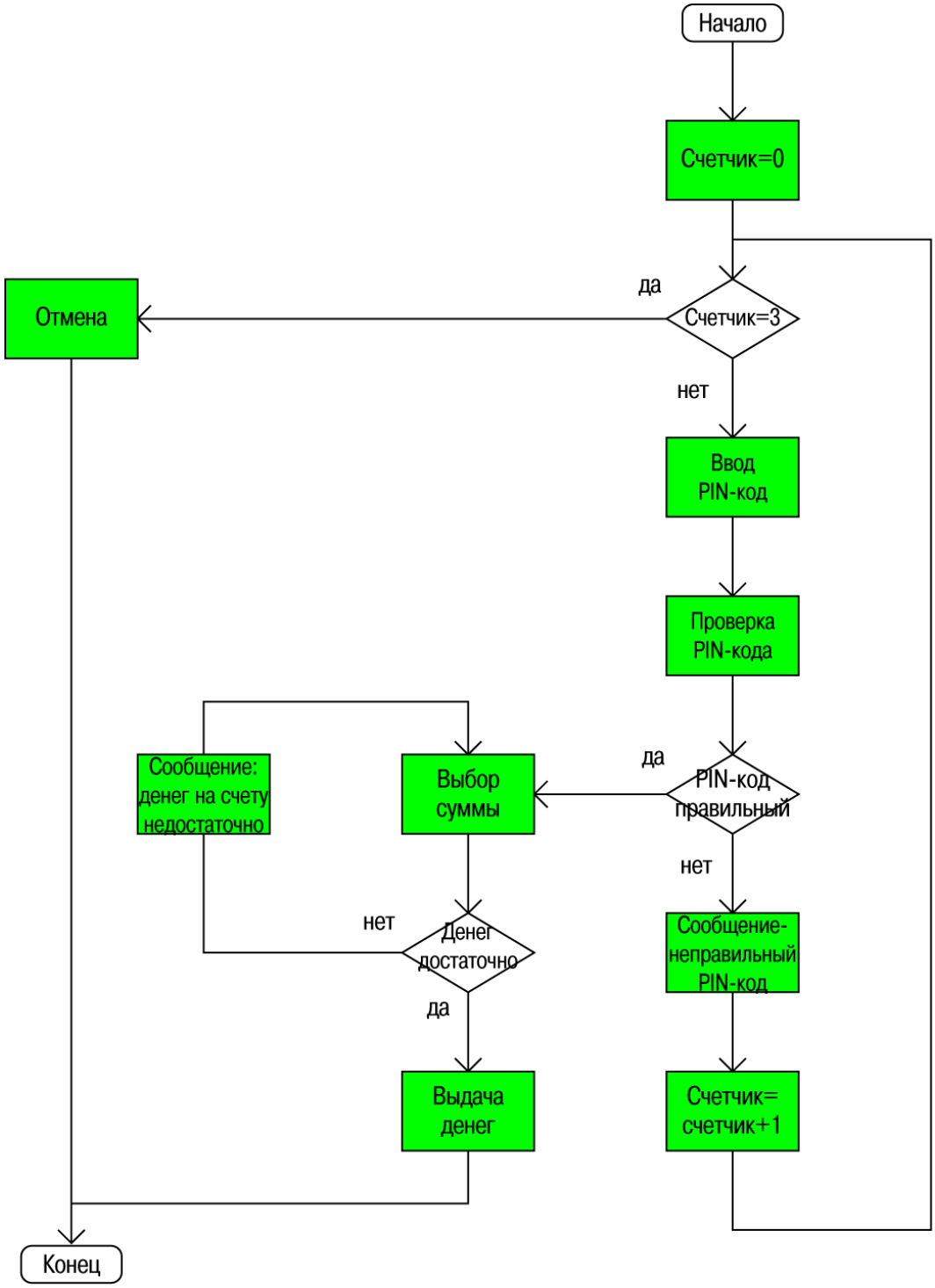


Рис. 4.4.2-6. Блок-схема программы

4.4.2.4. Диаграмма экрана

Диаграмма экрана используется для описания экранных форм при разработке программного обеспечения с целью автоматизации процесса порождения экранных форм из диаграмм экранов.

Диаграммы экранов представляют структуру и в некоторой степени функциональность экранной формы. Необходимо принимать во внимание, что слева направо и сверху вниз структура диаграммы экрана соответствует геометрии графического интерфейса пользователя.

Основным символом является «экран», который, согласно терминологии систем Windows, представляет окно. Это окно может содержать несколько tabs (символ *Страница*). В общем случае интерфейс может быть разделен «географически» на несколько областей с помощью табличного формата (символ *Секция* для строк и символ *Колонка* для столбцов). При построении сложных интерфейсов символы *Секция* и *Колонка* могут быть детализированы.

Вы можете расположить на интерфейсе таблицы (символ *Экранная таблица*), поля ввода текста (символ *СОТ-атрибут*), а также графику (символ *Bitmap*) и текстовые описания (символ *Текст*). Используя символ *Расположение*, вы можете связать свойства монитора с объектами *Экран, Страница, Секция, Колонка, Экранная таблица, СОТ-атрибут, Текст.*

Для описания интерфейса могут применяться и другие дополнительные символы.

На рис. 4.4.2-7 представлена диаграмма экрана, а на рис. 4.4.2-8 экран, сгенерированный из этой диаграммы.

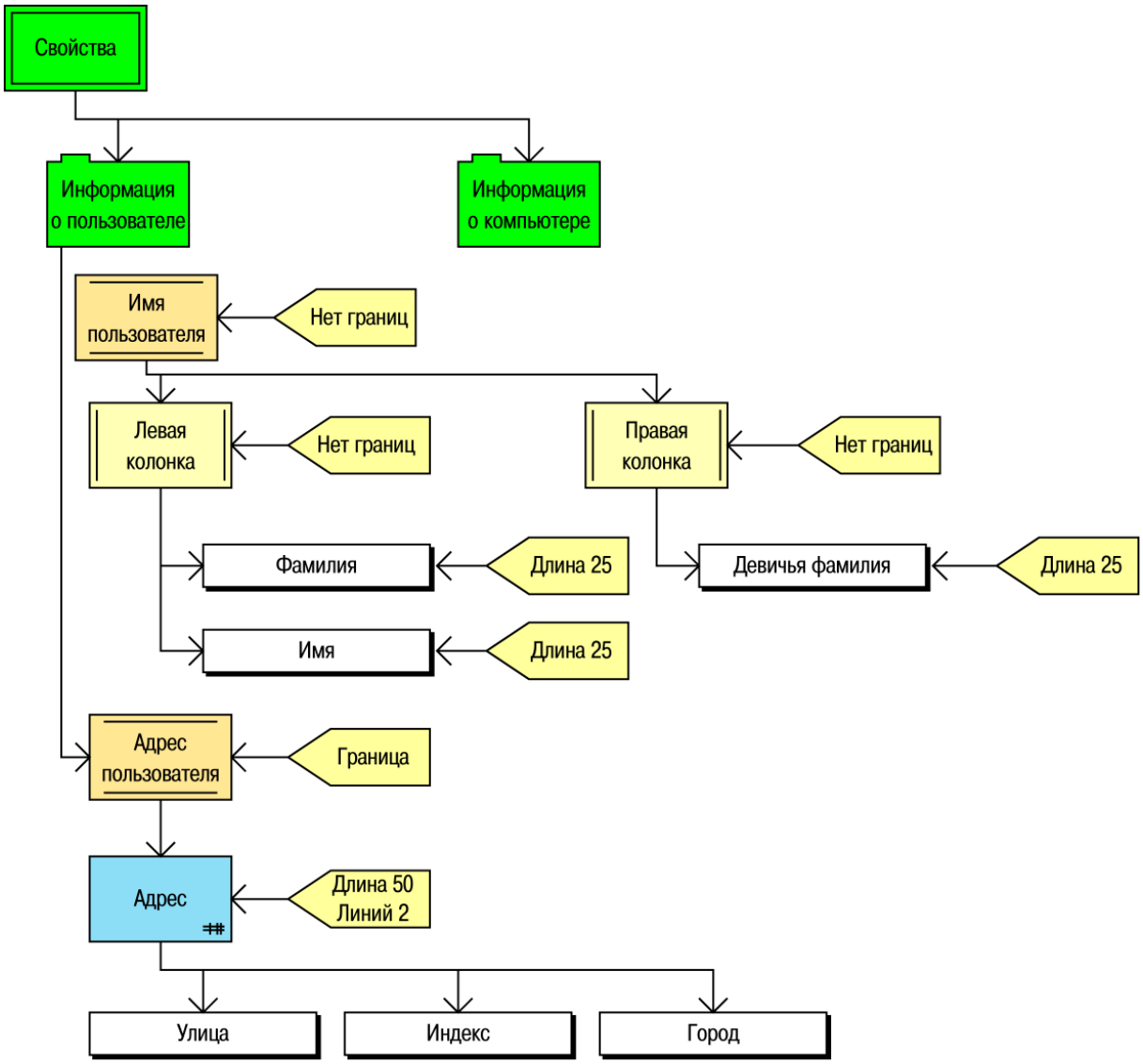


Рис. 4.4.2-7. Диаграмма экрана

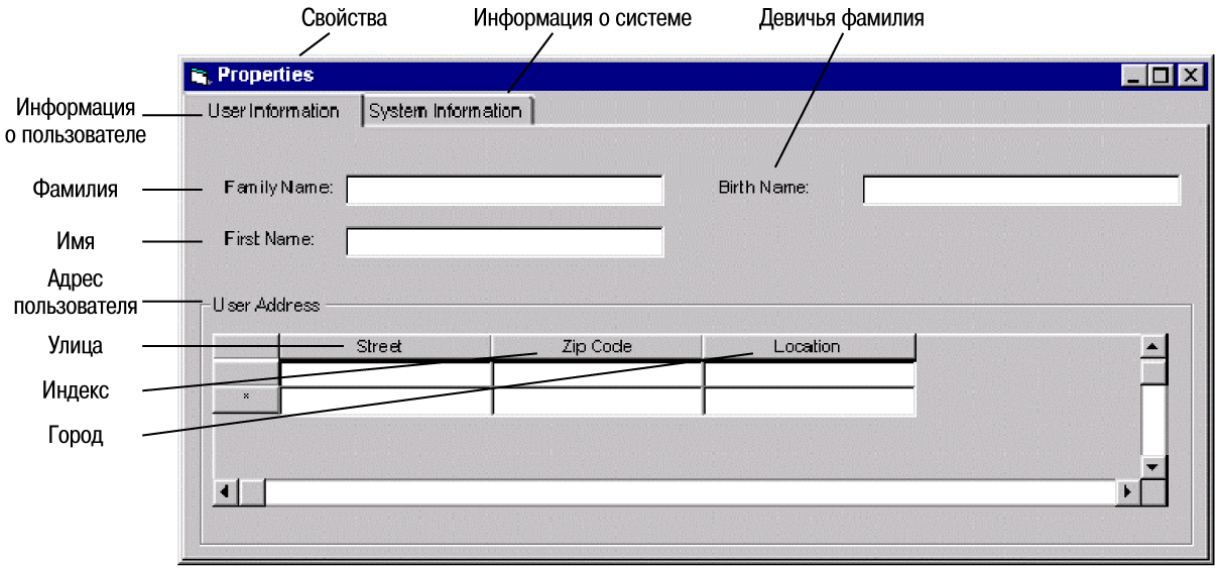


Рис. 4.4.2-8. Экран, сгенерированный из диаграммы экрана на рис. 4.4.2-7

4.4.3. Описание реализации. Диаграмма доступа (физического)

Вопросы, рассмотренные при спецификации проекта в рамках управляющей модели, имеют также отношение к уровню описания реализации. В отличие от уровня спецификации проекта здесь рассматриваются не типы объектов, а сами объекты. Например, мы имеем дело с взаимосвязями, которые существуют между конкретными прикладными системами и организационными единицами, а не между типами прикладных систем и организационными единицами.

Эти взаимосвязи, обсуждаемые в последующих разделах, моделируются с помощью диаграммы доступа (физического).

4.4.3.1. Объединение функций и данных

Для того чтобы определить, какие потоки данных существуют между прикладными системами и между объектами прикладной системы, в функциональную модель могут быть включены объекты потока данных. В отличие от уровня спецификации проекта эти объекты прикладных систем не эквивалентны типам прикладных систем, а представляют конкретные лицензированные прикладные системы. Это означает, что прикладные системы, модули и типы программных элементов могут быть взаимосвязаны посредством объектов потока данных.

Предположим, что на уровне спецификации проекта определено следующее: *могут* быть переданы данные к типам и от типов модулей *Продажи и Дистрибьюция (SD, версия 2.1)* и *Управление ммaтериалами (MM, версия 1.2)*. На уровне описания реализации это определение представляется так: между фактически установленными модулями *SD, License No. 1234,* и *MM, License No. 2352,* и *MM, License No. 34234*, происходит обмен данными. Два модуля MM принадлежат типу модуля *Управление материалами MM (версия 1.2)*. Соответствующий пример приведен на рис. 4.4.3-1.

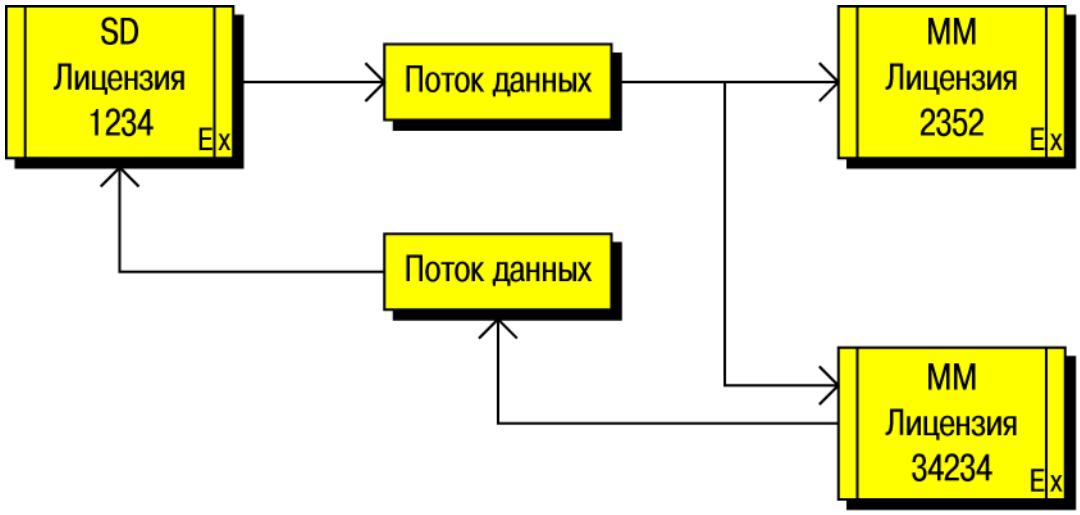


Рис. 4.4.3-1. Поток данных

При помощи связи объекта потока данных с соответствующими типами данных из модели данных передаваемые между системами объекты данных могут быть определены более точно.

Кроме потоков данных между прикладными системами, можно определить данные входа/выхода для каждой прикладной системы. Представление взаимосвязей в диаграмме доступа (физического) необходимо в двух случаях. В первом случае объектами данных являются объекты табличной диаграммы (таблица, поле, модель – физические), которые располагаются в модели данных на уровне описания реализации. Эти объекты данных могут быть связаны с объектами прикладной системы на уровне спецификации проекта или описания реализации посредством данных входа/выхода.

Во втором случае объектами прикладной системы являются конкретные прикладные системы или модули на уровне описания реализации, которые связаны с соответствующими объектами модели данных.

Таким образом, можно сформулировать следующее основное правило:

Если один из типов объектов, участвующих в отношении входа/выхода, взят из уровня описания реализации соответствующей модели, отношения в управляющей модели также представляются на уровне описания реализации (диаграмма доступа (физического)).

Рис. 4.4.3-2 иллюстрирует сказанное выше.

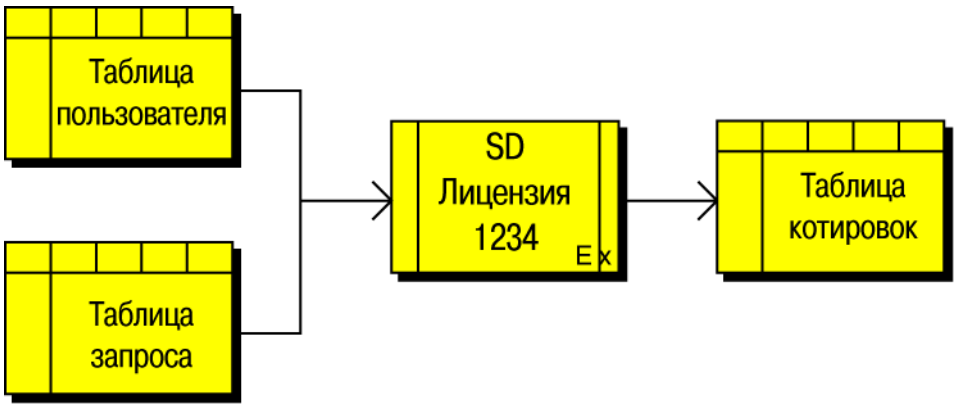


Рис. 4.4.3-2. Отношения входа/выхода

4.4.3.2. Объединение организационных единиц с данными

Вернемся к вопросам, с которыми мы имели дело на уровне спецификации проекта:

- Какие организационные единицы ответственны за объекты данных?
- Кто к каким объектам данных имеет авторизованный доступ?
- Какие объекты данных с помощью, каких компонент АО (аппаратного обеспечения) хранятся?

В отличие от спецификации проекта здесь устанавливаются отношения с объектами данных, отображаемыми на уровне описания реализации в рамках модели данных.

Это означает, что ответственность за объекты данных больше определяется не отношениями и атрибутами диаграммы отношений, а физическими структурами, т.е. таблицами, полями и их экземплярами (таблица (экземпляр), поле (экземпляр)).

Для представления этих зависимостей прорисовываются соединения между объектами организационной модели (организационная единица, должность, людские ресурсы и т.д.) и объектами табличной диаграммы (таблица, поле, модель (физическая) и т.д.) в диаграмме доступа (физического).

При отображении соединений между организационными единицами, таблицами и полями смысл каждой взаимосвязи должен определяться отдельно. Смысл *быть ответственным за* состоит в том, что отдельная организационная единица ответственна за содержимое соответствующей таблицы или поля. Смысл *иметь доступ*

к состоит в том, что эта отдельная должность или людской ресурс определяет авторизованный доступ к представленным объектам данных.

Кроме определения авторизованного доступа и ответственности, можно использовать объект *Компонента АО* (организационная модель/описание реализации), чтобы определить, какие реально существующие компоненты АО содержат определенные информационные объекты компании. Эти компоненты АО могут быть однозначно идентифицированы посредством инвентарных номеров, принятых в компании.

По этой причине объект *Компонента АО* может быть связан с информационными объектами на уровне описания реализации (таблицы, поля и т.д.), на уровне спецификации проекта (отношения, атрибуты) или на уровне формулировки требований (типы сущностей, кластеры данных и т.д.) в диаграмме доступа (физического).

Пример, иллюстрирующий сказанное, приведен на рис. 4.4.3-3.

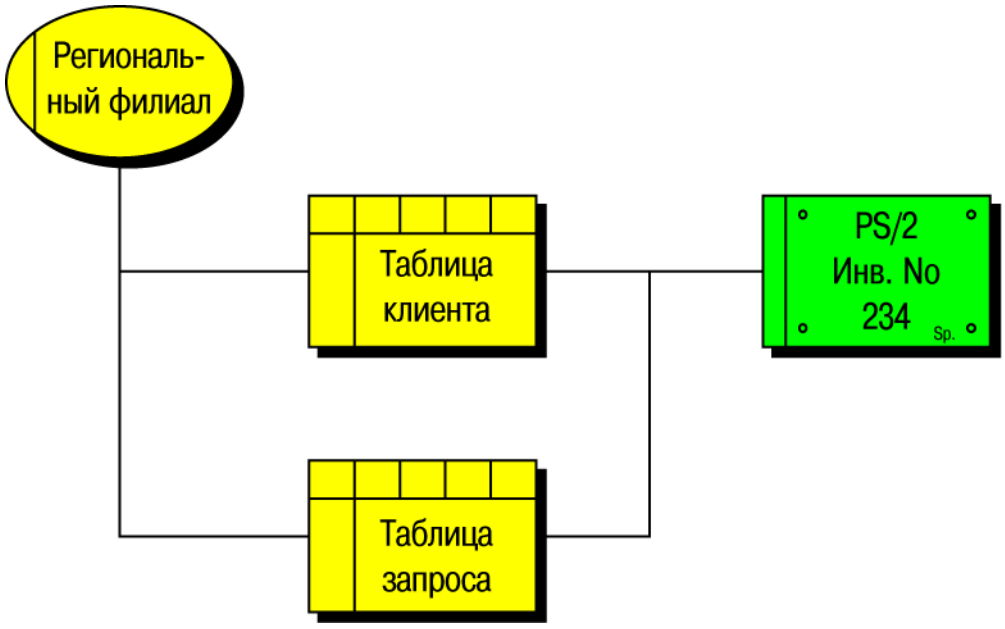


Рис. 4.4.3-3. Привязка к компоненте АО

4.4.3.3. Объединение организационных единиц с функциями

Взаимосвязи между объектами организационной модели и объектами функции, определенные в диаграмме доступа (физического), позволяют ответить на следующие вопросы:

1. Какие прикладные системы, на каких компонентах АО уже работают и какие типы прикладных систем *могли бы* работать на них?

Для того чтобы проиллюстрировать эти зависимости, между объектами прикладной системы на уровне описания реализации (прикладная система, модуль, программный элемент и т.д.) или объектами на уровне спецификации проекта (тип прикладной системы, тип модуля и т.д.), а также объектом типа *Компонента АО* в рамках организационной модели могут быть установлены две взаимозависимости: *является платформой* и *может быть платформой* (см. рис. 4.4.3-4).

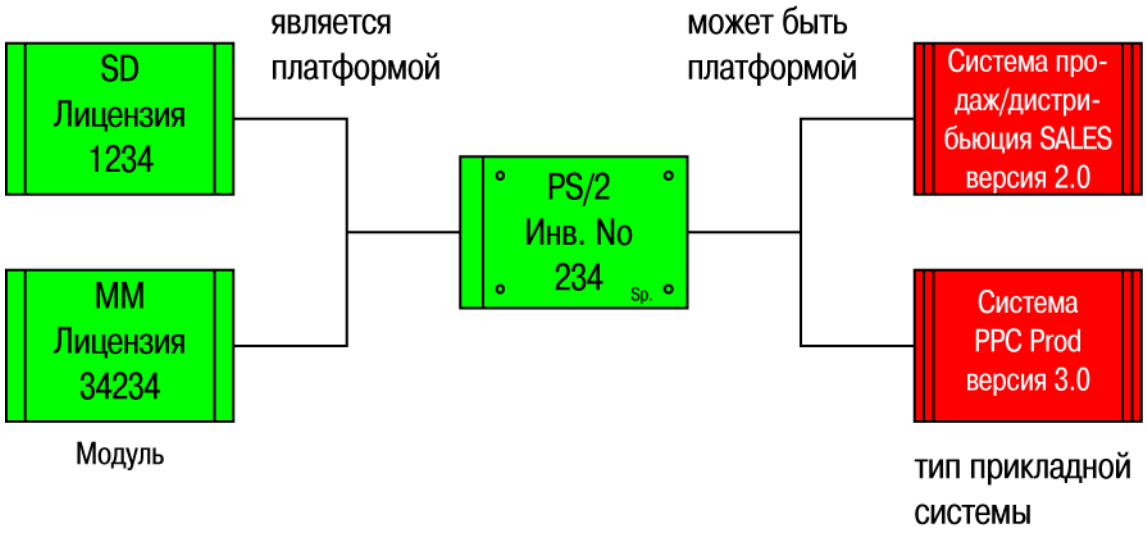


Рис. 4.4.3-4. Компоненты АО в качестве платформы

2. Какая организационная единица, какую конкретную прикладную систему использует?

До тех пор пока пользователи, которые имеют доступ к определенному типу прикладной системы, специфицированы на уровне спецификации проекта, имеется возможность определить эту взаимосвязь на уровне описания реализации для конкретных прикладных систем (индивидуальные лицензии). Таким образом, в одной компании могут быть доступны несколько лицензий на прикладную систему *ARIS Toolset версия 4.1,* в различных конфигурациях. При помощи диаграммы доступа (физического) возможно показать, кто использует эти лицензии. Для этого объекты типов *Организационная единица*, *Должность* и *Сотрудник* могут быть связаны с объектами типов *Прикладная система* и *Модуль* через соединение *Использует* (см. рис. 4.4.3-5).

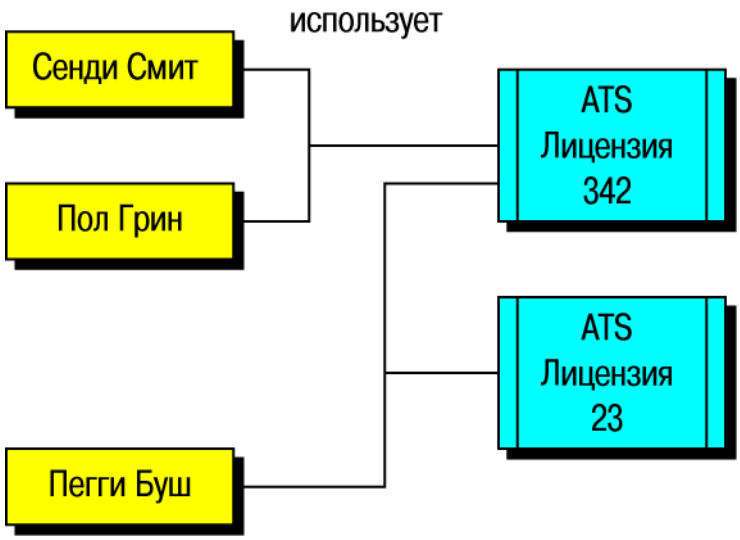


Рис. 4.4.3-5. Пользователи и прикладные системы

3) На каких рабочих местах в компании, какие прикладные системы установлены?

При спецификации проекта взаимосвязь *Тип прикладной системы – Местоположение* позволила определить, какие типы прикладных систем могут быть расположены на конкретных рабочих местах в компании. Для того чтобы фактически специфицировать, в каких местах компании используются индивидуальные лицензии на приобретенные типы прикладных систем, можно связать местоположение с типами объектов *Прикладная система*, *Модуль* и *ИТ-функция* в диаграмме доступа (физического). Это иллюстрирует пример на рис. 4.4.3-6.

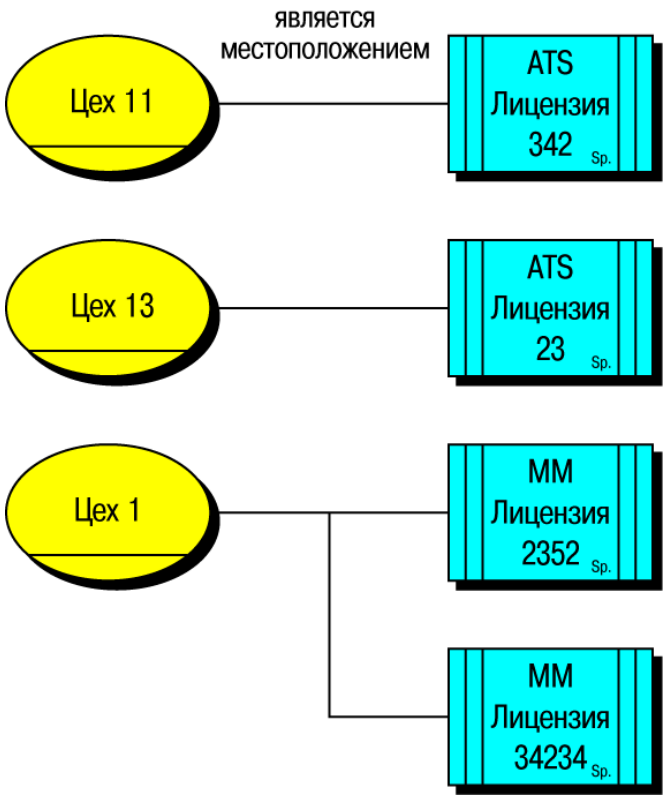


Рис. 4.4.3-6. Привязка местоположения

Все взаимосвязи в диаграмме доступа (физического) описываются в приложении к английской версии руководства «Методы ARIS».

4.5. Моделирование результатов

4.5.1. Диаграмма обмена продукт/услуга

Диаграмма обмена продукт/услуга предназначена для представления процесса создания продуктов/услуг, а также их обмена внутри компании. В качестве результата может рассматриваться как услуга, так и продукт. В диаграмме это обозначается соответствующим символом. Продукт может быть типом материала, типом операционного ресурса, типом технического операционного обеспечения и/или типом упаковочного материала, которые известны, например, из диаграммы eEPC с потоками материалов. Результаты в качестве входа/выхода функций могут быть соединены с начальными и конечными событиями для этих функций.

Обмен продукт/услуга между отдельными бизнес-функциями может быть весьма полезен даже на абстрактном уровне, например, между диаграммами цепочек добавленного качества и eEPC. Отношения обмена могут быть представлены не только с функциональной, но и с организационной точек зрения. Для этой цели в диаграмме обмена продукт/услуга существует несколько опций моделирования.

Диаграмма обмена продукт/услуга представлена на рис. 4.5-1.

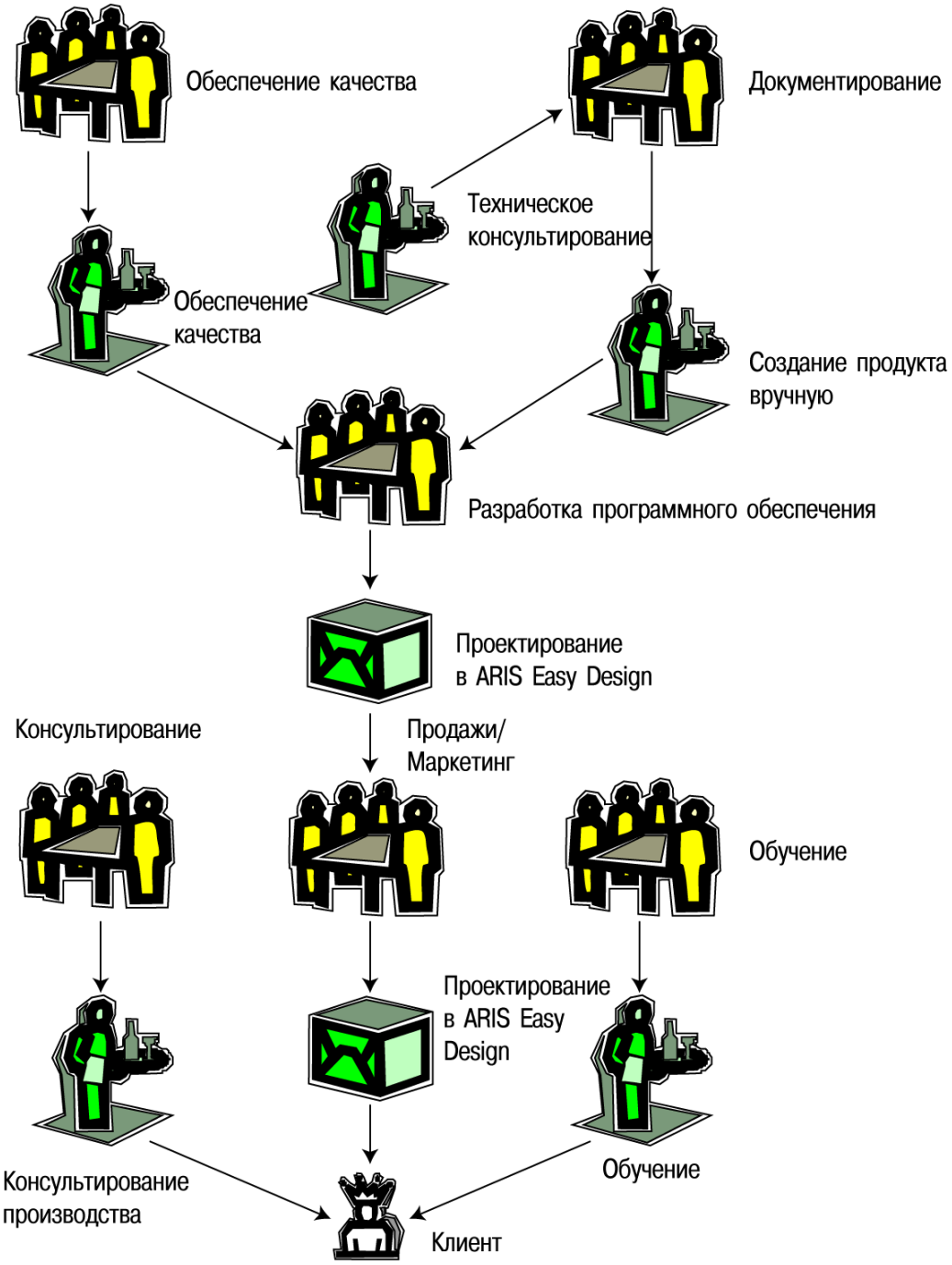


Рис. 4.5-1. Диаграммы обмена продукт/услуга в софтверной компании

4.5.2. Дерево продукт/услуга

Результаты могут рассматриваться на различных уровнях абстракции. Эти отношения полезно включить в модель, где представлены отдельные части результата, которые в дальнейшем составят общий результат. Этот статичный аспект описывается деревом продукт/услуга. Например, сложный продукт содержит несколько модулей, каждый из которых имеет различные составляющие части. Каждый такой элемент может пониматься как элемент результата.

В дереве продукт/услуга могут быть представлены отношения замещения результатов, когда используется (потенциальный) заменяющий продукт или услуга.

Связь результатов с целями компании также описывается в этой статичной модели. На рис. 4.5-2 приведено дерево продукт/услуга.

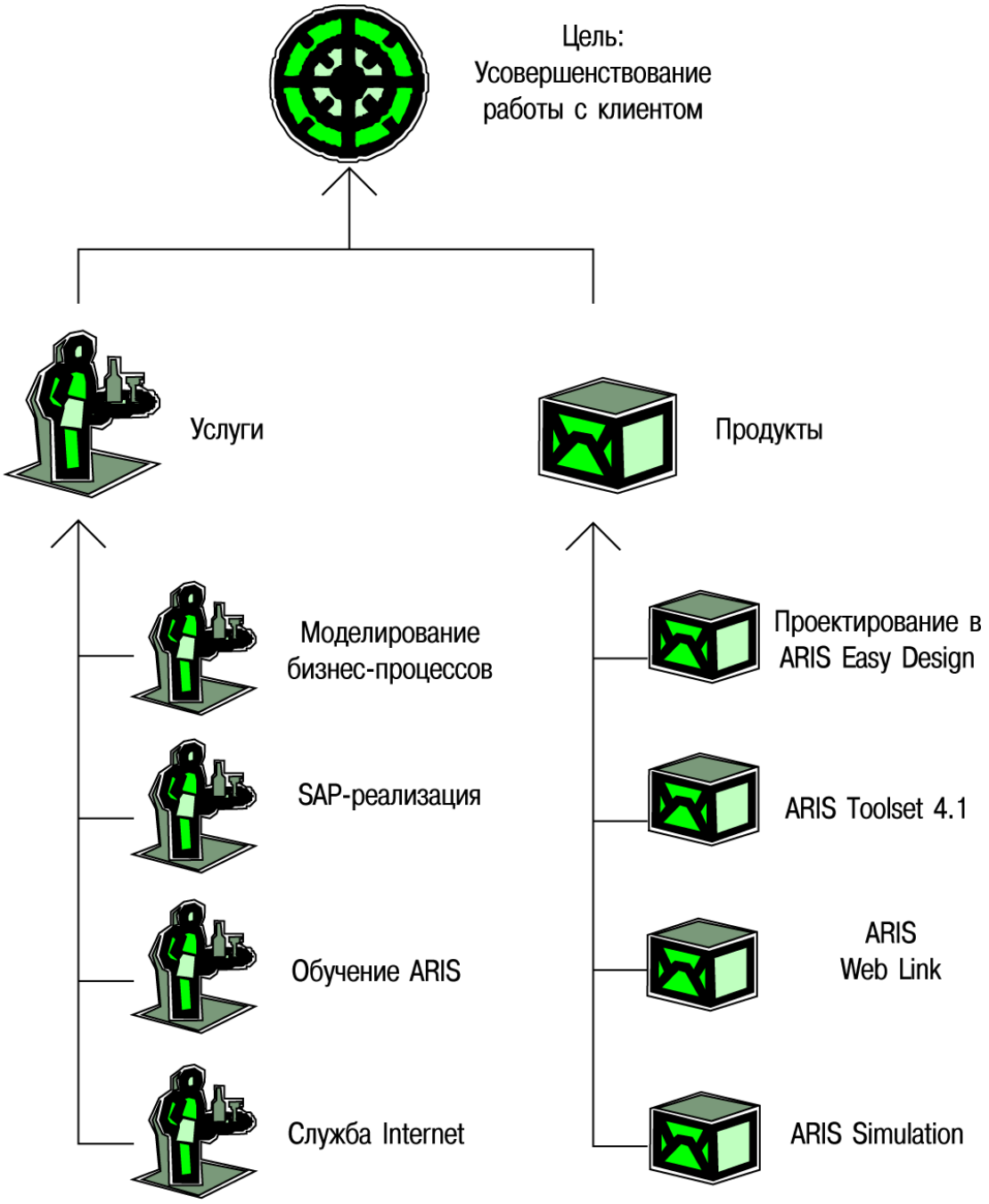


Рис. 4.5-2. Дерево продукт/услуга

4.5.3. Диаграмма описания продукта

В дополнение к общим диаграммам продукт/услуга, которые относятся к конкретным моделям, модели продуктов дают возможность создавать более абстрактные представления. С помощью диаграммы описания продукта проводится анализ процесса создания продукта. Аналогично диаграмме обмена продукт/услуга этот тип модели может применяться для отображения того, какие организационные единицы предоставляют или используют продукты, какие функции требуются для создания продукта, а также для каких функций продукты являются входом. Кроме того, здесь отражаются (легальный) базис с точки зрения заказа на соответствующий продукт. В диаграмме могут быть описаны и цели, достижению которых способствуют различные продукты.

На рис. 4.5-3 приведен фрагмент диаграммы описания продукта службы работы с населением.

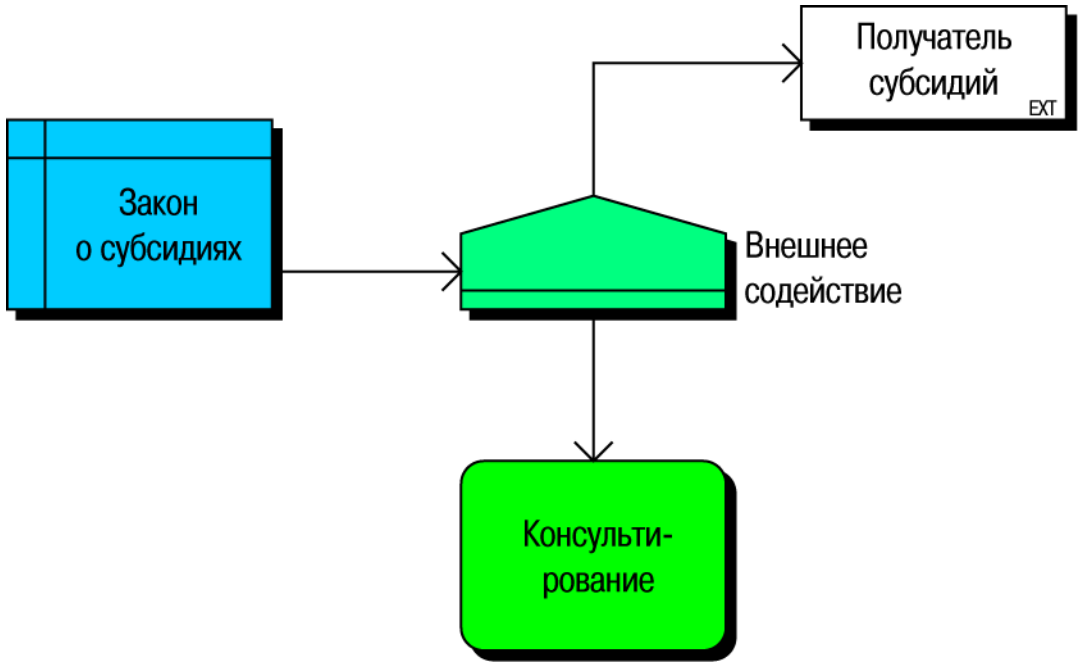


Рис. 4.5-3. Диаграмма описания продукта

4.5.4. Дерево продукта

Дерево продукта предназначено для анализа совокупности продуктов в службах работы с населением. Модель в основном состоит из дерева продукт/услуга, благодаря чему можно легко смоделировать продукты замены.

На рис. 4.5-4 представлен пример дерева продукта.

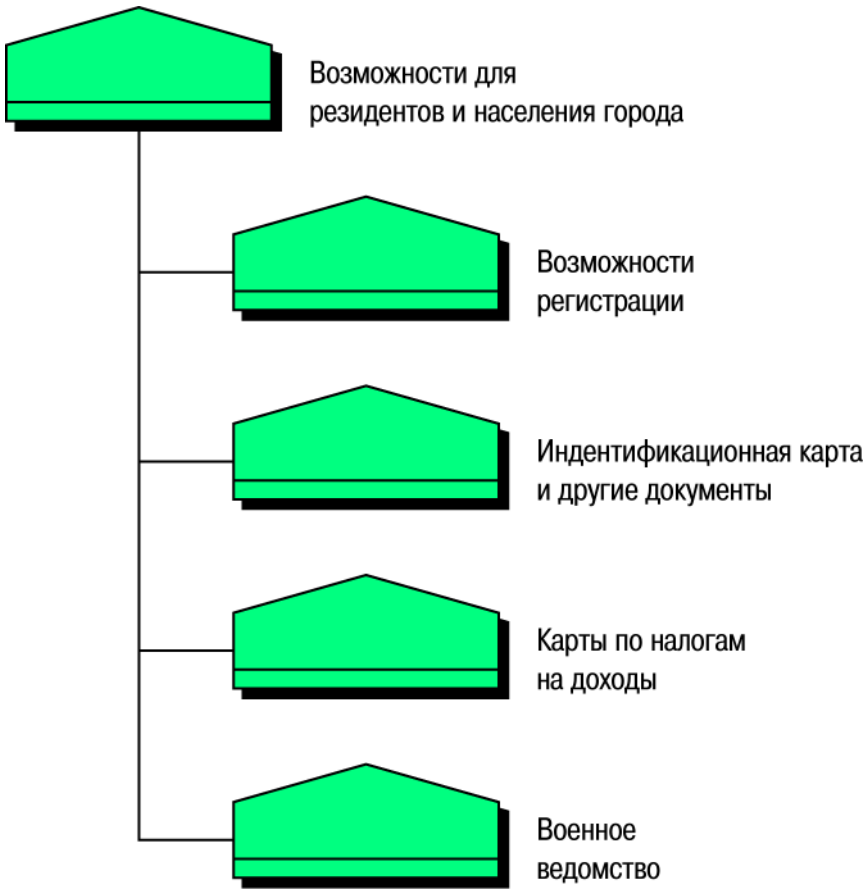


Рис. 4.5-4. Классификация продуктов «Население» и «Обеспечение населения», проведенная с помощью дерева продукта

4.5.5. Матрица выбора продукта

Матрица выбора продукта описывает организационное подразделение и продукты, за которые это подразделение несет ответственность. Продукты могут быть привязаны к функциям, необходимым для их производства. Данная модель – начальная точка, с которой становятся доступными организационные схемы, деревья продуктов и процессы, связанные с созданием этих продуктов. Дерево продуктов приведено на рис. 4.5-5.

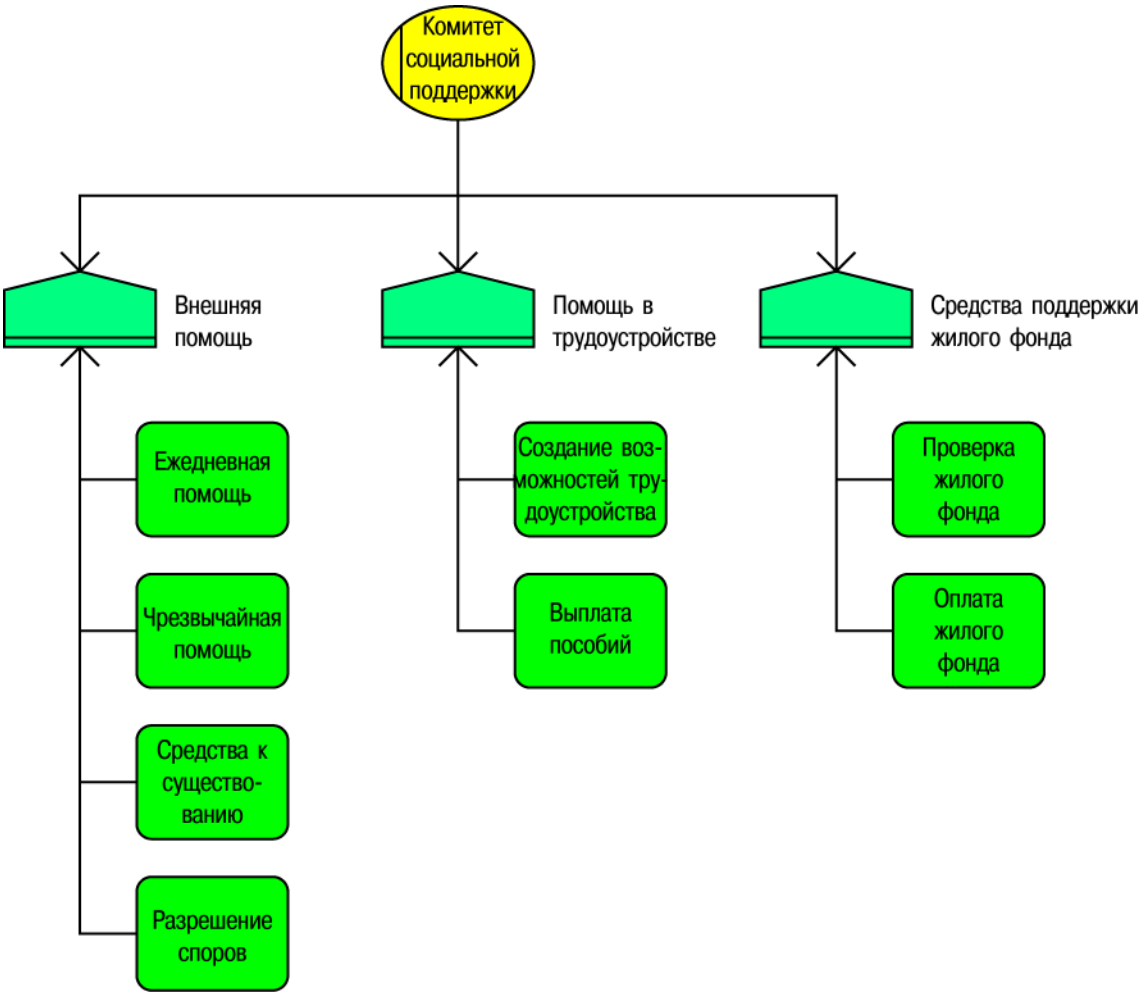


Рис. 4.5-5. Матрица выборки продукта для службы социальной поддержки

4.5.6. Модели конкуренции

Модель конкуренции поддерживает анализ и оценку конкурентной ситуации, в которой находится компания. Структура производства сильно влияет на стратегии, которые потенциально могут быть приемлемыми для компании.

Модель позволяет описать отношения внутри компании, конечные продукты и услуги, а также партнеров по рынку. Можно также представить, какие клиенты, какие результаты работы компании используют, какие результаты доступны для поставщиков и какие заменяющие продукты/услуги предлагаются от (потенциальных) конкурентов.

Таким образом, будет отображена ситуация на той части рынка, где работает компания. Модель конкуренции приведена на рис. 4.5-1.

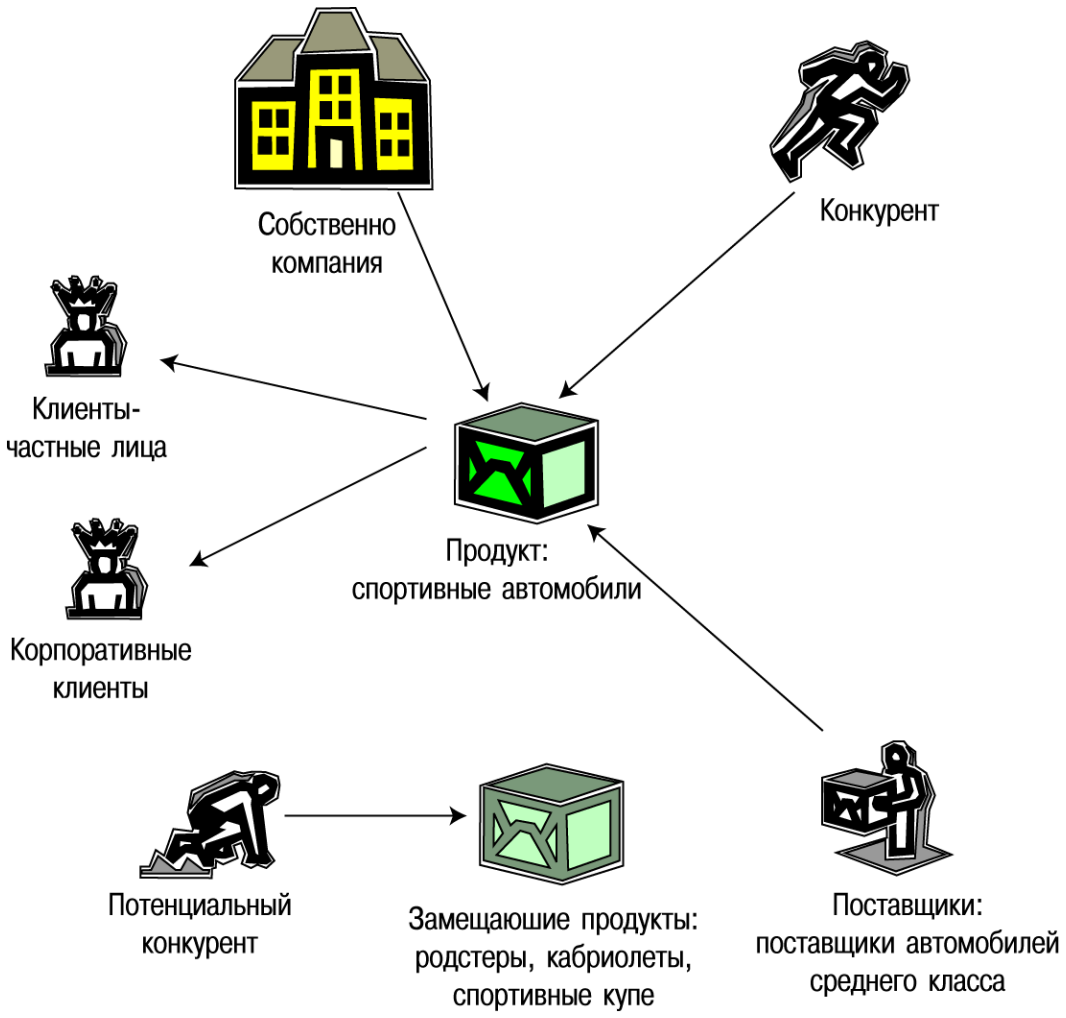


Рис. 4.5-6. Конкуренция на рынке спортивных автомобилей

5. Унифицированный язык моделирования ARIS

5.1. Введение

Унифицированный язык моделирования (UML - Unified Modeling Language) – это объектно-ориентированный язык моделирования. Рабочая группа OMG (Object Management Group) стандартизировала разработанный ею язык. UML базируется на объектно-ориентированных подходах OMT, Booch и OOSE.

Основой для моделей типа ARIS UML является определение унифицированного языка моделирования, состоящее из документов *UML Резюме*, *UML Семантика* (включая *UML Глоссарий*) и *UML Нотация*, каждый - версии 1.1 (сентябрь 1997).

Текущая информация о структуре UML находится в Интернет по адресу http://www.omg.org и http://www.rational.com/uml. Поскольку UML-модели ARIS ориентированы на стандартную (не немецкую) транскрипцию, существующие стандарты, типы моделей (диаграмм), типы объектов, типы отношений и типы атрибутов имеют английские идентификаторы даже в немецкой версии программы (язык интерфейса – немецкий).

Типы моделей UML описывают управляющие модели на уровне формулировки требований. ARIS предлагает следующие типы моделей: *UML-диаграмма деятельности*, *UML-диаграмма класса*, *UML-диаграмма описания класса*, *UML-диаграмма сотрудничества*, *UML-диаграмма компонент*, *UML-диаграмма состояний* и *UML-диаграмма выбора*.

5.2. UML-диаграммы

5.2.1. UML-диаграмма класса

Тип модели *UML-диаграмма класса* отражает статичные отношения между такими элементами модели, как *класс*, *объект* и *интерфейс*.

UML-диаграмма класса определяет классы, к которым с помощью отношения *иметь участником* могут быть привязаны соответствующие *операции* (методы) и *атрибуты*.

Отношения, которые имеют друг с другом классы, моделируются в *UML-диаграммах класса* с помощью соединения *ассоциироваться*. Соединения осуществляются непосредственно между классами для бинарных отношений. *Ассоциация*, изображаемая в виде ромба, используется для соединения нескольких отношений. Если *ассоциация* является *классом*, может быть использовано соединение *передавать свойства*. Множество соединений *ассоциироваться* может быть введено в

атрибуты *Множество (Источник)* и *Множество (Цель),* описывающие соединение *ассоциироваться*.

В UML *Агрегация* и *Композиция* отражают специальные отношения *ассоциироваться*. Они специфицируются с помощью входа в атрибут *Тип агрегации* соединения *ассоциироваться* и изображаются небольшим белым (агрегация) или черным (композиция) ромбом на конце соединения *ассоциироваться*. Пример представлен на рис. 5.2.1-1.

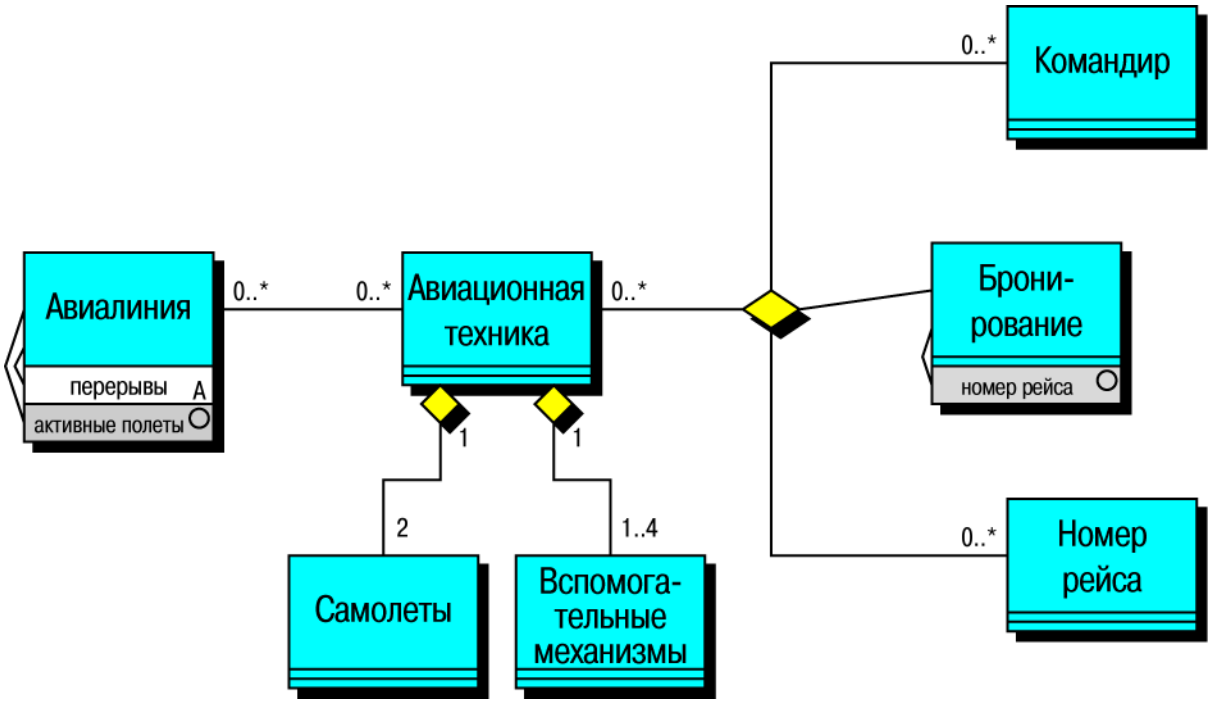


Рис. 5.2.1-1. UML-диаграмма класса: «Ассоциации»

Отношения наследования, которые существуют между классами, представляются отношением *обобщать* и изображаются треугольником. *Атрибуты* и *операции*, которые были привязаны к старшему классу, передаются подчиненным классам. Пример приведен на рис. 5.2.1-2.

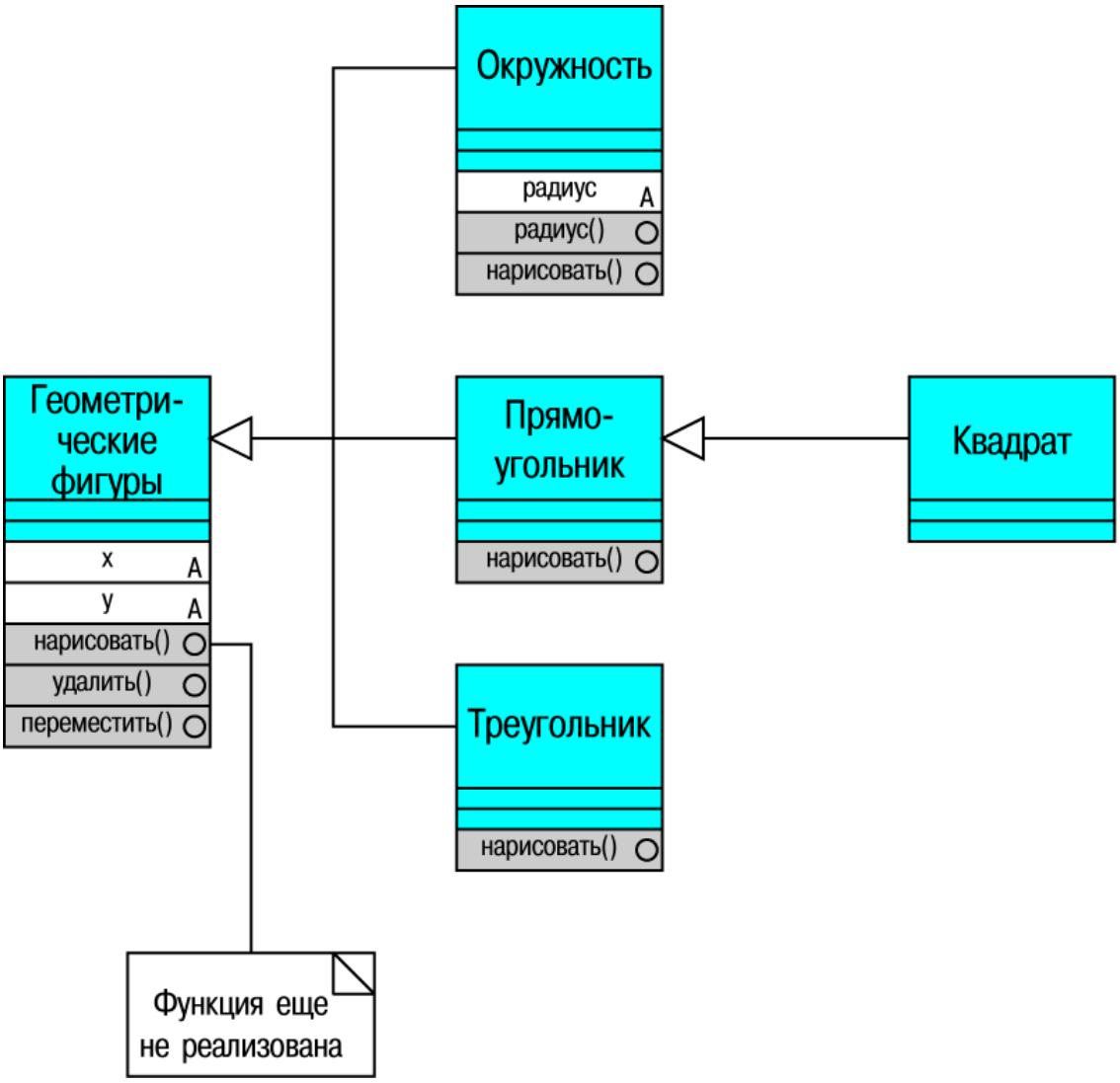


Рис. 5.2.1-2. UML-диаграмма класса: «отношения наследования»

Дополнительными элементами модели, доступными в *UML-диаграмме класса,* являются *пакеты*, используемые для группировки элементов модели; *примечания*, которые предоставляют дополнительную информацию для модели; *объекты* для экземпляров *классов* и *интерфейсы*. *Интерфейс* обозначает интерфейс класса (соединение *поддерживать*). При помощи вызова *интерфейса* (соединение *вызвать*) другие классы могут использовать *класс*, соответствующий *инферфейсу*.

5.2.2. UML-диаграмма описания класса

Тип модели *UML-диаграмма описания класса* – это дополнение к стандартным диаграммам UML, которое позволяет более точно определить классы. Параметры моделирования *UML-диаграммы описания класса* – подмножество параметров *UML-диаграммы класса*. Другими словами, все параметры моделирования в *UML-диаграмме описания класса* доступны и в *UML-диаграмме класса*. Классы *атрибуты*, *операции*, *примечания*, *объекты* и *интерфейсы* могут быть описаны в *UML-диаграмме описания класса*.

Это описание может быть сделано также в *UML-диаграмме класса*, но моделирование с помощью *UM-диаграммы описания класса* предлагается в случае, если *UML-диаграмма класса* становится слишком перегруженной графически. В этом случае *UML-диаграмма описания класса* должна быть определена как присоединение к классу *UML-диаграмма класса*. Классы *атрибуты*, *операции*, *примечания*, *объекты* и *интерфейсы*, которые имеют к ней отношение, но не являются необходимыми в *UML-диаграмме класса*, могут быть перемещены в *UML-диаграмму описания класса*.

5.2.3. UML-диаграмма использования приложений

В *UML-диаграмме использования приложений* описываются варианты используемых приложений и *исполнители*, т.е. привлеченные объекты, на которые воздействуют выбранные приложения. *Исполнители* изображают пользователей, которые используют прикладную систему для выполнения своих обязанностей. *UML-диаграмма использования приложений* описывает внешнее поведение системы с точки зрения пользователя. В ARIS *исполнители* представляются специальными символами типа объекта *тип участника*.

Соединения между *исполнителями* и *используемыми приложениями* определяются как отношение *взаимодействовать с*. Они отображают, что *исполнитель* выполняет *используемое приложение*. Связь между *использовать приложение* устанавливается с помощью отношения *обобщить*, соединение с которым изображается треугольником. По желанию к атрибуту *Стереотип* этого отношения может быть привязана семантика. Стандарт UML предлагает стереотипы *Расширить* и *Использовать*. *Расширить* отображает расширенное отношение, в котором, например в исключительной ситуации, одно *использование приложений* расширяет другое *использование приложений*.

На рис. 5.2.3-1 представлена *UML-диаграмма использования приложений*. В *UML-диаграмме использования приложений* доступны также типы объектов *Пакет* и *Примечание*.

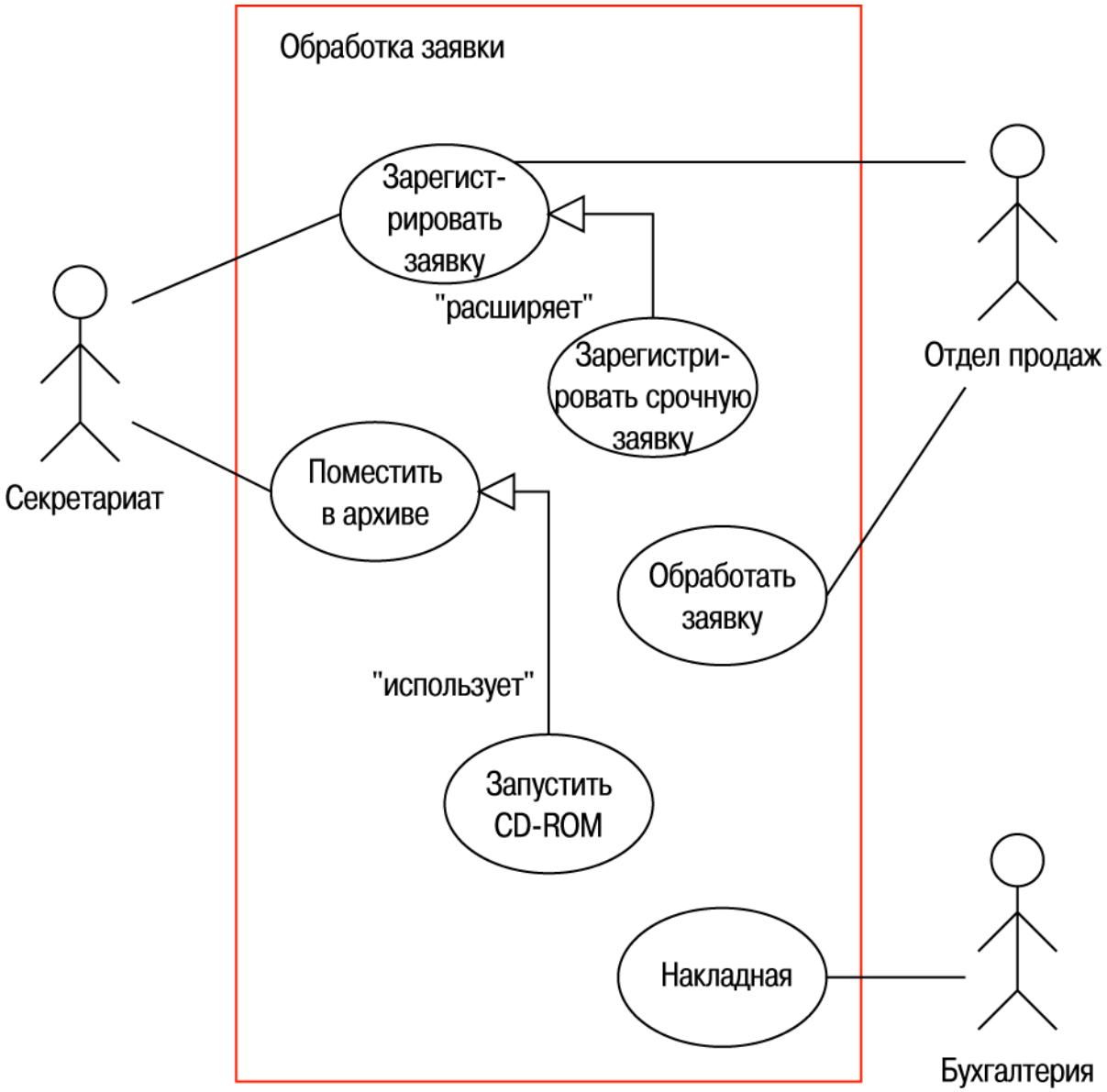


Рис. 5.2.3-1. UML-диаграмма использования приложений

5.2.4. UML-диаграмма действий

UML-диаграмма действий описывает процесс как последовательность действий.В UML *действия* всегда относятся к объектам. Таким образом, диаграммы действий привязываются к *классу*, *операции* или *использованию приложений* и описывают относящийся к ним внутренний процесс.

Поскольку *диаграммы действий* рассматриваются как специальная форма автоматического состояния, процесс, описываемый диаграммой действий, должен начинаться с *начального состояния* и оканчиваться *конечным состоянием*. *Действия* представляют состояние с каким-то внутренним действием и одним или несколькими переходами. Переходы обозначаются соединениями, соответствующими отношениям между *действиями*. *Действия* могут иметь простые отношения с другими действиями, так же как и множественные входящие и выходящие отношения:

1. Множественные внешние отношения могут быть сформулированы как условия. Для их представления используется символ *Решение* (в форме ромба). Моделирование условий с помощью символа *Решение* не обязательно; в качестве альтернативы пользователи могут моделировать просто несколько внешних соединений. На диаграмме рекомендуется показывать условие в атрибуте *Роль соединения* для отношений *является предшественником* и *активизирован*.
2. Символ *Разделить/Синхро* (вертикальная или горизонтальная черта) может быть использован для активизации в одно и то же время нескольких последовательных *действий* или *действия*, зависящего от переходов в рамках нескольких предыдущих *действий*.

Действия могут предполагать особое начальное состояние объекта и создавать особое конечное состояние объекта. Состояния объектов изображаются типом объекта *Состояние объекта*, который описывается соединениями *имеет входом* или *имеет выходом* (пунктирные стрелки) в виде отношения с *действиями*.

UML отображает организационную ответственность за выполнение *действий* с помощью так называемого «плавающего коридора».

Плавающий коридор – это столбец, где перечислены все *действия*, за которые ответственна организационная единица. Для этой цели *UML-диаграмма действий* ARIS содержит предопределенную таблицу с двумя линиями. Ответственная организационная единица (*Внутренний участник*, *Место*, *Тип участника*, *Организационная единица* или *Группа*) располагается на верхней линии, в то время как нижняя линия отводится для понятий *Действие*, *Решение*, *Разделить/Синхро*, *Состояние объекта* и *Примечание*.

На рис. 5.2.4-1 представлена *UML-диаграмма действий* с соответствующими компонентами.

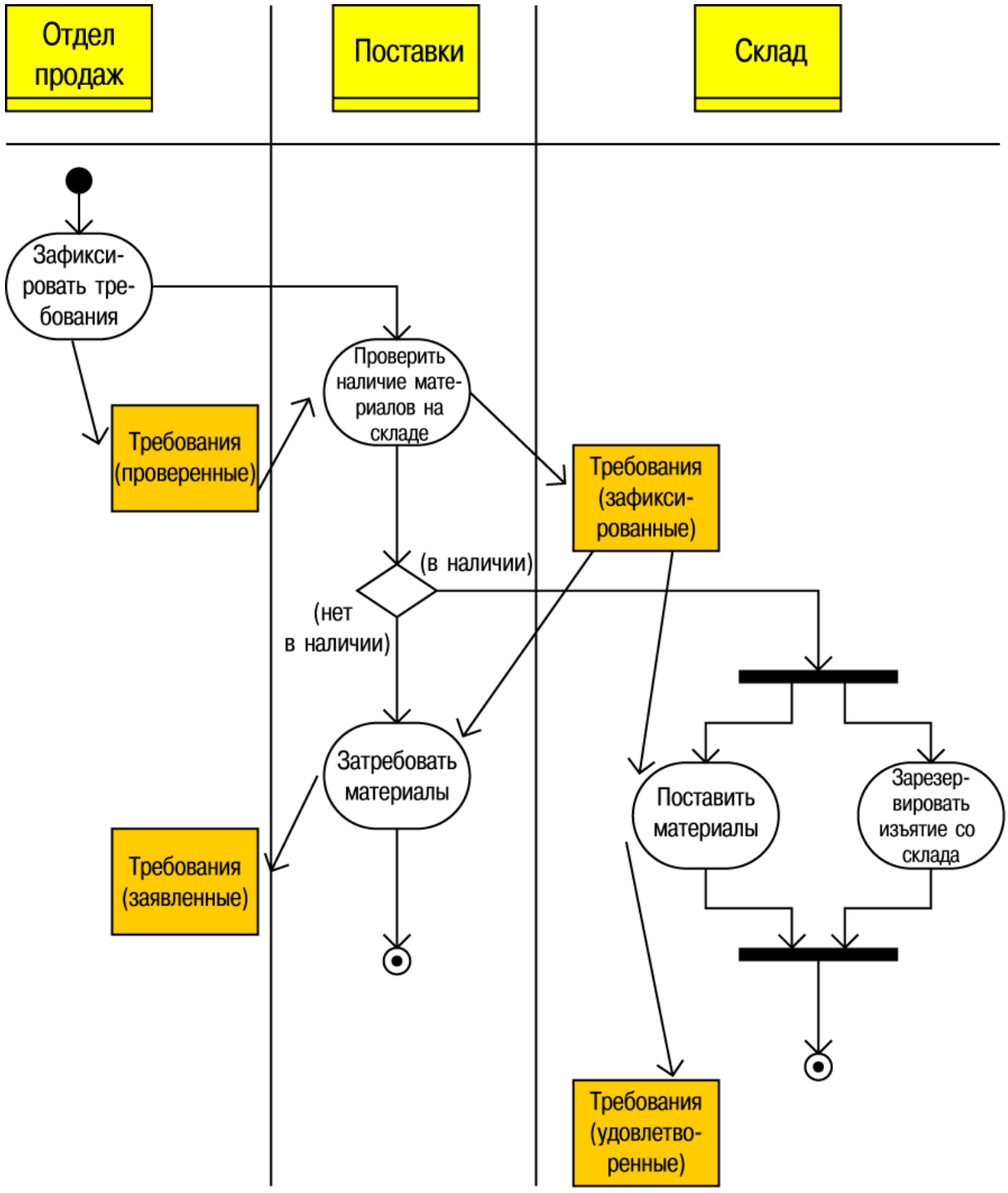


Рис. 5.2.4-1. UML-диаграмма действий

5.2.5. UML-диаграмма состояний

UML-диаграмма состояний подобно *UML-диаграмме действий* изображает автоматическое состояние и описывает аналогичную ситуацию. Однако *диаграмма состояний* сфокусирована на состояния объектов. Более того, она может содержать действия, относящиеся к состоянию. Действия – предопределенные значения на входе в состояние (*вход/*), которые выполняются при нахождении в состоянии (*выполнить/*) или при выходе из состояния (*выйти/*).

UML-диаграмма состояний ARIS включает символ *Состояние*. Переходы между состояниями соединяют состояния с помощью направленных соединений (*перейти на*). Так же, как и в случае *UML-диаграммы действий*, диаграмма состояний должна начинаться с *начального состояния* и оканчиваться *конечным состоянием*. На рис. 5.2.5-1 представлена *UML-диаграмма состояний*.

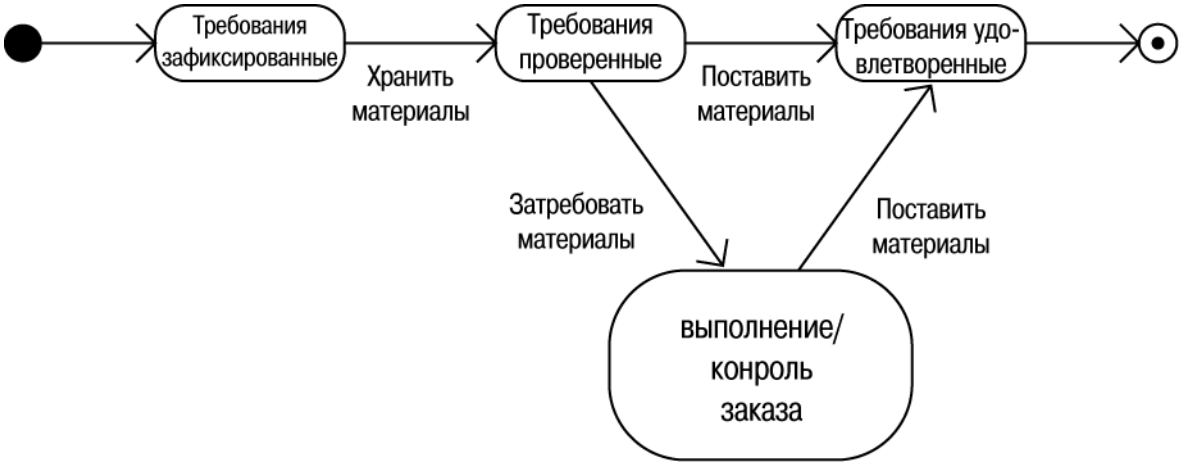


Рис. 5.2.5-1. UML-диаграмма состояний

5.2.6. UML- диаграмма взаимодействия

Взаимодействия в форме обмена сообщениями между объектами отображаются в *UML-диаграмме взаимодействия*. *Объекты*, называемые также экземплярами, - это конкретные экземпляры классов.

Обмен сообщениями моделируется с помощью соединения *взаимодействовать с*. Точный смысл соединения *взаимодействовать с* указывает конкретные значения атрибутов *Условие*, *Номер Сообщения*, *Операция* и *Параметр*. Эти атрибуты определяются следующим образом.

Условие: Принимает вид других сообщений, которые должны быть посланы перед посылкой текущего сообщения. Другие сообщения и соответствующие им

номера сообщений заданы в виде списка. Если не существует никаких предпосланных сообщений, *условие* становится не нужным. Условие отделяется от номера сообщения косой чертой (/).

Номер сообщения: Это уникальный номер, идентифицирующий сообщение на диаграмме. Сообщения сортируются в порядке возрастания номеров. Если операция, в данный момент обрабатывающая полученное сообщение, посылает несколько дополнительных сообщений, то старый номер дополняется «подномером». (*Пример.* Операция получает сообщение 3.4 и посылает два сообщения с номерами 3.4.1 и 3.4.2) *Номер сообщения* отделяется от операции двоеточием (:).

Операция: Отображает *операцию* для *класса* заданного *объекта*, которая должна быть выполнена.

Параметр: Определяет список параметров для вызываемой операции. Список параметров заключается в круглые скобки.

Пример: 1.3, 2.1 / 3.2.1: calculate net (gross, rate), где: сообщения 1.3 и 2.1 – это условия; само сообщение имеет номер 3.2.1; выполняемая операция – расчет чистой зарплаты; «грязная» зарплата и ставка налога – параметры операции.

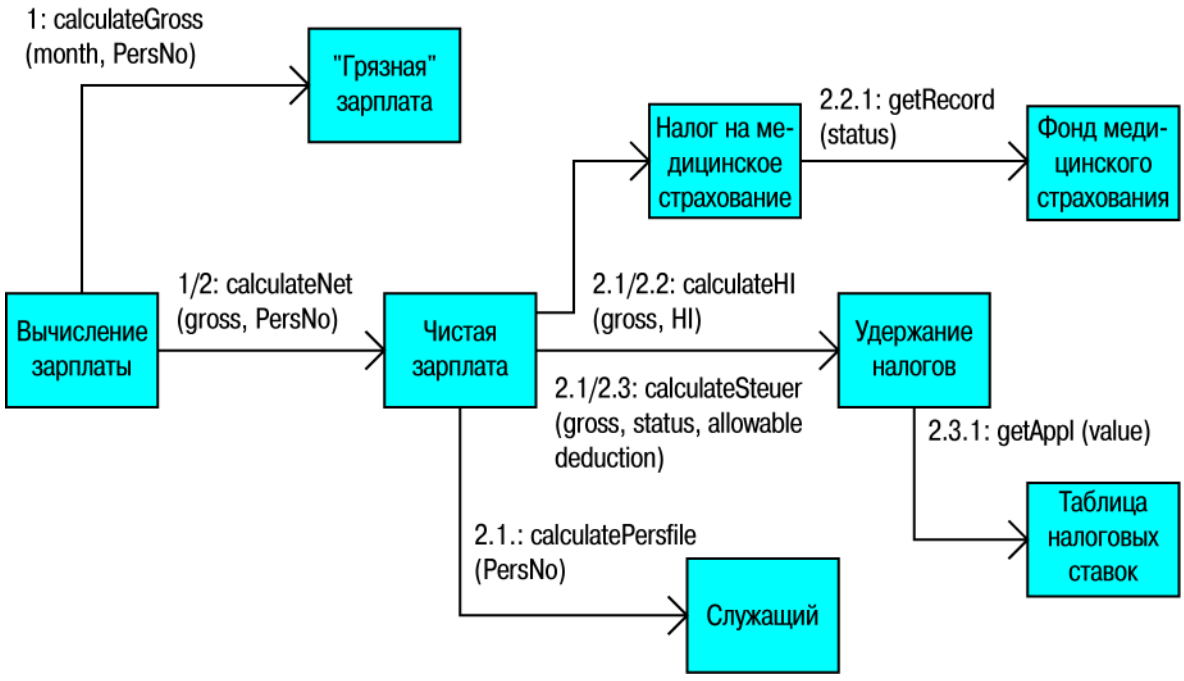


Рис. 5.2.6-1. UML- диаграмма взаимодействия

5.2.7. UML-диаграмма компонент

UML имеет возможность отображать в виде диаграммы, такие относящиеся к реализации аспекты, как структура кода (компонента) и структура работы системы (развертывания). Для этой цели ARIS использует *UML-диаграмму компонент*.

Компоненты – это элементы, которые формируют блоки в процессе компиляции или компоновки, а также во время, например, выполнения системных операций. Первый тип отношений между компонентами представляет физическую структуру компонент. Компонента может содержаться в другой компоненте. Этот факт отображается соединением между ними *содержит*.

Второй тип взаимосвязи между *компонентами* – это отношение вызова. Одна компонента вызывает другую через *интерфейс*, изображаемый небольшим кружком. *Компонента*, обеспечивающая *интерфейс*, соединяется с помощью отношения *поддерживает* (сплошная стрелка), а компонента, использующая интерфейс, соединяется через отношение *вызывает* (пунктирная стрелка).

Конфигурация компонент может по-прежнему сохраняться в процессе работы системы (развертывания). Для этого *компоненты* группируются и привязываются к *пакетам* (называемым также узлами). Привязка осуществляется с помощью соединения *содержит* между *компонентой* и ее *пакетом*. При графическом представлении рекомендуется помещать компоненты в объекты типа *пакет*.

UML-диаграмма компонент приведена на рис. 5.2.7.

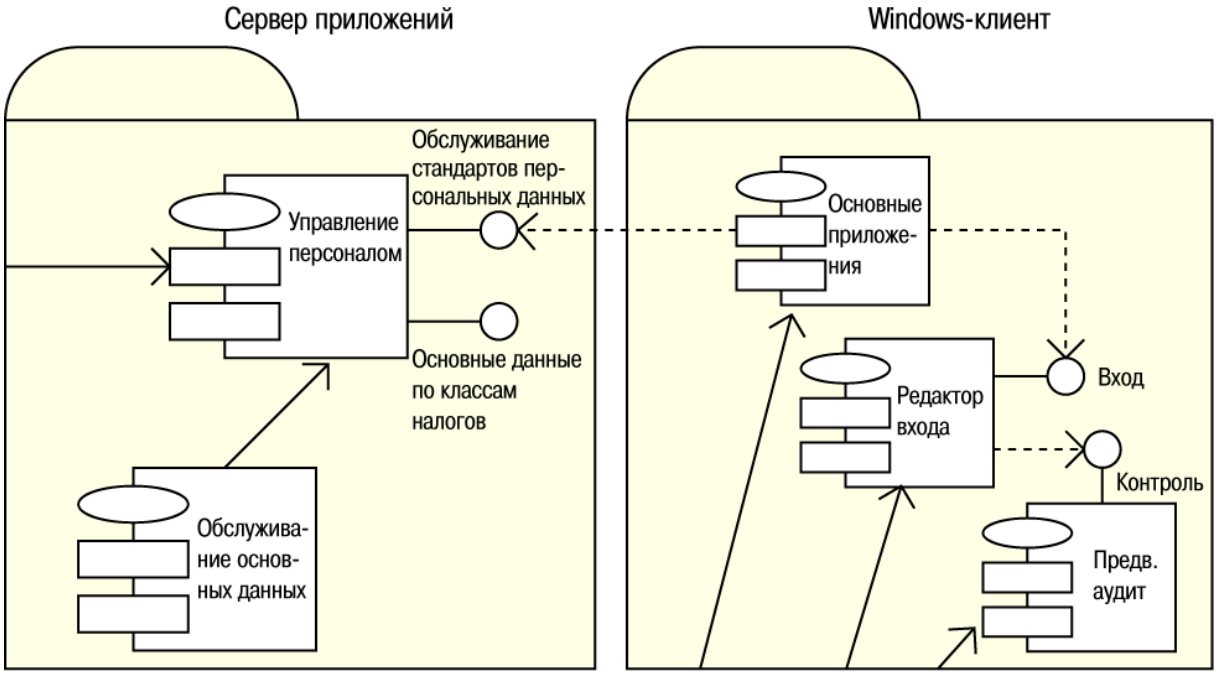


Рис. 5.2.7-1. UML-диаграмма компонент

5.3. Интеграция UML- диаграмм с другими моделями ARIS

5.3.1. Фундаментальные отношения между моделями

Различные модели ARIS связаны друг с другом. Моделирование и непосредственное описание соединений между объектами предоставляет два основных механизма для отображения этих отношений:

1. **Одни и те же объекты** могут использоваться в различных моделях. Например, тип объекта *Функция* существует как в *eEPC*, так и в *Функциональном дереве*. Одни и те же типы объектов могут изображаться различными символами и именами символов в различных моделях. Это происходит обычно в том случае, когда символ и его имя используются в одном методе (например, UML), но типы объектов с той же семантикой существуют в моделях других типов. Например, *Функция* (*eEPC Функция, Функциональное дерево* и т.д.) и *Операции* (в *UML-диаграмме класса*) имеют различные обозначения, но оба понятия отображают один и тот же тип объекта. Более того, один и тот же объект может встречаться как в *Функции*, так и в *Операции*. Его атрибуты идентичны во всех моделях. Идентичные объекты могут появиться в результате операций *Copy (Копировать)* и *Paste*

(Вставить) или при создании объекта с тем же именем, что и существующий объект, в диалоговом окне *Object Selection (Выбор Объекта)*.

2. Привязка объекта другой модели (команда *Create Object Assignment (Создать привязку объекта)* в диалоговом окне *Properties (Свойства) – Object (Объект)*, вкладка *Assignment (Привязка)*. В принципе спецификация объекта может быть выполнена более точно с помощью привязки модели.

Типы объектов в UML-диаграммах, идентичные типам объектов в других моделях ARIS, перечислены в следующей таблице:

Типы объектов в UML-диаграммах	Типы объектов в других моделях ARIS
Действие (UML-диаграмма действий)	Функция (eEPC, Функциональное дерево и т.д.)
Операция (UML-диаграмма класса, UML-диаграмма описания класса)	Функция (eEPC, Функциональное дерево и т.д.)
Исполнитель (UML-диаграмма использования приложений)	Тип сотрудник (eEPC, Организационная схема)
Состояние (диаграмма состояний)	Состояние объекта, результат (eEPC и т.д.)
Состояние объекта (диаграмма действий)	Состояние объекта, результат (eEPC и т.д.)

Идентичность типов объектов *Операции* и *Действия* следует из идентичности типов объектов для *Действия* и *Функции*. То же можно сказать о *Состоянии* и *Состоянии объекта*.

В UML-диаграммах типы объектов могут иметь следующие привязки:

Типы объектов	Привязываемые модели
Класс	eEPC, eERM, модель технических терминов, Модель данных IEF, OMT-модель описания класса, динамическая модель OMT, модель SAP-SERM, SeDaM, UML-диаграмма действий, UML-диаграмма описания класса, UML-диаграмма класса, UML-диаграмма состояний
Операция / Деятельность / Функция	eEPC, eEPC с потоком материалов, функциональное дерево, диаграмма описания функции, промышленный процесс, диаграмма информационных потоков, LLOVC, диаграмма потоков материалов, офисный процесс, PLOVC, матрица выбора процесса, модель SAP ALE Filter, функциональная модель SAP ALE, модель потока сообщений SAP ALE, диаграмма приложений SAP, системные атрибуты, UML-диаграмма действий, UML-диаграмма использования приложений, диаграмма цепочки добавленного качества
Состояние / Состояние объекта / Результат	eEPC, eEPC с потоком материалов, функциональное дерево, промышленный процесс, диаграмма обмена результатов, диаграмма обмена результатов (графическая), дерево обмена результатов, (графическая), офисный процесс, матрица выбора продукта, дерево продукта, диаграмма привязки продукта, матрица выбора процесса, UML-диаграмма состояний
Пакет	UML-диаграмма действий, UML-диаграмма описания класса, UML-диаграмма класса, UML-диаграмма взаимодействия, UML-диаграмма компонент, UML-диаграмма состояний, UML-диаграмма использования приложений
Исполнитель / Тип сотрудника	Ежегодный календарь, календарь смен
Решение / Разделить/ Синхро / Правило	Диаграмма правил
Использование приложений	eEPC, eEPC с потоком материалов, UML-диаграмма действий, UML-диаграмма взаимодействий, UML-диаграмма использования приложений, PCD, PCD с потоком материалов

5.3.2. Отношения между UML-диаграммами

Далее рассматриваются возможные связи и описываются рекомендуемые отношения.

5.3.2.1. UML-диаграмма класса и UML-диаграмма описания класса

UML-диаграмма описания класса применяется для привязки класса к *UML-диаграмме класса*. Класс может быть скопирован в диаграмму описания и в дальнейшем специфицирован. Как в *UML-диаграмме описания класса*, так и в *UML-диаграмме класса* могут использоваться одни и те же атрибуты и операции.

Пакеты могут быть привязаны с помощью *UML-диаграммы класса*, которая позволяет смоделировать привязку к классу. Для всех типов объектов, содержащихся в привязанной модели, автоматически создаются соединения *Содержит*.

5.3.2.2. UML-диаграмма класса и UML-диаграмма действий

Класс *UML-диаграммы класса* или *UML-диаграммы описания класса* может быть привязан с помощью *UML-диаграммы действий* для моделирования внутреннего процесса. Затем операции класса могут быть использованы как действия в *UML-диаграмме действий*.

5.3.2.3. UML-диаграмма класса и UML-диаграмма состояний

Для моделирования отдельных состояний, в которых может находиться класс, класс *UML-диаграммы класса* или *UML-диаграммы описания класса* может быть привязан с помощью *UML-диаграммы состояний*. Состояния *UML-диаграммы состояний* могут быть идентичны *Состояниям объектов* для *UML-диаграммы действий*, также привязанных к классу. Имена соединений (*иметь переход к*) между состояниями должны быть идентичны операции класса и/или действиям в *UML-диаграмме действий*.

5.3.2.4. UML-диаграмма класса и UML-диаграмма взаимодействия

UML-диаграмма взаимодействия показывает взаимосвязь между экземплярами объектов (*Объект*); таким образом, рекомендуется моделирование отношений объектов классов объектов (*Класс*) *UML-диаграммы классов*.Для этой цели должно быть создано отношение между классом и соответствующими *объектами* в *UML-диаграмме класса* или *UML-диаграмме описания класса* с помощью соединения *иметь экземпляр*. После

привязки класса к экземпляру объект может быть использован в *UML-диаграмме взаимодействия*. Имена операций в спецификации соединения *взаимодействует с* (атрибут *Операция*) между объектами должны быть опередедены как операция в присоединенном классе.

5.3.2.5. UML-диаграмма использования приложений

Варианты *Использования приложений* в *UML-диаграмме использования приложений* могут быть упорядочены иерархически. Они могут состоять из дополнительных вариантов использований (под)приложений. Это отношение устанавливается привязкой *Использования приложений* к *UML-диаграмме использования приложений.*

При моделировании процесса *использования приложений* может быть также связано с *UML-диаграммой действий* или *UML- диаграммой взаимодействия* .

5.3.3. Отношения с другими моделями ARIS

Интегрирование объектной ориентации UML и процессной ориентации при моделировании бизнес-процессов в наших рассуждениях в основном касалось отношений с другими моделями ARIS. Интеграция *UML-диаграммы класса* с событийной цепочкой процесса (eEPC) заслуживает специального рассмотрения. *eEPC* используется для моделирования процессов, в особенности в контексте бизнес-процессов.

5.3.3.1 UML-диаграмма класса и eEPC

Диаграмма *eEPC* содержит множество типов объектов, которые также используются в UML-диаграммах. Более того, как описано в разделе 5.3.1, некоторые типы объектов eEPC идентичны типам объектов UML; единственным различием между ними является графическое представление.

Классы с помощью соединений *является входом для* и *имеет выходом* могут быть использованы в *eEPC* как источники информации и как приемники функций. Если требуется более грубое или более детальное описание входа и выхода функций, могут быть созданы отношения входа и выхода от *функции* к *пакету* и *атрибуту*. Вместо отображения физического чтения или записи в этом случае будет представлено, каким образом создается или используется информация.

Функция может быть реализована с помощью одного или нескольких классов *операций*. Для достижения этого пользователи могут либо соединить *функции* и *операции* с помощью соединения *вызывает*, либо вследствие идентичности типа

объекта использовать *функции* непосредственно как операции в *UML-диаграмме класса.*

Диаграмма *eEPC* позволяет описывать процесс в пределах одного *класса* или одной *операции*. Для этих целей *eEPC* может быть привязана к обоим типам объекта.

5.3.3.2. UML-диаграмма состояний и eEPC

Состояния в *UML-диаграмме состояний* могут быть использованы в *eEPC* в качестве символов *Состояния объекта* или *Цикла разработки*. Эти символы могут быть привязаны к функции с помощью соединений *имеет выходом* и *является входом*. Поскольку *события* могут быть представлены состояниями объекта, при описании всех состояний объекта в *eEPC* может возникнуть семантическая избыточность. Если же необходимо смоделировать как *событие*, так и *состояние объекта*, семантическое дублирование изображается с помощью соединения *соответствует*.

5.3.3.3. UML-диаграмма использования приложений и eEPC

Пользователи, которые желают применить *UML-диаграмму использования приложений*, могут реализовать связь с *eEPC*-процессами двумя способами:

1. Описать процесс с помощью *диаграммы использования приложений*, причем *eEPC* связывается непосредственно с используемым приложением.
2. Более подробно специфицировать функцию для *eEPC* с помощью *UML-диаграммы использования приложений,* а затем связать *UML-диаграмму использования приложений* этой функцией.

При моделировании исполнителей в *UML-диаграмме использования приложений* оперируйте идентичностью типов объектов *исполнители* и *типы сотрудников,* для того чтобы иметь уверенность в согласованности процедурной организации *eEPC* с организационной структурой (например, в *организационной схеме*).

5.3.3.4. UML-диаграмма действий и eEPC

Процесс, отображенный в *UML-диаграмме действий*, может быть также описан с помощью *eEPC.* При этом будут реализованы все преимущества мощности *eEPC*. Если оба типа модели избыточно изображают одну и ту же фактическую информацию, то избыточность должна быть сведена к минимуму посредством введения одних и тех же объектов в обе модели (например, *Организационная единица*) или приминение идентичной символики для типов объектов, имеющих различное обозначение (например, *Действие* и *Функция*). Их содержимое может быть скопировано с помощью

операций *Copy (Копировать) и Paste (Вставить)* из *eEPC* в *UML-диаграмму действий* и наоборот, при этом значительно облегчится одновременное управление обеими моделями.

Обозначения для *Действия* и *Функции* при копировании изменяются автоматически. Однако содержимое модели может копироваться только в типы объектов, которые доступны в обоих типах моделей. Поскольку диаграмма *eEPC* имеет большое число типов объектов, значительная часть ее семантического содержания (например, все информационные носители, такие, как *Файл*, *Документ* и *Знания*, а также детальная организационная привязка *Функции* и т.д.) может быть потеряна при пересылке в *UML-диаграмму действий.*

5.3.3.5. UML-диаграмма класса и eERM

Если информационные системы на уровне формулировки требований, разработанных с объектно-ориентированной перспективой, должны быть конвертированы с помощью реляционной СУБД, структуры данных могут быть смоделированы в *eERM* с использованием моделей *сущность-отношение* (eERM). Возможна привязка класса *UML-диаграммы класса* или *UML-диаграммы описания класса* к диаграмме eERM.

5.3.3.6. UML-диаграмма использования приложений и eEPC

Если *UML-диаграмма использования приложений* должна применяться совместно с *eEPC,* рекомендуемая в этом случае процедура - моделирование процесса в рамках *Использования приложений* через привязку к *eEPC*.

6. Метод объектного моделирования

6.1. Введение

ARIS дополнительно поддерживает объектно-ориентированное моделирование, при котором используются графические обозначения *Техники объектного моделирования* (*OMT*).Однако OMT, наряду с методами Booch и OOSE, был интегрирован и преобразован в унифицированный язык моделирования - Unified Modeling Language (UML), также реализованный в ARIS (см. гл. 5). OMT все еще предлагается в данной версии инструментария ARIS для его совместимости с другими версиями, но мы рекомендуем унифицированный язык моделирования, а не OMT.

Далее описаны компоненты метода OMT и его реализация в ARIS.

6.2. Краткое описание метода OMT

OMT разработан для представления различных точек зрения при описании систем. Для этой цели применяются следующие методы:

- объектное моделирование,
- динамическое моделирование,
- функциональное моделирование.

Эти три метода моделирования ортогональны друг другу, но они не могут рассматриваться как полностью независимые друг от друга.

Объектное моделирование - представление статичных, структурных, а также ориентированных на данные аспектов системы, позволяющие отображать структуру объектов, их отношения к другим объектам и их атрибуты.

Динамическое моделирование - отображение временных, поведенческих и управляющих аспектов системы, позволяющие описывать последовательность операций с помощью последовательности событий.

Функциональное моделирование - отображение временных и функциональных аспектов системы, с помощью которого можно описывать преобразование значений.

Каждая из этих моделей содержит перекрестные ссылки на другие модели. Например, объектная модель описывает структуры данных, которые используются в динамической и функциональной моделях. Процессы в функциональной модели соответствуют операциям в объектной модели. Диаграмма состояний динамической модели целиком или частично описывает поведение объекта класса в объектной модели.

6.3. Использование OMT-диаграмм в ARIS

В дальнейшем мы покажем, как конструкции, предусмотренные методом OMT, отображаются, используются и соотносятся в ARIS друг с другом. Конструкции, применяемые при моделировании и определенные для OMT (например, *Класс*, *Процесс*, *Состояние*), не перекрывают другие конструкции ARIS (например, *Сущность*, *Функция*, *Тип сущности* и т.д.) и, таким образом, могут быть только повторно использованы в OMT-диаграммах. Следовательно, метод OMT должен пониматься как «независимый» метод.

6.3.1. Объектная модель

Представление экземпляров

При объектно-ориентированном моделировании объекты в основном документируются на уровне типа (например, на уровне класса). Однако может иметь смысл моделирование отдельных его экземпляров. В ARIS это изображается голубым прямоугольником с закругленными углами (см. рис. 6.3.1-1).

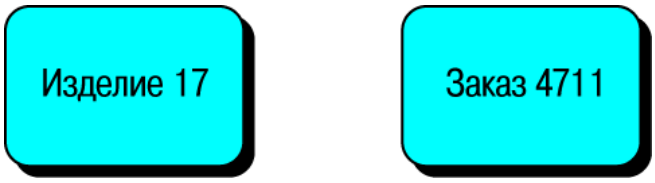


Рис. 6.3.1-1. Представление экземпляров

Представление классов

Классы описывают основные структуры моделируемого приложения. Графически они представляются в ARIS голубым прямоугольником с горизонтальными линиями (см. рис. 6.3.1-2).

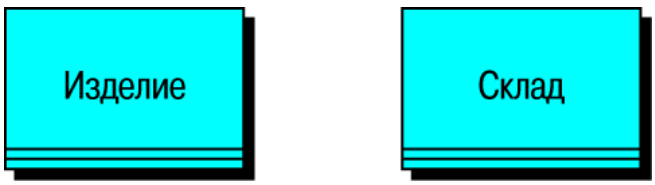


Рис. 6.3.1-2. Представление классов

Привязка экземпляров к классам

Если необходимо описать экземпляры, то можно графически представить их привязки к соответствующим классам. В этом случае соединению присваивается значение *является экземпляром* (см. рис. 6.3.1-3).

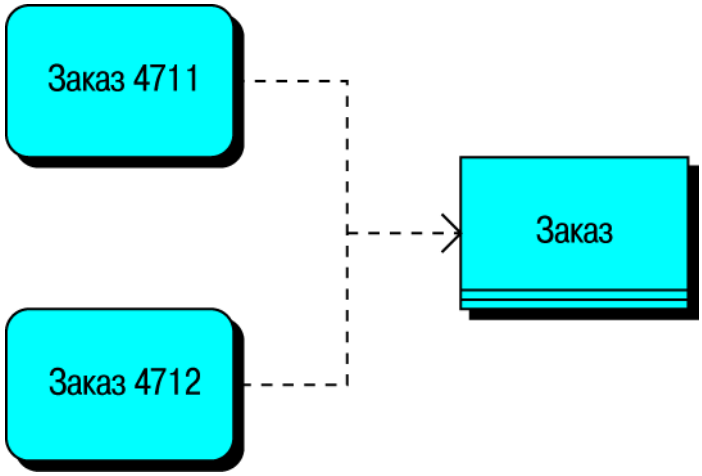


Рис. 6.3.1-3. Привязка экземпляров к классам

Привязка атрибутов к классам

Характеристики классов описываются с помощью атрибутов. При моделировании в ARIS – это отдельные объекты, обозначаемые специальными символами, которые присоединяются к соответствующим классам посредством соединений *имеет атрибут* (см. рис. 6.3.1-4). Разделение двух различных типов объектов (классы и атрибуты) необходимо для того, чтобы полностью реализовать возможности ARIS в навигации моделей и создании отчетов.

Для каждого атрибута можно определить, является ли он классом (это значение относится ко всем экземплярам класса) или атрибутом экземпляра.

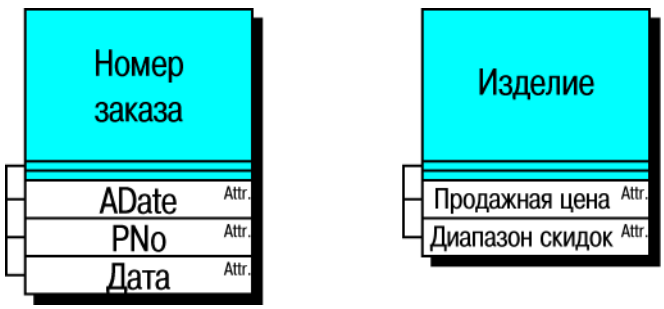


Рис. 6.3.1-4. Привязка атрибутов к классам

Привязка операций к классам

Функциональность, связанная с классами, описывается с помощью определения операций (методов). С этой целью определяются отдельные типы объектов, которые могут иметь отношения с классами (со смыслом *имеет операцию,* см. рис. 6.3.1-5).

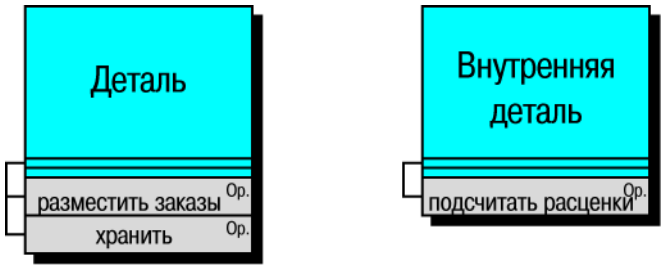


Рис. 6.3.1-5. Привязка операций к классам

Ассоциации между экземплярами

Отдельные экземпляры могут быть связаны друг с другом. Эти связи отображаются в ARIS с помощью ненаправленных соединений типа *быть связанным с* (см. рис. 6.3.1-6).

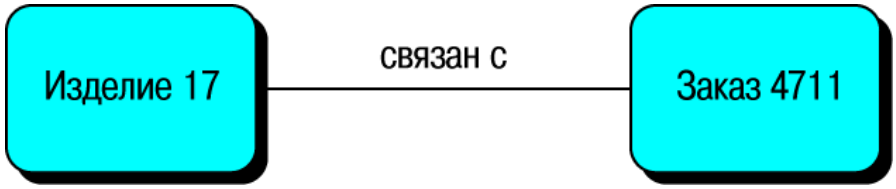


Рис. 6.3.1-6. Ассоциации между экземплярами

Ассоциации между классами

Классы также могут быть связаны в «ассоциации». Эти связи нам знакомы по модели «сущность-отношение». Они отображаются характерным символом (желтый ромб), что позволяет представлять n-отношения тем же способом (см. ниже). Соединения рисуются от символа класса к ромбу. Им может быть присвоена степень сложности, которая описывается атрибутом соединения *Множественность*. *Множественность* определяться следующими элементами, среди которых могут быть результаты соответствующего графического представления соединения:

- **1**
- **• • c**

- **cn**
- **n**

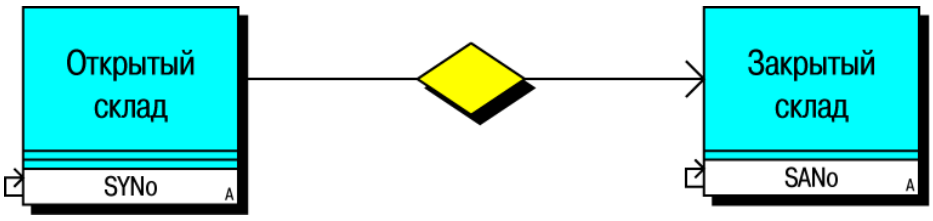


Рис. 6.3.1-7. Ассоциации между классами

N-Ассоциации между классами

Ассоциации между тремя (или n) классами отображаются присоединением третьего (или четвертого, пятого и т.д.) класса с помощью символа в виде ромба, определяющего связь (см. рис. 6.3.1-8).

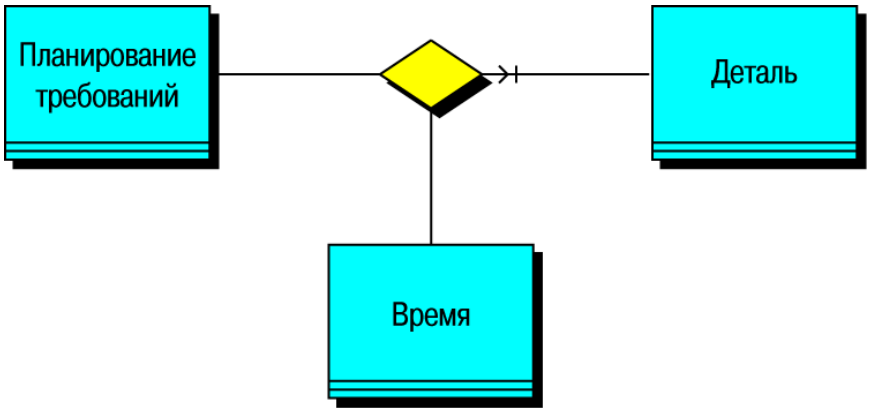


Рис. 6.3.1-8. Отношения между классами с тремя участниками

Моделирование ассоциации как класса

Ассоциация может пониматься как независимый объект и интерпретироваться как класс. Это может быть представлено графически направленными соединениями от символа в виде ромба к одному из символов классов, где затем перечисляются все атрибуты и операции (см. рис. 6.3.1-9). Этот «реинтерпретированный» класс, конечно же, может быть связан с другими классами.

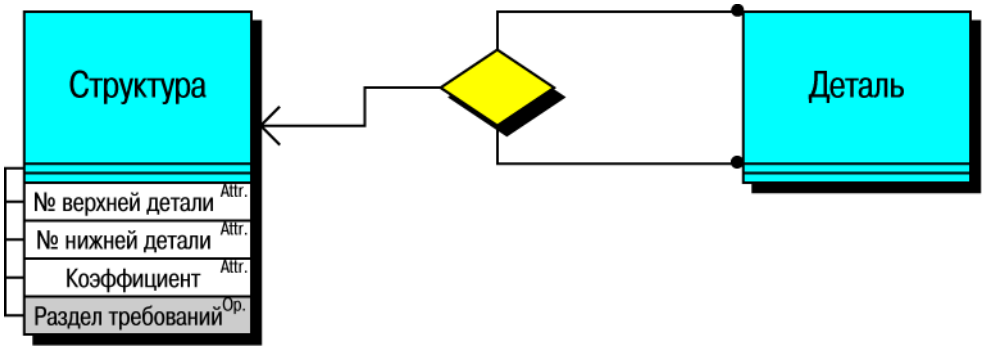


Рис. 6.3.1-9. Моделирование ассоциации как класса

Представление квалифицирующей ассоциации

Ограниченная ассоциация добавляет квалифицирующую информацию к нормальной ассоциации. Квалифицирующая информация помечается атрибутом, который уменьшает мощность ассоциации. Это имеет смысл для ассоциаций типа 1:m и n:m, поскольку объекты со стороны m-ассоциации дифференцируются тем же способом.

Квалифицирующая ассоциация отображается добавлением квалификационной информации вслед за соединением. Для этой цели вводится независимый атрибут *Квалификатор*, который (как и все атрибуты) может быть отображен графически (см. рис. 6.3.1-10).

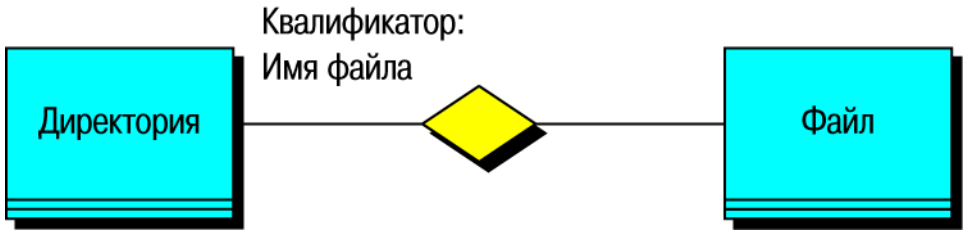


Рис. 6.3.1-10. Представление квалифицирующей ассоциации

Представление порядка ассоциаций

Если объекты ассоциации со стороны n упорядочены определенным образом, это можно представить графически (см. рис. 6.3.1-11). Для этой цели вводится независимый атрибут, располагающийся сразу за соединением между символами *Класса* и *Ассоциации*.

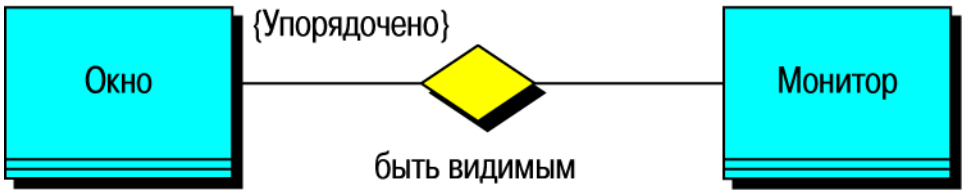


Рис. 6.3.1-11. Представление порядка ассоциаций

Агрегация между классами

Агрегация представляет отношение «часть-целое» и может рассматриваться как специальная форма ассоциации. Это отношение моделируется как направленная связь между классами (с помощью соединения типа *агрегировать*). Для графического описания класса, представляющего «целое», служит символ в виде белого ромба (группа компонент), как показано на рис. 6.3.1-12.

Рис. 6.3.1-12. Агрегация между классами

Обобщение и наследование

Одной из основных конструкций объектно-ориентированного моделирования является определение иерархии между классами, которая включает передачу атрибутов и операций от старшего класса к подчиненному. В ARIS это отображается независимым типом объекта (зеленый треугольник), связанным с отдельными «участвующими» классами (см. рис. 6.3.1-13). Кратное наследование также может быть отображено.

При необходимости для оператора обобщения можно добавить атрибут, который характеризует отношение обобщение/специализация, а также указывает, является ли специализация разъединяющей или неразъединяющей.

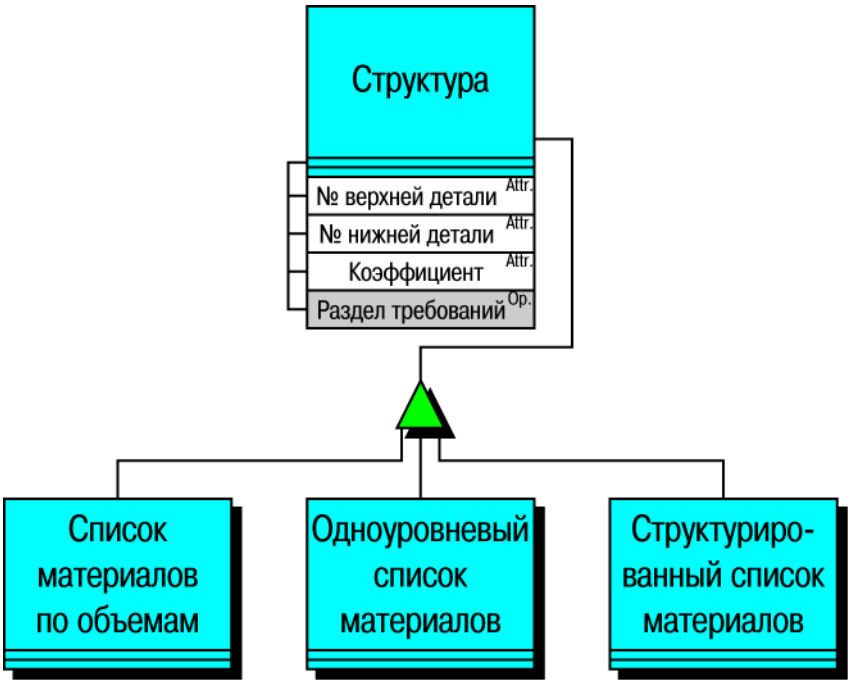


Рис. 6.3.1-13. Представление отношений между классами типа обобщение/специализация

Ограничения для классов, атрибутов и ассоциаций

Ограничения – это функциональные отношения между классами, атрибутами и ассоциациями в объектной модели, разработанной по методу OMT. В ARIS определяются независимые типы объектов (черная точка) для описания ограничений на атрибуты. На рис. 6.3.1-14 приведен пример, показывающий, что отношение высота/ширина для окон может принимать значения от 0,7 до 1,7.

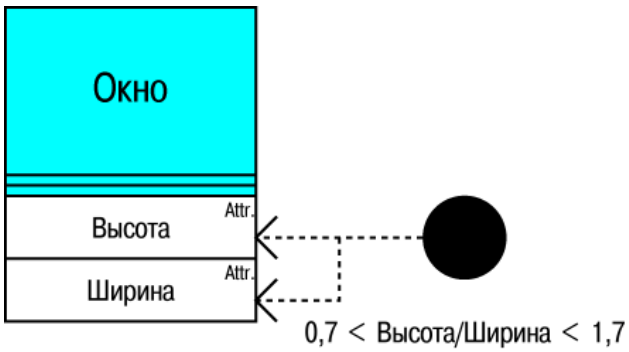


Рис. 6.3.1-14. Представление ограничений на атрибуты

Можно также определить ограничения на ассоциации. На рис. 6.3.1-15 приведен пример, отображающий, что множество лиц, образующих руководство комитета, естественно является подмножеством всех его членов. Этот факт может быть представлен направленной связью между символами ассоциаций.

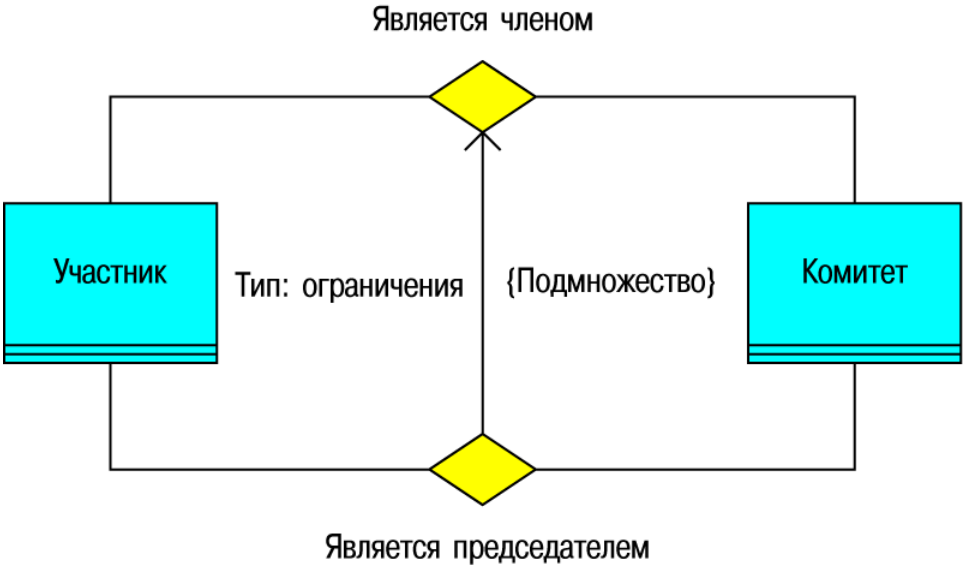


Рис. 6.3.1-15. Представление ограничений на ассоциации

Пример объектной модели

На рис. 6.3.1-16 показана типичная объектная модель, содержащая основные моделирующие конструкции.

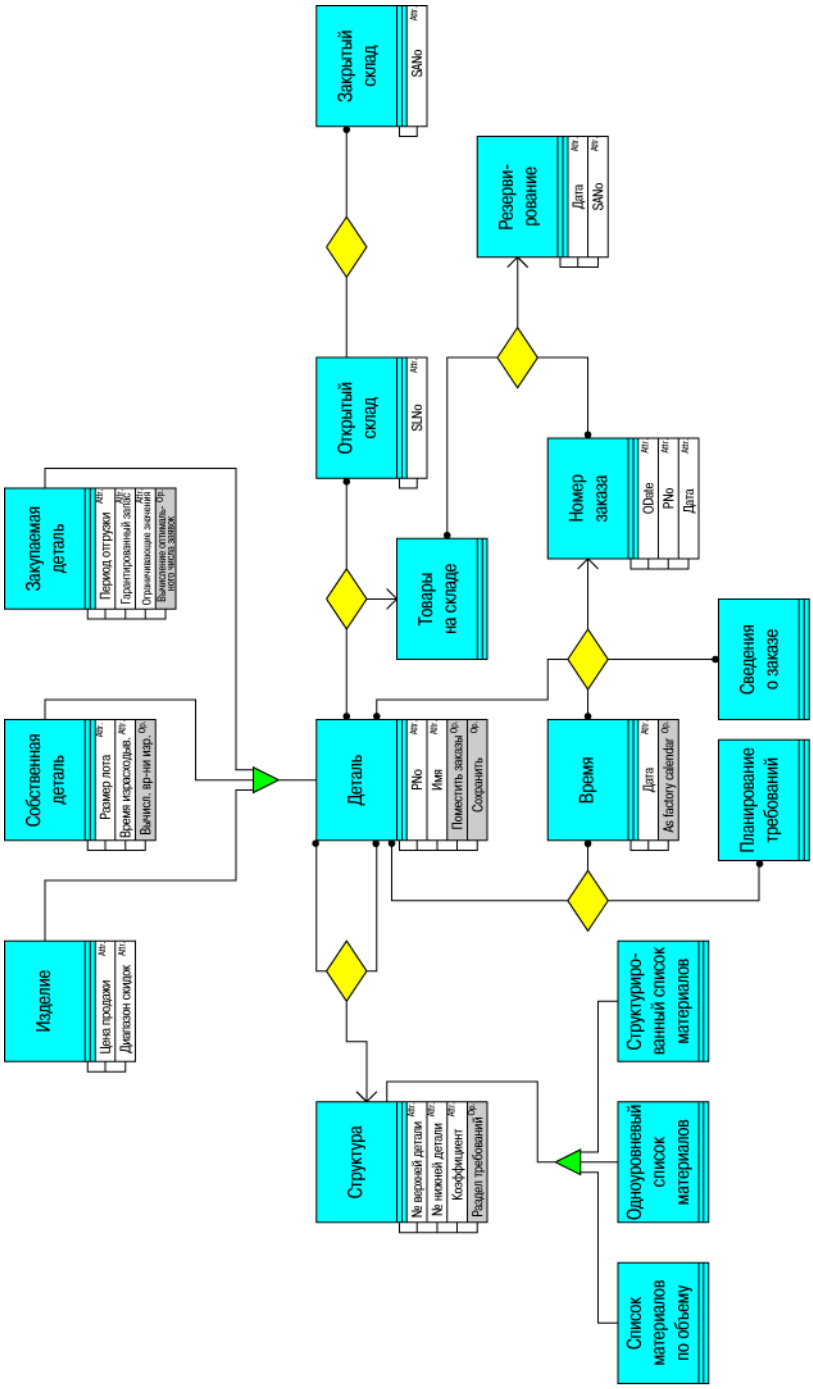


Рис. 6.3.1-16 Объектная модель

6.3.2. Динамическая модель

Динамическая модель – это совокупность диаграмм перехода состояний, где одна диаграмма перехода состояний обычно описывает поведение одного класса. Состояния

взаимосвязаны посредством направленных соединений, которые представляют события.

Представление начального и конечного состояний и состояний перехода

ARIS предлагает три графических символа для представления начальных состояний, конечных состояний и обычных состояний (см. рис. 6.3.2-1).

Рис. 6.3.2-1. Представление начального и конечного состояний, а также состояний перехода

Переходы между состояниями

Переходы между состояниями переключаются событиями. Соединение между двумя состояниями имеет тип *иметь переход в* (см. рис. 6.3.2-2).

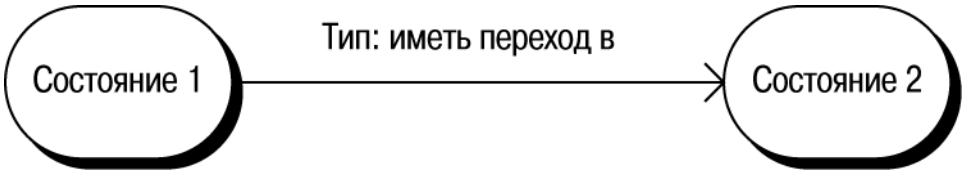


Рис. 6.3.2-2. Представление переходов между состояниями

К состояниям и переходам можно добавлять другую информацию. Для описания начальных, конечных и промежуточных действий используются атрибуты *выполнить/действие*, *вход/действие*, *выход/действие* и *событие/действие*. Условия перехода между двумя состояниями можно описать более подробно (см. рис. 6.3.2-3).

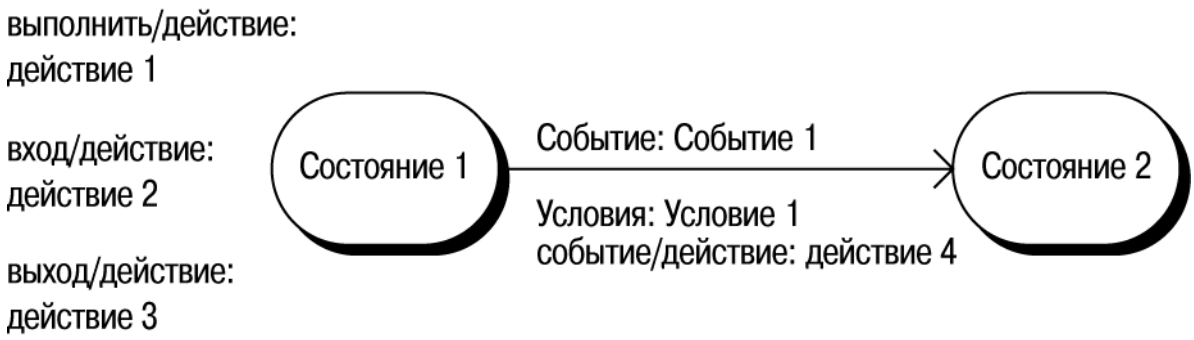


Рис. 6.3.2-3. Представление подробной информации о переходе состояний

Пример динамической модели

На рис. 6.3.2-4 приведен пример типичной динамической модели.

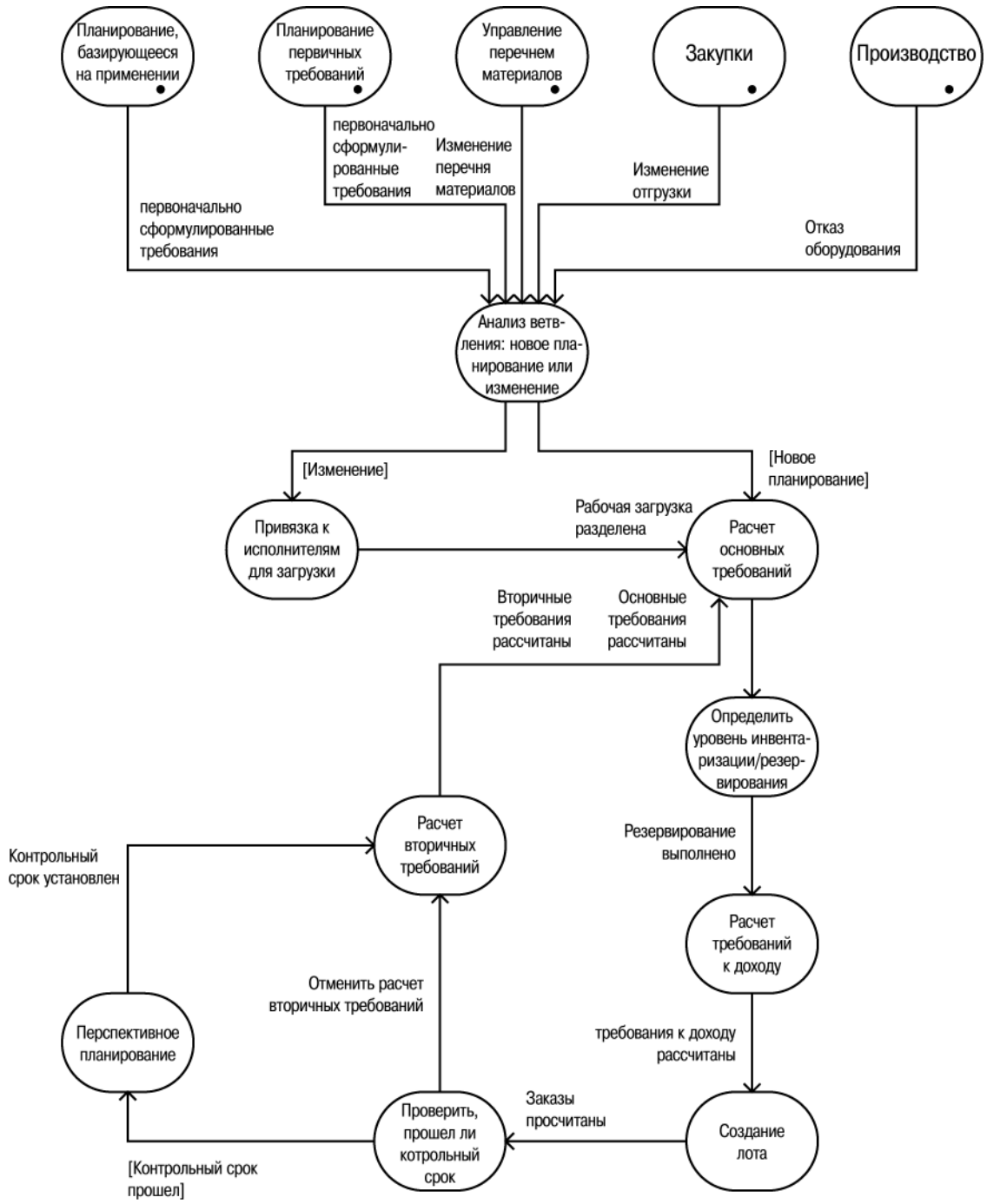


Рис. 6.3.2-4. Динамическая модель

6.3.3. Функциональная модель

Функциональная модель с помощью диаграмм потоков данных показывает, как в процессе вычислений получаются выходные данные из входных данных.

Представление хранилищ данных

Хранилища данных используются для пассивного хранения данных. В ARIS они представляются двумя горизонтальными линиями (см. рис. 6.3.3-1).

Рис. 6.3.3-1. Представление хранилища данных

Представление процессов

Процессы, преобразующие данные, представляются в ARIS желтыми овалами (см. рис. 6.3.3-2).

Рис. 6.3.3-2. Представление процессов

Представление исполнителей

Исполнитель – это объект, который активизирует поток данных путем создания и использования значений данных. Таким образом, исполнитель может рассматриваться в рамках графического представления как источник и приемник. Он отображается в виде белого квадрата (см. рис. 6.3.3-3).

Рис. 6.3.3-3. Представление исполнителей

Представление потоков данных

Потоки данных соединяют выход одного процесса или объекта с входом другого. Они моделируются такими объектами, как *Значение данных,* и обычно сопровождаются описанием данных (см. рис. 6.3.3-4).

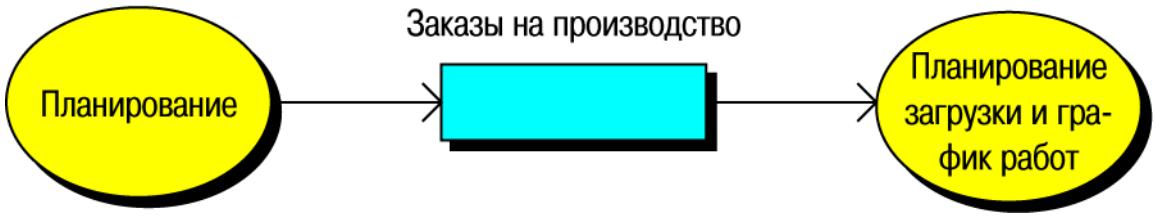


Рис. 6.3.3-4. Представление потока данных

Разделение потоков данных

Если значения данных должны быть посланы в различные места, то потоки данных могут быть разделены. ARIS использует свои символы (соединения) для представления такого разделения (см. рис. 6.3.3-5).

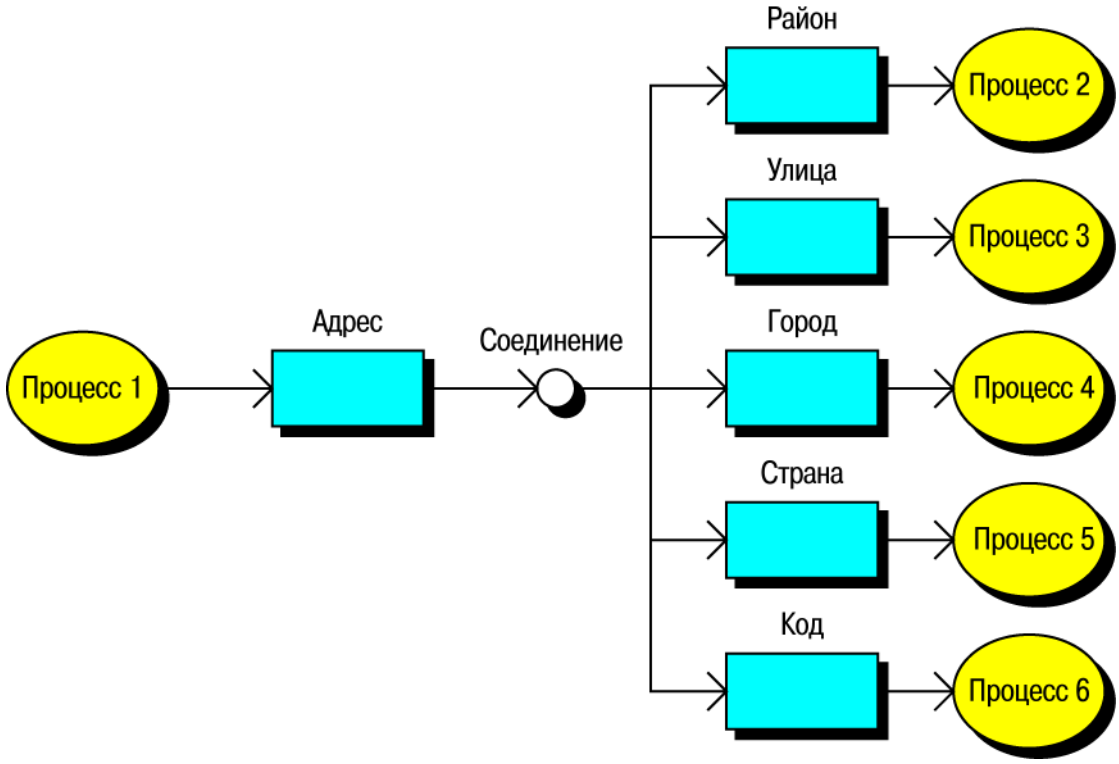


Рис. 6.3.3-5. Представление разделения потоков данных

Пример функциональной модели

На рис. 6.3.3.6 представлен пример функциональной модели.

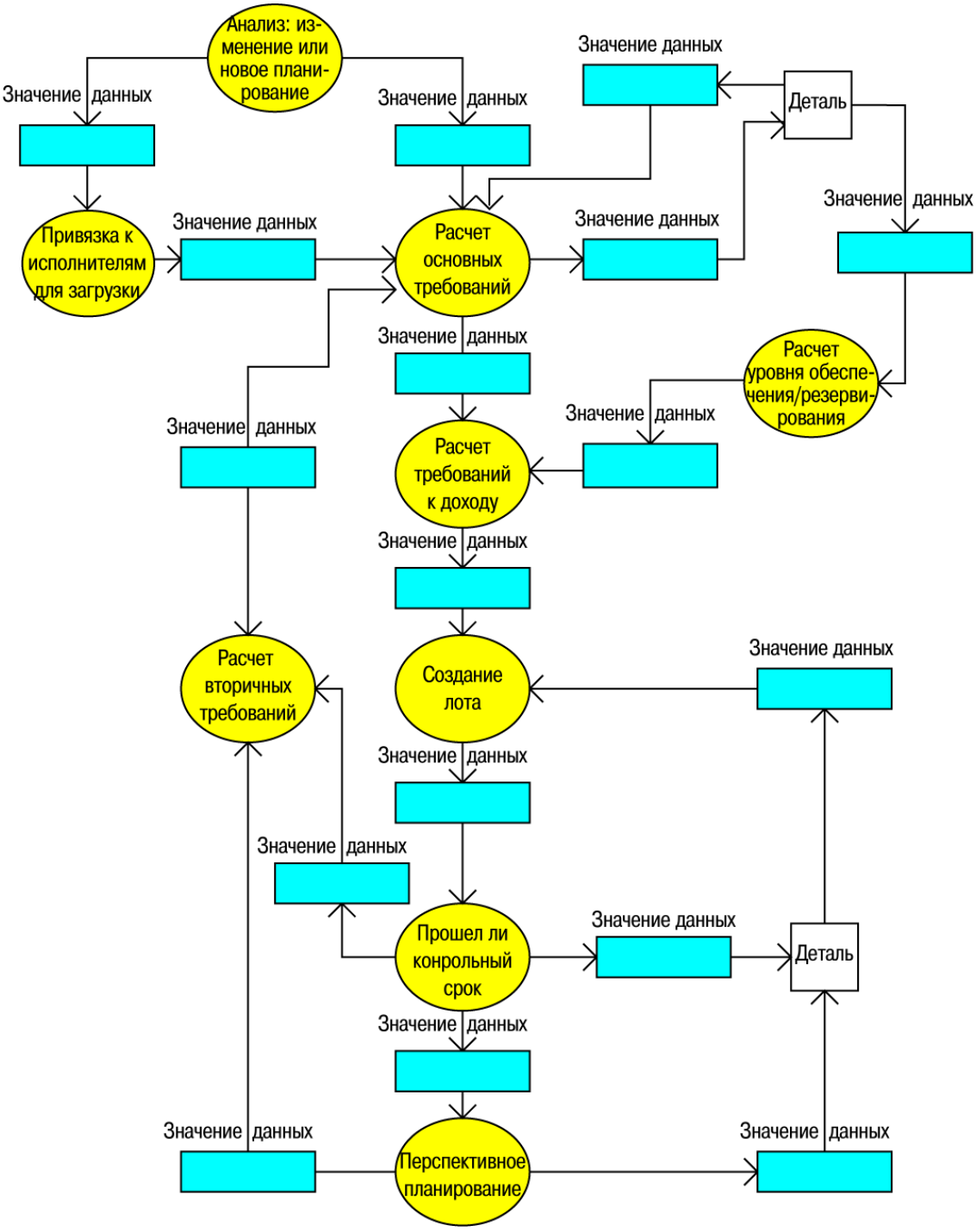


Рис. 6.3.3-6. Функциональная модель

6.3.4. Как можно упорядочить объекты иерархически

- Тип объекта *Класс* может быть иерархически упорядочен с помощью привязки динамической модели к перемещениям документа в пределах этого класса.
- Тип объекта *Класс* может быть иерархически упорядочен с помощью модели описания класса, привязывающей атрибуты и операции к этому классу. Возможные отношения в этой модели представляют подмножество объектной модели.
- Типы объектов *Состояние*, *Конечное состояние* и *Начальное состояние* могут быть представлены более подробно с помощью динамической модели, реализующей переходы на различных иерархических уровнях.
- Тип объекта *Процесс* может быть представлен более подробно с помощью функциональной модели, реализующей описание потоков данных на различных уровнях.
- Тип объекта *Хранилище данных* может быть иерархически упорядочен с помощью объектной модели.
- Типы соединений в функциональной модели могут быть представлены более подробно с помощью объектной модели и диаграммы декомпозиции значений данных, документирующей детали потока данных.

7. Методы управления знаниями

7.1. Введение

Целью управления знаниями является систематическое обновление знаний как ресурса компании, важность которого постоянно возрастает. Термин «управление знаниями» охватывает разработку, мониторинг, поддержку и совершенствование сратегий, процессов, организационных структур и технологий для эффективной обработки знаний в рамках компании. Он включает также все действия, имеющие отношение к приобретению знаний, их подготовке, передаче и использованию. Действия по управлению знаниями обычно не совершаются в изоляции. Они сопровождают главным образом оперативные и директивные бизнес-процессы в компании. Следовательно, на бизнес-процессы, обработку знаний, организационные структуры, информационные системы и т.д. необходим интегральный взгляд.

Большинство из этих аспектов может быть отражено с помощью таких известных методов ARIS, как eEPCs, организационные схемы, диаграммы описания функций, eERMs и других. Однако если цель заключается в точном представлении, анализе и совершенствовании дисциплины управления знаниями, необходимы дополнительные средства представления для идентификации и структурирования содержимого соответствующих категорий знаний, описания и распределения знаний в пределах организации, моделирования создания и использования знаний в бизнес-процессах.

По указанной причине добавляются два новых типа объектов – *Категория знаний* и *Документированные знания*, а также два новых типа моделей – *Диаграмма структуры знаний* и *Карта знаний*. Кроме того, существующие типы моделей для представления бизнес-процессов (eEPC, PCD, офисные процессы и т. п.) расширяются конструкциями для управления созданием и использованием знаний.

Новые типы объектов и моделей полностью и методично интегрируются в наиболее важные типы моделей на уровне формулировки требований (такие, как eERM, организационные схемы и функциональные деревья), предоставляя возможность для дальнейшей интеграции. Это позволяет, например, использовать модели оптимизации бизнес-процессов применять для анализа и совершенствования управления знаниями.

Диаграмма структуры знаний располагается в модели данных на уровне формулировки требований. Карта знаний, подобно расширенным типам моделей при моделированиии бизнес-процессов, помещается в управляющую модель на уровне формулировки требований.

7.2. Типы объектов для моделирования обработки знаний

7.2.1. Категории знаний

Тип объекта *Категория знаний*, графически обозначаемый овалом с клапаном (см. рис. 7.3.1-1), представляет объект с содержимым, относящимся к конкретным знаниям. Примерами *Категорий знаний* являются знание об управлении проектом, конкретные знания о производстве, конкретные знания о технологии, знания о клиенте и конкурентах и т.д. Эти категории помогают в классификации знаний, которыми компания обладает или которые ей необходимы.

Знания, распределяемые по отдельным категориям знаний, могут быть 1) неявными знаниями, которые невозможно полностью документировать, 2) знаниями о квалификации служащих или групп или 3) явными знаниями, которые могут быть документированы в виде описаний или технической документации (чертежей). Категории знаний часто содержат более одного вида знаний. Например, знания об управлении проектом включают опыт руководства проектом и советы по управлению проектом.

Приведенные в таблице дополнительные атрибуты, так же как и основные атрибуты *Описание*, *Комментарий*, *Источник* и т.д., служат для более подробного описания категорий знаний.

Наименование атрибута	**Область**	**Описание / Пример**
Частота корректировки	Градации: ежечасно, ежедневно, еженедельно, ежемесячно, ежегодно, редко, никогда	Частота корректировки описывает, как часто знания отдельной категории должны обновляться, чтобы отражать текущее состояние. Например, основные тригонометрические знания должны редко обновляться, если вообще должны, в то время как знания о ценах на товар необходимо обновлять ежедневно или ежечасно.
Важность	Процент: 0-100	Категория *Важность* может изменяться от 0 (совсем неважно) до 100% (чрезвычайно важно).
Степень охвата	Процент: 0-100	Текущая степень охвата для отдельных знаний может меняться от 0 (совсем не охвачено) до 100% (максимально возможный охват).

Наименование атрибута	Область	Описание / Пример
		Степень охвата категории знаний для отдельной организационной единицы или лица отображается в карте знаний с помощью соответствующего атрибута типа соединения *распорядиться*.
Преимущество	Процент: 0-100	Относительное преимущество вашей компании над конкурентами в терминах знаний может изменяться от 0 (конкуренты имеют максимально возможное преимущество над вашей компанией) до 100% (ваша компания имеет максимально возможное преимущество над конкурентами).
Использование знаний	Процент: 0-100	Степень применения отдельной категории знаний может изменяться от 0 (конкретные знания не используются совсем) до 100% (оптимальное использование конкретных знаний).
Желаемая степень охвата	Процент: 0-100	Желаемая степень охвата отдельных знаний может изменяться от 0 (не охвачены совсем) до 100% (максимально возможная степень охвата).
Важность в будущем	Градации: резкое уменьшение, уменьшение, постоянная, увеличение, резкое увеличение	*Важность в будущем* отображает ожидаемую тенденцию в изменении важности данной категории знаний для компании.
Скорость структурных изменений	Процент: 0-100	Скорость структурных изменений отображает, насколько быстро должны изменяться методы, направленные на приобретение конкретных знаний (0%: не требуется изменений, 100%: максимальная скорость структурных изменений).

Указанные выше атрибуты служат для доступа к конкретной *Категории знаний* применительно к вашей компании. Следовательно, они могут быть использованы как основа при идентификации важных или крайне необходимых оценок для усовершенствования управления знаниями. Часто оказывается полезным графическое представление их величины.

Простейший способ отображения – скопировать и вставить значения из окна *Attribute (Атрибут)* в программу обсчета таблиц, которая может построить нужную диаграмму. Например, может быть сгенерирована гистограмма сравнения текущей и желаемой *Степеней охвата* для рассматриваемой *Категории знаний*.

7.2.2. Документированные знания

В отличие от типа объекта *Категория знаний*, который охватывает неявные и явные знания, тип объекта *Документированные знания* касается исключительно *Категории знаний*, которая явно документирована или которая может быть в принципе документирована. Примером такого типа знаний является знание об использовании программного обеспечения, которое документировано в руководстве. При распределении информации по категориям, осознание различий между основными категориями знаний и документированными знаниями способствует выявлению возможностей и ограничений в поддержке информационных систем, если только документированные знания могут быть сохранены, переданы или обработаны электронным способом.

Тип объекта *Документированное знание* обозначается треугольником с клапаном. Он содержит те же самые типы атрибутов, что и тип объекта *Категория знаний*.

7.3. Типы моделей для моделирования обработки знаний

7.3.1. Структурная диаграмма знаний

С помощью структурной диаграммы знаний пользователь может поместить категории знаний в подгруппы, основываясь на их содержимом. Пример структурной диаграммы знаний приведен на рис. 7.3.1-1. Категория знаний может включать другие категории знаний, а также документированные знания. Документированные знания можно разделить на несколько подкатегорий документированных знаний, но они не могут включать общие знания.

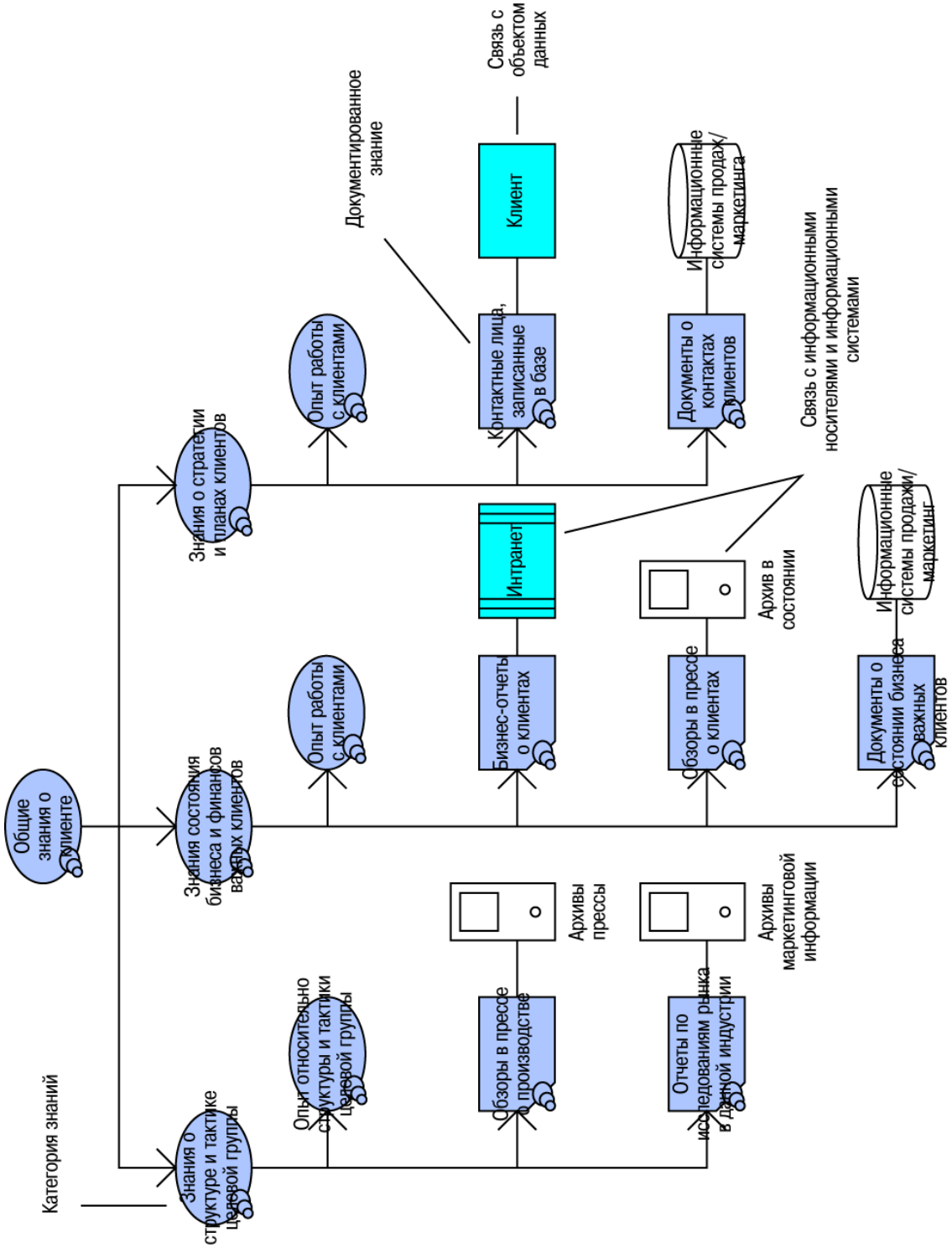


Рис. 7.3.1-1 Структурная диаграмма знаний

Для документированных знаний структурная диаграмма знаний может отображать информационную среду, где знания документированы, или указывать, какие информационные объекты в модели данных (или классах объектно-ориентированных систем) служат для документирования этих знаний. Наконец, могут быть также смоделированы типы или классы прикладных систем для управления знаниями.

7.3.2. Карта знаний

Карта знаний отображает распределение различных категорий знаний в рамках организации. Типы объектов в организационной модели (например, *Организационная единица*, *Должность*, *Сотрудник*, *Оборудование*, *Группа*) могут быть привязаны к категориям знаний с помощью соединения *распорядиться*. Кроме факта, что отдельный сотрудник или организационная единица обладает знаниями конкретной категории, может быть также описана и степень охвата.

Соединение *распорядиться* содержит атрибут *Степень охвата*, который может иметь процентное выражение степени охвата знаний в категории знаний, выбранной для организационной единицы. Значение 100% соответствует максимальному охвату, а значение 0% означает полное отсутствие знаний. 0% соответствует отсутствию соединения.

Кроме такой количественной оценки, возможна качественная оценка в виде графика. Для этой цели служит атрибут соединения *Качество охвата*, который может принимать значения *низкий*, *средний*, *высокий* и *максимальный*. Эта информация может быть представлена с помощью графического символа на соединении, как это показано на рис. 7.3.2-1. Значения атрибутов *Степень охвата* и *Качество охвата* не являются взаимосвязанными. Если используются оба атрибута, то предполагается, что значение *низкий* относится к степени охвата <25%, *средний* – 25-50%, *высокий* – 50-75%, и *максимальный* – 75-100%.

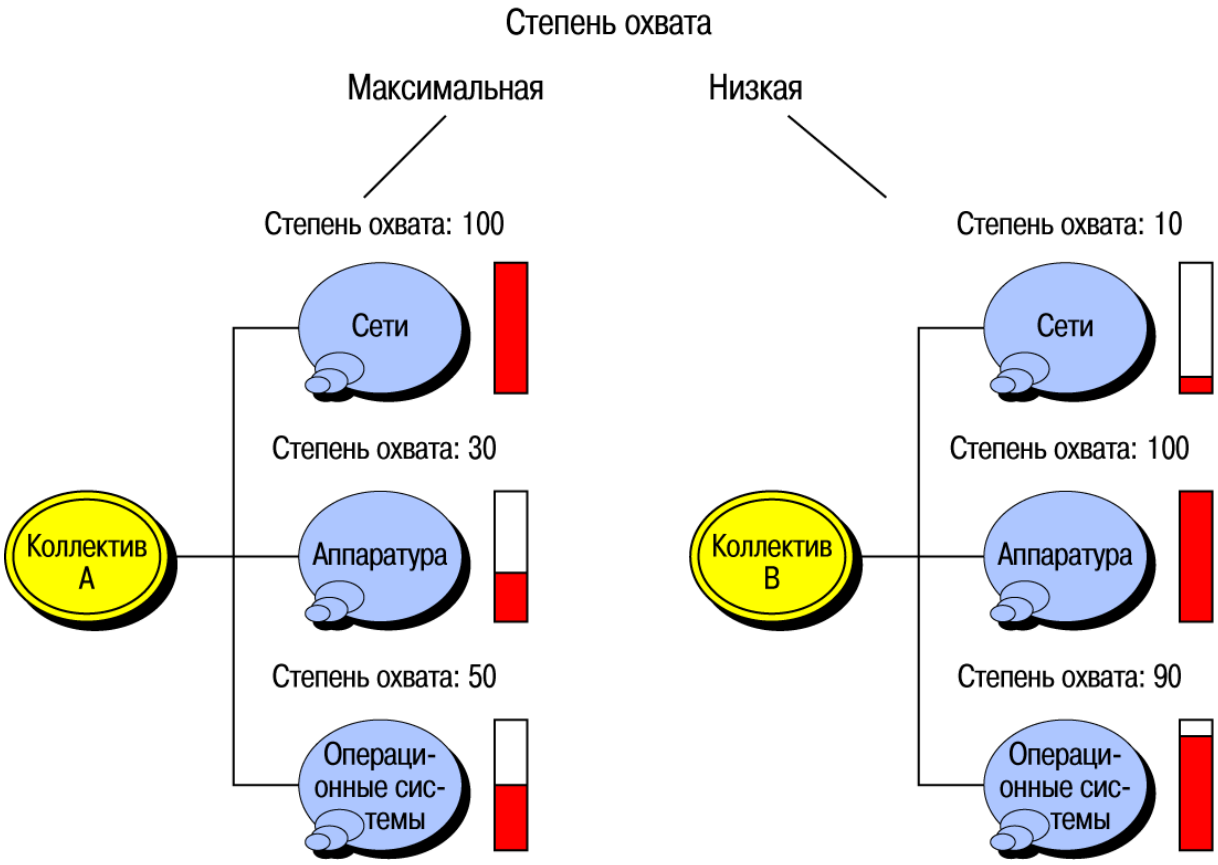


Рис. 7.3.2-1. Карта знаний, ориентированная на организационную единицу

Карта знаний, приведенная на рис. 7.3.2-1, имеет ориентированную на организационные единицы структуру, т.е. соответствующая *Категория знаний* «связывается» с каждой организационной единицей. Конечно, возможно также выбрать *Категории знаний* в качестве центральной модели и моделировать вокруг них соответствующие организационные единицы. Параметры навигации ARIS (вкладка *Relationship (Отношения)* в диалоговом окне *Properties - Object (Свойства – Объект)*) позволяют легко найти другие связи для организационной единицы или *Категории знаний*.

Для карт знаний часто используется матричное представление. Это может быть достигнуто организацией нескольких экземпляров одной и той же категории знаний в столбцы, как показано на рис. 7.3.2-2. В этом примере даются только имена *Категорий знаний*, расположенные в верхней части рисунка, как это принято обычно в таблице. Для других экземпляров имена и атрибуты опущены. На рисунке также показано альтернативное визуальное представление для различных степеней охвата: *Категории знаний* изображаются графическими фигурами разного размера.

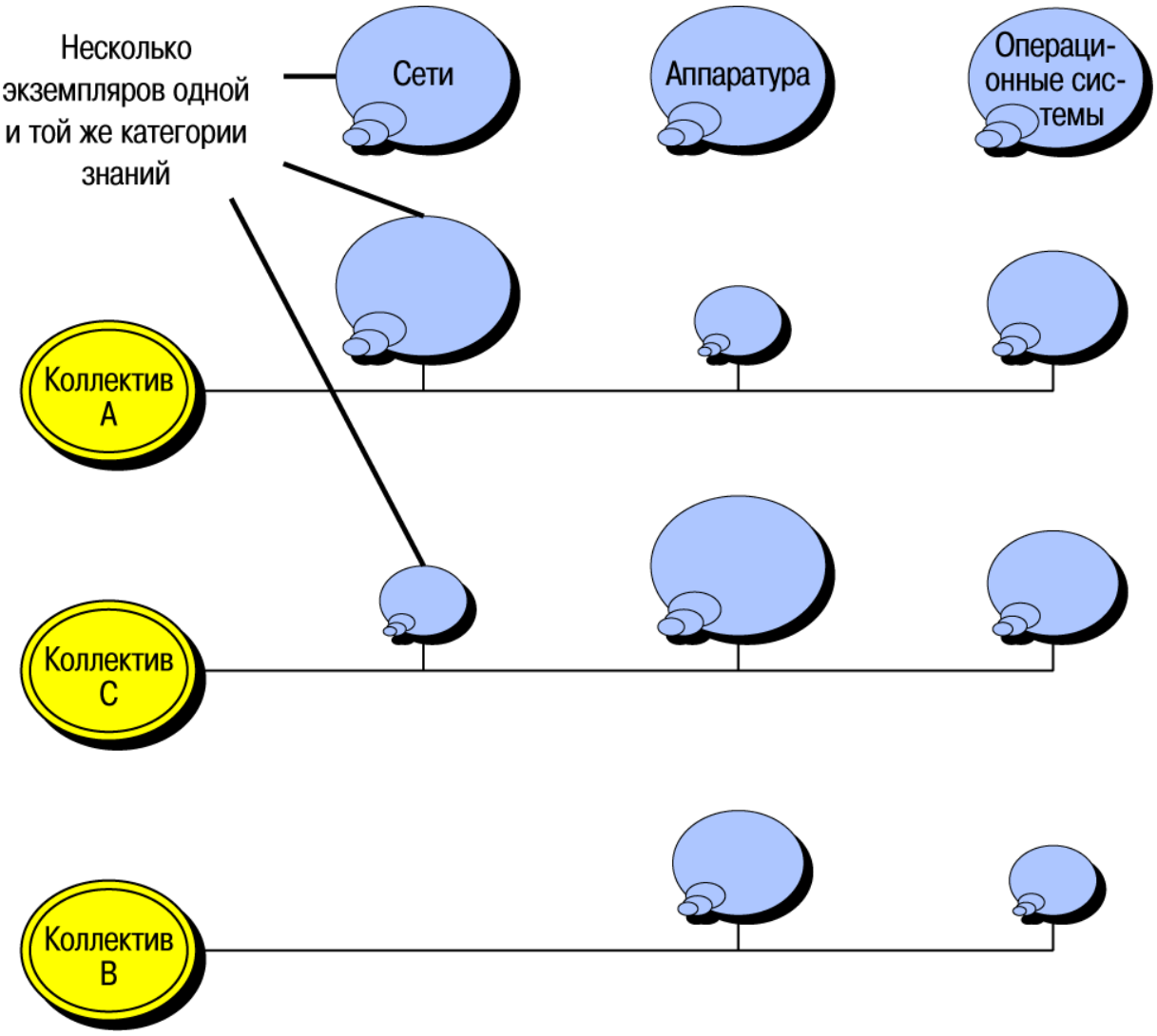


Рис. 7.3.2-2. Карта знаний. Матричное представление

При привязке категорий знаний к отдельным сотрудникам и оценке этих категорий отдавайте себе отчет в том, что сбор, документирование и в особенности электронная обработка данных, относящихся к служащим – это предмет многих ограничений, связанных с законами и/или политикой компании. При создании, использовании или распространении карт знаний такого типа необходима уверенность в том, что это полностью согласуется с существующими законами или политикой компании.

7.3.3. Представление обработки знаний в бизнес-процессах

Создание и применение знаний в бизнес-процессах компании моделируется с помощью типов моделей, используемых для представления бизнес-процессов (eEPC, eEPC с потоком материалов, офисный процесс, промышленный процесс, PCD, PCD с потоком материалов). Типы объектов *Категория знаний* и *Документированные знания* теперь доступны в указанных диаграммах. Можно определить, какой тип знаний (общие или документированные) необходим для выполнения функции, и отметить, какие знания создаются и/или документируются, когда функция выполняется.

Этот тип представления позволяет изучать бизнес-процессы в терминах обработки знаний. Например, может быть обнаружен пробел в необходимых знаниях и определен уровень повышения квалификации, который требуется при выполнении отдельной функции.

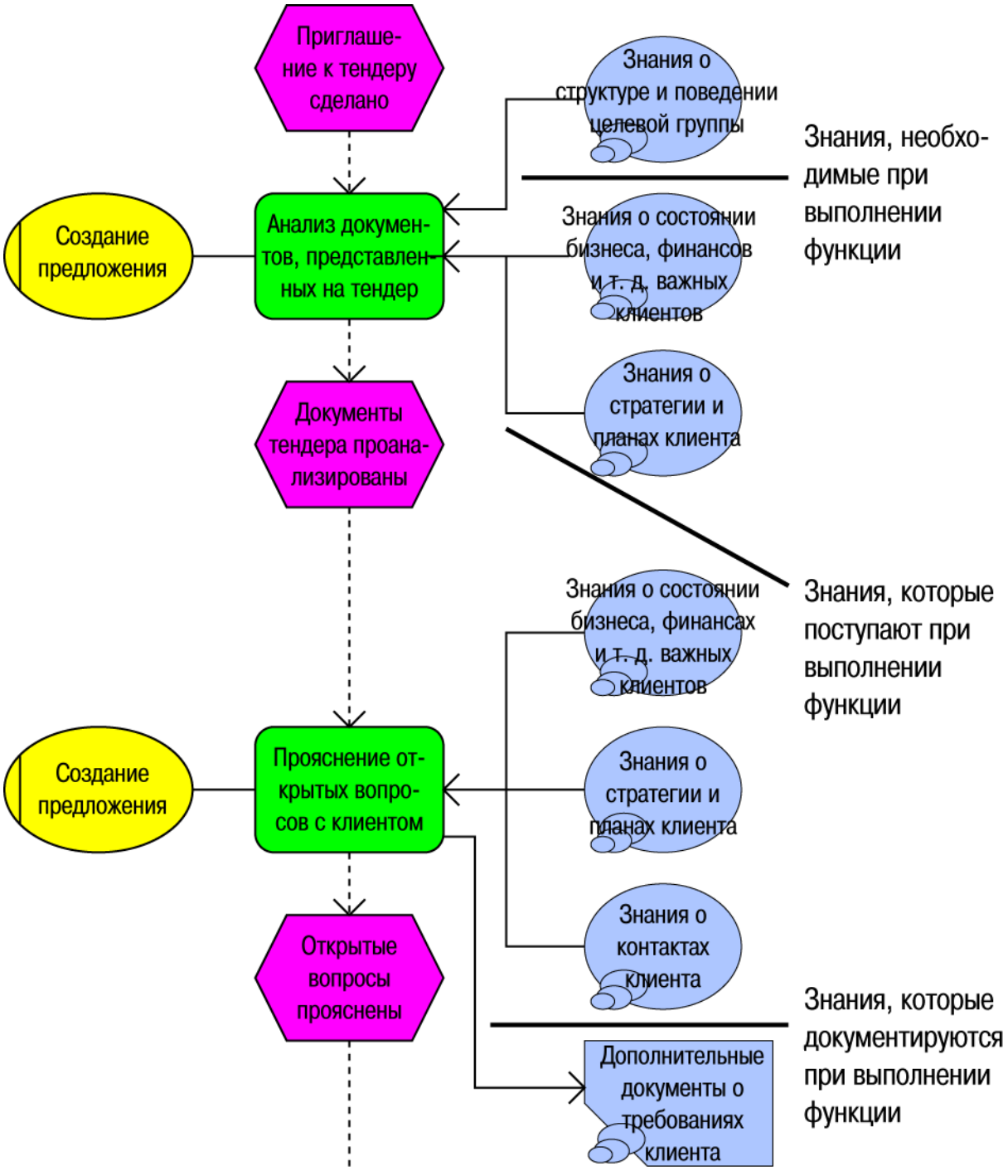


Рис. 7.3.3-1. Обработка знаний в модели eEPC