Август-Вильгельм Шеер

Моделирование бизнес-процессов

Издание 2-е, переработанное
и дополненное

Перевод с английского

Научная редакция
канд. техн. наук
Каменнова М. С.,
канд. хим. наук
Громов А. И.

Весть-МетаТехнология
Москва, 2000

Август-Вильгельм Шеер. «Моделирование бизнес-процессов».
Издание 2-е, переработанное и дополненное. Пер. с англ.

Научная редакция и предисловие:
канд. техн. наук Каменнова М. С., канд. хим. наук. Громов А. И.
Переводчик: Михайлова Н. А.
Оригинал-макет: Каменнов Н. Г.

ISBN 5-89163-049-4

ARIS (Архитектура интегрированных информационных систем) представляет собой уникальный метод оптимизации бизнес-процессов и реализации прикладных систем, завоевавший международное признание. В этой книге подробно описывается использование методов ARIS для моделирования и реализации бизнес-процессов при помощи унифицированного языка моделирования (UML). Создаваемая в результате информационная модель служит краеугольным камнем для систематизированного и интеллектуального метода разработки прикладных систем. Внедрение методов ARIS иллюстрируется на множестве примеров, взятых из реальной жизни, включая управление знаниями, реализацию систем workflow и стандартных программных решений (в частности, SAP R/3).

ISBN 3-540-64438-5 © Springer-Verlag Berlin - Heidelberg 1999
ISBN 5-89163-049-4 © ОАО «Весть» 2000, перевод, предисловие

«Весть-МетаТехнология», 115446, Москва, Коломенский пр. 1а,
т. (095) 115-60-01, ф. (095) 112-23-33, e-mail: consulting@vest.msk.ru

ISBN 5-89163-049-4

ООО «Издательство «Серебряные нити»
129515, а/я 34

Предисловие к русскому изданию

Искусство прогресса заключается в поддержании порядка среди изменений и продолжении изменений среди порядка.

Альфред Вайтхэд

Менеджер современного предприятия тяжело вздохнул... Еще бы! Новая эра глобальной конкуренции и технологических перемен привела к необходимости удовлетворения серьезных и в то же время противоречивых требований. Себестоимость деятельности должна быть снижена до минимума, но при этом для сохранения конкурентоспособности требуются постоянные инвестиции в развитие и совершенствование технологий рабочих процессов. К этому следует добавить качество, которое требует рынок, невзирая на все инновации и дешевизну того, что ему предлагается. Плюс требования государственных и международных стандартов. Немногие сегодня могут себе позволить не обращать внимания на это. Даже не слишком поворотливый ВАЗ, долгое время делавший удивленный вид, как бы говоря: «Мы ведь выпускаем самые дешевые в мире автомобили, так какого же качества вам нужно?», стал задумываться над этой проблемой. Как может сегодняшнее предприятие быстро уменьшить расходы на свое производство и заниматься инновациями, и при этом не просто сохранять, а повышать совокупное качество производства и продукции, да и к тому же еще соответствовать все возрастающим требованиям со стороны стандартов?

Один из подходов к этой проблеме - методология процессного управления. Практикой современного управления, или менеджмента, признано, что прежние представления о бизнесе или производственной деятельности – сквозь призму схемы организационной структуры или изолированных подразделений – не отвечают задачам предприятия и бизнеса в нынещеих условиях. Процессное же управление фокусируется на ключевых процессах предприятия, которые предоставляют добавленное качество потребителю или рынку. В зависимости от школы и тенденций на рынке консалтинга методология процессного управления имеет много имен: разработка бизнес-процессов, улучшение бизнес-процессов, реинжиниринг бизнес-процессов - это лишь некоторые из них.

Перечисленные тенденции отчетливо прослеживаются во всем мире на протяжении уже 20 лет. Благодаря реальному совершенствованию управления предприятиями и бизнесом с помощью инновационных технологий западная экономика не только стабилизировалась, но и был выровнен экономический дисбаланс во многих странах. В результате за очень короткий (с точки зрения истории) срок было создано Европейское Экономическое Сообщество с введением моновалюты.

Россия, в отличие от Запада, последнее десятилетие самозабвенно занималась выборными технологиями и публичной политикой, которые полностью захватили внимание и силы общества. Когда же грянул гром Интернета и понятия е-бизнеса и е-коммерции стали настолько обыденными, что непонятно, как раньше можно быть жить без них, Россия, опираясь на дюжину-другую неизменно алчущих хакеров, решила этот раунд не пропускать. Стали появляться и до поры до времени действовать Интернет-проекты, очень близкие по своей сути к известным всем «пирамидам».

Проблема не в том, что в России нельзя реализовать Интернет-проект - с технической точки зрения здесь нет никаких ограничений, а в том, что любой бизнес-проект, и Интернет в особенности, требует выверенной технологии управления. Ина-

че говоря, должна возникнуть синергия между процессами, объективно присутствующими в бизнесе, людьми, занятыми в этих процессах и действующими часто весьма субъективно, технологиями управления и законами, управляющими отношениями между предприятиями, людьми, рынком и государством, т.е. речь идет о повышении эффективности процесса за счет взаимовлияния участвующих в нем функций и объектов.

Итак, мы имеем сложную систему: «процессы - люди - технологии - законы» со множеством прямых и обратных связей, нарушение любой из которых неизбежно выводит систему из продуктивного развития. Западное общество инстинктивно или сознательно, но очень аккуратно и бережно подошло к смене технологий управления, предварительно создав модели и описание бизнес-процессов, а затем, подготовив людей и подправив законы, оно начало внедрять новые технологии.

Однако недавно в Европе разорился первый Интернет-магазин. По мнению экспертов, это не единичный случай, а уже наметившаяся тенденция. В 1999 г. этот магазин был назван одним из самых богатых Интернет-проектов Европы. Его ликвидатор, компания KPMG, намерена продать магазин одному из 30 покупателей, но аналитики из компании Merrill Lynch предсказывают, что в ближайшие год-два будут поглощены либо обанкрочены 75% европейских Интернет-проектов. Сегодня в Интернет активно выходят традиционные ритейлеры, имеющие хорошо отлаженные механизмы выполнения заказов. Согласно прогнозам экспертов, инвесторы в первую очередь будут уделять внимание реалистичным бизнес-планам и профессиональной подготовке молодых Интернет-компаний в области менеджмента.

Банкротство европейских и американских Интернет-проектов, равно как и российских, проистекает все из того же желания заработать как можно больше, думая как можно меньше. Молодые зарубежные Интернет-компании, как и их российские коллеги, не утруждают себя анализом и построением бизнес-процессов, как с точки зрения самого бизнеса, включая бизнес- риски, так и с точки зрения законодательства - внутреннего и международного.

Казалось бы, если все так плохо у всех, то что же грозит России при ее отставании в эпоху Интернета? Ответ прост. В Европе и Америке созданы базовые предпосылки для успешного ведения Интернет-бизнеса, выстроена организационная и правовая инфраструктура существования подобных проектов. Больше половины экономических субъектов ориентированы на процессное управление и готовят свои команды для оптимизации процессов управления. Доктриной стала жизненная продуктивная философия развития бизнеса «стратегия подчиняет себе процессы, а процессы – структуру», а не наоборот, как до сих пор в России.

Стоимость процессной инновации в Америке, Японии, Англии и Германии с каждым кварталом года становится все ниже, и, следовательно, она все более привлекательна для мелкого бизнеса. Если рассматривать это явление с точки зрения НВП (национального валового продукта), то как следствие процессной инновации наблюдается снижении себестоимости НВП при повышении качества. А при отложенном снижении рыночной стоимости НВП (что вполне естественно, поскольку никто, снизив себестоимость, не спешит снизить рыночную стоимость) наблюдается рост экономического потенциала страны в целом.

В то же время, по мнению авторитетных экспертов из Garthner Group, 75% расходов в структуре себестоимости НВП кроется в организации управления ресурсами, а оптимизация этих расходов за счет процессной организации управления и перевода большинства процессов в Интернет-процессы (или е-процессы) позволяет

высвободить до 80% от суммарной составляющей. Это означает, что только благодаря переходу на процессное управление и повышению эффективности ресурсов мы можем высвободить 60% из структуры себестоимости, т.е. дать экономике двойное ускорение. Эти данные справедливы и для благополучных Европы и Америки, где стабильная экономика и сложившиеся традиции в рыночных отношениях. Относительно России такие оценки делать трудно, так как у нас нет процессного управления как явления, эффективность ресурсов не обсуждается, а, значит, нечего и оптимизировать.

Для России возникла серьезная угроза потери потенциала для воспроизведения НВП, если в ближайшие два-три года не произойдет переосмысление и принятие необходимости скорейшего перехода в масштабе страны на процессное управление и эффективное управление ресурсами. Этот процесс уже начался, и все больше раздается голосов представителей промышленной России, которые понимают, зачем и часто знают, как нужно двигаться в этом направлении. Настоящая книга как раз для них - для тех, кто осознает неизбежность решения давно назревших проблем и готов быть первым среди первых.

Книга профессора А.-В.Шеера «Моделирование бизнес процессов» призвана выполнить сложную миссию. Она устанавливает мосты между специалистами АСУ, информационных департаментов, традиционно либо просящих средства на развитие, либо отчитывающихся за средства, и руководством предприятий (бизнеса), проводящим в жизнь линию стратегического развития их деятельности и стремящим сократить расходы. Ее не просто читать ни тем, ни другим, однако она необходима и тем, и другим и заслуженно может стать их настольной книгой. Чтение этой книги будет значительно более продуктивным с точки зрения количества практических выводов, если читатель предварительно познакомиться с первой книгой профессора А.-В.Шеера «Бизнес-процессы. Основные понятия. Теория. Методы», которая в переводе на русский язык издана в июне 1999 г.

Это фундаментальный труд по теории бизнес-процессов, посвященный именно теоретико-методологическим аспектам их описания и анализа. В ней изложена концепция архитектуры бизнес-инжиниринга (АБИ), которая представляет собой четко сформулированную методологию проведения всего цикла работ по совершенствованию бизнеса - от формирования стратегических целей компании до подробнейшей спецификации проекта информационной системы. Методология дана в аспекте новейших достижений в области информационных технологий и тенденций их развития.

Вторая книга сосредоточена на моделировании бизнес-процессов. Проблема состоит в том, что при любом моделировании, а бизнес-процессов в особенности, существует две серьезные опасности. Первая заключается в том, что всегда хочется охватить моделью как можно больше и в ширь, и в глубь. Результат получается плачевный: такие модели не существуют, они не жизнеспособны из-за своей неоправданной сложности. Другая опасность противоположна первой: модель оказывается слишком поверхностной и не описывает важнейших свойств реальной системы, вывод – она не нужна. Книга профессора А.-В. Шеера, которую мы имеем честь представлять, позволяет удержать одних от чрезмерной глобализации, а других - от увлечения рисованием вертикальных переходов при детализации процессов до «чиха вахтера».

Книга требует «осторожного» чтения и предполагает осмысление прочитанного материала с позиции опыта читателя. Очень велико искушение, назвав ARIS инструментом для моделирования, сразу броситься рисовать свои процессы, рас-

считывая увидеть в них что-либо новое или неожиданное. Разочарование здесь неизбежно: процессы не будут нарисованы, и ничего нового обнаружить не удастся. Но если вкусить архитектурного видения процессов, а отсюда - использовав тот или иной метод, будь то ролевое или событийно-функциональное моделирование, или любой другой из набора объемом в 82 подхода, - сделать шаг к пониманию текущей ситуации, то можно легко увидеть все несовершенство, всю избыточность и негибкость собственной деятельности. То, что раньше казалось очевидным, может быть сомнительным, а то, с чем были связаны сомнения, окажется реальным. Таким образом, возникает возможность пластической хирургии деятельности с предсказуемым результатом без каких-либо потерь и приостановки производства.

При рассуждениях о процессном управлении и ресурсной эффективности можно легко впасть и в другую крайность - отказавшись от функциональной модели организации деятельности, рассматривать предприятие, ориентированное исключительно на процессы. В этом случае мы снова получим отрицательный результат: эффективность процессов начинает доминировать над эффективностью ресурсов. Глубокий философский смысл кроется в том, что любая крайность, будь то чисто функциональный подход или чисто процессный, не позволяет добиться синергии в реальном процессе. Так же обстоит дело и с выбором технических решений для информационной поддержки деятельности, осуществляемым на основании стратегического планирования и определения критических факторов успеха. Попытки глобальной автоматизации, равно как и игнорирование рутинных функций для нее, неизбежно приводят к увеличению накладных расходов и дискриминации самой идеи. Этот вопрос очень квалифицированно освещен в книге.

Книга А.-В. Шеера претендует стать не только бестселлером в данной специализированной номинации, но и реальным пособием по практическому освоению науки моделирования бизнес-процессов. Автор ведет терпеливого читателя через все тернии анализа деятельности под углом процессного рассмотрения к созидательному искусству стратегического и тактического моделирования, предлагая путь от общего видения через стратегическое понимание к процессной декомпозиции и ролевому моделированию, от формирования фундаментальных требований к спецификации служебных обязанностей и определению структуры СУБД, от удовлетворения протоколов специализированных систем класса R/3 к построению информационных порталов. Все это сегодня становится доступным российскому читателю, который не сидит сложа руки, а думает о завтрашнем дне России.

Александр Громов, канд. хим. наук,
Мария Каменнова, канд. тех. наук

Москва, май 2000 г.

Предисловие ко второму изданию

С момента первой публикации в 1992 году книга «Архитектура интегрированных информационных систем» *(Architecture of Integrated Information Systems)* пользуется огромной популярностью. Использование бизнес-моделей для документирования стандартного программного обеспечения дало блестящие результаты. Программный продукт ARIS Toolset, разработанный фирмой IDS Prof. Scheer GmbH на базе концепции ARIS, сегодня занимает лидирующее положение на мировом рынке инструментальных средств для инжиниринга бизнес-процессов. ARIS Toolset нашел применение в университетах США, Европы, Южной Африки, Бразилии и стран Азиатско-Тихоокеанского региона, предоставив в распоряжение научных, исследовательских и проектно-конструкторских учреждений, занимающихся вопросами корпоративной организации и информационных технологий для бизнеса, современный инструмент для инжиниринга бизнес-процессов.

Бурное развитие в области информационных технологий (ИТ), происшедшее с момента выхода в свет первого издания этой книги, привело к появлению такой массы новых аспектов и такого количества дополнительной информации, что мы сочли необходимым полностью переработать содержание книги, разбив ее на две: «Бизнес-процессы. Основные понятия. Теория. Методы» (ARIS – Business Process Frameworks), «Моделирование бизнес-процессов» (ARIS – Business Process Modeling).

Каждая книга рассчитана на свой контингент читателей. Если первая предназначена для тех, кого интересуют вопросы проектирования бизнес-процессов, стандартных приложений и их аспекты, связанные с бизнесом, то вторая подробно знакомит с методами моделирования и информационными технологиями.

Об этой книге

Книга «Моделирование бизнес-процессов» знакомит читателя с методами моделирования, разработкой соответствующих метамоделей и их объединением в информационную модель ARIS.

По сравнению с первым изданием, представленные здесь методы моделирования значительно расширены. Так, более подробно рассматривается моделирование планирования стратегических бизнес-процессов и методы объектно-ориентированного моделирования, особенно унифицированный язык моделирования (UML).

Ввиду грядущей стандартизации UML, метамодели ARIS в соответствии с UML представлены как диаграммы классов, хотя по содержанию они ничем не отличаются от тех, что были представлены в первом издании в виде моделей сущность–отношение (ERM).

Мы особо стремились к тому, чтобы обеспечить применимость этих моделей к конфигурированию бизнес-приложений. Поэтому все бизнес-приложения, бизнес-объекты, различные возможности конфигурирования программного обеспечения для планирования и мониторинга бизнес-процессов, а также системы workflow рассматриваются здесь в рамках концепций архитектуры бизнес-инжиниринга («ARIS – здание бизнес-инжиниринга»).

Изложение концепции мы дополнили главами, где рассматриваются вопросы прикладного характера, в частности, внедрение моделей ARIS для реализации систем workflow и бизнес-приложений SAP R/3, использование инфраструктуры ARIS Framework для разработки программ и разработка приложений с помощью UML.

Книга может заинтересовать менеджеров по информационным технологиям, консультантов, преподавателей и студентов, занимающихся информатикой, бизнес-информатикой и родственными дисциплинами.

Я благодарен Dipl.-Kff. Урсуле Маркус за координацию и правку перевода с немецкого на английский, Dipl.-Kfm. Франку Хаберманну — за тщательное редактирование немецкой рукописи, cand. rer. oec. Стефану Андресу и cand. rer. inform. Йохену Кунце — за подготовку иллюстраций на английском языке. Неоценимую техническую помощь оказали Dipl.-Inform. Томас Фельд, Dipl.-Kfm. Йенс Хагемейер, Dipl.-Hdl. Майкл Хоффманн и д-р Маркус Нюттгенс.

Я также хотел бы выразить благодарность Кристиану К. Тьюсу из «The Localizer» за скрупулезный перевод текста на английский язык.

Саарбрюккен, Германия, октябрь 1998 г.

Август-Вильгельм Шеер

Классификация содержания

Написанные автором книги по инжинирингу бизнес-процессов образуют стройную систему, которая представлена на рис. I.

Бизнес-информатика перекидывает мостик между теорией бизнеса и информационными и коммуникационными технологиями, устанавливая между ними двунаправленную связь. Информационные и коммуникационные технологии следует анализировать с точки зрения того, каким образом новые технические процедуры позволяют реализовать новые концепции приложений для бизнеса. «Направление влияния» показано стрелкой в левой части рис. I. В бизнес-информатике необязательно разбираться во всем спектре информационных технологий — достаточно знать, как применять сегмент, отвечающий за изменения в концепциях бизнес-приложений. В этой области бизнес-информатика играет особенно важную роль.

Стрелка в правой части рис. I иллюстрирует влияние потребностей бизнеса на развитие информационных и коммуникационных технологий.

Оба направления отношения рассматриваются в книге «Principles of Efficient Information Management» («Принципы эффективного управления информацией»), второе издание которой было опубликовано в 1991 году.

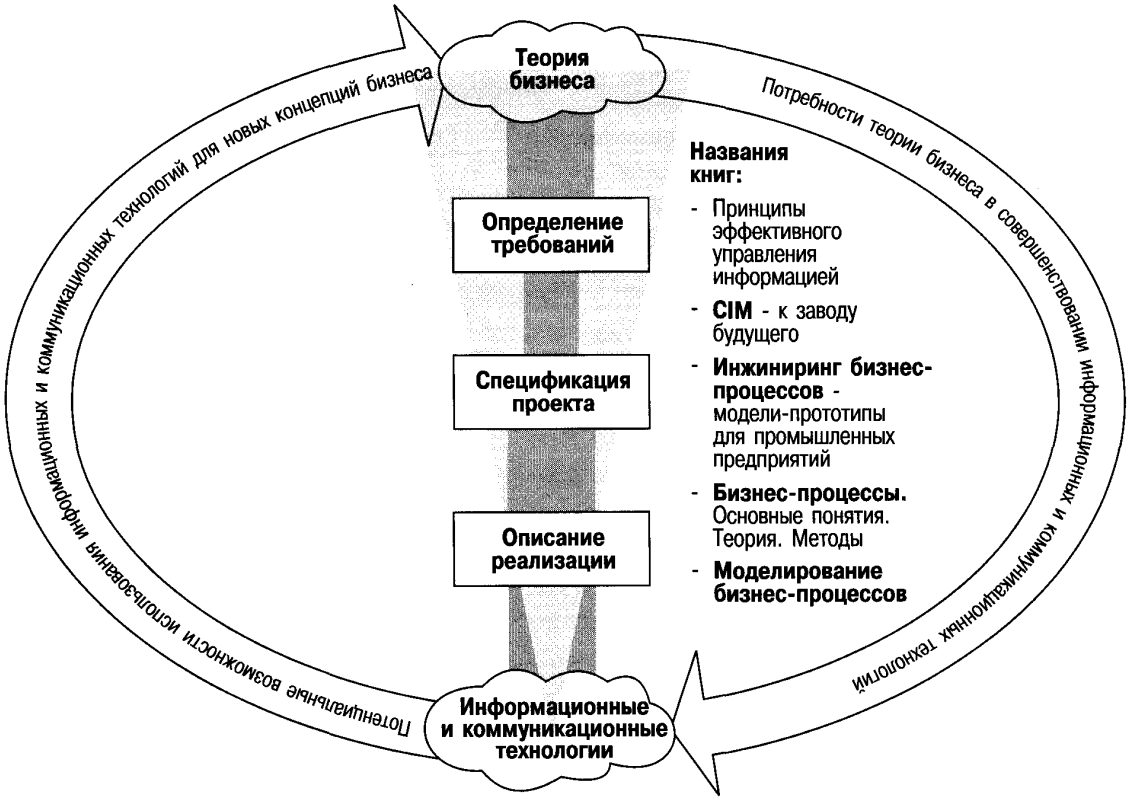


Рис. I. Технический профиль книг автора

Важнейшие аспекты влияния информационных технологий на бизнес-процессы рассматриваются в книге «CIM (Computer Integrated Manufacturing) – Towards the Factory of the Future» («CIM (компьютеризованное управление производством) – к заводу будущего»), которая в 1994 году вышла в третьем издании.

Обе книги посвящены инфраструктурам, ориентированным на ИТ, и могут служить надежным фундаментом для создания специализированных корпоративных систем.

В информационных системах эти инфраструктуры воплощаются в инструменты ИТ. Таким образом, информационные системы становятся реальным мостиком между бизнес-приложениями и информационными технологиями.

Концепция, изложенная в книге «Architecture of Integrated Information Systems – ARIS» («Архитектура интегрированных информационных систем – ARIS»), была разработана для комплексного описания информационных систем. Первое издание этой книги выпущено в 1992 году. Второе издание выходит в двух книгах: ARIS – Business Process Frameworks (Бизнес-процессы. Основные понятия. Теория. Методы) и ARIS – Business Process Modeling (Моделирование бизнес-процессов).

В книге «Business Process Engineering – Reference Models for Industrial Enterprises» («Инжиниринг бизнес-процессов – модели-прототипы для промышленных предприятий»), второе издание которой опубликовано в 1994 году, предлагается интегрированная информационная система для промышленных предприятий на базе концепции ARIS, где используются модели функций, данных, организации и процессов.

Коммерческая ценность описания информационных систем снижается по мере развития и совершенствования технических средств реализации. Одновременно ослабевает и стабильность концепций, поскольку стремительное развитие ИТ, как правило, не может не сказываться на технической реализации информационных систем. При подготовке книг к печати автор учитывал это обстоятельство в соответствии с «весовым коэффициентом» каждого аспекта. На рис. I этот «весовой коэффициент» представлен треугольником.

Все книги автора изданы и на немецком языке. Книга «Инжиниринг бизнес-процессов» вышла в свет на китайском, а «CIM» переведена на португальский. В настоящее время готовятся переводы на другие языки.

Содержание

A. ARIS – моделирование бизнес-процессов

В Архитектуре интегрированных информационных систем (ARIS) можно выделить четыре аспекта:

- концепцию ARIS («здание» ARIS), представляющую собой архитектуру для описания бизнес-процессов;
- концепцию ARIS, предлагающую методы моделирования, метаструктуры которых представлены в информационных моделях;
- концепцию ARIS как фундамент прикладной системы ARIS Toolset, разработанной фирмой IDS Scheer AG для поддержки процесса моделирования;
- ARIS — «архитектуру» бизнес-инжиниринга (АБИ), представляющую концепцию комплексного автоматизированного управления бизнес-процессами.

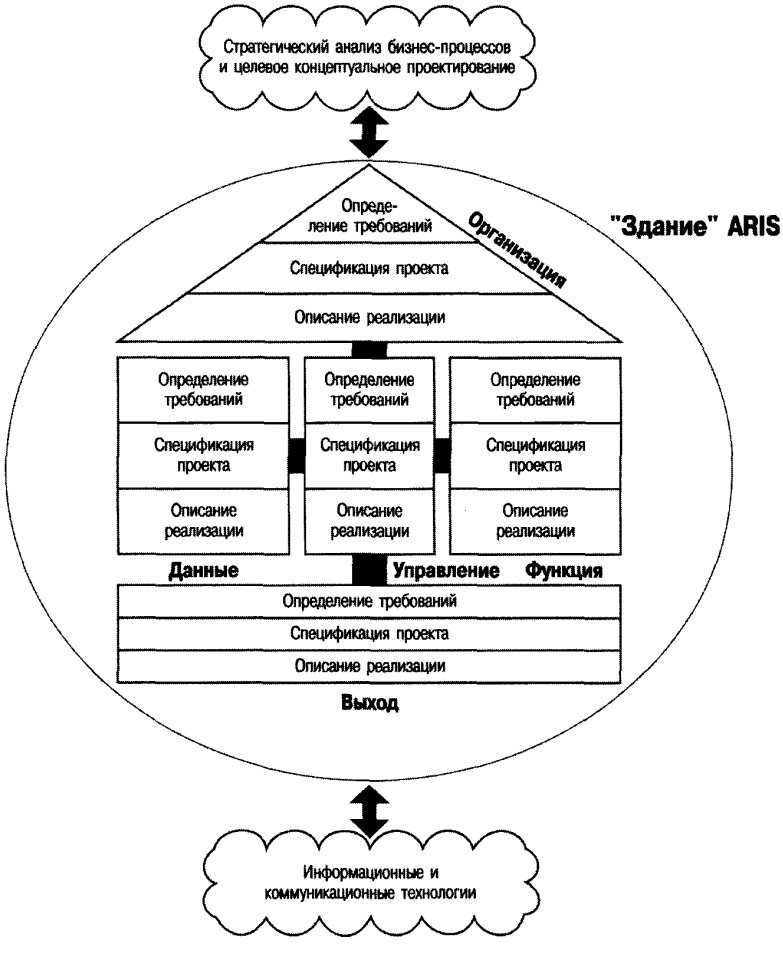


Рис. 1а. Здание ARIS *(Scheer. ARIS — Business Process Frameworks. 1998, рис. 17)*

Концепция ARIS рассматривалась ранее в нашей книге ARIS — Business Process Frameworks[1], где мы показали, как можно упростить описание бизнес-процессов, введя различные уровни представления и фазы модели жизненного цикла. Эта концепция, известная как здание ARIS, проиллюстрирована на рис. 1а.

Методы моделирования классифицируются с помощью различных моделей и уровней здания ARIS, которые являются главной темой этой книги. Их метаструктуры подробно описываются в рамках информационной модели ARIS.

В 1992 г. фирма IDS Prof. Scheer GmbH (Саарбрюккен, Германия) разработала на базе концепции ARIS пакет ARIS Toolset, предназначенный для создания и сопровождения моделей программного обеспечения. (На рис. 1б показан пользовательский интерфейс ARIS-Easy Design.) Методология ARIS облегчает разработку, последовательно направляя действия пользователя.

Пакет ARIS Toolset, завоевавший огромную популярность у широкого круга экспертов, занимающихся сопоставлением различных методов моделирования, является мировым лидером в области программных средств моделирования и анализа бизнес-процессов. Как это ни странно, но иногда упускают из виду два первых аспекта, упомянутые в начале данной книги (архитектура и информационная модель), а ARIS зачастую приравнивают к понятию «инструмент». Между тем мы всецело поддерживаем точку зрения Баха, Брехта и других специалистов. *(Bach, Brecht, Hess, Österle. Enabling Systematic Business Change. 1996, с. 28)*, которые, об-

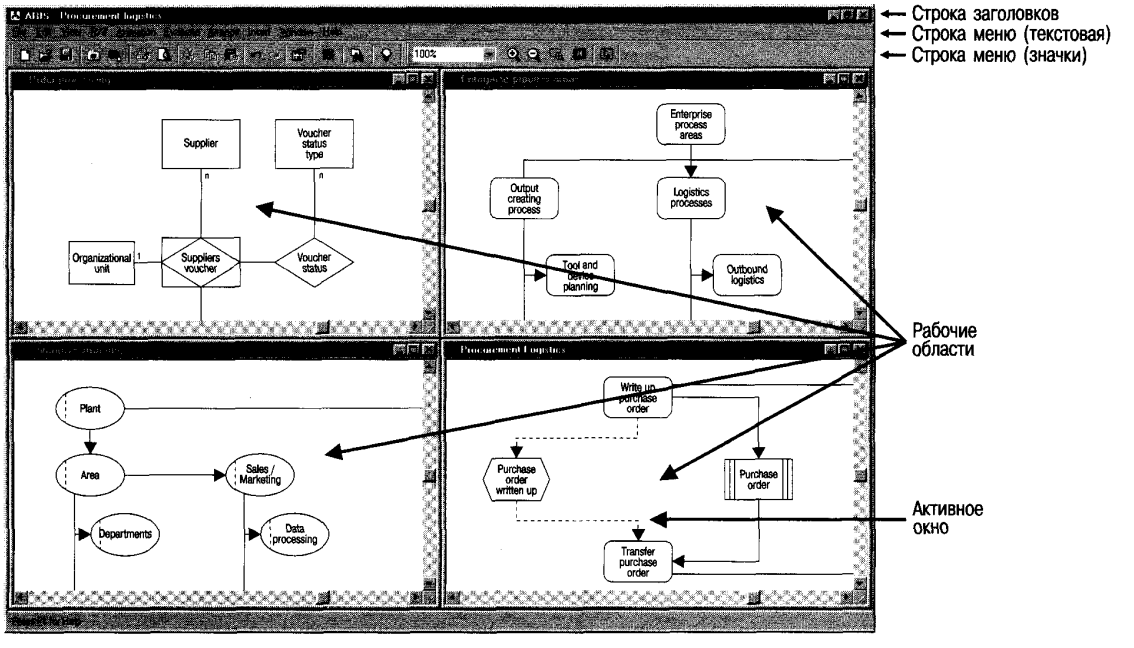


Рис. 1б. Пользовательский интерфейс ARIS Easy Design

[1] *В русском издании книга А.-В. Шеера вышла под названием «Бизнес-процессы. Основные понятия. Теория. Методы». (М., Вестъ-МетаТехнология, 1999). - Прим. ред.*

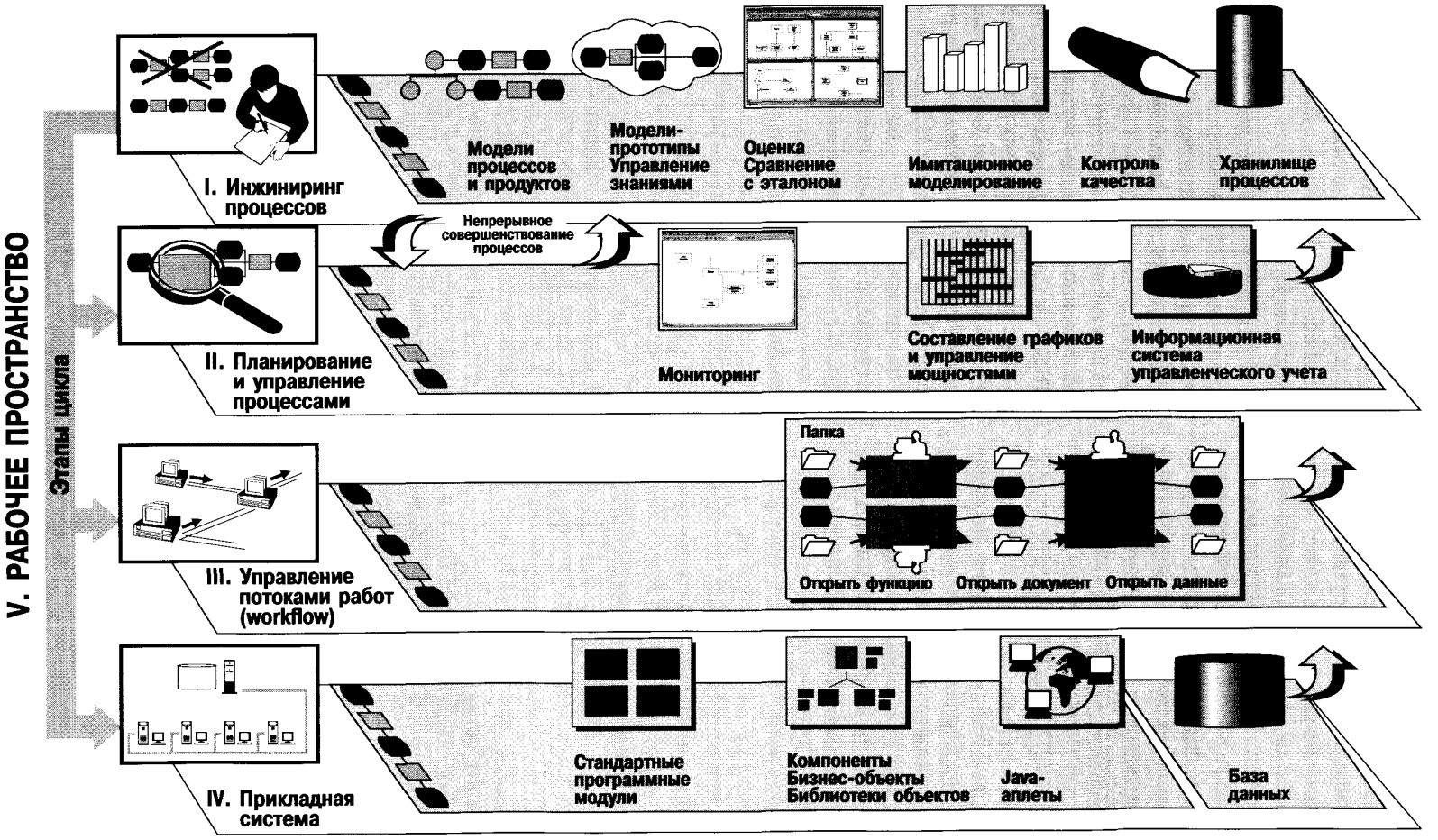


Рис. 1в. Управление процессами на базе концепции ARIS. *(Scheer, ARIS — Business Process Frameworks 1998, рис. 24)*

суждая последовательность этапов проектов, связанных с моделированием, утверждают, что выбору инструментов всегда должен предшествовать выбор методов («Прежде — методы!»). Собственно говоря, мы готовы пойти еще дальше и взять на себя смелость утверждать, что при оценке пригодности и полноты методов необходимо сначала рассмотреть архитектуру и лишь потом выбрать методы. Таким образом, тезис «Прежде — методы!» следовало бы уточнить и дополнить: «Прежде всего — архитектура!».

Концепция ARIS АБИ (см. рис. 1в) отражает систему обратных связей: от создания бизнес-процессов, через стадию планирования и управления, - к реализации с помощью систем автоматизации потоков работ (workflow) и функциональных компонентов. В частности, связывание уровней I и II позволяет проводить непрерывное совершенствование процессов. Более подробно концепция АБИ наряду с другими вопросами рассматривается в работе: Scheer. ARIS — Business Process Frameworks. 1998.

В первых разделах этой книги обсуждаются методы моделирования стратегических бизнес-процессов. Далее рассматриваются различные модели (типы представлений) ARIS. Описание каждой модели построено в соответствии с последовательными фазами жизненного цикла ARIS — от определения требований до описания реализации. Сначала мы подробно изложим методы моделирования на уровне описания требований, а затем покажем, как концепция HOBE используется на уровнях II–IV, выведенных при конфигурации системы, базирующейся на моделях, которые построены на уровне определения требований. Это показано стрелками на рис. 2.

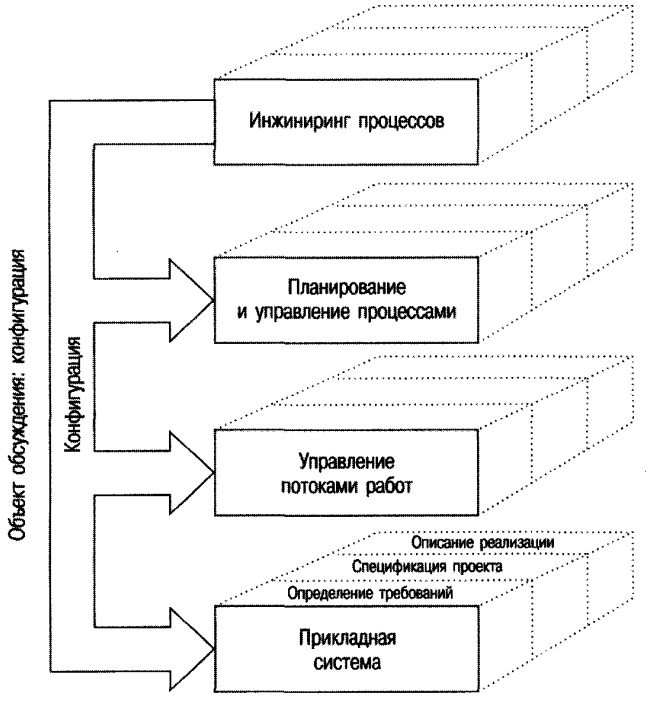


Рис. 2. Структура книги, отражающая концепцию ARIS

Таким образом, программное обеспечение на уровнях II–IV можно охарактеризовать как оболочку. Для связи с конкретными приложениями эту оболочку следует наполнить содержанием, представленным в моделях бизнес-процессов на уровне I.

Например, информационную систему планирования мощностей на уровне II, прежде чем ее можно будет использовать в качестве системы управления для больничной операционной, необходимо сначала привязать к модели процессов, выполняемых в больнице. Точно так же, прежде чем информационную систему планирования мощностей можно будет применить к управлению логистикой, ее необходимо привязать к модели процессов, связанных с парком грузовых автомобилей. Таким образом, обобщенное понятие «ресурс», фигурирующее в системе планирования мощностей, на уровне I модели бизнес-процессов наполняется конкретным содержанием, в данном случае «хирург, прибор для искусственного дыхания и койка» или «экспедитор, грузовик и склад».

Аналогично структуры процессов используются для конфигурирования систем workflow на уровне III и прикладных систем на уровне IV. Отсюда с очевидностью вытекает необходимость в соответствующих интерфейсах для этих систем. Современные стандартные приложения типа SAP R/3 или BAAN IV имеют собственные инструменты конфигурирования и настройки, а системы workflow — собственные языки конфигурирования, например FDL (язык описания потоков) в системе IBM FlowMark.

Поскольку конфигурирование осуществляется на уровне модели бизнеса, мы рассматриваем его как расширенный вариант описания на уровне определения требований. О конфигурациях мы поговорим после того, как рассмотрим определение требований.

Затем мы коснемся отображения моделей бизнеса в объекты на уровнях спецификации проекта и описания реализации.

При реализации информационных систем на всех четырех уровнях АБИ ARIS переход от определения требований к описанию реализации обычно связан с программным обеспечением на каждом уровне. Эта процедура не зависит от типа реализуемых систем. На рис. 2 показан процесс реализации на уровне IV, представленный в виде следующих одна за другой фаз (определение требований, спецификация проекта и описание реализации). То же самое относится к программному обеспечению на других уровнях. Мы лишь кратко обсудим вопросы реализации в соответствии с нашим пониманием *бизнес-информатики.*

Для описания каждой метамодели используются диаграммы классов с обозначениями, принятыми в унифицированном языке моделирования *(UML, см. UML Notation Guide. 1997).* Этот тип представления аналогичен модели Чена сущность–отношение *(ERM, см. Chen. Entity Relationship Model 1976),* которая рассматривалась в предыдущих изданиях этой книги.

На рис. 3 объекты моделирования на языке UML, приведенные в настоящем издании, сравниваются с объектами ERM. Обозначения UML представлены в разделе А.3.2.1.1.1.

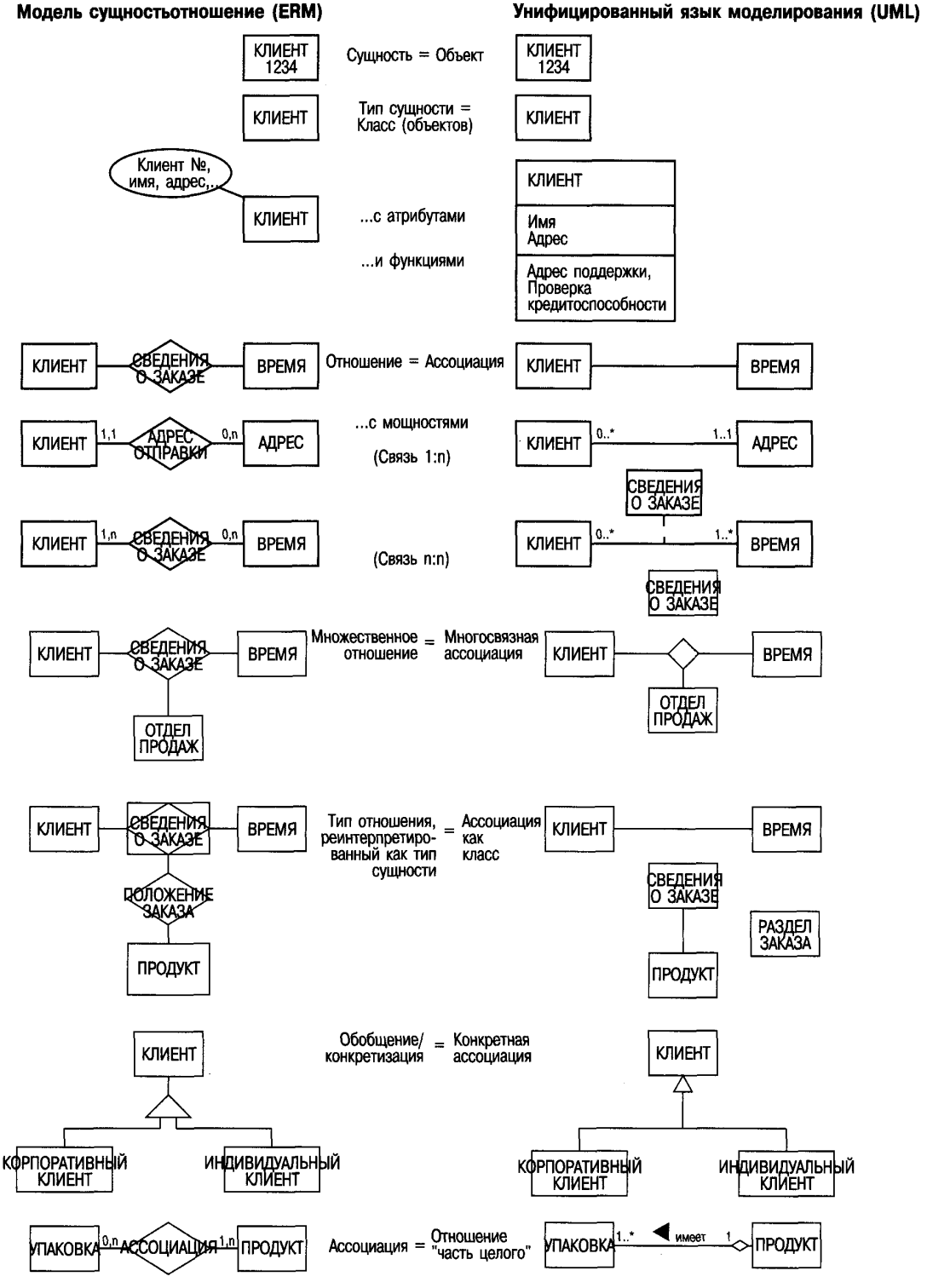


Рис. 3. Сравнение диаграмм ERM с диаграммами классов UML

A.1. Стратегический анализ бизнес–процессов

Поскольку представление организации в виде совокупности бизнес-процессов является новой парадигмой в корпоративном инжиниринге, прежде чем перейти к описанию здания ARIS (см. рис. 1а), рассмотрим стратегический анализ бизнес-процессов. Если в основе тейлоризма лежал принцип структурирования функций в соответствии с объемом необходимых усилий, что, в свою очередь, вело к узкой специализации работников, то при организации бизнес-процессов объектом внимания является синергический эффект, т.е. повышение эффективности процесса за счет взаимовлияния участвующих в нем функций. Идея заключается в том, чтобы обеспечить стабильность процесса в результате достижения высочайшего уровня квалификации персонала.

Эти основополагающие изменения в принципах корпоративной организации носят стратегический характер, оказывая влияние на конкурентоспособность предприятия. С наибольшей очевидностью это проявляется в философии «подчинения структуры процессу, а процесса — стратегии» *(Osterloh, Frost. Prozeßmanagement. 1996, с. 3)*. Иными словами, при инжиниринге основных процессов используются концепции стратегического планирования. Таким образом, стратегический анализ бизнес-процессов и определение стратегической концептуальной схемы позволяют идентифицировать ключевые цели и секторы бизнеса, предварительно наметить новые бизнес-процессы (которые предстоит разработать) и даже выявить слабые места.

Кроме того, стратегический анализ бизнес-процессов дает возможность определить, какие новые информационные технологии целесообрзно внедрять. Стратегический анализ как активизирующее начало играет основополагающую роль и в организации бизнес-процессов.

A.1.1. Моделирование стратегических бизнес–процессов

Организационная модель не является самоцелью. Она должна отражать требования, выполнение которых необходимо для обеспечения эффективности, в частности эффективности ресурсов, эффективности процессов и рыночной эффективности *(Frese. Grundlagen der Organisation. 1995, с. 26)*. В рыночном отношении предприятие считается эффективным, если оно в полной мере реализует потенциальные возможности рынка, а различные организационные единицы, взаимодействующие с клиентом, не являются зависимыми друг от друга, что уменьшает потребность в координации. Под эффективностью ресурсов предприятия понимается их эффективное использование. Особенно это относится к таким потенциальным факторам, как человеческие и производственные ресурсы. Эффективность процессов предполагает фокусирование их на корпоративных целях.

Параметры эффективности, как правило, вступают в противоречие друг с другом. Гутенберг, например, говорит о дилемме, возникающей при одновременном стремлении к минимальной продолжительности производственного цикла, т.е. к эффективности процесса, и максимальному использованию мощностей, т.е. к эффективности ресурсов *(Gutenberg. Die Produkton. 1983, с. 216)*.

На рис. 4а представлена схема предприятия, ориентированного на функции. Речь идет о предприятии, где для выполнения каждой функции создана соответствующая организационная единица. Теоретически такая организационная структура должна способствовать повышению эффективности ресурсов. Однако из-за необходимости координации функций бизнес-процессы, направленные на создание продукта/услуги (П1–П4), на-

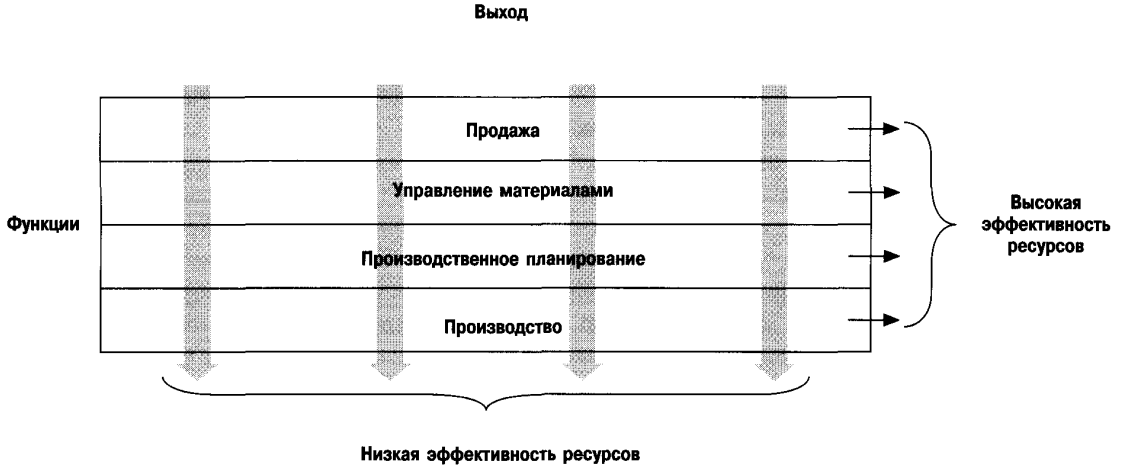


Рис. 4а. Предприятие, ориентированное на функции

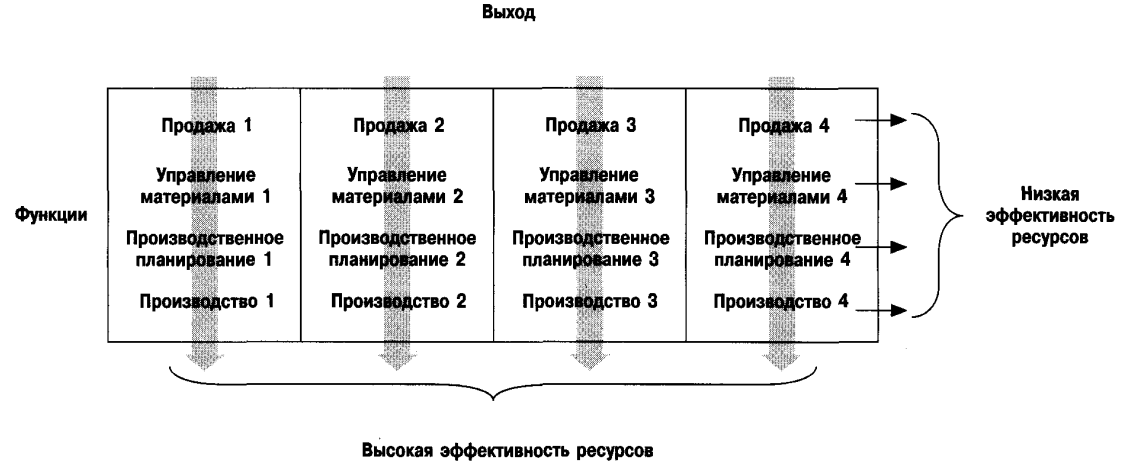


Рис. 4б. Предприятие, ориентированное на процессы

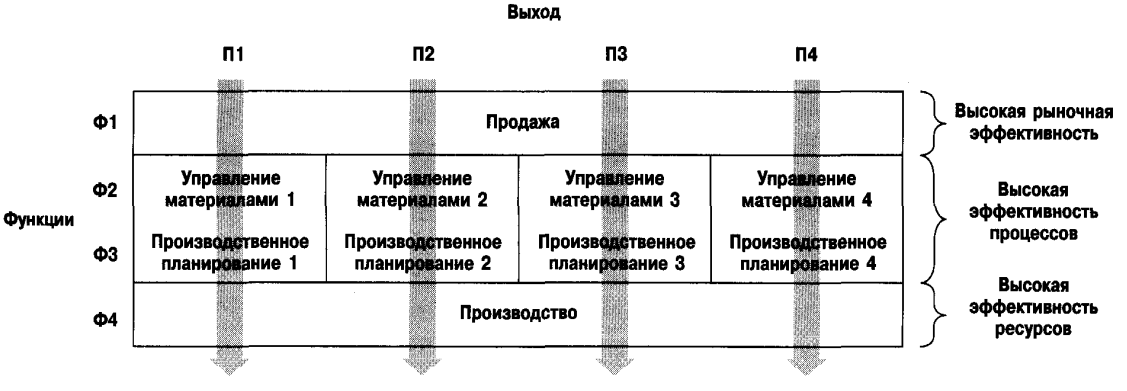


Рис. 4в. Смешанное предприятие

пример обработка заказов, оказываются малоэффективными.

На рис. 4б представлена организация, ориентированная исключительно на процессы. Иными словами, организационные единицы группируются по процессам. Здесь эффективность процессов доминирует в ущерб эффективности ресурсов.

На рис. 4в приведена схема смешанного предприятия, стремящегося удовлетворить всем трем критериям с целью обеспечить рыночную эффективность за счет централизованной организации продаж («одно лицо для клиента»). Здесь процессы материально-технического обеспечения по разным группам продуктов являются кросс-функциональными и, следовательно, эффективными. При этом в производстве, использующем ресурсы, учитываются все потенциальные факторы эффективности.

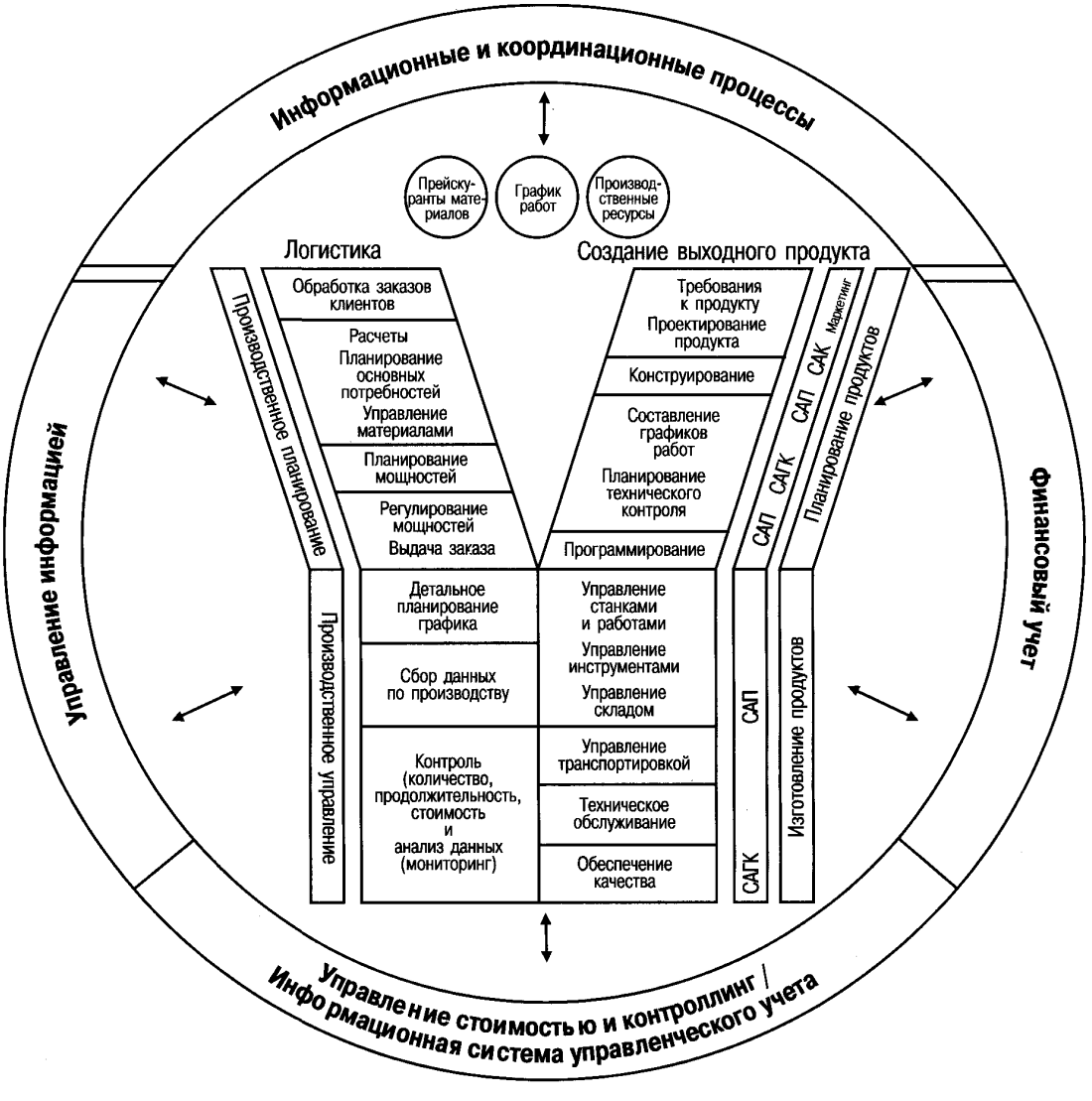


Рис. 5. Основные и вспомогательные процессы

Несмотря на то, что параметры эффективности иногда вступают в конфликт друг с другом, ориентация на процессы мощно вторгается на сегодняшние предприятия. Одна из причин этого заключается в том, что эффективность корпоративных процессов (скажем, в области материально-технического обеспечения или разработки продуктов) представляет несомненную ценность для клиентов, а также в том, что руководство получает возможность управлять корпоративными процессами, изменяя масштаб задачи, например, сокращая процессы путем исключения того или иного фактора.

Предприятия пытаются достичь рыночной эффективности, делая акцент на конкурентоспособности (и, таким образом, на рыночных критериях) при создании стратегически важных (основных) процессов. Эффективность ресурсов достигается за счет предельной гибкости потенциальных факторов, например, за счет способности легко перераспределять функции и включать их в другие процессы при нехватке корпоративных мощностей.

Основные процессы влияют на конкурентоспособность предприятия, они являются кросс-функциональными и взаимодействуют как с клиентами, так и с поставщиками. Выделяют две ключевые категории основных процессов, вокруг которых группируются информационные и координационные процессы — логистика (материально-техническое обеспечение) заказов и разработка нового изделия (см. рис. 5). Эта классификация изначально носит общий характер и допускает дальнейшую детализацию *(Osterloh, Frost. Prozeßmanagement. 1996, с. 50)*. Основные процессы можно дифференцировать по степени сложности (например, срочные, приоритетные и обычные заказы) или по группам клиентов (индивидуальные и корпоративные).

Для анализа конкурентоспособности можно использовать концепции стратегического планирования *(Krcmar. Informationsmanagement. 1997, с. 203)*.

Интересна концепция Рокарта, который использует понятие критических факторов успеха *(Rockart. Critical Success Factors. 1982)*. Критические факторы успеха — это цели, достижение которых так же необходимо для успешной деятельности предприятия, как и высокое качество, оперативность доставки, конкурентное превосходство в области научных исследований и опытно-конструкторских разработок (НИОКР), высокая гибкость. Именно эти цели должны стоять в центре внимания при разработке бизнес-процессов.

Предложенная Портером цепочка добавленного качества (см. рис. 6) представляет собой инфраструктуру, показывающую значимость функций. Первичными являются функции, предназначенные непосредственно для создания или использования результатов деятельности предприятия (корпоративного выхода). Вторичные функции играют вспомогательную роль, обеспечивая необходимую инфраструктуру и средства управления при выполнении первичных функций. Определив, каким образом первичные функции взаимодействуют с процессами добавления качества, можно разработать предварительную структуру процессов, которая, в свою очередь, будет служить первой ступенью к организации процессов.

На рис. 7 приведен пример реального процесса с указанием соответствующих подпроцессов и их связей. В описании на верхнем уровне определяется каждый тип выходного продукта для процессов, связанных с «узлами», и процессов, связанных с готовыми грузовиками. Для бизнес-процессов, связанных с обслуживанием клиентов, типы выходных продуктов идентичны, в то время как бизнес-процессы обработки заказов делятся на два варианта — «узлы» и «автомобили».

На рис. 8 приведена метамодель этих понятий.

Рис. 6. Цепочка добавленного качества, предложенная Портером *(Porter. Wettbewerbsvorteile.1992, с. 62)*

Класс ПАРАМЕТР ЭФФЕКТИВНОСТИ обрзуется из параметров эффективности организационных подразделений (по категориям «процессы», «ресурсы» и «рынок»). Иерархические сегменты, представленные на рис. 4а-в, составляют класс ОРГАНИЗАЦИОННЫЙ СЕГМЕНТ. Этот класс делится на два подкласса: ФУНКЦИОНАЛЬНЫЙ СЕГМЕНТ — для предприятий с горизонтальной структурой и СЕГМЕНТ БИЗНЕС-ПРОЦЕССА — для предприятий с вертикальной структурой. Таким образом, на рис. 4в каждый организационный сегмент представляет отдельную область. Вклад организационных сегментов в параметры эффективности описывается реляционным классом ВКЛАД В ЭФФЕКТИВНОСТЬ.

Вклад можно представить также в виде таблиц с помощью словесного описания или (если это возможно) количественных показателей. Рис. 9 иллюстрирует оценочную таблицу для примера, приведенного на рис. 4в.

Фокус деятельности организаций, ориентированных на функции или процессы, становится очевидным из числа экземпляров связей между ОРГАНИЗАЦИОННЫМ СЕГМЕНТОМ и классами ФУНКЦИЯ, ДОБАВЛЯЮЩАЯ КАЧЕСТВО, и ОБЛАСТЬ ВЫХОДА.

Термин «функция, добавляющая качество», означает, что мы имеем дело с мощными функциями (или, иначе, — подчиненными процессами), представленными на рис. 6.

Типы выходных продуктов характеризуют важные группы продуктов и услуг на предприятии (см. рис. 10) и могут быть связаны друг с другом.

Организации, ориентированные на функции, характеризуются лишь несколькими связями (ассоциациями) между ФУНКЦИОНАЛЬНЫМ СЕГМЕНТОМ и ФУНКЦИЕЙ, ДОБАВЛЯЮЩЕЙ КАЧЕСТВО, (в пределе их число сводится к 1) и множеством связей между ФУНКЦИОНАЛЬНЫМ СЕГМЕНТОМ и ТИПОМ ВЫХОДНОГО ПРОДУКТА. Организации, ориентированные на процессы, напротив, характеризуются лишь несколькими связями с ФУНКЦИОНАЛЬНЫМ СЕГМЕНТОМ (в пределе их число сводится к 1) и множеством связей с ФУНКЦИЯМИ, ДОБАВЛЯЮЩИМИ КАЧЕСТВО.

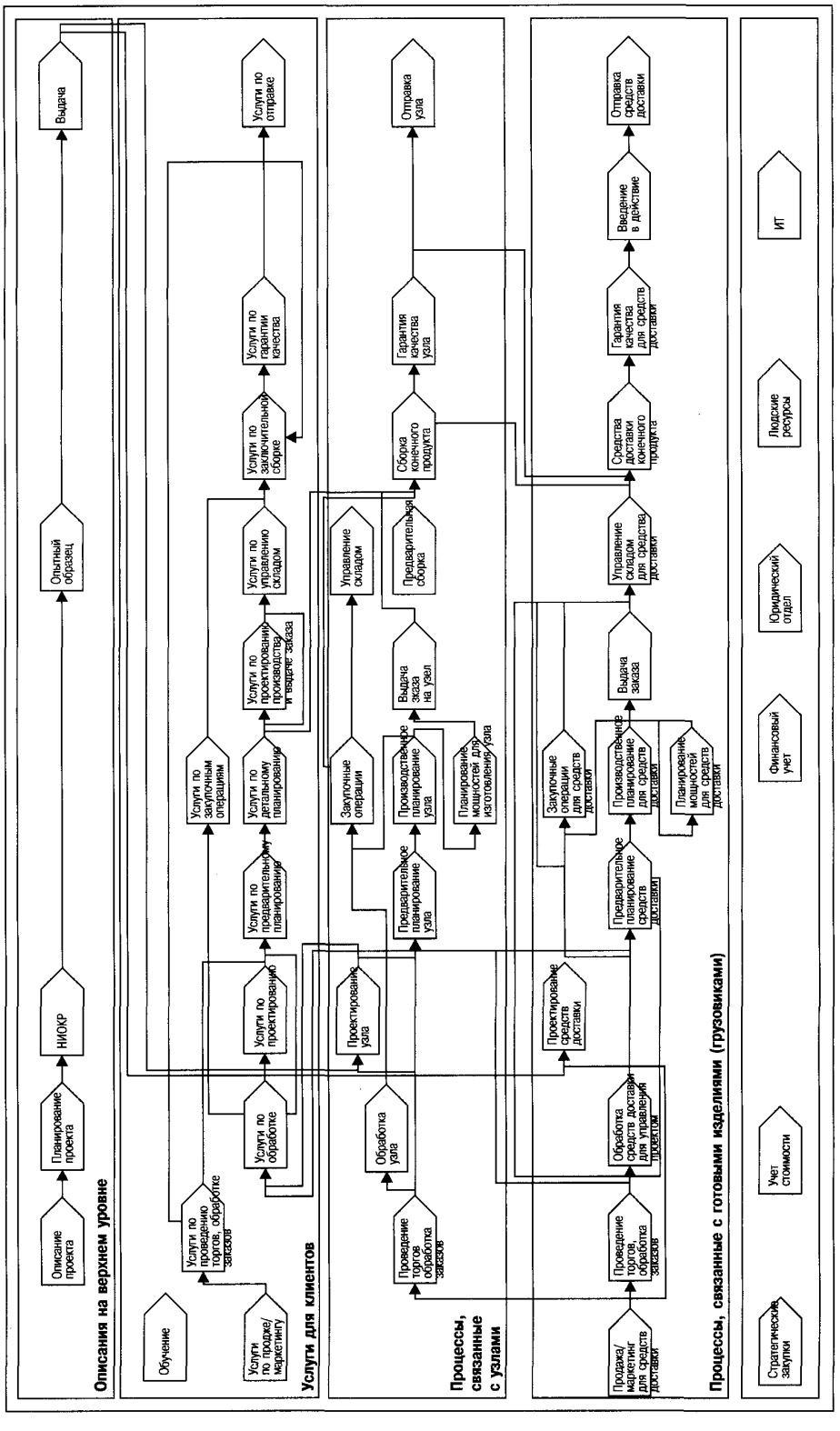


Рис. 7. Описание бизнес-процессов *(Kirchmer. Definition von Geschäftsprozessen. 1995, с. 271)*

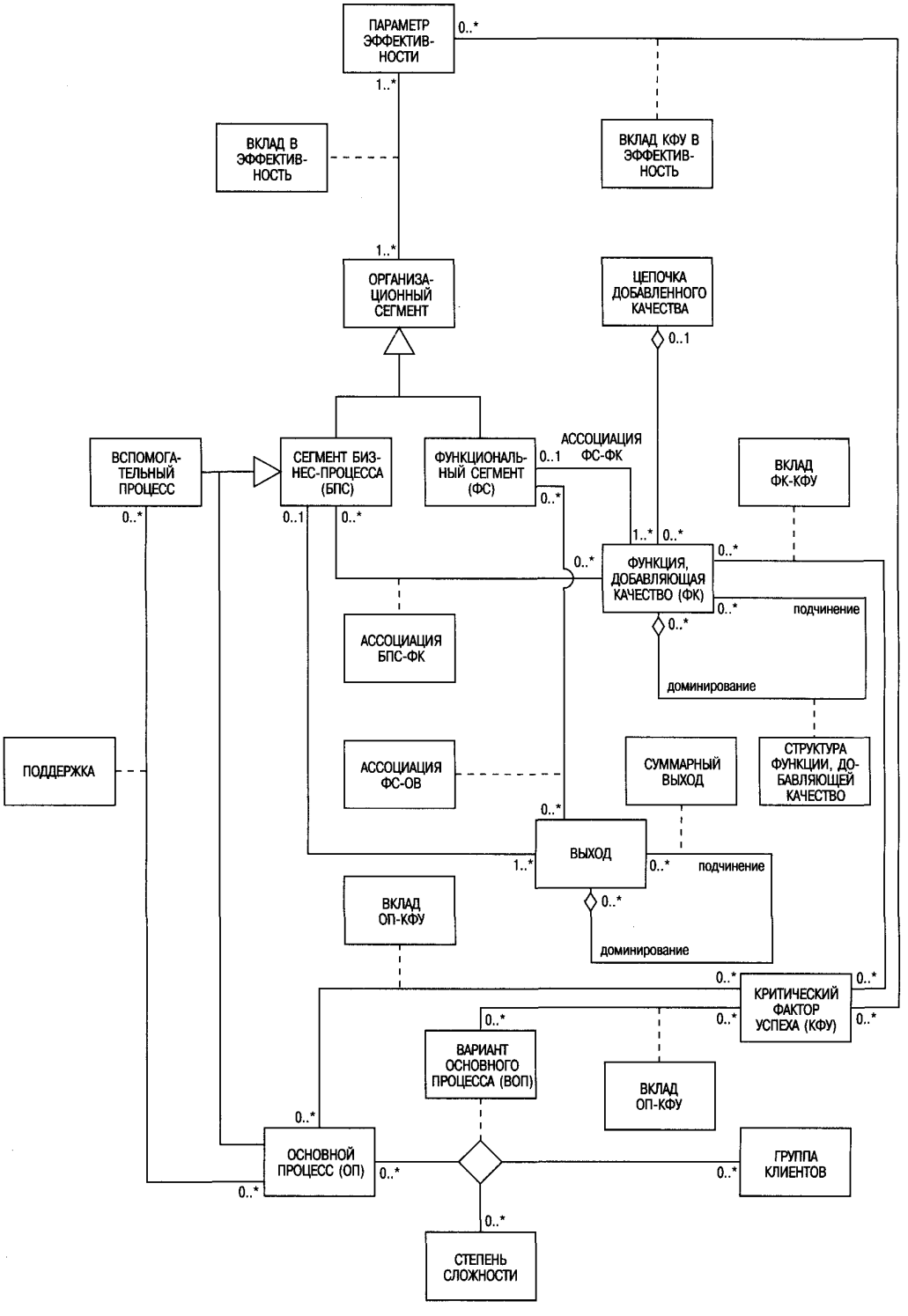


Рис. 8. Метамодель стратегии бизнес-процессов ARIS

Параметр эффективности / Организационный сегмент	Рыночная эффективность	Эффективность процессов	Эффективность ресурсов
Продажа	Высокая: координация почти не требуется, хорошее использование возможностей рынка благодаря принципу "одно лицо для клиента" лишь вскользь	Средняя: множественность систем материально-технического обеспечения обусловливает необходимость координации	Хорошее использование ресурсов отдела продаж, его сотрудников, рекламных возможностей и т.д.
Логистика	Средняя: необходима координация с централизованным отделом продаж	Высокая в отделе логистики, средняя в смежных подразделениях, т.е. в отделе продаж и на производстве	Средняя: ресурсы распределяются по отдельным сегментам, возможности корректировки незначительны
Производство	Средняя: производство сфокусировано на всех продуктах, индивидуальное изготовление сопряжено с громоздкой процедурой	Средняя: разнородные заказы создают друг другу помехи	Высокая: все производственные ресурсы используются совместно

Рис. 9. Вклад организационных сегментов в эффективность

Связь между ТИПОМ ВЫХОДНОГО ПРОДУКТА и СЕГМЕНТОМ БИЗНЕС-ПРОЦЕССА показывает, какие бизнес-процессы необходимы для различных типов выходных продуктов. Это требуется, например, для того, чтобы решить, следует ли привлекать ресурсы со стороны. Когда те или иные типы выходных продуктов перестают существовать, отпадает и необходимость в некоторых процессах. В то же время возникают новые процессы, связанные с привлечением ресурсов извне. Взаимоотношения между типами выходных продуктов выражаются связью КОРРЕЛЯЦИЯ ТИПОВ ВЫХОДНЫХ ПРОДУКТОВ. Таким образом, в метамодели фиксируется структура типа выходного продукта, представленная на рис. 10.

В некоторых бизнес-процессах функции, добавляющие качество, характеризуются отношениями типа «предшествующая-последующая». Такие отношения описываются связями СТРУКТУРА ФУНКЦИИ, ДОБАВЛЯЮЩЕЙ КАЧЕСТВО, и образуют фундамент для стратегических моделей бизнес-процессов. Это отражено в метаструктуре, приведенной на рис. 7.

В соответствии с предложенной Портером моделью добавления качества бизнес-процессы можно разделить на два типа: ОСНОВНОЙ ПРОЦЕСС и ВСПОМОГАТЕЛЬНЫЙ ПРОЦЕСС.

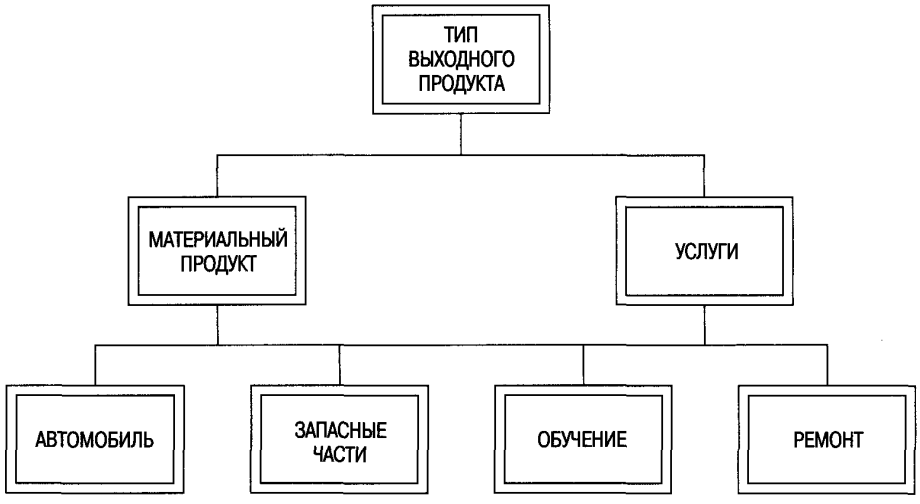


Рис. 10. Области выхода

Критические факторы успеха / Варианты процесса обработки заказов клиентов	Гарантия качества продукта	Оперативность доставки	Эффективность затрат
Обработка заказов клиентов Стандартные изделия	Высокая	Высокая	Очень высокая
Обработка заказов клиентов Специальные изделия	Очень высокая	Очень высокая	Высокая
Обработка заказов клиентов Заказы на запчасти	Чрезвычайно высокая	Чрезвычайно высокая	Средняя

Рис. 11. Оценка критических факторов успеха для различных вариантов процесса

Критические факторы успеха (КФУ) можно связывать с основными процессами в целом или с отдельными функциями, добавляющими качество, фиксируя таким образом их ВКЛАД В КФУ.

Варианты основных процессов, классифицируемые по группам клиентов или по степени сложности, создают связи между сущностями ОСНОВНОЙ ПРОЦЕСС, ГРУППА КЛИЕНТОВ и СТЕПЕНЬ СЛОЖНОСТИ. Различные оценки критических факторов успеха можно также относить к ассоциативному классу ВАРИАНТ ОСНОВНОГО ПРОЦЕССА, что позволит в дальней-

шем определить цели, относящиеся к совершенствованию процесса.

Зависимость процессов от критических факторов успеха можно описать с помощью таблиц. На рис. 11 приведен пример оценки трех вариантов процесса обработки заказов клиентов. Связав критические факторы успеха с параметрами эффективности в процессе создания организационной структуры, можно показать значения параметров эффективности для деятельности предприятия.

Выбранные варианты бизнес-процессов разбиваются на отдельные операционные этапы с помощью диаграмм процессов (PCD). Эти описания соответствуют стратегическому уровню, хотя они и используются на уровне описания требований.

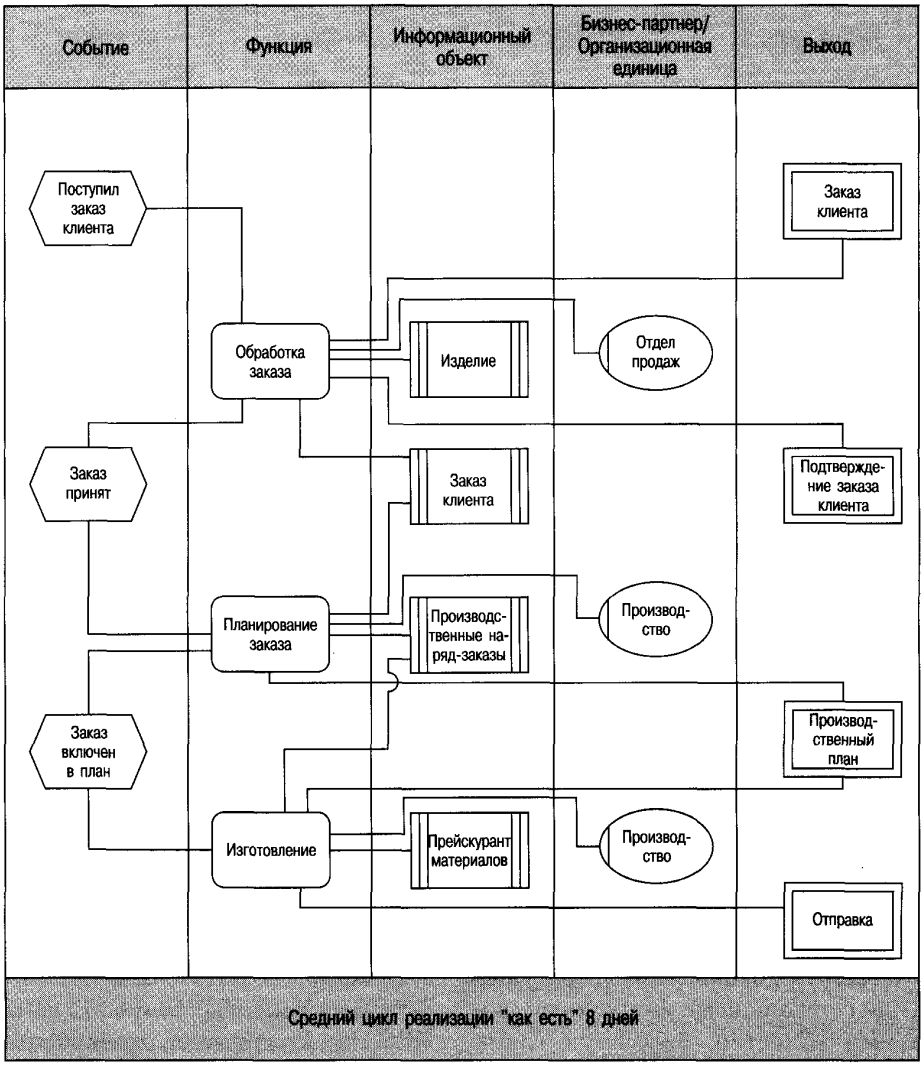


Рис. 12а. Диаграмма стратегического процесса «как есть»

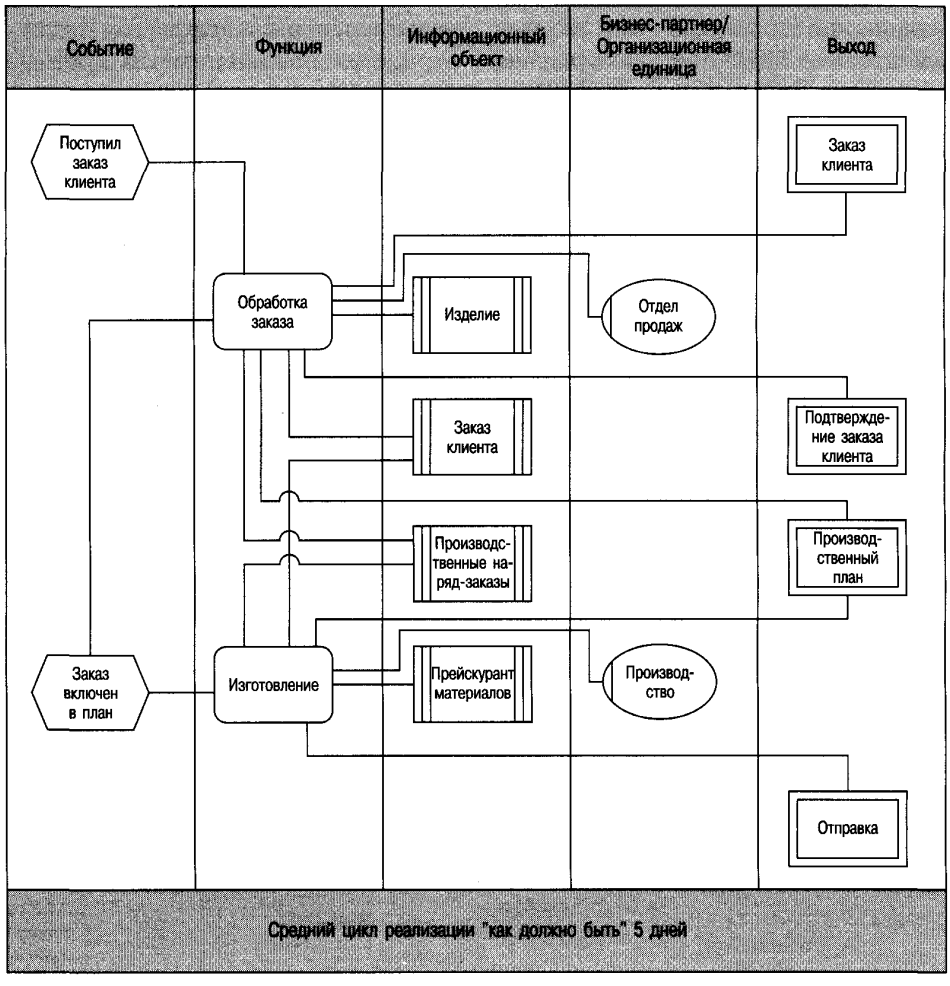


Рис. 12б. Диаграмма стратегического процесса «как должно быть»

Представления процессов детализируются лишь тогда, когда нужно указать ключевые стратегические недостатки текущего процесса и преимущества будущего. На диаграммах цепочек процессов бизнес–процессы представлены в виде последовательности «черных ящиков» (простые версии диаграмм процессов описаны в работе: *Scheer. Principles of Efficient Information Management. 1991, с. 22).*

На рис. 12а и 12б представлены предварительные процедуры для данных «как есть» и «как должно быть» применительно к бизнес-процессу обработки заказов. Диаграммы процессов отражают сокращенные версии ключевых взаимосвязей ARIS-модели бизнес-процессов. В отличие от произвольных схем таблицы делают событийные диаграммы процессов (EPC) более удобочитаемыми (см. раздел A.3.2.1.2), но в них трудно отобразить сложные процедурные структуры, например, замкнутые контуры.

Первые две колонки на рис. 12а и 12б представляют первичный поток управления вкупе с информационными объектами, характеризующими поток данных. Внешние бизнес-партнеры и внутренние организационные единицы, участвующие в выполнении функций, перечислены в колонке 4. Поток выходов показан в колонке 5.

При сравнении процессов «как есть» и «как должно быть» последний выигрывает уже за счет того, что отделы продаж напрямую сравнивают новые клиентские заказы с существующим производственным планом. Исключение лишних перемещений данных в рамках одного подразделения позволяет сэкономить несколько дней, устранив необходимость дожиожидания завершения планирования заказа в производственном секторе. Благодаря такой стратегии сокращается и среднее время цикла.

Уже на этом этапе в диаграммы цепочек процессов можно включить общие понятия ИТ. Это позволяет идентифицировать серьезные отклонения в системе или подчеркнуть цели стратегического управления информацией в целевых концепциях.

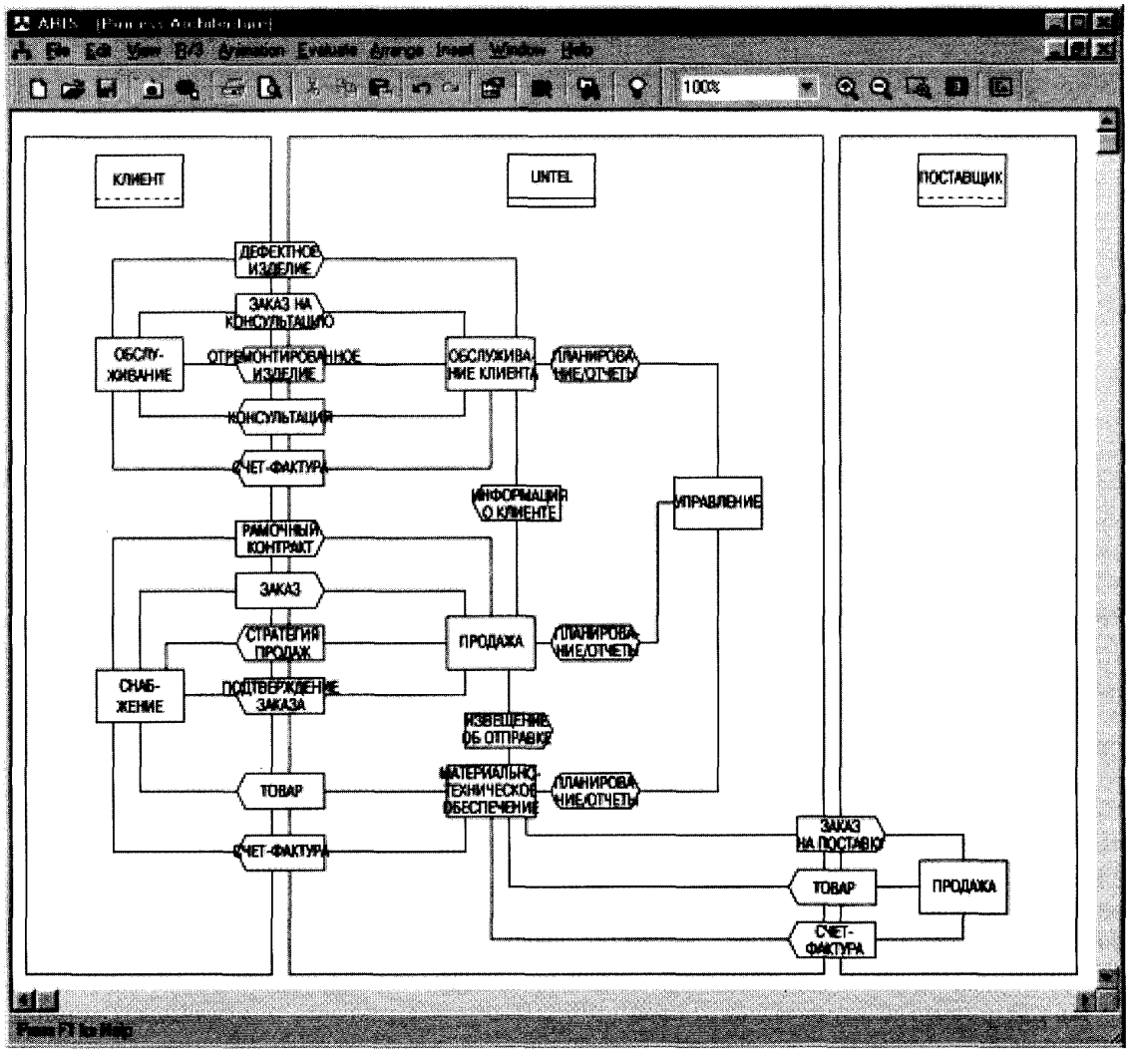


Рис. 13. Последовательность выполнения процесса в рамках метода PROMET
(Bach, Brecht, Hess, Österle. Enabling Systematic Business Change. 1996, c. 271)

При описании требований и построении метаструктуры вновь используются диаграммы цепочек процессов.

A.1.2. PROMET

Метод PROMET разработан в Санкт-Галленском университете, Швейцария *(Information Management Gesellschaft. PROMET. 1994; Österle. Business Engineering 1. 1995; Österle, Vogler. Praxis des Workflow–Managements. 1996)*. В основе метода лежат различные сетевые диаграммы и стандартные матрицы для описания отношений и весов, позволяющие проектировать бизнес-процессы, опираясь на стратегическое корпоративное планирование, и увязывать их с информационными технологиями. Метод PROMET поддерживается ARIS Toolset.

На первом этапе определяется корпоративная стратегия, т.е. решения относительно альянсов, организационных структур, направлений рзвития бизнеса и инструментов управления. Сетевая диаграмма, соответствующая SWOT-анализу (SWOT — аббревиатура, образованная от слов Strengths — достоинства, Weaknesses — недостатки, Opportunities — возможности, Threats — опасности), иллюстрирует взаимоотношения между конкурирующими силами внутри предприятия. Секторы других сетевых диаграмм описывают различных участников рынка и взаимоотношения между ними.

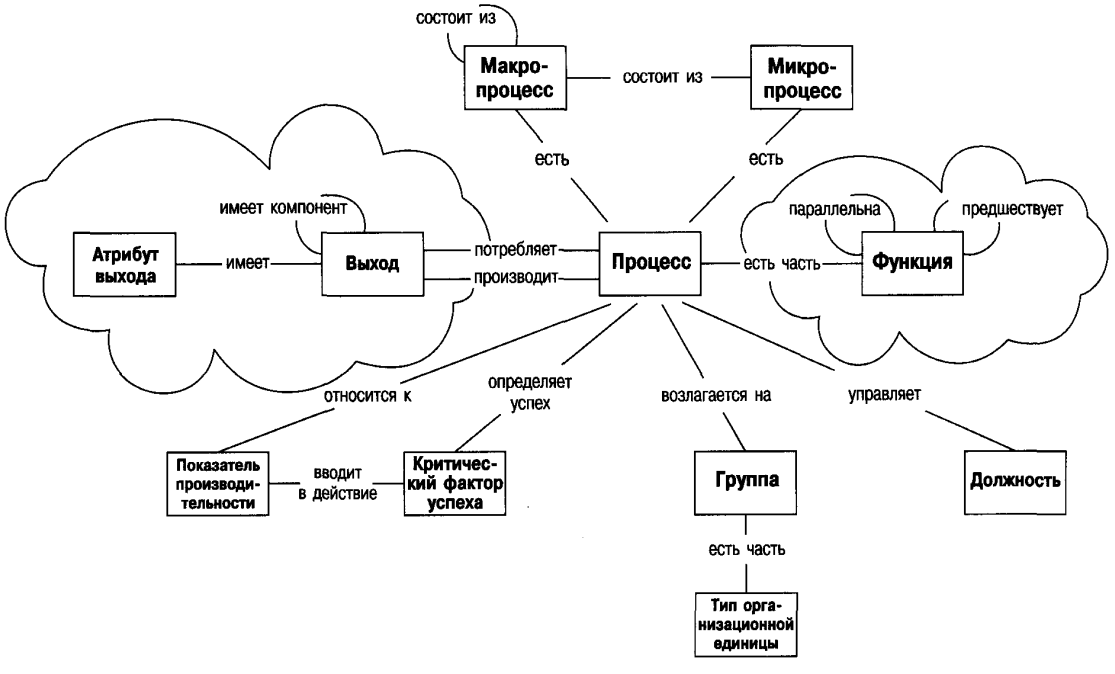


Рис. 14. Метамодель PROMET *(Bach, Brecht, Hess, Österle, Enabling Systematic Business Change. 1996, с. 270)*

Комбинации продукт/рынок привязываются к каналам сбыта в матрице направлений бизнеса.

После определения и описания корпоративной стратегии бизнес–процессы распределяются по направлениям бизнеса. Затем процесс, а также качество его отдельных компонентов сравниваются с аналогичными параметрами конкурентов. Последовательность выполнения операций в рамках процессов предварительно описывается с помощью документированных «цепочек задач», с учетом тех информационных технологий, которые позволят затем программно реализовать эти задачи (см. рис. 13). Приведенная здесь таблица сопоставима с диаграммами процессов в концепции ARIS.

На рис. 14 представлена метамодель PROMET, содержащая некоторые усовершенствования по сравнению с метамоделью на рис. 8. Эти усовершенствования самоочевидны и не нуждаются в пояснениях.

Преимущество метода PROMET заключается в жестком структурировании результатов проекта по реструктуризации (совершенствованию) бизнеса и в последовательном применении стратегических концепций (например, концепции Портера). Эффективно используя мощные возможности ИТ, он также поддерживает процессы и генерирует документацию, с помощью инструментальных средств моделирования. Однако для улучшения процессов необходимо связать PROMET с критическими факторами успеха (целями).

A.1.3. Другие методы стратегического моделирования бизнес-процессов

Существует широкий спектр дополнительных методов оценки, расстановки приоритетов и создания стратегических бизнес-процессов во многих из этих методов используются графические представления или таблицы. Вот некоторые из популярных методов стратегического моделирования:

- анализ Portfolio бизнес-процессов *(см. Mc Farlan, Mc Kenney, Pyburn. Information Archipelago. 1983)*;
- информационная насыщенность областей выхода по Портеру *(см. Porter, Millar. How Information Gives You Competitive Advantage. 1985)*;
- описание Portfolio, разработанное Boston Consulting Group.

A.2. Моделирование на разных уровнях представления ARIS

A.2.1. Моделирование на уровне функционального представления

На рис. 15 показано место «функционального» кирпичика в здании ARIS. Функции часто описываются в контексте их отношений с другими компонентами. Они тесно связаны с данными, поскольку офисные функции описывают процесс преобразования информации, т.е. преобразуют входные данные в выходные. Нередко описание функций привязано к организационным объектам, особенно при описании должностных обязанностей.

В концепции ARIS функции рассматриваются как отдельный уровень представления бизнес-процессов.

A.2.1.1. Определение требований на уровне функциональной модели

Стратегия бизнес-процессов обусловливает те функции, которые предприятие должно эффективно выполнять *(Mertens. Wirtschaftsinformatik. 1995, с. 40)*. Термин «функция» не имеет общего определения и употребляется как синоним процесса, операции или задачи. Названия сложных функций, например, обработка заказа, используются также для обозначения бизнес-процесса. Описание поведения функций, т.е. динамичное управление функциями от начала до конца, также является частью бизнес-процесса. Однако при непосредственном описании функций основной акцент делается на представлении их статичной структуры.

Для изображения функций применяются различные символы. На рис. 16 представлены некоторые общеупотребитель-

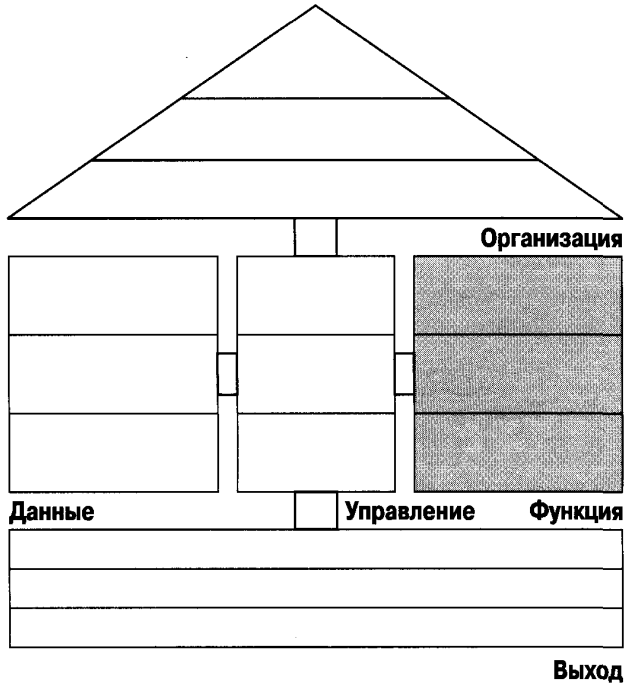


Рис. 15. Классификация функциональной модели в ARIS

ные обозначения. В этой работе мы будем пользоваться скругленными прямоугольниками.

В инжиниринге бизнес-процессов моделирование функций выполняется в соответствии со стратегической концепцией компании, определяющей в том числе и перспективы использования ИТ. На базе этой концепции описываются цели, которые должны быть реализованы при помощи данных функций. Цели можно формулировать, исходя их концепции критических факторов успеха, разработанной Рокартом *(Rockart. Critical Success Factors. 1982)*.

Под функциями понимаются операции, выполняемые с объектами для достижения одной или более целей. Цели могут быть иерархически связаны друг с другом (см. рис. 17), при этом одна подчиненная цель может быть направлена на поддержку нескольких доминирующих (главных) целей. Таким образом, в рамках класса ЦЕЛИ структура взаимосвязанных целей характеризуется ассоциацией *:* (см. рис. 18). Чтобы дифференцировать эти две связи между ЦЕЛЬЮ и СТРУКТУРОЙ ЦЕЛЕЙ, мы присваиваем им ролевые имена. Главные цели верхнего уровня не имеют над собой других доминирующих целей, но при этом одна цель может выступать подчиненной по отношению к нескольким доминирующим, поэтому для связи «доминирование» указывается в виде атрибута (минимальная и максимальная) мощность (0..*). Связь «подчинения» имеет такую же мощность, поскольку подчиненные цели на низшем уровне не имеют никаких дополнительных подчиненных им целей.

Функции могут поддерживать несколько целей. Связь между функциями и целями может наследоваться более высокими уровнями, т.е. отношение, установленное на лежащем ниже уровне, может переходить на лежащие выше. Так, функция «Управление производством», представленная на рис. 17, поддерживает также доминирующую цель «Снижение стоимости».

Рис. 16. Обозначения функций

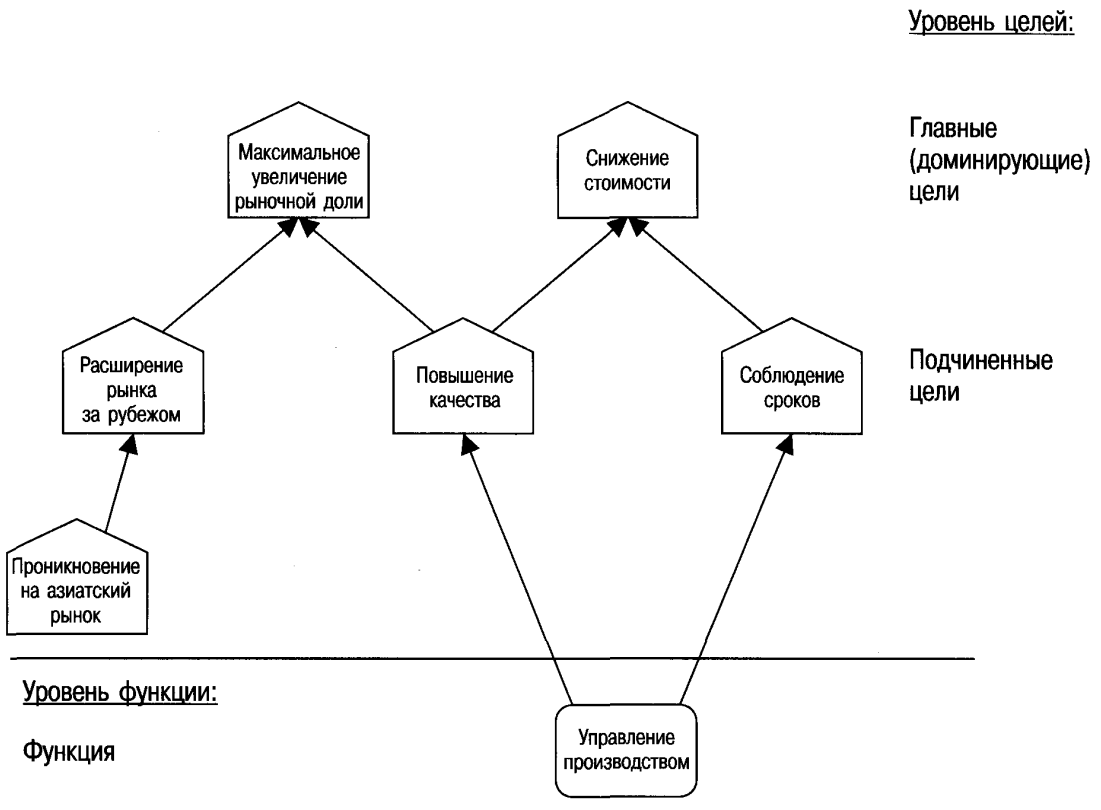


Рис. 17. Структуры целей и функций

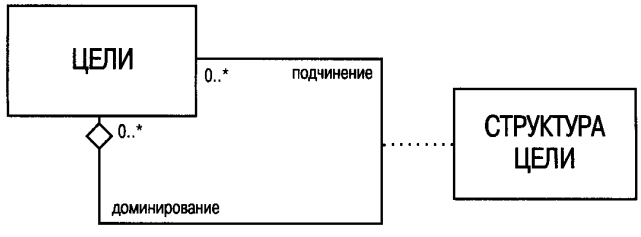


Рис. 18. Диаграмма классов для моделирования структуры цели

A.2.1.1.1. Структура функций

Функции можно описывать на разных уровнях агрегирования. Верхний уровень агрегирования — и, следовательно, отправная точка нашего обсуждения — состоит из сложных совокупностей функций (так называемых комплексных функций), которые для упрощения картины структурируются, т.е. представляются в виде иерархических диаграмм с разбивкой на подфункции. На рис. 19а приведена иерархическая диаграмма для функций, (преимущественно) преобразующих информацию, а на рис. 19б — для функций, (преимущественно) преобразующих материалы.

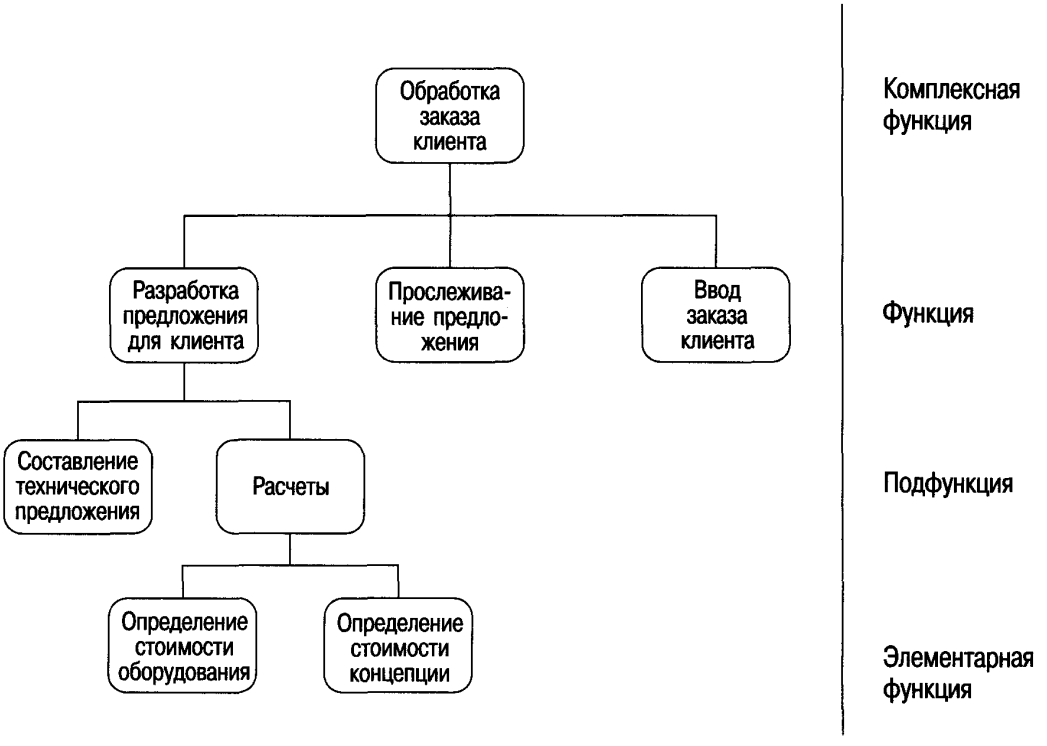


Рис. 19a. Иерархическая диаграмма для функций, преобразующих информацию

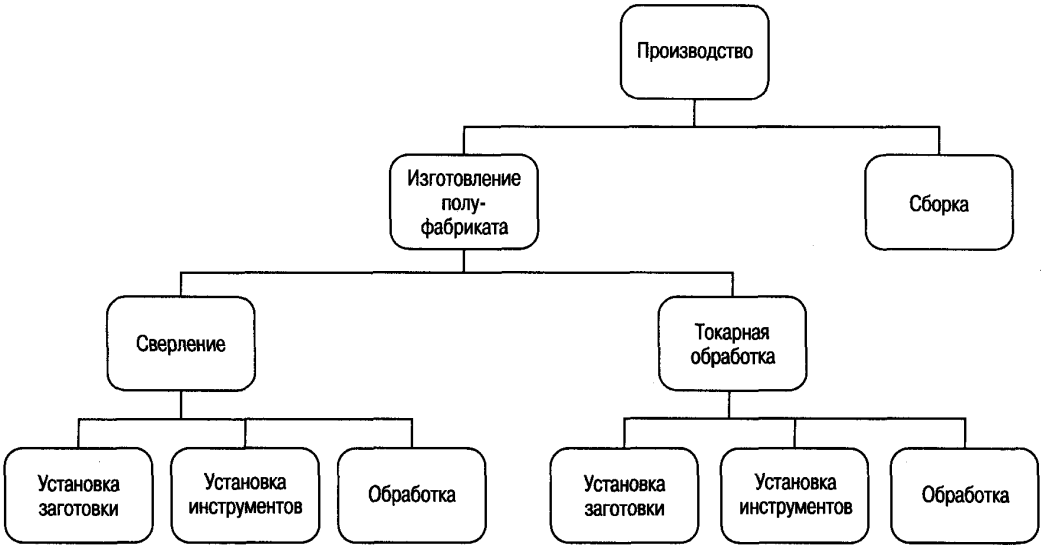


Рис. 19b. Пример иерархической диаграммы для функций, преобразующих материалы

Как видим, здесь представлены два основных класса функций, различающихся по обрабатываемому объекту. Мы сосредоточим внимание на функциях, преобразующих информацию в условиях офиса, хотя все изложенное ниже справедливо и для функций, выполняемых на производстве.

Вообще понятием «функция» можно пользоваться на любом уровне иерархии, хотя нередко его расчленяют на более мелкие составляющие в зависимости от степени детализации, используемой при обсуждении (см. рис. 19а):

- **комплексная функция:**
  сложная мульти-функция, состоящая из множества операций;
- **функция:**
  сложная операция, которую можно разложить на составляющие; непосредственно входит в комплексную функцию;
- **подфункция:**
  операция, которую можно разбить на подфункции или элементарные функции; входит в доминирующую функцию;
- **элементарная функция:**
  операция, которую нельзя разбить на составляющие; например, операции, выполняемые на одном рабочем месте, или внутренние процедуры, не имеющие альтернативы.

Эта классификация носит весьма условный характер и нередко допускает произвольное толкование, поэтому здесь мы будем довольствоваться общим понятием «функция». Разбивка функций на составляющие обычно производится методом «сверху вниз» — при помощи иерархических диаграмм. Однако этот метод имеет свои недостатки. Например, здесь часто отсутствуют строгие правила классификации, что на определенном уровне затрудняет контроль согласованности функций. Противоположный метод - группировка элементарных функций в более крупные функциональные блоки - отличается большей систематичностью. Именно поэтому в практических приложениях следует использовать оба метода. При этом сначала производится разбивка по принципу «сверху вниз», чтобы вычленить элементарные функции, которые затем подвергаются перегруппировке по принципу «снизу вверх». Мартин, Олле и другие авторы приводят интересные примеры применения функциональных иерархий *(Martin. Information Engineering II. 1990, с. 45; Olle et al. Information Systems Life Cycle. 1988, с. 57)*.

Иерархии функций можно создавать в соответствии со следующим принципом: «идентичные процедуры, идентичные информационные объекты и идентичные описания должны применяться к идентичным бизнес-процессам» *(Nüttgens. Koordiniert-dezentrales Informationsmanagement. 1995, с. 97)*. С учетом идентичности «бизнес-процесса» (т.е. параметров, отражающих эту идентичность) группируются только статичные функции в отличие от динамичного описания бизнес-процессов. На рис. 20 приведено несколько примеров классификационных параметров. При дальнейшей дифференциации параметров получают подгруппы.

Выбор классификационных параметров зависит от предназначенности данной модели. При реинжиниринге бизнес-процессов функции удобно классифицировать по этим параметрам. Если впоследствии систему предполагается развивать далее, то функции следует классифицировать по видам работ (операций), что удобно при повторном использовании функциональных модулей. Это позволяет одновременно создавать разные классификации функций и подфункций.

С учетом этого обстоятельства при построении метамодели, приведенной на рис. 21 была приведена классификациия функций по видам операций и процессам. Каждая функция вводится только один раз, поэтому мы создаем класс ОБЩАЯ ФУНКЦИЯ, описывающий действия независимо от параметров классификации. Таким образом,

Параметры классификации	Характеристика	Пример
Операция (работа)	Групповые функции с идентичными или сходными правилами образования "вход-выход"	Обработка списка счетов-фактур клиента Обработка списка клиентов Обработка по заработной плате
Обрабатываемый объект	Групповые функции, обрабатывающие одни и те же объекты	Ввод заказа Отмена заказа Выполнение заказа
Бизнес-процесс	Групповые функции, участвующие в процессе	Выбор поставщиков, обсуждение условий поставки Составление заказа на поставку

Рис. 20. Параметры классификации функций

функции «принятие заказа» и «проверка готовности» описываются только один раз как экземпляры класса ОБЩАЯ ФУНКЦИЯ.

Каждое свойство функции, не включенное во взаимоотношения процессов, описывается как атрибут данного класса. В соответствии с предложением Олле и его соавторов мы можем разграничить имена новых элементов и сами элементы, т.е. имена общей функции и саму общую функцию *(см. Olle et al., Information Systems Life Cycle. 1988)*. Это облегчает оперирование синонимами и ононимами в многоязычных концепциях. Однако в данной работе для простоты мы опустим такую дополнительную дифференциацию.

Для разграничения функций, преобразующих информацию, и функций, преобразующих материалы, мы создадим подклассы.

Классификационные структуры, ориентированные на операции, изображаются ассоциативным классом СТРУКТУРА ОБЩЕЙ ФУНКЦИИ (ориентированная на операции). Мощности отношений *:* предполагают сетевую структуру, т.е. некая функция может быть включена в несколько доминирующих функций. Если функции представлять исключительно в виде древовидных индексов, это приведет к бессчетному количеству кортежей.

Бизнес-процессы поддерживают доминирующие корпоративные цели. Эту взаимосвязь отражает элемент ПОДДЕРЖКА БИЗНЕС-ЦЕЛЕЙ, помещенный между БИЗНЕС-ПРОЦЕССОМ и КОРПОРАТИВНЫМИ ЦЕЛЯМИ. Минимальная мощность отношения равна 1. Процесс, не поддерживающий корпоративную цель, лишен смысла, точно так же, как лишена смысла корпоративная цель, с которой не связана та или иная бизнес-цель. Бизнес-процессы можно разбивать на подпроцессы. Это находит отражение в ассоциативном классе СТРУКТУРА БИЗНЕС-ПРОЦЕССА. В этом контексте подпроцессы могут приводить к нескольким доминирующим (например, основным) процессам.

Общие функции, относящиеся к соответствующим бизнес-процессам, связываются с последними при помощи ассоциативного класса ФУНКЦИЯ. Поскольку представление функций ориентировано на бизнес-процессы, понятие «функция» определяется только на том этапе, где оно связывается с бизнес-процессом. Ассоци-

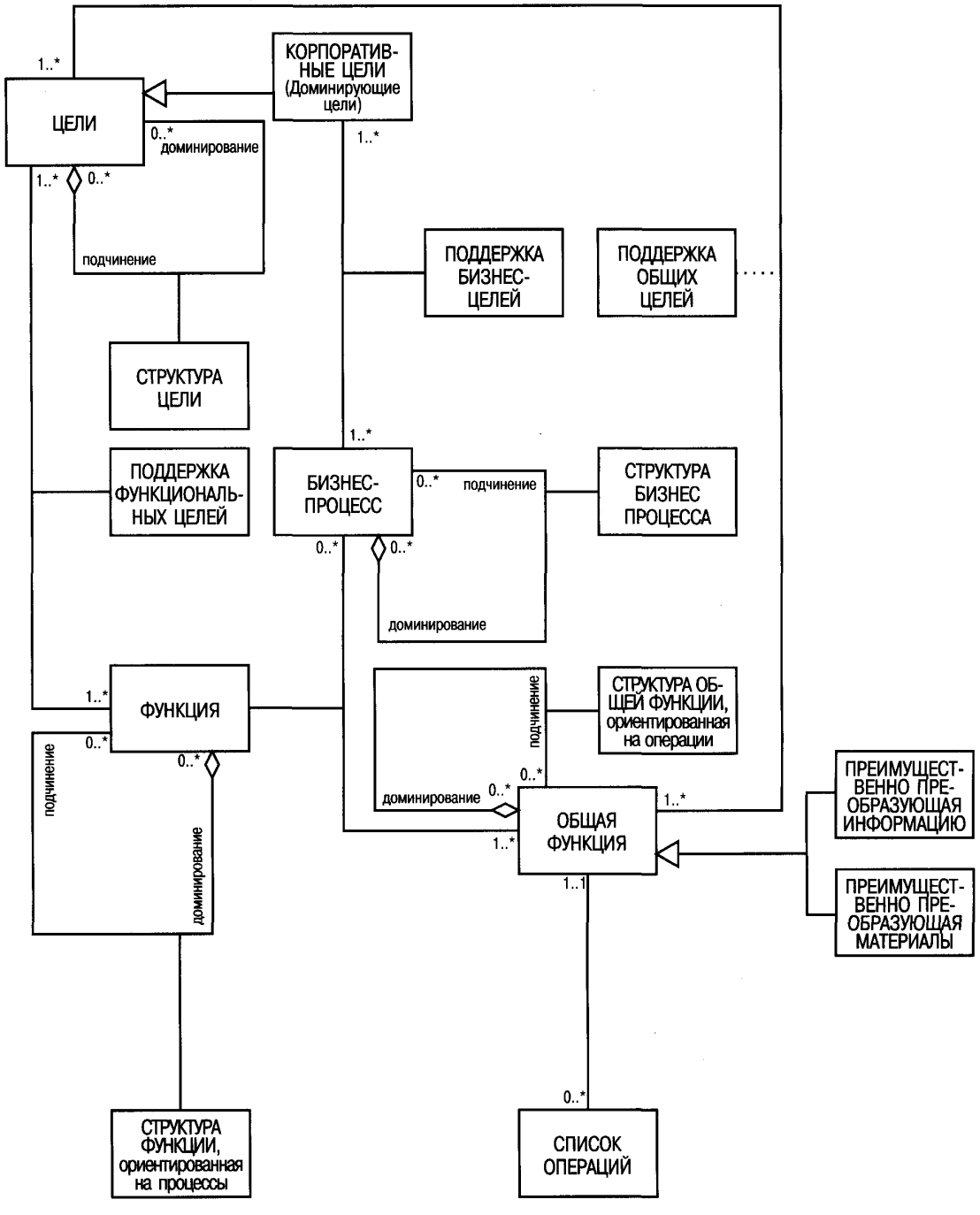


Рис. 21. Метамодель, описывающая структуры функций и целей

ативный класс ФУНКЦИЯ может иметь атрибуты общей функции. С другой стороны, ассоциативному классу можно присвоить дополнительные атрибуты, присущие конкретному процессу. Функции разбиваются на дальнейшие составляющие в зависимости от конкретного процесса (т.е. контекста). Поскольку та или иная функция может фигурировать несколько раз в рамках одного и того же процесса, здесь также используются сетевые структуры.

Функции на верхнем уровне иерархии включаются непосредственно в основные бизнес–процессы и, следовательно, не имеют над собой доминирующей функции. Соответственно элементарные функции нижнего уровня не разбиваются на дальнейшие составляющие и не имеют подчиненной функции. Следовательно, их минимальная мощность равна 0. Цели также связываются с конкретными функциями.

На первый взгляд, статичные структуры бизнес–процессов и структуры функций, ориентированные на процессы, могут быть очень похожи. На рис. 22а представлен бизнес–процесс «планирование и управление производством» (П и УП) с изображением ряда функций.

На рис. 22б изображена сетевая структура с древовидным индексом без избыточных функций, что стало возможным благодаря отношениям, имеющим мощность (0..*):(0..*).

Если функции, фигурирующие в нескольких подпроцессах требуется дифференцировать, им можно присвоить ролевые имена (см. рис. 23). Таким образом, функции идентифицируются по ролевым именам, по бизнес–процессам, в которых они участвуют, и по их базовому назначению. Ролевые имена можно использовать и вместо других, еще не описанных элементов бизнес–процесса. Например, на рис. 22 при «распределении» процессов ПиУП между иерархическими уровнями организационной структуры ролевое имя могло бы служить для обозначения организационной единицы, выполняющей данную функцию. Здесь проверка среднесрочной готовности проводилась бы на уровне отдела, а проверка краткосрочной готовности — на уровне операции.

Предлагаемое решение позволяет придать каждой общей функции специальные свойства в контексте конкретного процесса. С другой стороны, если функции описаны настолько четко, что ими можно оперировать в контексте любого приложения, и к тому же состоят из одинаковых подфункций, то их можно описать как общие конструктивные блоки. На рис. 22б такими функциональными объектами могут быть «проверка готовности» и «создание резервов», поэтому они заключены в рамку вместе со своими подфункциями. Применительно к диаграмме классов на рис. 21 это означает, что ассоциативный класс СТРУКТУРА ОБЩЕЙ ФУНКЦИИ представляет собой описание в виде конструктивных блоков, а мощность отношения между ОБЩЕЙ ФУНКЦИЕЙ и БИЗНЕС–ПРОЦЕССОМ равна (1..*):(0..*). Это позволяет привязывать к процессам не каждую общую функцию, а только главные конструктивные блоки, одновременно являющиеся функциональными объектами.

В метаструктурах понятие ФУНКЦИОНАЛЬНЫЙ ОБЪЕКТ представляют в виде конкретной версии класса ОБЩАЯ ФУНКЦИЯ (см. рис. 24). Затем из функциональных объектов можно компоновать сложные функциональные структуры. Позже мы раздвинем рамки этого сценария, связав функции с другими моделями ARIS для формирования бизнес-объектов.

Помимо классификаций, ориентированных на операции, ассоциативный класс СТРУКТУРА ОБЩЕЙ ФУНКЦИИ позволяет также моделировать классификации, ориентированные на процессы или объекты.

Описание функции на уровне макромодели детализирует, «что» делается, т.е. задачу, выполняемую данной функцией (например, «обработка заказа»). Микромодель

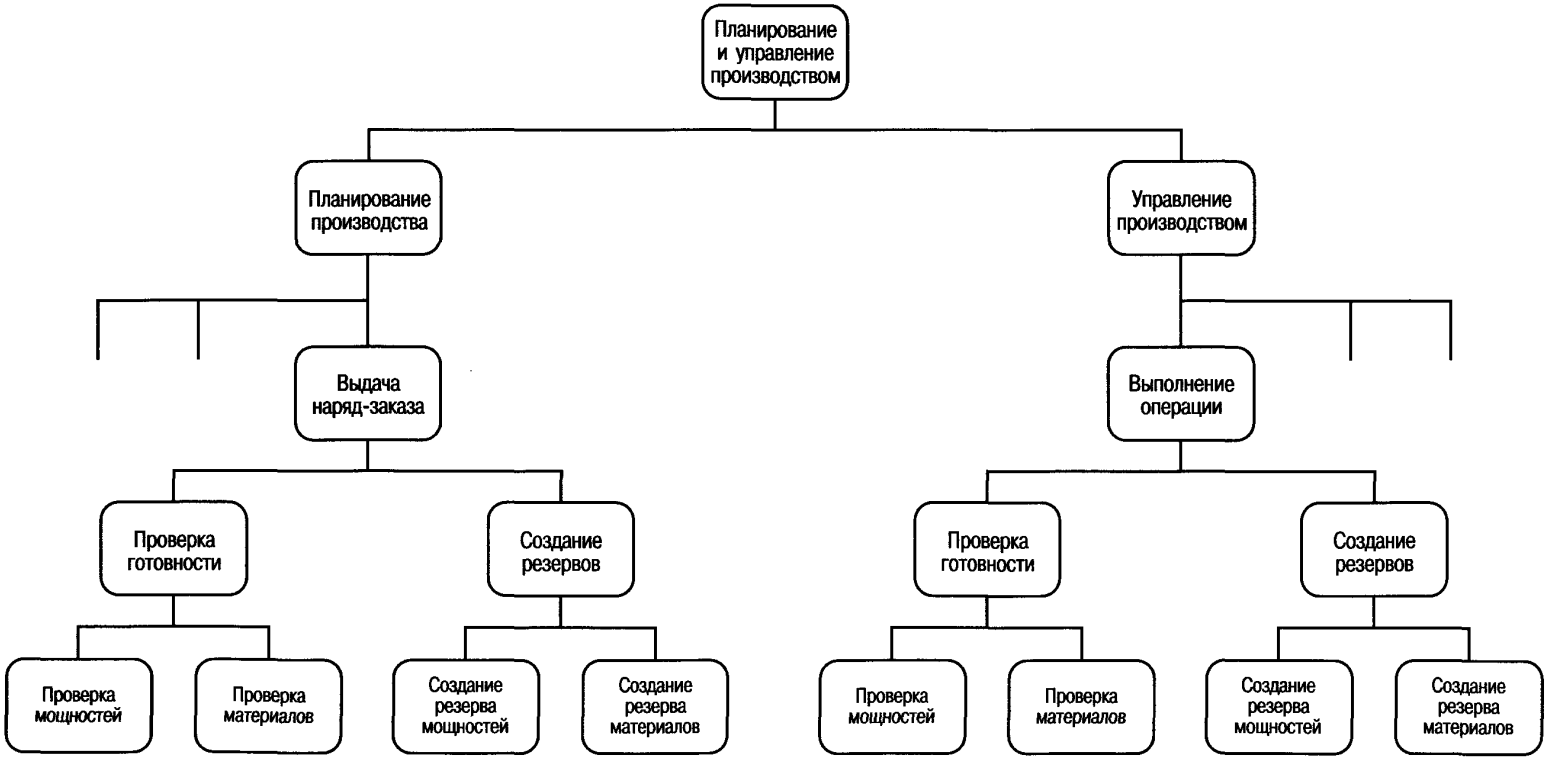


Рис. 22а. Планирование и управление производством (П и УП): структура функций, ориентированная на бизнес-процесс

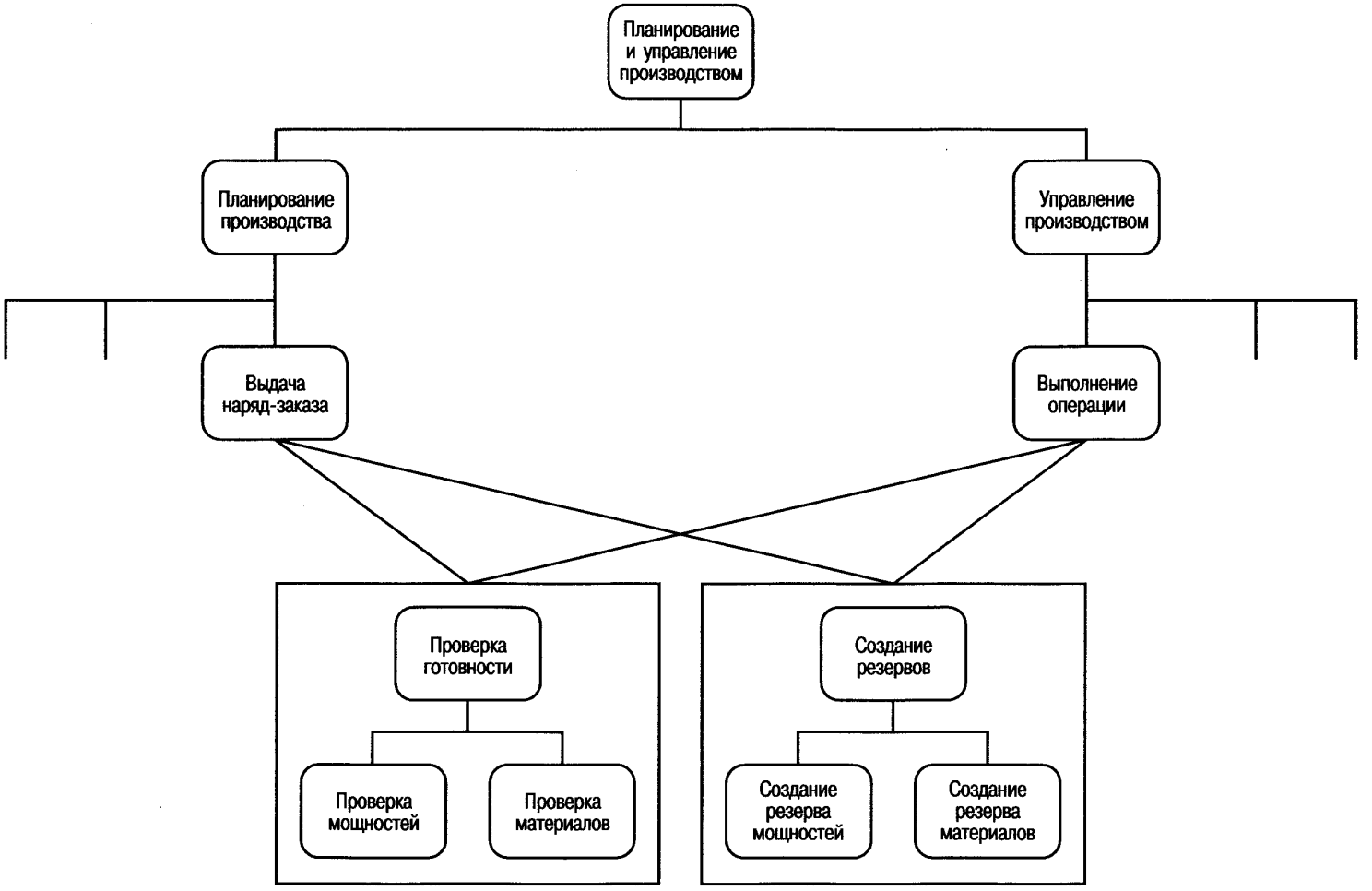


Рис. 22б. Производственное планирование и управление (ППУ): структура без избыточных функций

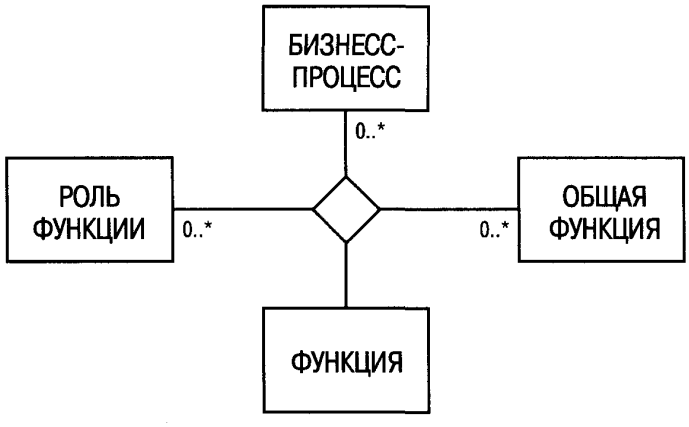


Рис. 23. Присвоение ролевых имен

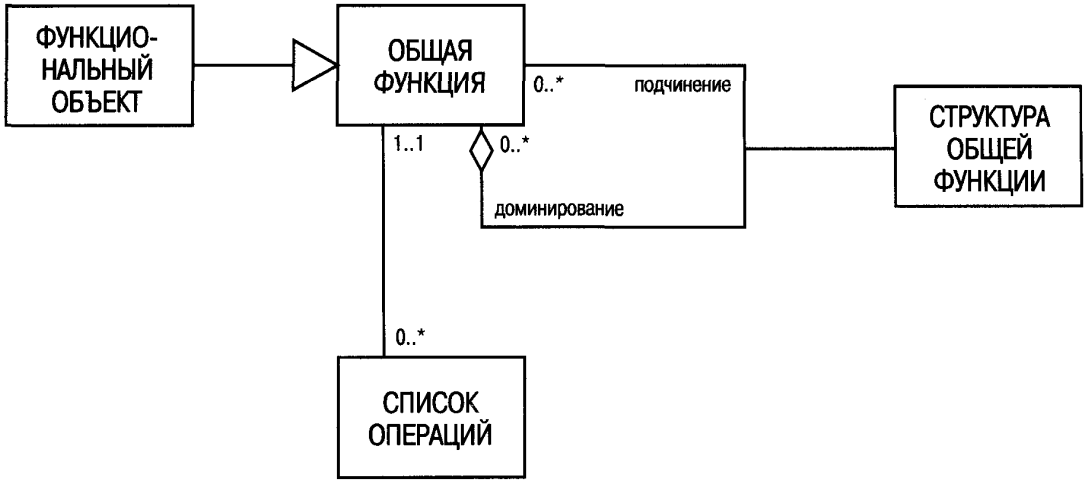


Рис. 24. Функциональный объект метамодели

описывает, «как» это делается, т.е. правила, которые необходимо соблюдать для выполнения данной функции. По-другому их называют списками операций. Например, функцию «командировка» можно разбить на следующие получение командировочных документов; проверка соответствия командировки корпоративным правилам; проверка расходов; выдача разрешения.

Эти операции выполняются в контексте поставленной задачи. Они не классифицируют функцию, а являются ее составляющими. Соответствующая документация представляет собой своего рода перечень этапов, необходимых для выполнения функции. Его можно моделировать в виде текста, структурограмм или таблиц решений *(Nüttgens. Koordiniert-dezentrales Informationsmanagement. 1995, с. 95).*

В метаклассах СПИСКИ ОПЕРАЦИЙ представляются как отдельный класс, связанный с общей функцией.

Если документальное описание внутренних процессов отсутствует, как это обычно бывает с такими творческими функциями, как принятие решений или сбор идей, то в результате вы имеете внутренние неструктурированные функции, которые могут поддерживаться только инструментальными средствами для групповой работы (Groupware), а не прикладными системами, ориентированными на четкое выполнение операций.

A.2.1.1.2. Последовательности процедур

Помимо описания структуры функций, цель этапа формулировки требований заключается в установлении процедурной последовательности функций, что фактически прокладывает путь к описанию процессов. В отличие от описания процессов на уровне модели управления, которое выполняется позже, на этом этапе описываются логические последовательности функций, но не их активизаторы (т.е. события). Это рекомендуется в тех случаях, когда активизирующие события или сообщения носят настолько рутинный характер, что не вносят никакой дополнительной информации на этапе определения требований, или когда события, активизирующие функцию, добавляются позже — на стадии проектирования.

Для описания процедур можно воспользоваться методами сетевых диаграмм, представляющих различные отношения типа предшествующая–последующая, метрики расстояний, возможные наложения и минимальные расстояния между событиями. Кроме того, можно описать логические связи между входящими и исходящими элементами соответственно входящих и исходящих отношений.

На рис. 25 часть древовидного индекса, приведенного на рис. 19а, представлена в виде процедуры. Это подтверждает, что изначально описать контекст процедуры на основании древовидного индекса нельзя. После соответствующих расчетов, для которых необходимо знать «приближенные показатели» (например, ставки заработной платы) и стоимость заказываемого оборудования, описывается узел решений с тремя альтернативными исходящими ветвями: составление нового технического предложения, если вычисленная цена нереальна; отказ от выполнения, если есть основания полагать, что предложение не будет принято; выполне-

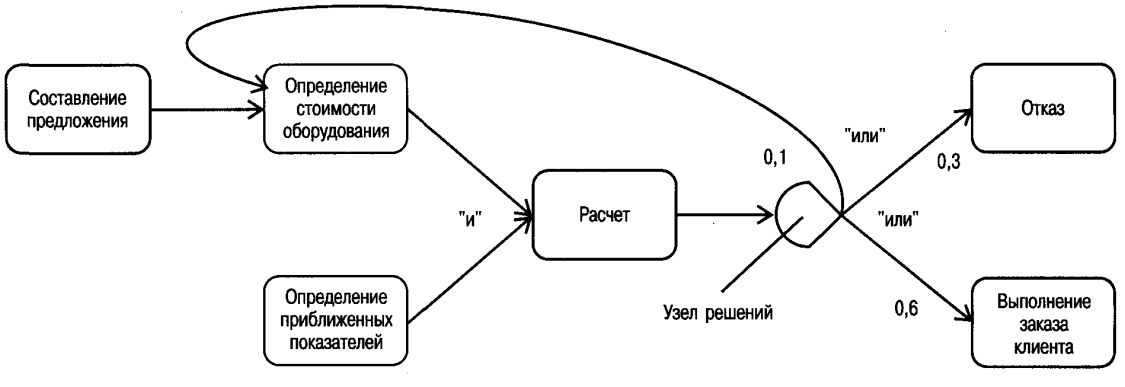


Рис. 25. Процедурная последовательность функций

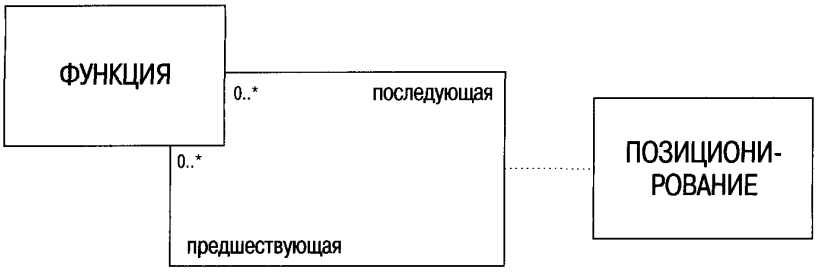


Рис. 26. Учет позиционных отношений

ние заказа, поскольку клиент принял предложение. Вероятность каждой альтернативы можно привязать в качестве атрибута. Поскольку эти альтернативы взаимоисключающие, их максимальная мощность должна равняться 1.

Эта концепция из GERT (графический метод оценки и анализа; *см. Elmaghraby. Activity Networks. 1977; Scheer. Projektsteuerung. 1978).*

Введенный здесь узел решений может быть выделен в отдельный элемент, но может интерпретироваться и как нормальное событие с нулевой временной продолжительностью либо как часть предыдущего события.

Упорядоченные отношения образуют в рамках класса ФУНКЦИЯ новую ассоциацию — ПОЗИЦИОНИРОВАНИЕ. Каждое отношение позиционирования можно идентифицировать по предыдущему и последующему этапу функции. Добавление на рис. 26 ассоциативного класса ПОЗИЦИОНИРОВАНИЕ позволяет присваивать в качестве атрибутов метрики расстояния для наложений, задержки или коэффициенты (их значения помещаются на соответствующие ветви).

Логические зависимости между ребрами соответствующего графа присваиваются отношениям позиционирования в качестве атрибутов.

Для функций, преобразующих материалы, процедурные последовательности описываются в графиках работ. Понятие «график работ» уже предполагает описание некоего процесса, особенно в силу того, что такие графики содержат элементы других моделей ARIS (организационные единицы, ресурсы, материальные выходы). Однако в график работ явным образом не входит управление событиями, поэтому динамика процесса, описываемая позже в модели управления, тоже не находит в нем явного отражения.

Графики работ относятся к изготовлению деталей и могут составляться применительно к разным уровням их описания. Графики работ для подклассов создаются на уровне типов, как показано в упрощенном примере на рис. 27а. Каждая операция описывает функцию, а график работ, как она включается в процесс.

Операции соответствуют техническим процедурам. Технические процедуры можно описывать независимо от контекста графика работ, а затем уточнять их в контексте этого графика или процесса на более позднем этапе. Диаграммы классов соответствуют рис. 27б. С технической точки зрения, эта диаграмма идентична метаструктуре функции, показанной на рис. 21 (БИЗНЕС–ПРОЦЕСС, ОБЩАЯ ФУНКЦИЯ, ФУНКЦИЯ). Таким образом, последовательности технических функций можно рассматривать так же, как последовательности административных функций.

Документирование графиков работ - одна из классических задач планирования

График работ для изделий из листового металла			
Номер операции	Название операции	Продолжительность (средняя)	Группа производственных ресурсов
1	Сверление	1	ГПР 1
2	Фрезерование	2	ГПР 5
3	Снятие заусенцев	3	ГПР 4
4	Промывка	4	ГПР 7

Рис. 27а. График работ на уровне типов деталей

Рис. 27б. Диаграмма классов для графиков работ

и управления производством в информационных системах. Графики работ анализируются на уровне деталей с привлечением экземпляров классов деталей и заполнением соответствующих баз данных.

Диаграмма классов для администрирования графиков работ, относящихся к изготовлению деталей, представлена на рис. 27б (Scheer. Business Process Engineering. 1994, с. 216). Контекст процесса становится ясен из описания отношения между деталями, которое теперь становится подэкземпляром.

Обычно в моделировании бизнес-процессов не принято использовать функции, связанные с деталями. Если это и делается, то лишь применительно к особо важным конечным продуктам.

Графики работ на уровне типов и экземпляров, которые мы рассматривали до сих пор, аналогичны эталонным данным в том смысле, что они не зависят от контекста заказов, привязанных к фактору времени, хотя экземпляры, управляемые системами workflow, привязаны к заказам. Здесь эталонные графики работ, оперируют

ющие типами и экземплярами, являются своего рода шаблонами. В системах ПиУП эталонные графики работ, соответственно оперирующие данными и заказами, также рассматриваются параллельно.

A.2.1.1.3. Типы обработки

Для того чтобы описать способ реализации функции (средствами ИТ или вручную), при спецификации понятия ФУНКЦИЯ можно разграничить СИСТЕМНУЮ ФУНКЦИЮ и РУЧНУЮ ФУНКЦИЮ (см. рис. 28).

Системные функции порождают заказы клиентов, сопровождают данные о клиентах, ведут статистику и т.п. при помощи информационных систем. Если ПРИКЛАДНАЯ СИСТЕМА уже известна, то эта информация также приобщается к системной функции. Однако такие сведения должны носить лишь общий характер (например, имя бизнес–приложения), чтобы заранее не предопределять описание на уровне спецификации проекта.

Этап определения требований включает также описание типа предварительной обработки для системных функций. Один из ключевых параметров типа обработки определяется в зависимости от ситуации: могут ли пользователи вносить коррективы в процесс (оперативная обработка) или же функции выполняются без вмешательства со стороны пользователей (пакетная обработка). Чтобы решить, подходит ли данная функция для оперативной обработки, можно воспользоваться параметрами, приведенными и на рис. 29. Исходя из этих соображений, мы подразделяем класс СИСТЕМНАЯ ФУНКЦИЯ на подклассы ОПЕРАТИВНАЯ ФУНКЦИЯ и ГРУППОВАЯ ФУНКЦИЯ.

A.2.1.1.4. Модели решений

Помимо административных целей, информационные системы используются также для поддержки решений, например, для оптимизации производственного планирования.

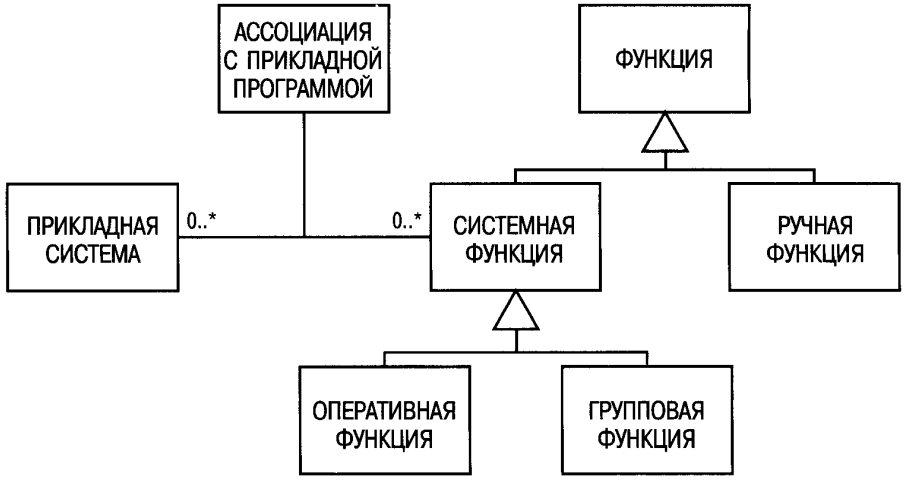


Рис. 28. Спецификация понятия «функция»

Свойства / Цели	Управляемость событиями	Возможность интеграции функции	Допускает интерактивные решения	Устраняет пиковые нагрузки	Позволяет вносить улучшения	Позволяет повышать качество
Экономия времени	●	●	●	●	●	
Экономия рабочей силы		●		●	●	
Получение Информации	●		●			●
Создание блгоприятных условий работы	●	●		●	●	●
Оптимизация организационных процессов		●	●		●	

Рис. 29. Параметры и цели оперативной обработки

В качестве примера приведем типичную структуру модели решеня и рассмотрим метод линейного программирования (ЛП). В моделях ЛП — при соблюдении всех вторичных условий — переменные задаются таким образом, чтобы максимизировать целевые функции (см. рис. 30). Структуры ЛП не связаны с каким–либо конкретным приложением и располагаются на метауровне описания моделей решений. На рис. 31 представлена модель ЛП на 2–ом уровне абстракции, относящаяся к приложению для планирования производства (уровни абстракции в моделировании см. Scheer. ARIS — Business Process Frameworks. 1998, с. 120–125; в русском издании с. 109-115).

В представлении модели ЛП как диаграммы классов внимание фокусируется на объектах метамодели ЛП (см. рис. 30). Модели ЛП состоят из элементов ПЕРЕМЕННАЯ, УРАВНЕНИЕ (вторичные условия и целевые функции) и КОЭФФИЦИЕНТ.

Для отдельных моделей решений формируется класс МОДЕЛЬ РЕШЕНИЙ (см. рис. 32). В одной ФУНКЦИИ (например, в планировании производства) можно использовать несколько моделей решений. И наоборот, одну модель решений можно применять к нескольким разным функциям. Мощности отношений равны соответственно, (0,..*).

С одной моделью решений связывается несколько уравнений, при этом одни и те же уравнения могут встречаться в разных моделях (например, потребность во вспомогательных мощностях может фигурировать как в модели краткосрочного планирования производства, так и в модели планирования инвестиций).

Различные переменные, такие как, например, объем производства, объем продаж и размер инвестиций, могут использоваться в нескольких моделях решений.

Отношение между заданными переменными (столбцы матрицы ЛП) и уравне-

Целевая функция:	$\sum_j c_j x_j \rightarrow \max$
Вторичные условия:	$\sum_j a_{ij} x_j \leqslant A_i$ для всех i $x_j \geqslant 0$ для всех j
Переменные:	x_j
Коэффициенты:	a_{ij}, x_j

Рис. 30. Структура модели ЛП

$\sum_j c_j x_j \rightarrow \max$	c_j = вклад j-го продукта
	x_j = объем производства j-го продукта
$\sum_j a_{ij} x_j \leqslant C_i$, (для всех i)I	a_{ij} = потребности в мощностях i-го типа на еденицу j-го продукта
	C_i = предельные мощности i-го типа
$x_j \leqslant M_j$, (для всех j)	M_j = максимальный объем продаж j-го продукта

Рис. 31. Модель ЛП для планирования производства

ниями (строки матрицы ЛП) устанавливается с помощью коэффициентов. В каждом столбце (т.е. для каждой переменной) коэффициенты можно подставлять во множество уравнений. И наоборот, в каждой строке (уравнении) можно рассматривать несколько переменных.

Генераторы матриц, переменные, уравнения и коэффициенты модели можно получить из базы данных, описав все допустимые индексные комбинации переменной на основе хранящегося в ней логического контекста (Scheer. Business Process Engineering. 1994, с. 525). Формат системы математического программирования (MPS) позволяет стандартизировать описание.

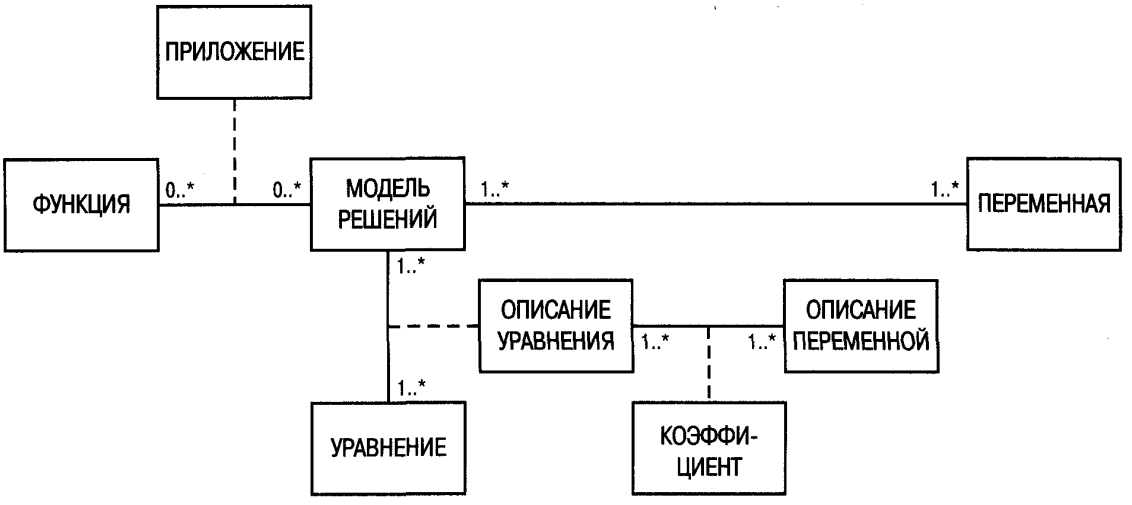


Рис. 32. Логическая структура моделей решений

Логическая структура модели решений, изображенная на рис. 32, представляет собой структуру данных для репозитория, где хранятся модели приложений (Scheer. Principles of Efficient Information Management. 1991, с. 157).

A.2.1.1.5. Объединение определения требований на уровне функциональной модели

На рис. 33 представлена объединенная метамодель определения требований на уровне функций.

A.2.1.2. Конфигурирование функций

Если мы хотим использовать собственные интерфейсы систем управления процессами, потоками работ и конфигурацией прикладных систем, необходимо следовать определенным требованиям моделирования. Однако модели должны по-прежнему оперировать только понятиями бизнеса.

Сначала формируются связи функций с классами прикладных систем (системы управления проектами, текстовые процессоры, бизнес-приложения), которые нужно сконфигурировать. Если типы прикладных систем уже можно описать более подробно (MS Project, MS Word for Windows, R/3 и т.д.), эти связи могут быть установлены. Кроме того, следует указать, предполагают эти системы обмен данными или нет (см. рис. 34).

Функциональные модели и списки операций создают основу для функционального анализа, оценки стоимостных характеристик функций и управления бизнес-процессами (например, методом пооперационного исчисления стоимости). Для пооперационного исчисления стоимости функциям присваиваются необходимые атрибуты (время, количество, стоимость за единицу продукции и т.д.).

Содержание соответствующих функций планирования мощностей конфигурируется согласно функциональным моделям.

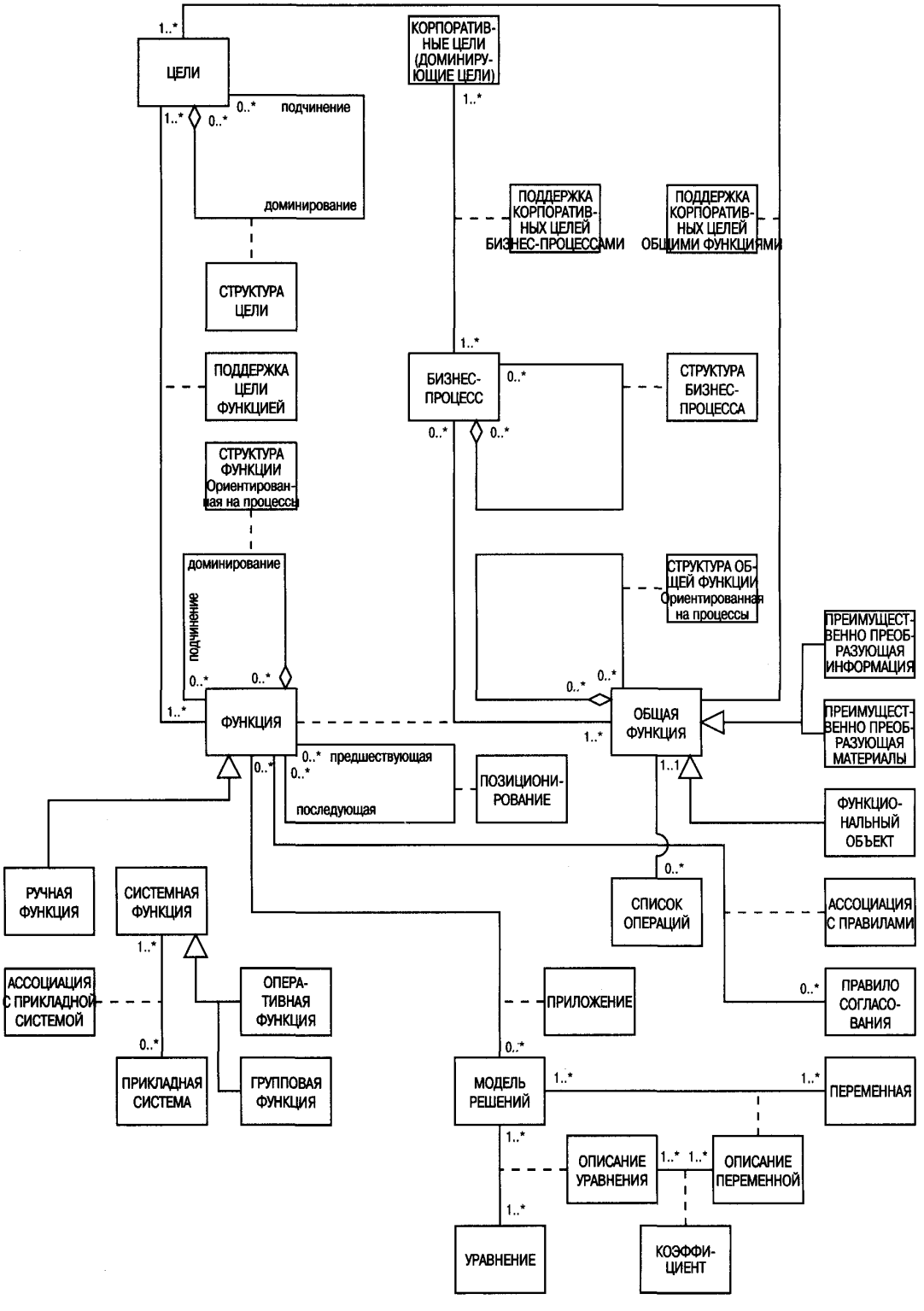


Рис. 33. Метамодель определения требований на уровне функционального представления

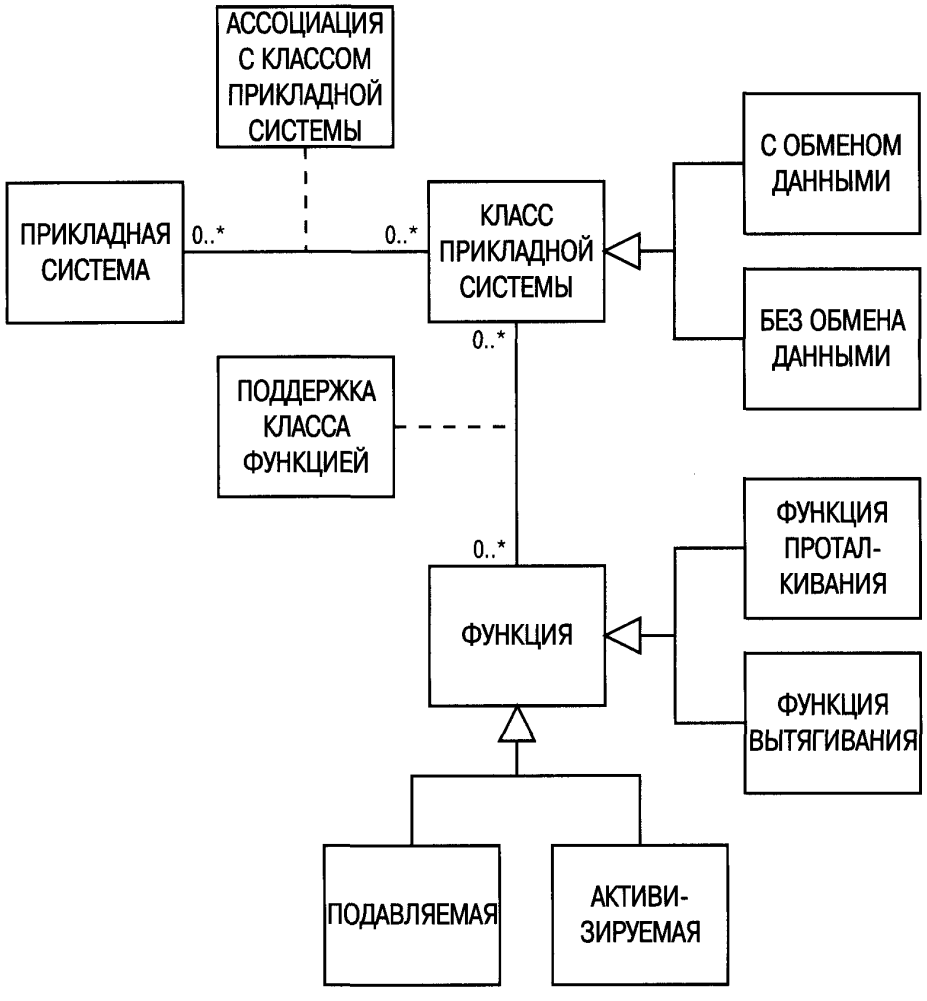


Рис. 34. Формирование связей прикладной системы

Описание функций необходимо для того, чтобы настроить систему планирования проектов на конкретное приложение в рамках проекта, связанного с производством или организационной структурой, например, при подготовке к торговой выставке.

Рассмотренные метамодели описания на уровне функций уже могут быть использованы для моделирования приложений workflow *(Galler. Vom Geschäftsprozeßmodellzum Workflow–Modell. 1997, с. 62)*. При описании атрибутов функций целесообразно внести конкретные уточнения, например предельные сроки (среднюю продолжительность функции, среднее время цикла, возможные наложения во времени и т.п.).

Вызов функции может осуществляться по принципу «вытягивания» («pull») или «проталкивания» («push»). В первом случае сотрудник извлекает соответствующую функцию из электронного почтового ящика вместе со списком операций, которые требуется выполнить. Во втором случае функции, управляемые событиями, присылаются ему на обработку.

Моделирование процессов workflow должно удовлетворять дополнительным требованиям на уровне экземпляров. Эти требования копируются из определения требований на уровне функций, но при необходимости их можно изменять таким образом, чтобы смоделировать сами экземпляры.

После того как бизнес-приложения связаны с функциями и сконфигурированы, особенно важно иметь возможность активизировать или исключать функции в рамках существующей первоначальной модели.

Если определенное бизнес-приложение использует определенную функциональную модель-прототип, модели для конкретных пользователей можно создавать путем активизации или «вычеркивания» соответствующих функций. Можно задать правила согласования, учитывающие логические связи между функциями. Например, при вычеркивании функции «управление складом грузов, предназначенных для отправки» необходимо удалить и функцию «заказ на отправку груза».

Чтобы обеспечить целостность модели-прототипа, каждую функцию нужно либо активизировать, либо исключить *(см. Scheer. ARIS — Business Process Frameworks. 1998, рис. 50)*. На рис. 35 представлен фрагмент модели-прототипа SAP R/3, иллюстрирующий редактирование с помощью ARIS Toolset. Структуру проекта для индивидуальной настройки SAP R/3 (нижнее окно) можно вызвать непосредственно из функциональной модели (верхнее окно). Внимательно изучив функцию, можно задать выбранные параметры в соответствии с определением требований. Программа SAP R/3 Business Engineer, обращающаяся непосредственно к системе настройки IMG (Руководство по управлению реализацией), предлагает для этой цели интерфейс, построенный по принципу вопросов и ответов.

На рис. 36 приведен фрагмент прототипа SAP AG Business Engineer. Модель проекта показывает функциональное дерево приложения. Активизация и деактивизация функций осуществляются путем установки флажков. Функции —в данном примере это «Credit processing» («Обработка кредитов») — отсылают пользователя непосредственно к нужным подфункциям настройки (нижнее окно слева), а отсюда можно выйти на нужные параметры (верхнее окно справа). Нижнее окно справа иллюстрирует, как функция встраивается в контекст процесса.

A.2.1.3. Определение требований на уровне функциональной модели

Определение требований к функциям можно проектировать на различных уровнях вертикальной иерархии, построенной по принципу «сверху вниз». По-другому это называется проектированием программного обеспечения, поскольку функции впоследствии реализуются в структуре подпрограммы. В этом значении широко применяется термин «крупномасштабное программирование», тогда как последующая реализация на языке программирования называется «мелкомасштабным программированием» *(Balzert. Lehrbuch der Software-Technik. 1996, с. 632, 927)*.

Ключевыми этапами проектирования являются построение структуры модуля, детальное проектирование содержимого модуля и выдача отчета.

Функции, преобразующие входные данные в выходные, особенно тесно связаны с моделью данных. При определении требований учитываются также существующие ограничения ИС. Эти связи ослабевают, если придерживаться принципа абстрагирования, что позволяет

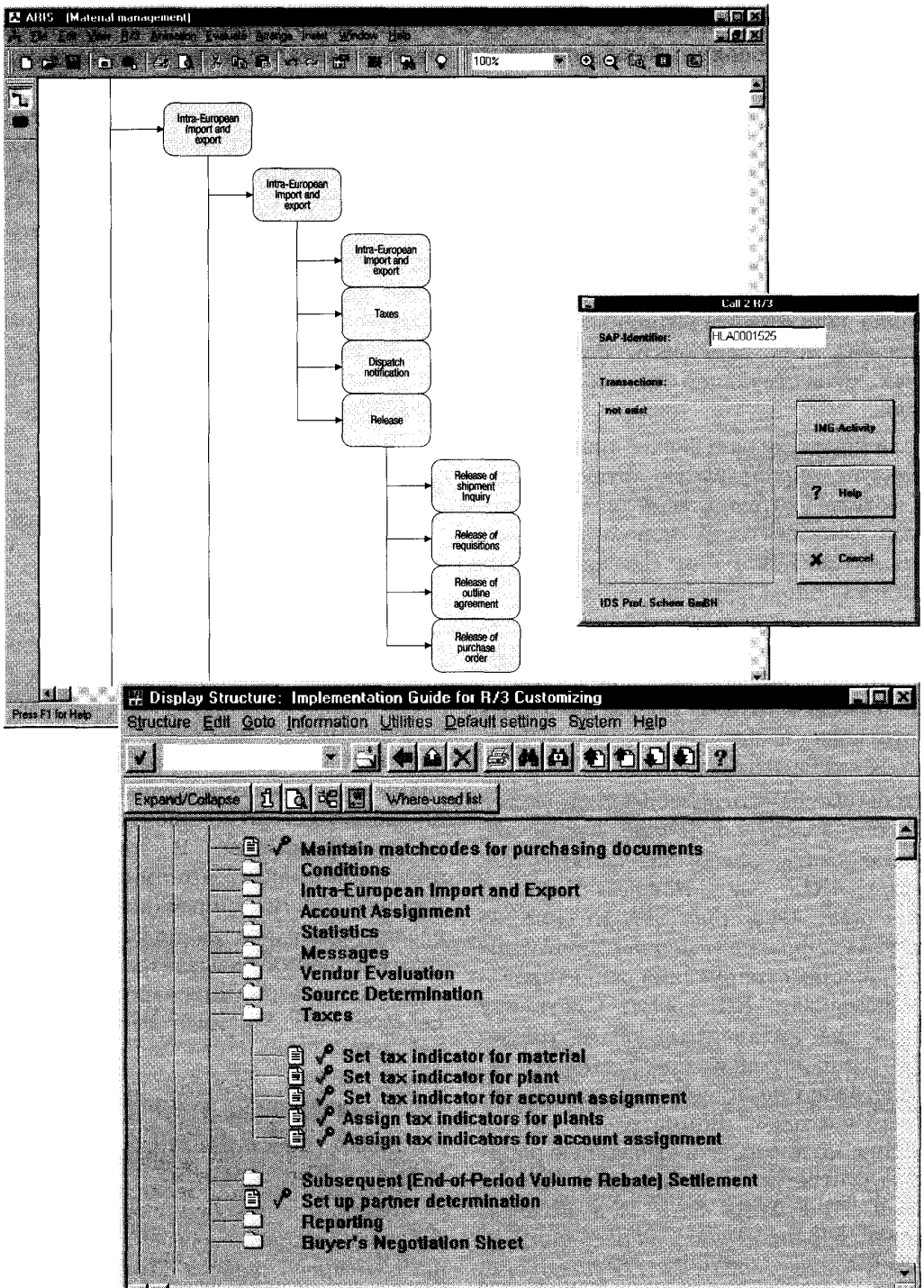


Рис. 35. Редактирование SAP R/3 с помощью ARIS Toolset

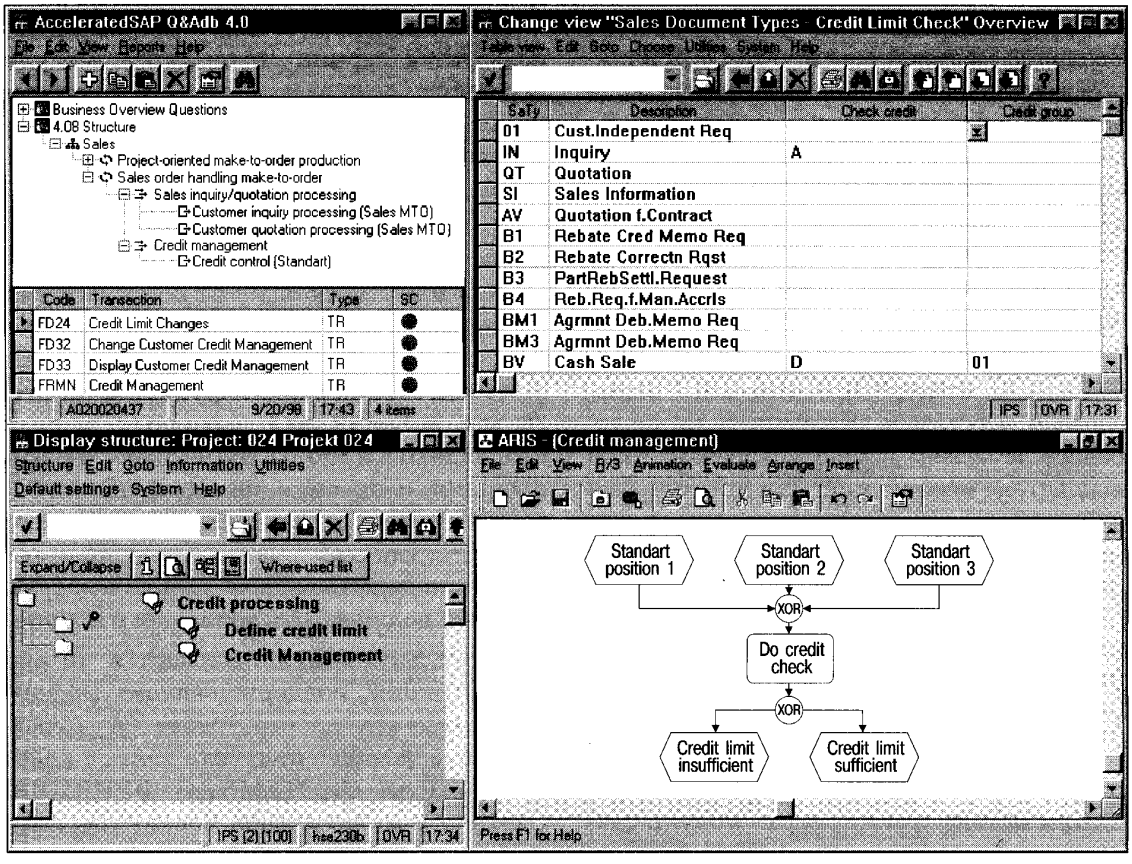


Рис. 36. Настройка функции в SAP R/3

создавать концепцию функции, не располагая знаниями о модели данных или о конкретной архитектуре оборудования. Иными словами, следует рассматривать только общие свойства смежных архитектурных моделей, а не их конкретные экземпляры. Это достигается абстрагированием данных, при котором описывается только тип данных, но не их физическая реализация. Скажем, для описания интерфейсов функции можно обозначить элемент данных «номер клиента» целым числом, не конкретизируя устройство, отвечающее за этот элемент, например, тип сущности КЛИЕНТ, имя отношения, тип записи или даже адрес записи и обозначение поля. То же самое справедливо для аппаратных компонентов, где при описании функций экранов можно задать виртуальный терминал с базовыми свойствами, но указывать конкретную модель терминала необязательно.

A.2.1.3.1. Проектирование модулей

Одним из центральных элементов проектирования программного обеспечения является модуль, четко описывающий автономный функциональный конструктивный блок для входящих и выходящих данных. Модули соответственно включают следующие компоненты: описание дан-

ных, управляющую логику и инструкции. Создание модулей согласуется с принципом локальности и многократного использования, поскольку модули можно использовать для разных прикладных функций. При описании входящих и исходящих данных, предоставляемых в распоряжение пользователя, соблюдается также принцип «сокрытия информации», т.е. описывается, что делает модуль, а не как он это делает.

Модули следует проектировать таким образом, чтобы максимизировать их «внутреннее действие» и минимизировать взаимодействие между ними. При проектировании модулей можно пользоваться методом «сверху вниз» или «снизу вверх». При методе «сверху вниз» проектирование начинается с самого верхнего уровня и постепенно детализируется по мере продвижения вниз. Конечным результатом являются базовые модули, которые можно реализовать с помощью существующих объектов базового программного обеспечения.

При методе «снизу вверх» модули сначала проектируются на самом нижнем уровне, а затем объединяются в модули следующего лежащего выше уровня. Методы «снизу вверх» особенно удобны для работы с уже заполненными архивами модулей, из которых извлекаются базовые модули, компонуемые затем в более крупные блоки (Blazer. Lerhbuch der Software–Technik. 1996, с. 853).

Применительно к модулям иногда используется термин «процедура»; модули верхнего уровня называют также программами. Возможен широкий ряд различных определений. На рис. 37 приведен пример очень детальной иерархии описания.

На уровне определения требований можно задать направление процесса проектирования, поскольку он допускает как восходящие, так и нисходящие методы. Выходные данные модуля проектируются в рамках этой иерархии функций. На рис. 38 мы выбрали для выходных данных класс ОБЩАЯ ФУНКЦИЯ. Здесь «общая функция» означает описание функции безотносительно к контексту конкретного бизнес–процесса. Это подчеркивает «принцип многократной применимости», который должен быть воплощен в модуле.

Поскольку модули создаются только для функций, поддерживающих информационные технологии, требуется уточнение, позволяющее получить связь с классом ОБЩАЯ СИСТЕМНАЯ ФУНКЦИЯ. Связь *:* с минимальной мощностью 1 означает, что благодаря многократной применимости один модуль можно использовать в разных системных функциях и одна системная функция может поддерживаться различными модулями. Связь *:* между системными функциями и модулями показывает также, что проектирование бизнеса и проектирование ИТ до определенной степени не зависят друг от друга.

- Класс прикладной системы — например, текстовый процессор Word и т.д.
- Тип прикладной системы — например, MS Word for Windows 6.0 и т.д.
- Прикладная система — например, MS Word for Windows 6.0 на ПК № 3417 и т.д.
- Класс модуля — например, программа проверки правописания и т.д.
- Тип модуля — например, программа проверки правописания для MS Word for Windows 7.0 и т.д.
- Модуль — например, программа проверки правописания для MS Word for Windows 6.0 на ПК № 3417 и т.д.

Рис. 37. Иерархия описания модулей

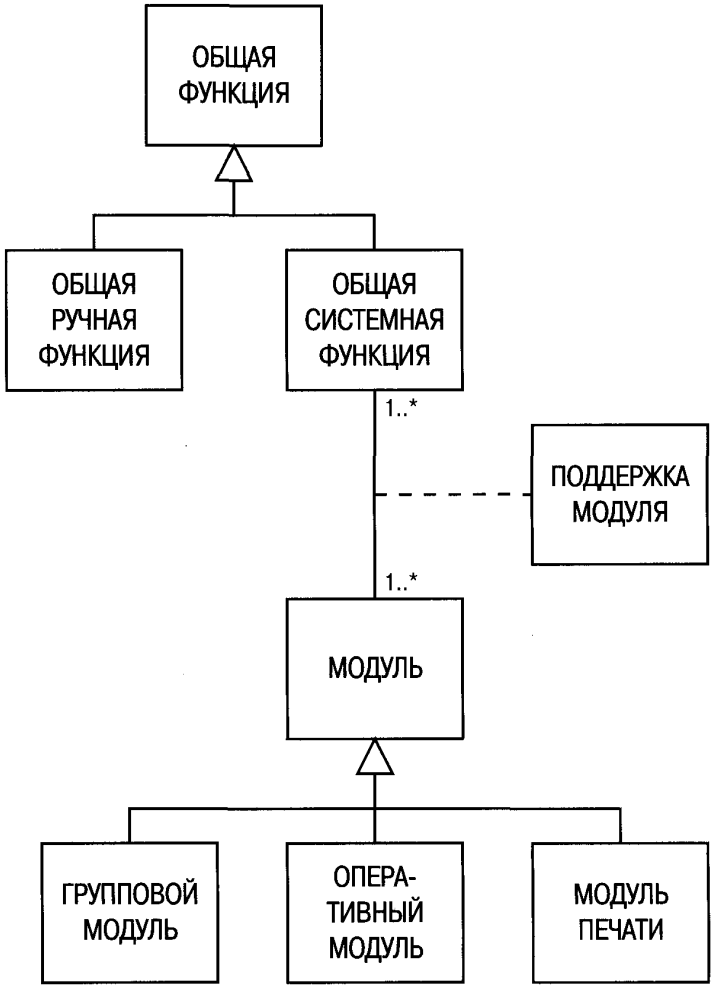
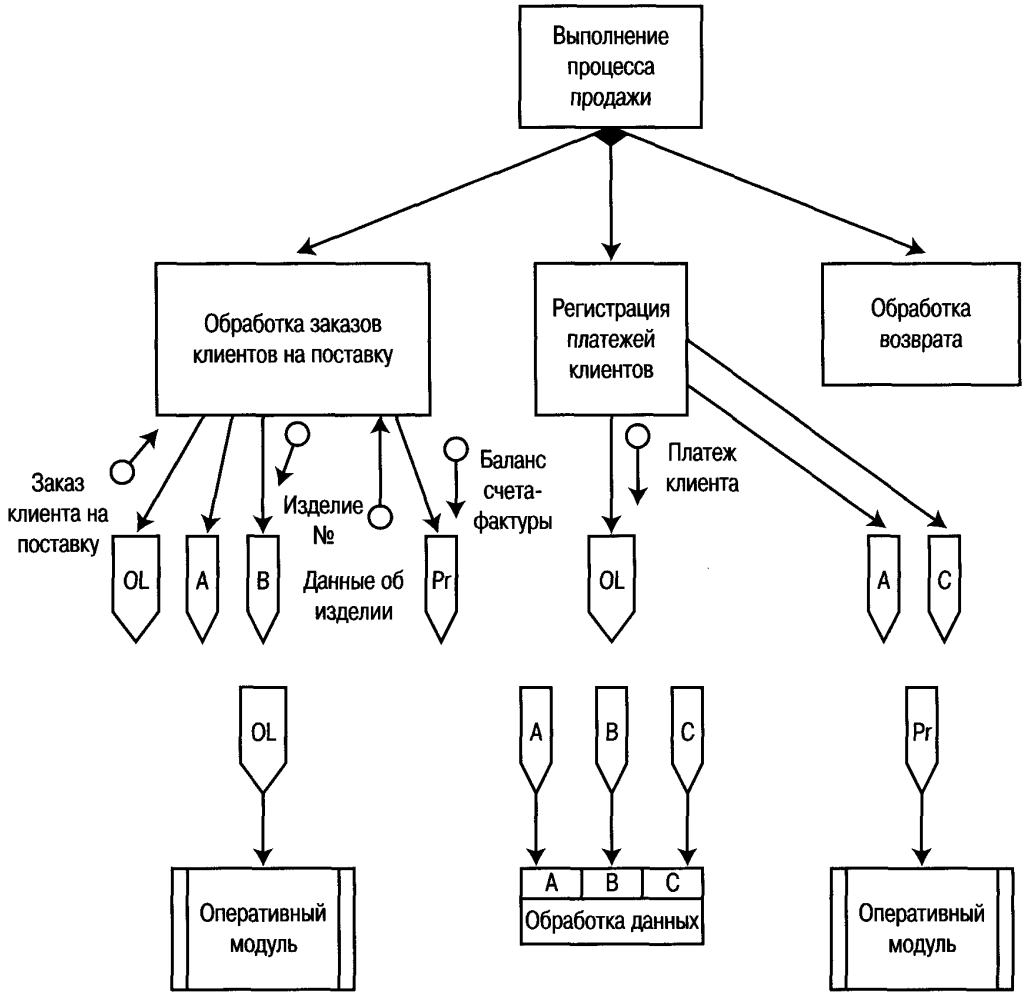


Рис. 38. Связь между модулями и функциями

Модули могут быть дифференцированы на подтипы и посредством отношений вызова соединены друг с другом в сети. На рис. 39 показана структурная диаграмма, где модули представлены прямоугольниками. Существующие модули, к которым имеется доступ, обозначены по краям двойными линиями.

Представление с помощью структурных диаграмм приобрело широкую популярность благодаря работе Константине и Йордона, посвященной сложному (структурированному) проектированию *(Constantine, Yourdon. Structured Design. 1979; о структурированном проектировании см. Page–Jones. Practical Guide to Structured System Design. 1980; Balzert. Lehrbuch der Software–Technik. 1996, с. 801–862; Sommerville. Software Engineering. 1987, с. 75–103).* Представление упрощается с помощью операторов (в данном случае - для оперативной обработки). Взаимодействие между модулями обозначено стрелками с указанием передаваемых данных, причем стрелки соответствуют простым связям данных. Мож-



A = считать и обновить эталонные данные о клиенте
B = считать и обновить данные об изделии
C = обновить счет клиента

: простая связь с данными
: управляющая связь
: динамичные параметры, т.е. входные и выходные значения

	вход	вход-выход	выход
1	Клиент №		Эталонные данные о клиенте
2	Сумма клиента №		Сальдо

Рис. 39. Представление модулей с помощью структурных диаграмм

но также использовать управляющие связи и динамичные параметры (т.е. параметры, требующиеся для ввода или вывода). При чрезмерно сложных взаимоотношениях данные можно пронумеровать и поместить в таблицы.

Представленные на рис. 39 операторы A, B и C связаны с операциями доступа и абстрагируют данные, т.е. обозначают данные вместе с операциями, для которых они предназначены. Ромб в модуле продаж обозначает управляющую структуру выбора.

Иерархические отношения между модулями вытекают из направления вызова (обращений). Модули на верхних уровнях вызывают модули следующего лежащего ниже уровня. Стрелка, связывающая модули, указывает направление процесса вызова.

Иерархии модулей создаются в соответствии с единообразными параметрами типа «отношение вызывает» или «имеет компонент, именуемый».

В рамках иерархий вызовов модули выполняют часть своих задач с помощью собственного программного кода; остальная часть реализуется путем вызова функций других модулей *(Lockemann, Dittrich. Architektur von Datenbanksystemen. 1987, с. 102).*

В иерархии компонентов только «листья» иерархии модулей описаны инструкциями. Таким образом, зависимости вызовов (обращений) в иерархии компонентов не всегда очевидны. Обзор различных этапов разбиения модулей дан в работе *Lockemann, Dittrich. Architektur von Datenbanksystemen. 1987, с. 103.*

На рис. 40 представлена классификация модулей при помощи связи 1:* между классами МОДУЛЬ и ТИП МОДУЛЯ с разбивкой на модули манипулирования данными, модули обработки данных и оперативные модули.

Отношения между модулями характеризуются связью КОММУНИКАЦИЯ. Класс ТИП КОММУНИКАЦИИ характеризует тип связи (например, простые связи данных или управляющие связи). Обмениваемые данные идентифицируются по атрибуту «имя данных». Хотя каждый коммуникационный набор может передавать только дату, этим наборам можно присваивать номера элементов. Для этой цели ассоциативный класс связывается с классом ЭЛЕМЕНТ.

Структурные диаграммы являются лишь одним из нескольких методов проектирования систем, однако разрабатываемые на их основе структуры классов носят настолько общий характер, что позволяют моделировать другие методы на базе той же логики (дополнительные языки спецификации приведены в работе: *Sommerville. Software Engineering. 1987, с. 77, 106).*

Помимо термина «модуль», мы можем также использовать понятие «программа». Вообще говоря, программы — это полные наборы инструкций, содержащие все необходимые требования для решения задач (Stetter. Softwaretechnologie . 1987, с. 12–16). Когда программы состоят из подпрограмм, взаимодействующих друг с другом, они образуют классы программ или прикладных систем. Подпрограммы, которые отвечают требованиям, предъявляемым к модулям, называются «модульными».

Степень детализации в проектах модулей зависит от типа обработки. Транзакции, предполагающие последовательные этапы обработки пользователем, предназначены для оперативной обработки. В зависимости от степени детализации проекта транзакции могут соответствовать нижней ступени иерархии в классификации бизнес-процесса (элементарная функция).

Количество классификаций в спецификации проекта зависит также от конкретных реализуемых информационных систем. Некоторые мониторы транзакций лучше справляются с обработкой множества мелких транзакций, другие — с обработкой более крупных транзакций, но в

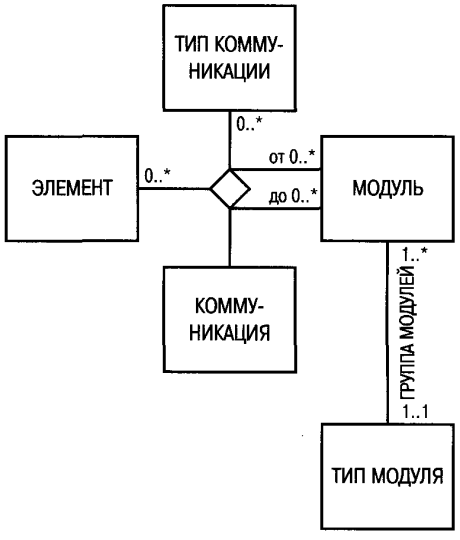


Рис. 40. Классификация модулей

меньшем количестве (Olle et al. Information Systems Methodologies. 1991, с. 256).

Однако при разработке спецификаций проекта влияние конкретных особенностей ИТ следует учитывать лишь до определенной степени.

A.2.1.3.2. Мини–спецификация

В проектах ИС содержимое модулей описывается посредством мини–спецификаций. К числу общеупотребительных методов относится описание управляющих структур (управляющих алгоритмическими процессами) и исполняющих инструкций при помощи псевдокода и структурограмм. В качестве управляющих структур могут служить последовательности, инструкции выбора и повторения. На рис. 41 приведена простая схема, иллюстрирующая использование структурограмм и псевдокода.

Инструкции состоят из вызовов процедур и модулей, а также арифметических операций. Последние выполняются на уровне элементов данных. В зависимости от сложности управляющие структуры в модулях могут вкладываться одна в другую. «Листья» модульной сети содержат наиболее детальное описание, в то время как на более высоких уровнях содержимое модулей обычно состоит из управляющих структур и вызывающих инструкций.

На рис. 42 приведена метамодель управляющей структуры, объединяющая классы ТИП УПРАВЛЯЮЩЕЙ СТРУКТУРЫ, УРОВЕНЬ ИЕРАРХИИ и МОДУЛЬ. При этом управляющая структура охватывает весь блок инструкций. Модули включают несколько управляющих структур, связанных с различными ступенями иерархии. Инструкции, принадлежащие управляющей структуре, обозначены связью 1:*.

A.2.1.3.3. Представление выхода

Требования к вводу и выводу определяются при описании дизайна и списков экрана. Примеры приведены на рис. 43 и 44.

Экраны можно использовать для целей ввода и вывода. Несколько модулей могут применять для ввода и вывода один тип экрана, поэтому на диаграмме классов, при-

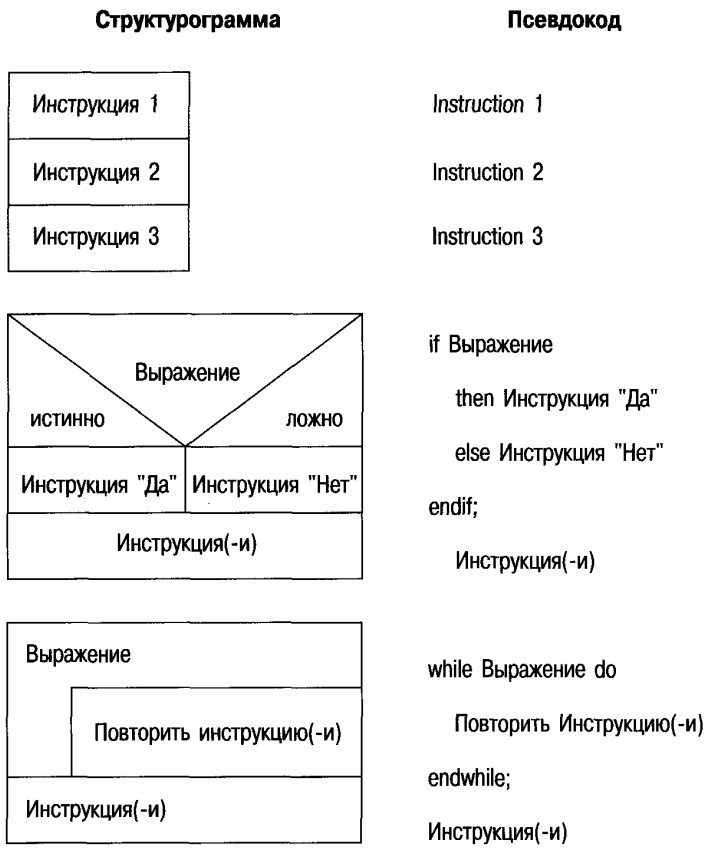


Рис. 41. Управляющие структуры

веденной на рис. 45, функции ввода и вывода характеризуются мощностью (0..*). Если типы экранов хранятся на местных языках, их конкретные экземпляры можно создавать путем различных комбинаций МЕСТНОГО ЯЗЫКА и ТИПА ЭКРАНА. Функциональные возможности Windows позволяют отобразить определенный ЭКРАН в рамках существующего экрана.

Экраны можно интерпретировать как представления модели данных. Подробнее мы раскроем эту тему, при обсуждении отношения между моделями данных и моделями функций. Говоря же о взаимоотношениях между функциональными и организационными моделями, мы рассмотрим экраны и списки с позиции тех, кто получает или вводит данные.

A.2.1.4. Реализация на уровне функциональной модели

На стадии описания реализации разрабатываются фактические программы, подлежащие выполнению. Для этой цели в соответствии со спецификациями модулей используется один или несколько языков программирования (например, Си, C++, Java, ABAP4, Кобол). Если изначальные спецификации достаточно детализированы, они могут быть реализованы генератором приложений. В этом случае связующим звеном между описанием модуля определения требований, языком программирования и утилитой служит МОДУЛЬ ИСХОДНОГО КОДА (см. рис. 46). Однако если программирование выполняется исключительно про-

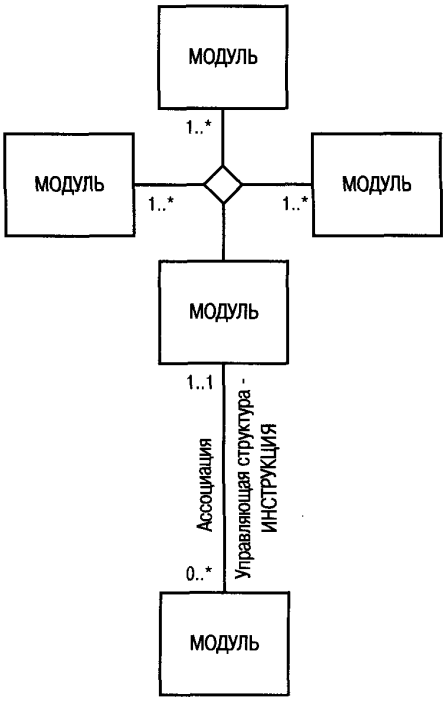


Рис. 42. Метамодель управляющей структуры

граммистами, то упоминать генератор приложений излишне.

Модули исходного кода могут храниться в библиотеке программ в рамках репозиториев. БИБЛИОТЕКИ ПРОГРАММ, где хранятся все существующие программы или модули, значительно повышают многократную применимость модулей. Библиотеки программ можно использовать для описания модулей и на уровне спецификации проекта. На рис. 46 представлена связь с описанием модулей на уровне спецификации проекта.

С помощью КОМПИЛЯТОРОВ или ИНТЕРПРЕТАТОРОВ модули исходного кода преобразуются в МОДУЛИ ОБЪЕКТНОГО КОДА. Для каждого ЯЗЫКА ПРОГРАММИРОВАНИЯ возможны несколько компиляторов или интерпретаторов, например, для каждой аппаратной платформы. Из одного модуля исходного кода можно получить разные модули объектного кода.

Вообще, для обработки целой задачи необходимо несколько модулей, скомпилированных в единую программу. Класс ПРОГРАММА на рис. 46, представляет собой структуру репозитория для хранения физических программ.

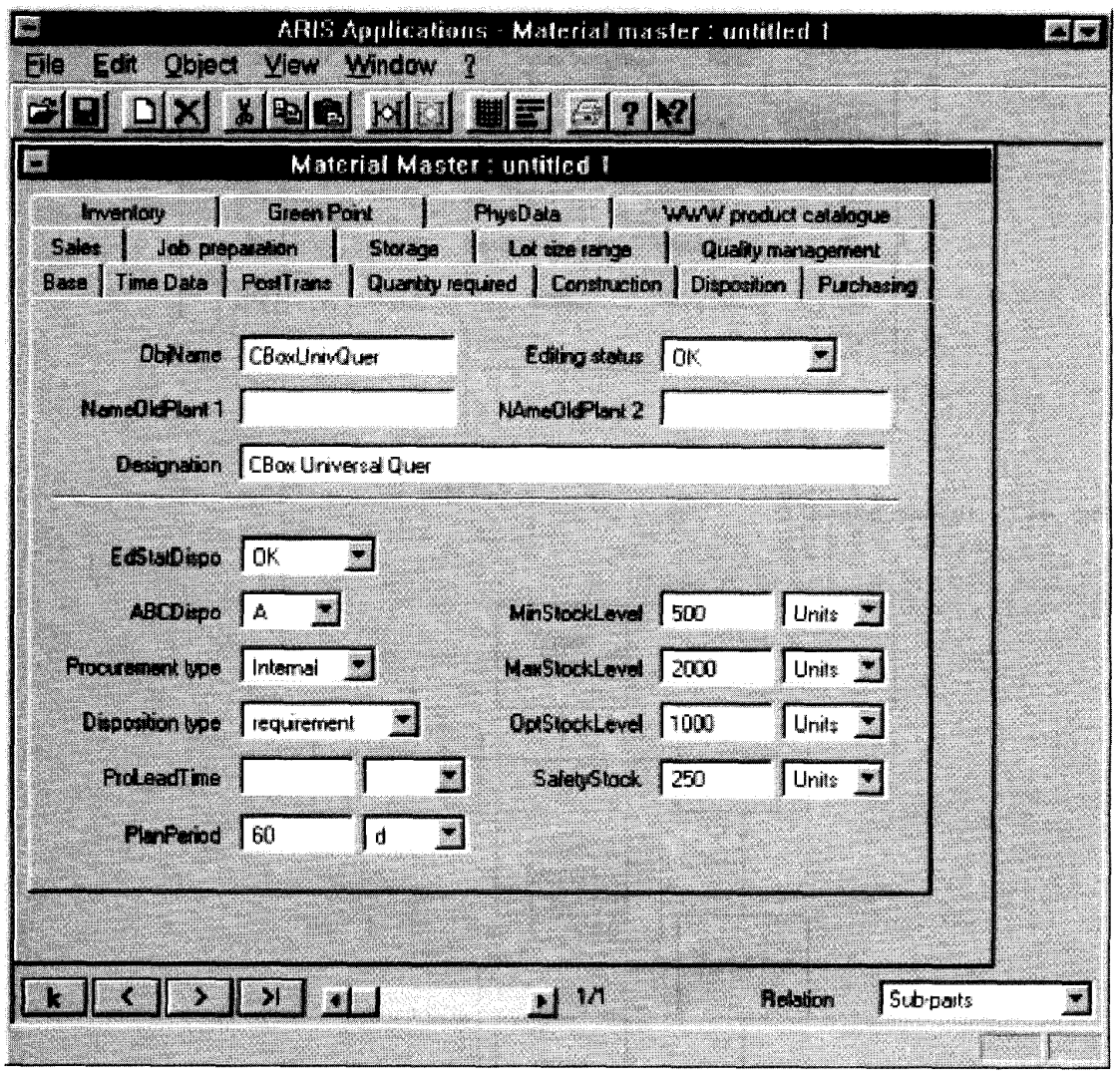


Рис. 43. Пример экрана

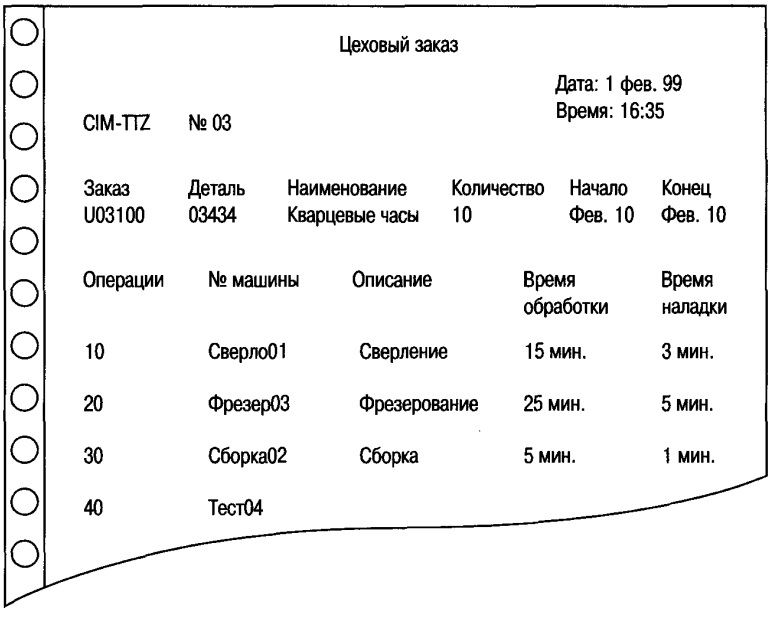


Рис. 44. Пример списка

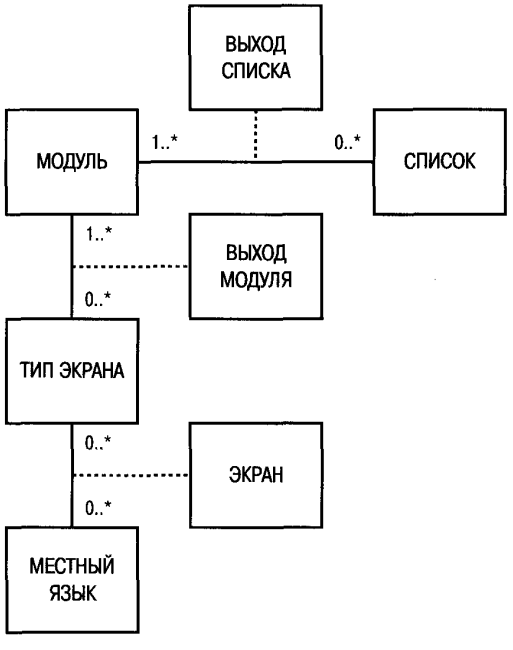


Рис. 45. Экраны и списки

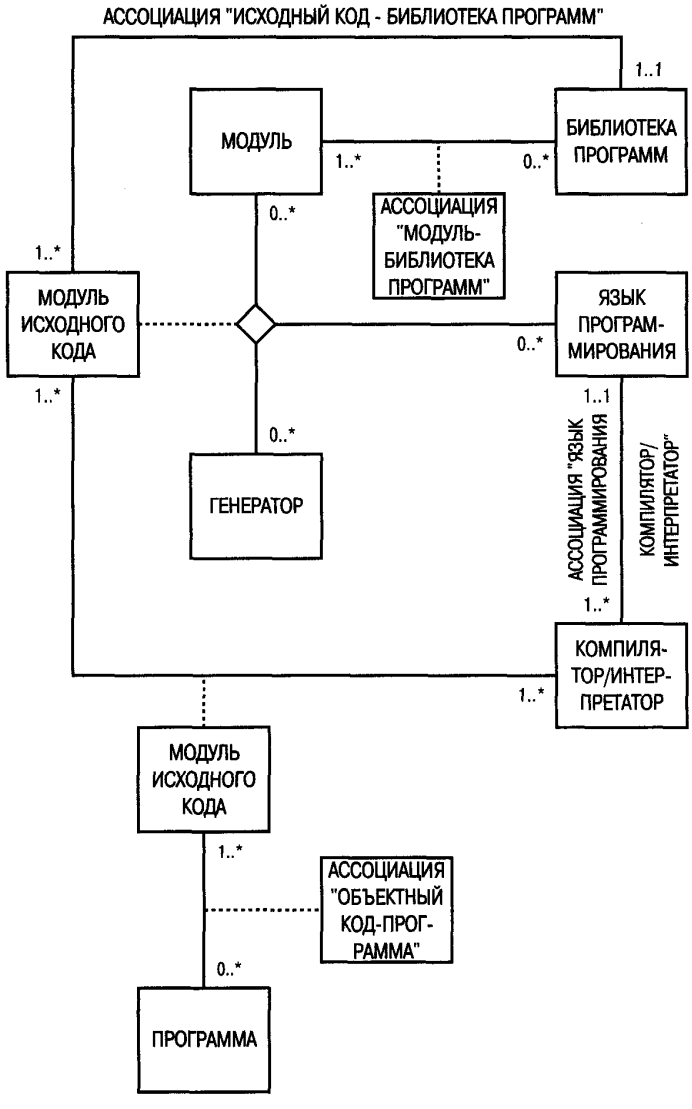


Рис. 46. Преобразование модулей в исходный код

A.2.2. Моделирование представления организации

На рис. 47 показано, какое место занимают в концепции ARIS этапы создания организационной модели. Блок, представляющий уже рассмотренную нами функциональную модель, заштрихован. Исходя из описания организационной иерархии на этапе спецификации проекта разрабатывается сетевая топология. Конкретные коммуникационные протоколы определяются на этапе описания реализации.

A.2.2.1. Определение требований на уровне организационной модели

Организационная модель описывает иерархическую структуру организации, т.е. организационные единицы, связанные между собой коммуникационными отношениями и отчетностью. Кроме того, ролевая концепция определяет профиль требований применительно к организационной единице, что особенно необходимо для приложений workflow.

В бизнес–приложениях иерархическая модель организации часто не имеет столь четкого описания, как данные, функции и процессы, «скрываясь» за такими понятиями, как «группа продаж», «код компании» или «производственный участок», с фиксацией конкретных субъектов ответственности в программах. Однако при реальном внедрении бизнес–приложения критическое значение приобретает согласование организационных структур.

A.2.2.1.1. Организационные структуры (иерархические организации)

Цель представления иерархической структуры организации состоит в том, чтобы упростить описание предприятия, объединив группы, выполняющие аналогичные задачи, в организационные единицы.

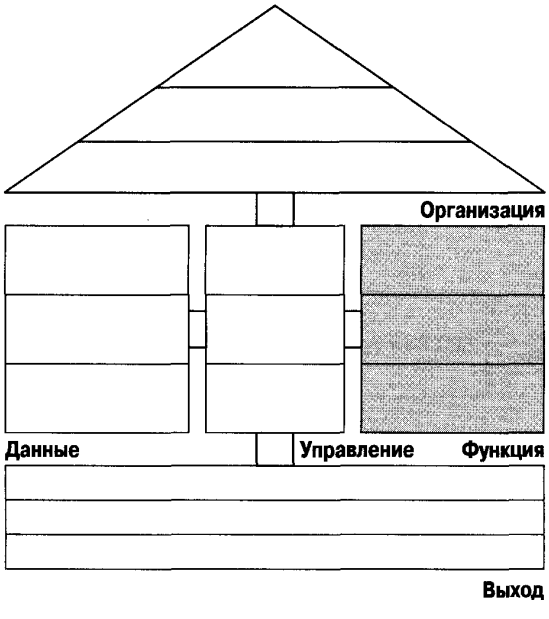


Рис. 47. Место организационной модели в здании ARIS

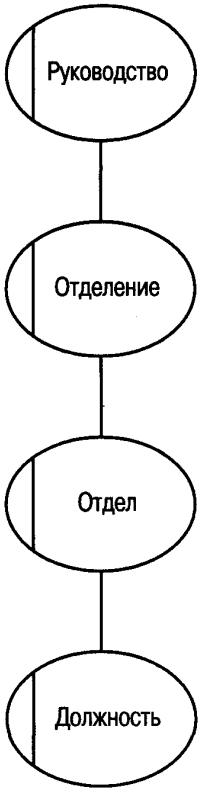


Рис. 48а. Органиграмма: уровень обобщенных типов

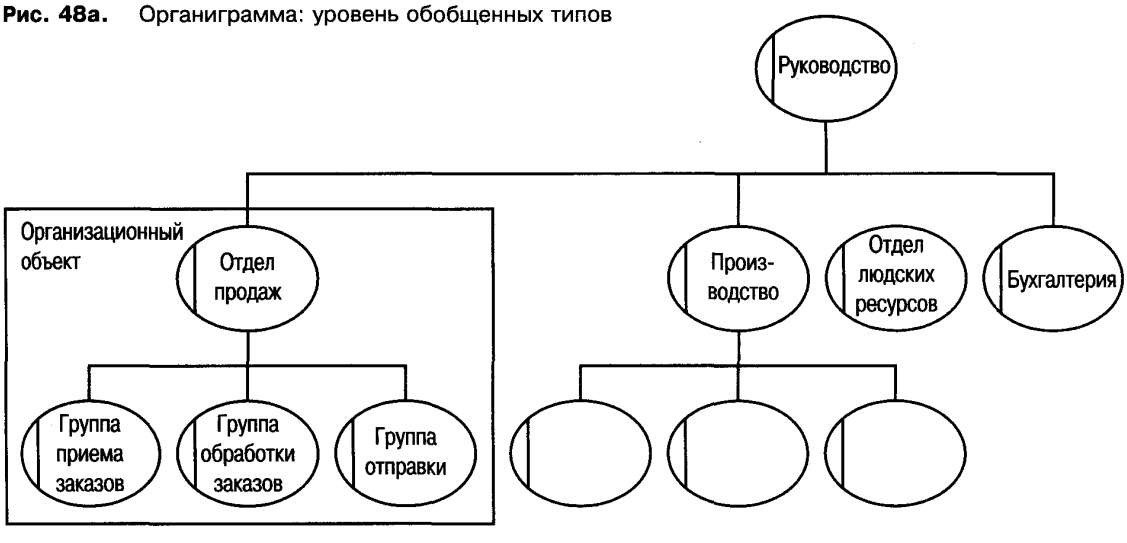


Рис. 48б. Органиграмма: уровень типов

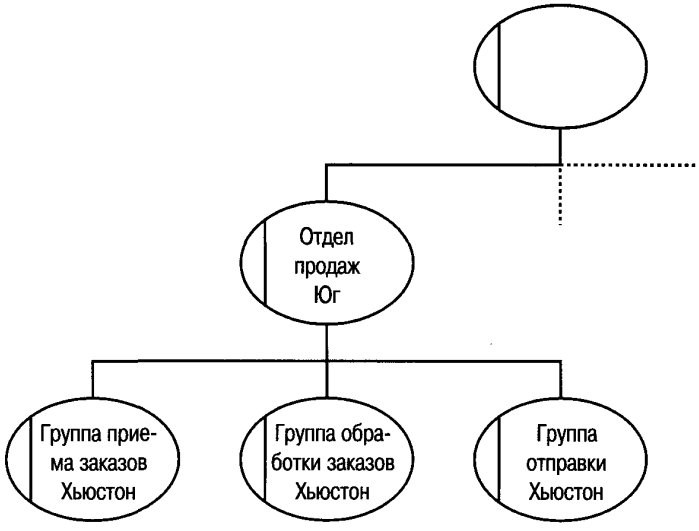


Рис. 48в. Органиграмма: уровень экземпляров

На рис. 48а–в приведены три примера органиграмм. На рис. 48а и 48б представление дается на уровне типов: на рис. 48а — в виде обобщенных понятий, на рис. 48б — в виде конкретных описаний, специфичных для приложения. Рис. 48в иллюстрирует конкретные экземпляры реальных организационных единиц.

При моделировании бизнес-процессов обычно используются понятия, фигурирующие в описании типов (например, отдел продаж, группа обработки заказов и т.д.). Однако если та или иная организационная единица занимает в бизнес-процессе особое место, можно моделировать и отдельные экземпляры (например, процедуры для конкретных производственных предприятий в Чикаго или Хьюстоне).

Как и функции, организационные единицы могут строиться в соответствии с ориентацией на операции, объекты или процессы. На рис. 48б представлена модель иерархической организации, построенная с ориентацией на операции.

На рис. 49 приведена модель, ориентированная на процессы. Эта модель содержит организационные единицы, связанные с бизнес-процессом логистики, который охватывает цепочку от клиента до производственного процесса. Организационные единицы, участвующие в этом бизнес-процессе, связаны отношением предшествующая-последующая.

Хотя обычно органиграммы представляют в виде древовидных индексов, здесь мы рассматриваем сетевые отношения, так как, например, один филиал по продаже может заниматься несколькими видами продуктов, а некоторые виды продуктов продаются через разные филиалы. Дополнительные сведения о процессах, связанных с поставками услуг *(Bullinger, Prozeßorientierte Strukturen 1994)*.

На метауровне моделируемые организационные единицы образуют класс ОРГАНИЗАЦИОННАЯ ЕДИНИЦА (см. рис. 50). В роли экземпляров обычно используются понятия, заимствованные из описания типов, хотя допустимы и реальные экземпляры.

Помимо человеческого вклада, можно структурировать и машинный вклад, а затем ввести его в такие организационные единицы, как группы машин, центры обра-

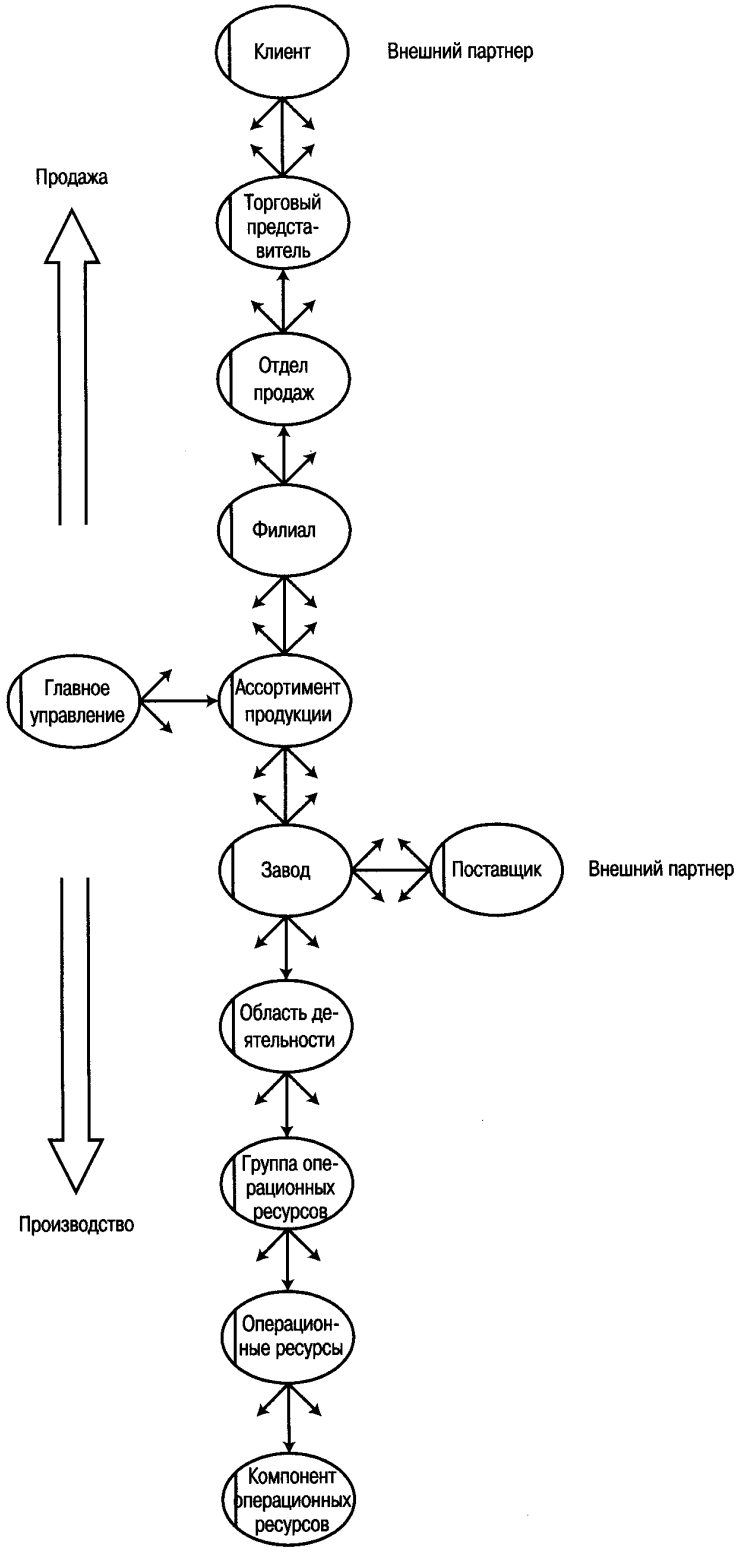


Рис. 49. Уровни планирования материалов, ориентированные на процессы

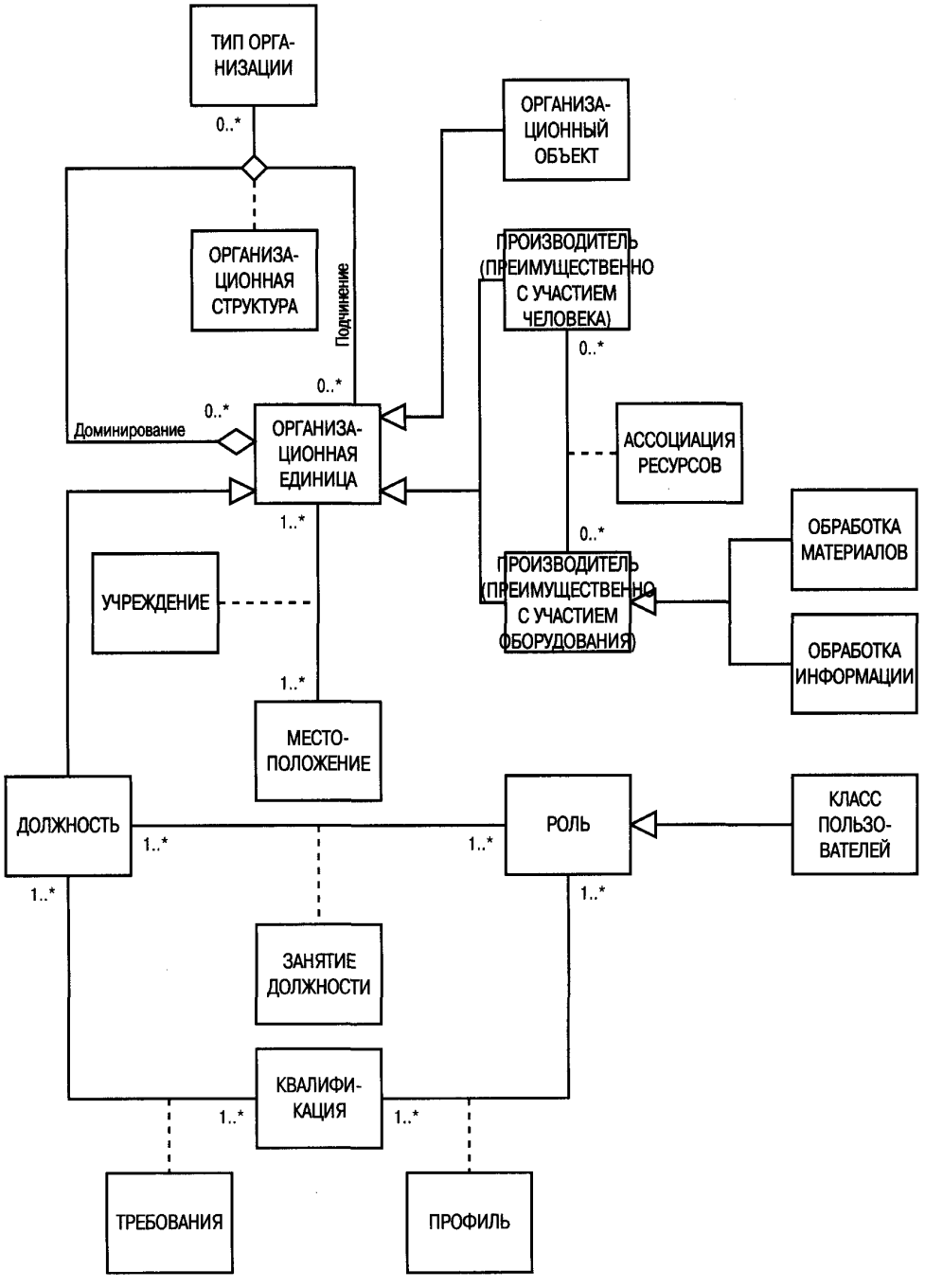


Рис. 50. Метамодель иерархической организации

ботки, складские системы или гибкие производственные системы, а в случае информационных ресурсов — в центры данных, рабочие станции и сети. Это позволяет привести структуру организационной модели в соответствие со структурой людских и технических ресурсов. В метамоделях это выражается с помощью подклассов ПРОИЗВОДИТЕЛЬ ВЫХОДА (ПРЕИМУЩЕСТВЕННО С УЧАСТИЕМ ЧЕЛОВЕКА) И ПРОИЗВОДИТЕЛЬ ВЫХОДА (ПРЕИМУЩЕСТВЕННО С УЧАСТИЕМ ОБОРУДОВАНИЯ). Последний разбивается на ОБРАБОТКУ МАТЕРИАЛОВ и ОБРАБОТКУ ИНФОРМАЦИИ (компьютеры). Поскольку организационные единицы можно создавать одновременно с учетом производителей выходов, эти связи отображают отношением АССОЦИАЦИЯ РЕСУРСОВ с добавлением соответствующего ассоциативного класса. Это позволяет связать ту или иную производственную систему с заводом (цехом), а компьютер — с отделом продаж. Экземплярами класса ОРГАНИЗАЦИОННАЯ ЕДИНИЦА являются также внешние деловые партнеры, такие как клиенты, поставщики или государственные учреждения.

Территориальное распределение организационных единиц реализуется отношением РЕГИОНАЛЬНЫЙ ОФИС между классами МЕСТОПОЛОЖЕНИЕ или ОРГАНИЗАЦИОННАЯ ЕДИНИЦА. Мельчайшей единицей в рамках организационной структуры является «должность», которая обычно описывается функцией, возлагаемой на конкретного сотрудника. Класс ДОЛЖНОСТЬ включает конкретные должности в виде экземпляров.

Должности связываются с более крупными единицами в соответствии с параметрами оргструктуры или структуры отчетности. Для того чтобы удовлетворять этим параметрам, необходимо ввести класс ТИП ОРГАНИЗАЦИИ. Помимо разделения на категории бизнеса и категории отчетности, можно уточнить возможности взаимного замещения между должностями, введя описания взаимоотношений и модели, ориентированные на процессы. Структурные отношения между организационными единицами описываются классом ОРГАНИЗАЦИОННАЯ СТРУКТУРА.

A.2.2.1.2. Ролевая концепция

Помимо организационных единиц, при моделировании цепочек процессов в категориях, связанных с бизнесом, описываются типы работников, например, специалисты по продаже, бухгалтеры, операторы машин, специалисты по закупке. Однако функции редко привязывают к конкретным индивидуумам, поскольку в этом случае при переводе сотрудника на другую должность или его уходе из компании пришлось бы вносить изменения в описание бизнес–концепции. Термин «роль» служит для описания типа работника, обладающего вполне определенной квалификацией и навыками *(Galler. Vom Geschaftsprozeßmodell zum Workflow–Modell. 1997, с. 52–58; Rupietta. Organisationsmodellierung. 1992; Esswein. Rollenmodell der Organisation 1992).*

Квалификационные параметры вводятся в класс КВАЛИФИКАЦИЯ и привязываются к РОЛИ посредством ассоциативного класса ПРОФИЛЬ (см. рис. 50). Например, роль «инженер по продажам» будет включать следующие квалификационные критерии: диплом по промышленному инжинирингу и практический опыт работы в области продаж. С точки зрения должности, к квалификации могут предъявляться определенные требования. Некоторые роли могут связываться с должностями по принципу «хорошего соответствия».

При описании ролей можно также дифференцировать классы пользователей - тех, кто проектирует ИС, и тех, кто ими пользуется. Позже такие классы пользователей могут служить для определения привилегий доступа к данным и функциям. В зависимости от навыков и частоты использования ИС можно выделить следующие категории:

- эпизодические пользователи;
- профессиональные пользователи;
- эксперты

(Martin. Application Development. 1982, с. 102–106; Davis, Olson. Management Information Systems. 1984, с. 503–533). Для конкретизации классов пользователей мы введем специальное отношение КЛАСС ПОЛЬЗОВАТЕЛЕЙ.

На прикладном уровне организационные метамодели аналогичны структуре данных систем управления персоналом *(Scheer. Business Process Engineering 1994, с. 491).* Однако хотелось бы отметить, что системы управления персоналом ставят в фокусе внимания конкретных индивидов, тогда как ролевые концепции описывают только типы работников, т.е. определенные классы. Кроме того, системы управления персоналом предполагают гораздо более дифференцированное описание фактов.

A.2.2.2. Конфигурирование организационной структуры

Организационные модели представляют описания стоимостных центров в терминах параметров стоимостного учета, позволяя конфигурировать управление бизнес–процессом. Описание организационных единиц как отделов, рабочих групп, секторов компании и т.д. позволяет адаптировать общие системы планирования (например, MS Project) к конкретным приложениям.

Рассмотренные нами метаструктуры могут быть использованы для моделирования рабочих потоков в рамках бизнес-процессов. Описание ролей в этом контексте является ключевым, так, как для реализации каждой функции необходимо сопоставить те или иные роли, которые, в свою очередь, найдут воплощение в конкретных исполнителях в реальном времени. Кроме того, существенное значение имеют описания замещений. Иногда целесообразно использовать экземпляры организационных единиц, например, указывать не роли, а конкретных сотрудников.

В бизнес-приложениях организационные понятия должны быть точно сформулированы и документированы, в соответствии с их связью с приложением и влиянием на программные процедуры.

Кроме того, в бизнес-приложениях организационные модели описывают такие важные параметры, как клиенты, коды компаний и заводы (цеха). Это закладывает фундамент для последующей привязки функций и данных к организационным единицам, определяя, таким образом, степень распределенности прикладной системы.

В приложениях для управления персоналом организационные модели определяют основу для всех функций планирования и учета. В сфере учета они предоставляют коды компаний и структуру стоимостных центров.

A.2.2.3.Спецификация проекта на уровне организационной модели

Рассматривая определение требований на уровне организационной модели, мы описали организационные единицы предприятия, включая их взаимоотношения. Спецификация проекта дополняет организационную модель бизнеса в топологию информационных и коммуникационных систем. В частности, определяются топо-

логии сетей, требуемые мощности, типы доступа пользователей к конкретным узлам и имеющиеся типы компонентов.

A.2.2.3.1. Топология сети

На рис. 51 представлена типичная конфигурация сети на промышленном предприятии, использующая такие сетевые топологии, как звезда, кольцо и шина (см. рис. 52).

Помимо топологии с присущими ей свойствами, определяющими безотказность, быстродействие и доступ к сети *(Hutchinson, Mariani. Local Area Networks. 1985; Sikora, Steinparz. Computer & Kommunikation. 1988; Sloman, Kramer. Verteilte Systeme und Rechnernetze. 1988; Tannenbaum. Computer Networks. 1988; Kauffels. Lokale Netze. 1997; Taylor. Network Architecture Design Handbook. 1997)*, сети можно характеризовать с других точек зрения, например, дифференци-

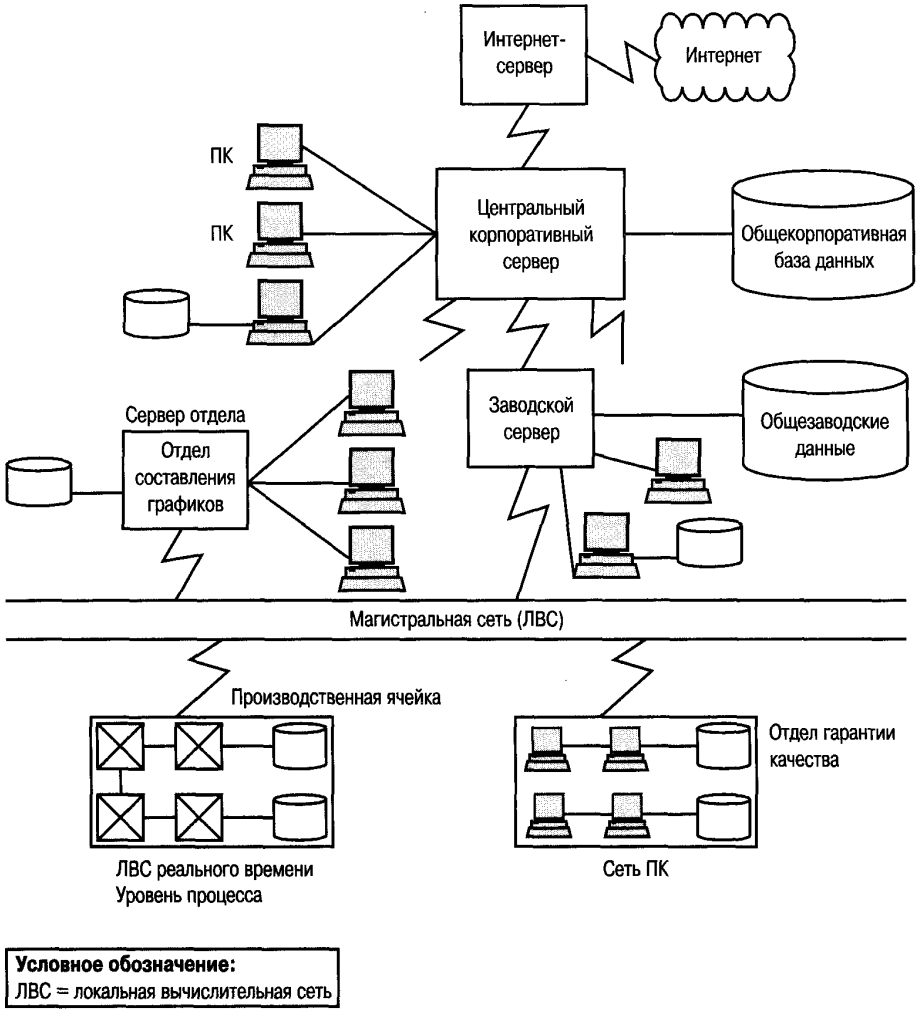


Рис. 51. Сетевая конфигурация

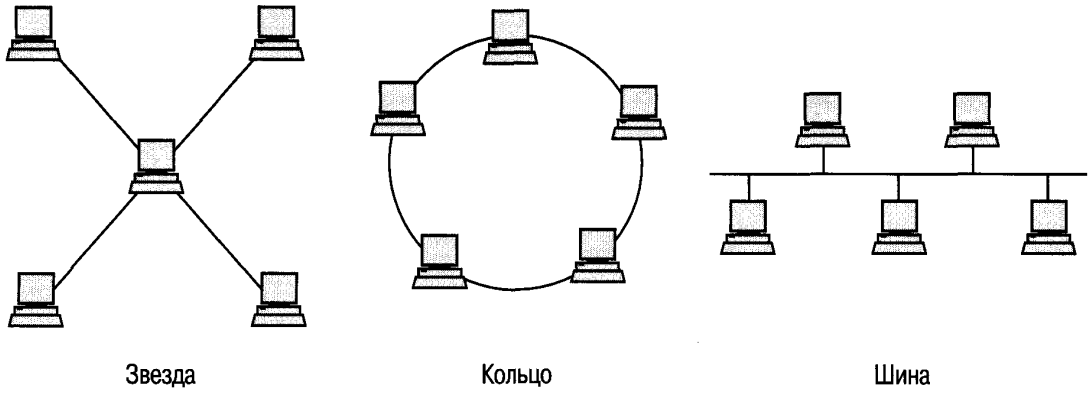


Рис. 52. Сетевые топологии

ровать глобальные вычислительные сети (ГВС), соединяющие удаленные друг от друга места, и локальные вычислительные сети (ЛВС), соединяющие узлы, сосредоточенные в одном месте.

В зависимости от типа подключаемых устройств можно провести дальнейшую дифференциацию между терминальными сетями, соединяющими терминалы («тонкие» клиенты) с серверами, и сетями, соединяющими интеллектуальные рабочие станции («толстые» клиенты).

«Хребет» магистральной сети образует соединение между сетями, обладающими различными свойствами. Одним из ключевых свойств является пропускная способность в реальном времени, характеризуемая такими параметрами, как скорость доставки и управление прерыванием для доступа к сети.

Другим, более специальным - свойством является ПРОТОКОЛ, выбираемый для конкретной сети. Существует несколько Интернет-протоколов, ориентированных на приложения, например, SMTP, FTP и HTTP, а также ряд протоколов ISO/OSI, например, X.400. Для доступа к данным подходят такие протоколы, как передача маркера и CSMA/CD.

Свойства сетевой архитектуры можно задавать независимо от конкретных аппаратных продуктов или управляющего программного обеспечения.

Для дифференциации различных типов сетей вводится класс ТИП СЕТИ, представленный на рис. 53. Этот класс можно разбить на подклассы в зависимости от топологии и типа протокола.

Для конкретных сетей мы предлагаем класс СЕТЬ.

На стадии спецификации проекта рассматривается только логическая схема сети без конкретизации физических сред (оптоволоконные кабели, коаксиальные кабели или радиопередача). Эти вопросы уточняются на стадии описания реализации. Поскольку понятие ТИП СЕТИ включает несколько классификационных параметров, одну и ту же сеть можно привязать к нескольким типам.

В описании сетей применяются понятия «узел» и «ребро». Местонахождение сетевого узла характеризуется понятием МЕСТОПОЛОЖЕНИЕ. Термином (сетевой) УЗЕЛ обозначается связь между МЕСТОПОЛОЖЕНИЕМ и СЕТЬЮ. Сети содержат от 1 до n узлов, хотя некоторые фрагменты сети могут включать 0 или n разных логических сетевых узлов.

Сетевые топологии описываются позиционными отношениями между узлами,

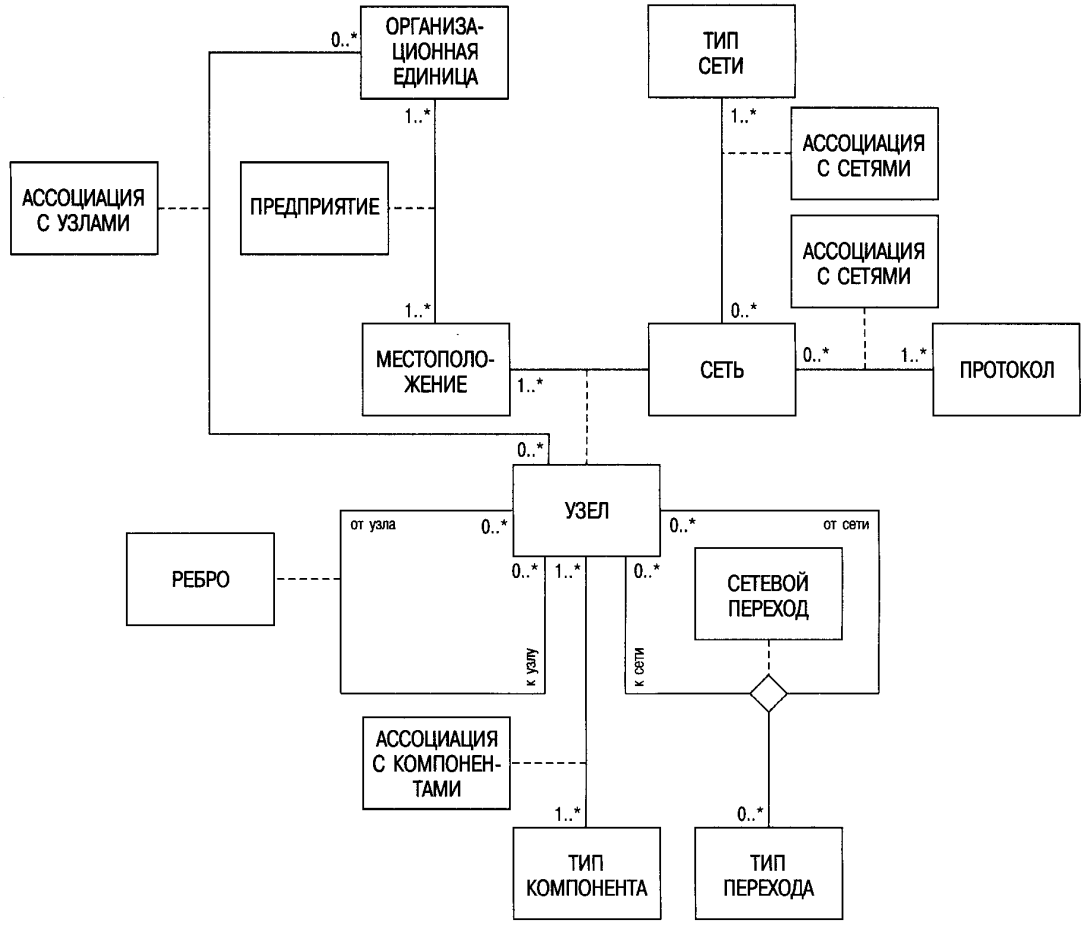


Рис. 53. Метамодель сетевой конфигурации

поэтому в рамках класса УЗЕЛ мы введем связь РЕБРО. От одного узла может исходить (и к нему может сходиться) от 0 до n узлов, при этом в сети с топологией «шина» 0 обозначает первый или конечный узел.

Если к РЕБРУ присоединяются в качестве атрибутов переносимые величины, имеет смысл дифференцировать направление ребер. Это делается с помощью ролевых имен «от узлов» и «к узлам».

Ключевые группы передаваемых данных (например, данные об изделии или наряд-заказы) можно перечислять в качестве атрибутов.

Сети, описываемые для предприятия, обычно не изолированы, а связаны друг с другом. Это иллюстрирует рис. 51, где показаны различные формы перехода между сетями в зависимости от уровня сетевого протокола. Так называемые шлюзы передают на каждом уровне сетевого протокола (например, все семь уровней модели-прототипа ISO/OSI) от протокола одной сети к протоколу другой. Если же несколько уровней (обычно верхних) совпадают (так что требуется передавать только протоколы нижних уровней), используются маршрутизаторы и мосты. Эти типы соединений представлены на рис. 53 классом ТИП ПЕРЕХОДА.

Переход между сетями осуществляется путем соединения узла одной сети с узлом другой. Он представлен связью СЕТЕВОЙ ПЕРЕХОД, описание которого включает ТИП ПЕРЕХОДА (шлюз, маршрутизатор и т.д.).

Определение требований на уровне организационной структуры связывается с сетевой топологией путем переноса классов МЕСТОПОЛОЖЕНИЕ и ОРГАНИЗАЦИОННАЯ ЕДИНИЦА, фигурирующих в определении требований. Если субъект ответственности уже описан на узловом уровне, т.е. если несколько организационных единиц совместно используют один узел сети или если какой-либо узел, расположенный в определенном фрагменте сети, доступен только некоторым организационным единицам, обычно привязанным к этому микрорайону, то необходимо ввести АССОЦИАЦИЮ между УЗЛАМИ и ОРГАНИЗАЦИОННЫМИ ЕДИНИЦАМИ. Эта ситуация также отражена на рис. 53.

A.2.2.3.2. Типы компонентов

До сих пор мы описывали узлы только с точки зрения их местоположения и принадлежности к той или иной сети. Новых аппаратных систем мы пока не касались.

Например, пока было неясно, идет ли речь о полной компьютерной системе, станции ввода, станции вывода, или, может быть, о децентрализованной рабочей станции с доступом к фоновым системам. Теперь мы введем понятие ТИП КОМПОНЕНТА, чтобы охарактеризовать предварительные типы устройств и определить, например, требуются ли нам комплексные компьютерные системы или на каком-то узле будет достаточно устройства вывода. На любом узле можно реализовать ряд компонентов разного типа. Характерным атрибутом типа компонента является описание.

Связь АССОЦИАЦИЯ С КОМПОНЕНТАМИ можно интерпретировать по-разному. Если описание узла настолько детализировано, что каждому узлу однозначно соответствует одно устройство (иными словами, каждая рабочая станция представляет собой отдельный узел сети), то мощность отношения с точки зрения узла равна (1..*). Это означает, что один узел соотносится с одним типом компонента. При этом один тип компонента может использоваться на нескольких узлах. Если, однако, узел рассматривается лишь как порт подключения в рамках сети и предназначен для использования несколькими устройствами (например, персональными компьютерами или устройствами вывода), то на одном узле можно установить несколько типов компонентов. В этом случае одним из атрибутов связи является количество устройств определенного типа, установленных на узле.

A.2.2.4. Реализация на уровне организационной модели

Реализация начинается с сетевой топологии спецификации проекта, представленной на рис. 54 в верхней части диаграммы классов. Физическая реализация сетей и узлов, описанных в спецификации проекта, может быть различной. Поэтому в спецификации проекта понятия, существующие в той же форме на уровне физической реализации, дополняются префиксом ЛОГ. (логический). Аналогично термины, относящиеся к уровню физической реализации, дополняются префиксом ФИЗ. (физический). Это указывает на то, что логически описанные компоненты теперь переносятся на физические объекты. На рис. 55 приведен пример гетерогенной сетевой архитектуры.

Предположим, некий сетевой протокол (например, TCP/IP) моделируется в физической сети на уровне реализации. Физическая сеть характеризуется определенным носителем (коаксиальный кабель, оптический кабель или телефонный ка-

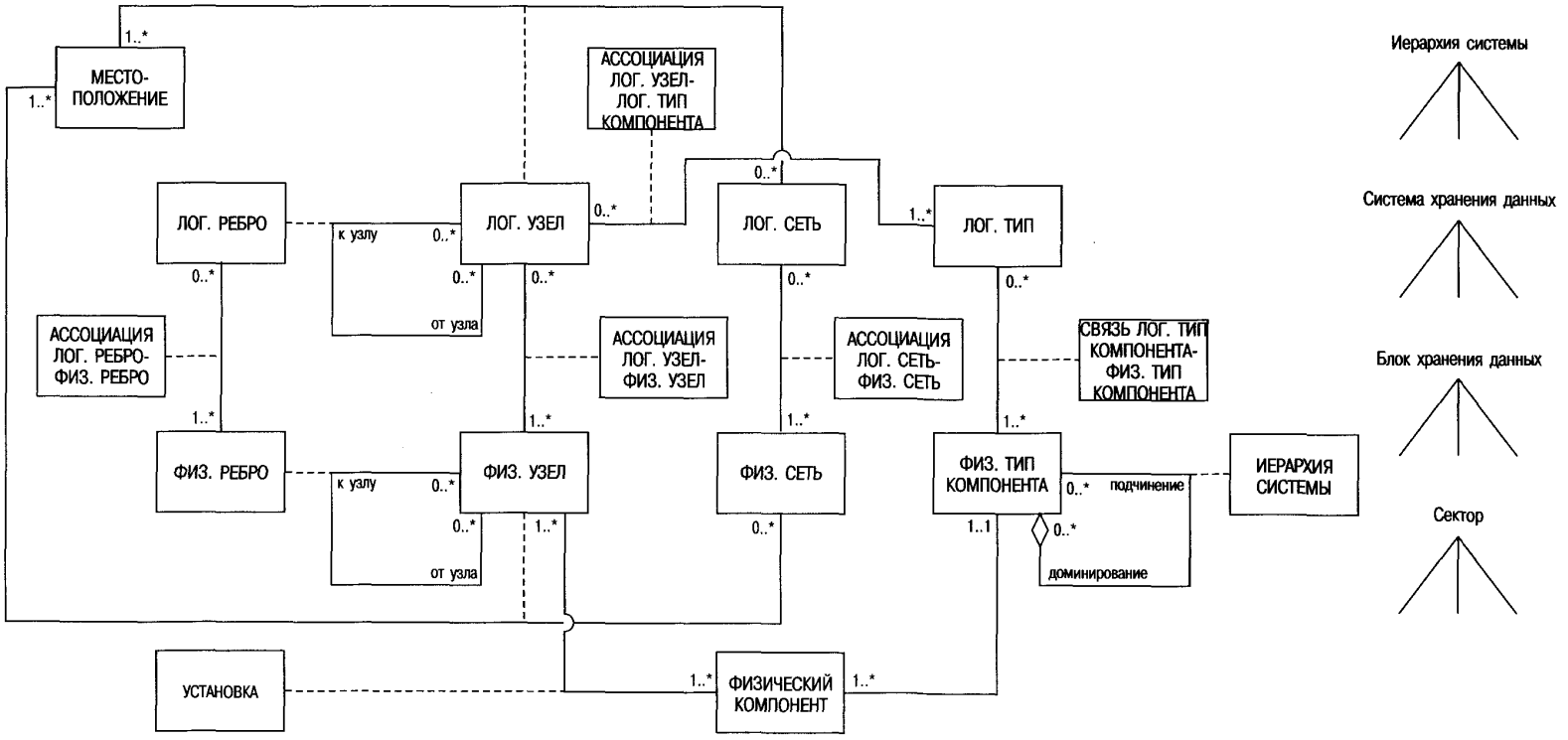


Рис. 54. Моделирование логических сетей и их физической реализации

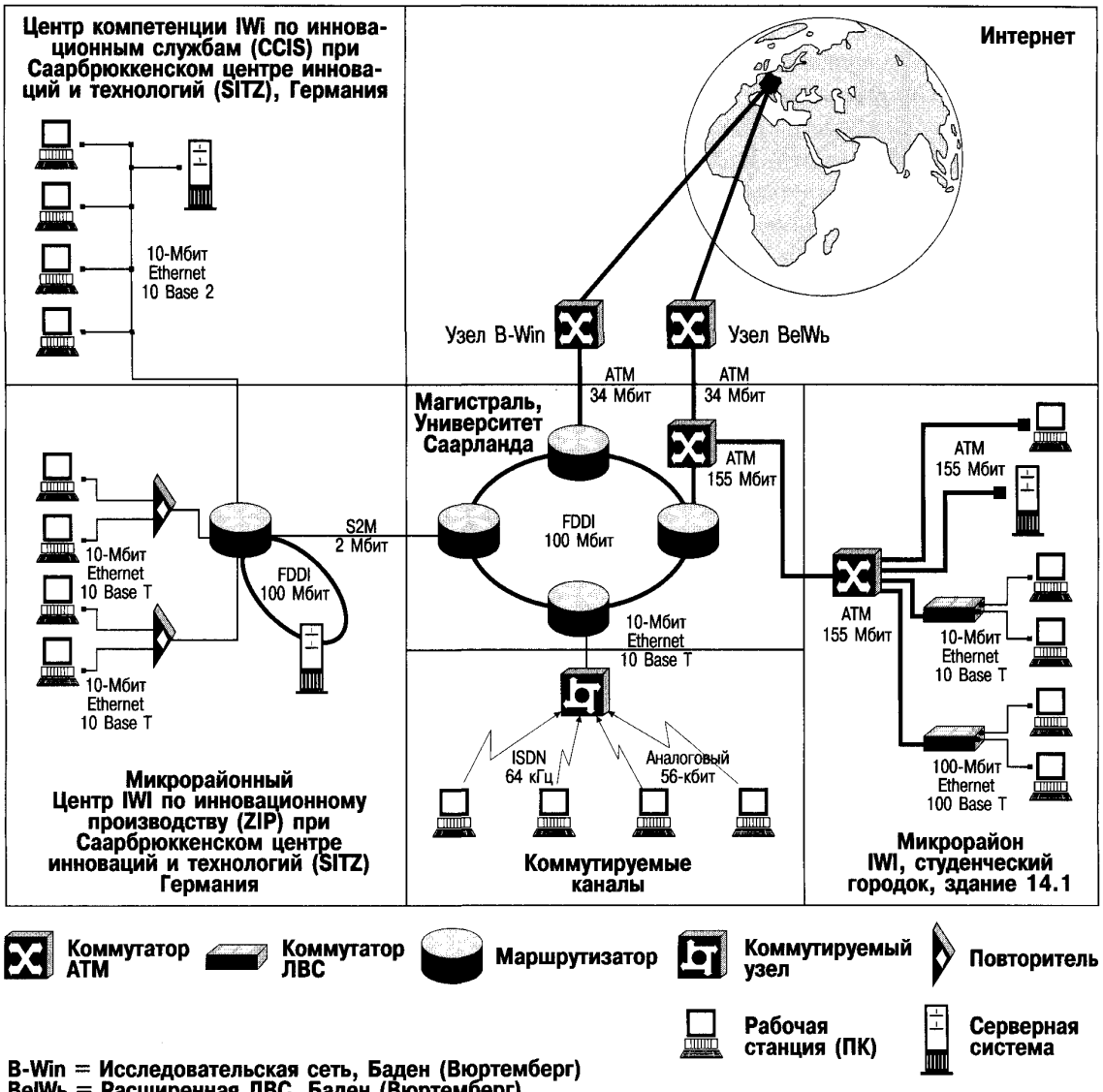


Рис. 55. Гетерогенная сетевая архитектура

бель типа «витая пара»). Между классами ЛОГ. СЕТЬ и ФИЗ. СЕТЬ, соответственно, возможна связь *:*. В одной физической сети можно смоделировать несколько логических сетей, а единую логическую сеть можно реализовать посредством нескольких взаимосвязанных физических сетей.

В соответствии с диаграммой классов на уровне спецификации проекта узел физической сети проектируется как связующее звено между классами ФИЗ. СЕТЬ и МЕСТОПОЛОЖЕНИЕ. Описание местоположения строится на основании проекта ИС, и при необходимости добав-

ляются конкретные экземпляры. Между понятиями «физический узел» (ФИЗ. УЗЕЛ) и «логический узел» (ЛОГ. УЗЕЛ) также существует отношение *:*, поскольку в одном логическом узле (с точки зрения спецификации проекта) можно реализовать несколько физических узлов.

В то же время возможны физические узлы, не связанные напрямую с каким-либо логическим узлом в спецификации проекта. Это бывает, например, когда физический узел является всего лишь техническим «усилителем» и к нему не привязаны какие-либо устройства или прикладные функции. Физические сети описываются с помощью понятий «физический узел» и «физическое ребро». Между классами «логическое ребро» (ЛОГ. РЕБРО) и «физическое ребро» (ФИЗ. РЕБРО) также создается связь *:*.

Однако мы хотели бы отметить, что для построения информационной модели эти отношения необязательны. Иногда достаточно описать связи между логической и физической сетями или между логическими и физическими узлами. Описания физических ребер необходимы только для целей реализации, при этом указывать связи с логическими ребрами необязательно.

Уровни устройств, т.е. компьютерные компоненты, характеризуются понятием ФИЗ. ТИП КОМПОНЕНТА, которое связано с уровнем спецификации проекта классом ЛОГ. ТИП КОМПОНЕНТА. Например, конкретное семейство устройств определенной марки и модели на логическом уровне «привязывается» к системе хранения данных.

Иерархическая связь между доминирующей системой и вложенными подчиненными системами представлена агрегацией ИЕРАРХИЯ СИСТЕМЫ, что позволяет «развертывать» сложные компьютерные системы или системы хранения данных в виде структуры, аналогичной прейскуранту материалов.

Различные физические компоненты каждого типа представлены классом ФИЗИЧЕСКИЙ КОМПОНЕНТ, который не только группирует их в типы, но и привязывает к определенному физическому узлу сети. Физические узлы можно связывать с несколькими физическими компонентами компьютерной системы. С другой стороны, один и тот же физический компонент (компьютер) можно связывать с несколькими сетями, т.е. физическими узлами.

Кроме того, программные компоненты системы можно рассматривать как аналоги аппаратно-ориентированных компонентов, что позволяет использовать соответствующее ПО для администрирования сетей. В данной работе мы не будем останавливаться на этих, в общем аналогичных, методах, поскольку исходим из предположения, что системное ПО является частью аппаратной платформы.

A.2.3. Моделирование на уровне представления данных

Модель данных включает описание объектов данных, которыми оперируют функции. Эти объекты данных, воспринимаемые последующими организационными единицами как информационные услуги, частично накладываются на модель выходов.

Объекты данных, описываемые на уровне определения требований, могут служить хорошей основой для описания класса объектно-ориентированного метода проектирования. На рис. 56 показано, какое место занимает модель данных в здании ARIS. Блоки, представляющие уже рассмотренные модели, заштрихованы.

A.2.3.1. Определение требований на уровне модели данных

Модель данных представляет различные объекты с разной степенью структурирования. Примеры таких объектов приведены на рис. 57. Для некоторых объектов показаны конкретные экземпляры. Моделирование бизнеса в основном сосредоточено на описании типов. Такие объекты, как <голос> и <система носителя>, типичны для макромодели, а объекты <тип сущности>, <атрибут> и <тип отношения>, будучи понятиями модели сущность-отношение, типичны для микромодели.

Применительно к данным термин <объект> имеет несколько значений. С одной стороны, он обозначает широкий диапазон типов документов, как показано на рис. 57. Однако понятие <объект> может

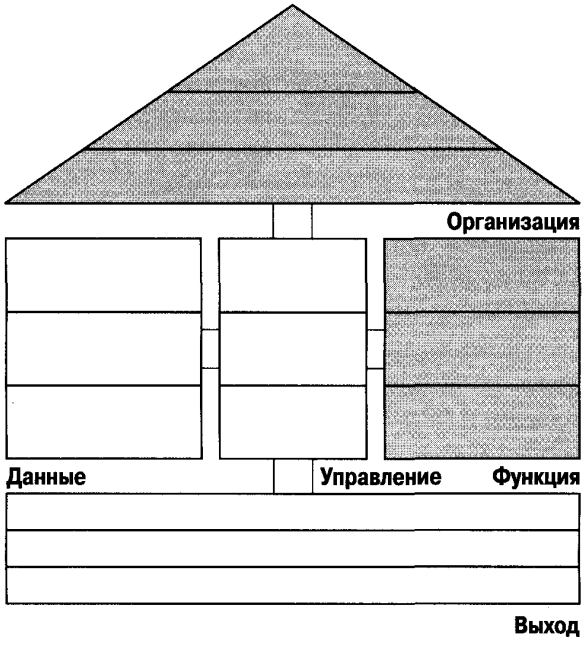


Рис. 56. Классификация модели данных в ARIS

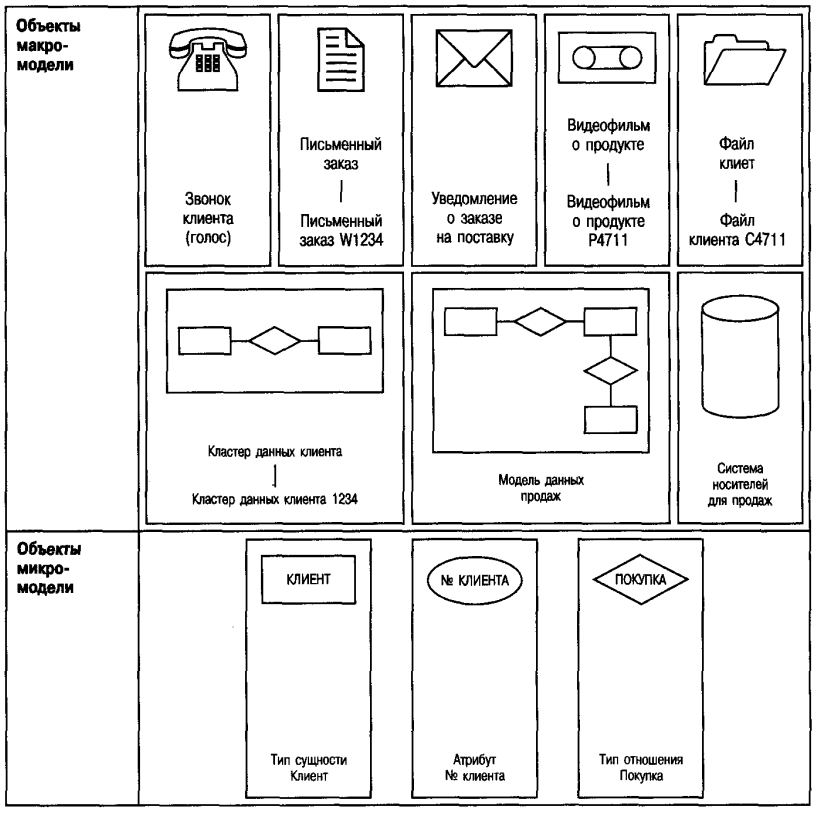


Рис. 57. Примеры типов данных

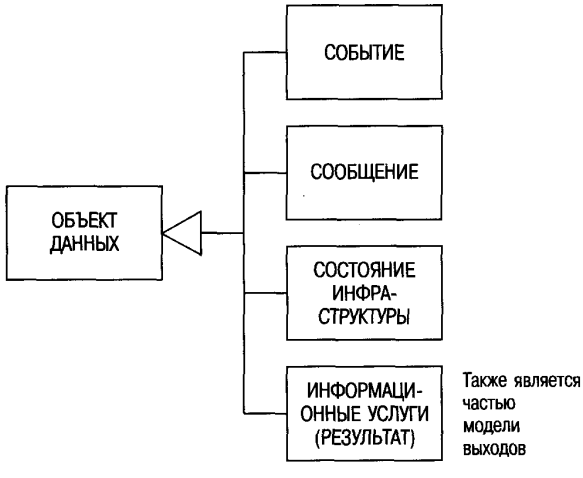


Рис. 58. Роли объектов данных

связываться и с системами управления объектно-ориентированных баз данных. Во избежание путаницы здесь иногда оговаривается, что речь идет именно об объекте данных.

Объекты данных, сведенные воедино в модели данных, выполняют различные роли (см. рис. 58). Они описывают события и сообщения, управляющие бизнес- процессом, т.е. поток управления. Кроме того, объекты данных отображают состояние среды (инфраструктуры), в которой протекает бизнес-процесс. Выход (результат) функций, обрабатывающих информацию, представлен документами и, таким образом, данными. Поскольку в ARIS выходные результаты описываются отдельно в модели выходов, эти типы представлений частично накладываются друг на друга.

Для начала мы рассмотрим метаструктуру макропредставления, а затем перейдем к микропредставлению.

A.2.3.1.1. Макроописание

Данные, которые можно разбить на более мелкие элементы (как в методе ERM), называются макроданными. Однако для описания данных, необходимых для бизнес-процесса, часто бывает целесообразнее и проще работать с предварительными объектами данных. Описание ERM можно отложить до стадии детального анализа.

В метамодели, показанной на рис. 59, термин МАКРООБЪЕКТ ДАННЫХ используется как общее обозначение совокупности данных.

Макрообъекты данных могут быть взаимосвязаны; например, файл клиента может содержать несколько типов писем (письменные заказы, напоминания об уплате, почтовые отправления и т.д.).

Корпоративные модели данных охватывают модели разных видов деятельности, состоящие из нескольких кластеров данных. Поскольку кластеры данных мо-

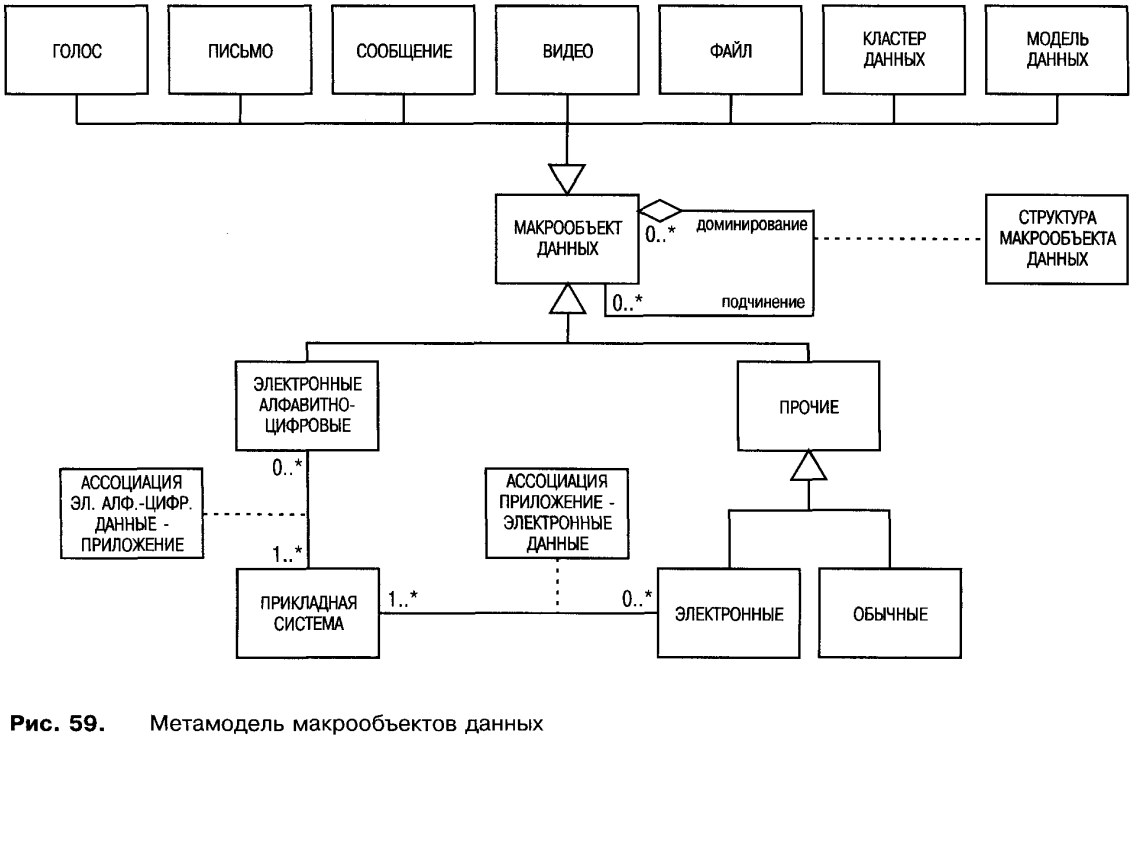


Рис. 59. Метамодель макрообъектов данных

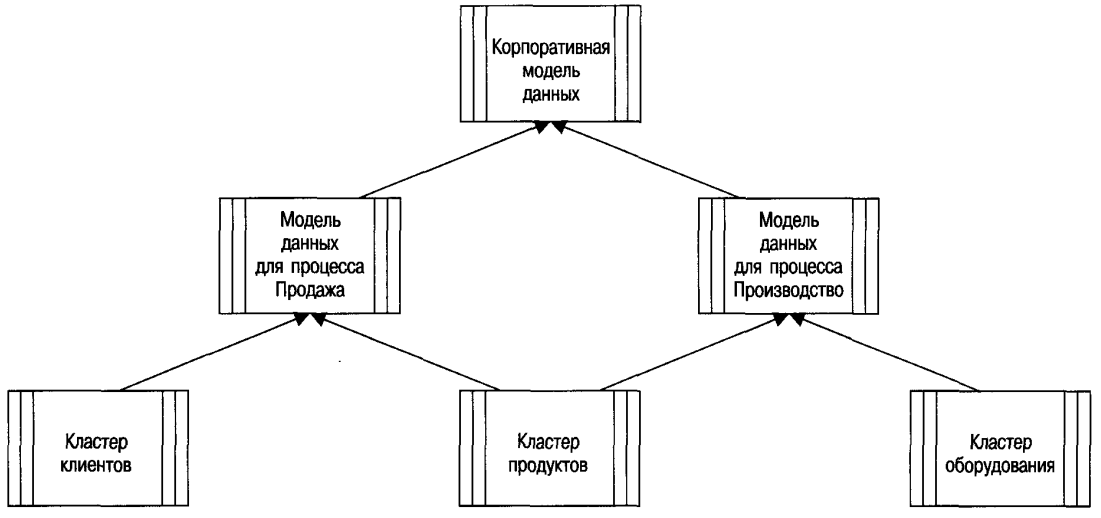


Рис. 60. Связь «:» между макрообъектами данных

гут быть составной частью нескольких моделей направлений деятельности компании, класс МАКРООБЪЕКТ ДАННЫХ характеризуется связью *:*, выражающей отношение «часть целого» (см. рис. 60).

Объекты данных могут быть электронными алфавитно-цифровыми или состоять из звуковых, битовых или обычных (бумажных) элементов. Поэтому класс МАКРООБЪЕКТ ДАННЫХ можно подразделить на ЭЛЕКТРОННЫЕ АЛФАВИТНО-ЦИФРОВЫЕ и ПРОЧИЕ, которые, в свою очередь, разбиваются на ЭЛЕКТРОННЫЕ и ОБЫЧНЫЕ элементы.

К объектам данных, хранящимся в электронной форме, например, к системам носителей, можно привязать прикладные системы. Здесь мы пользуемся только предварительными обозначениями, не вторгаясь в область спецификации проекта. Анализ ситуации «как есть» на стадии спецификации проекта позволяет специализировать системы и бизнес-приложения, используемые в текущий момент. Стратегические решения, касающиеся, например, внедряемых бизнес-приложений, находят отражение в целевых исследованиях.

A.2.3.1.2. Микроописания

Разбиение макрообъектов данных на более мелкие единицы позволяет получить описание на микроуровне. Детальную структуру данных для того или иного типа бизнес-приложения можно моделировать с помощью объектно-ориентированных диаграмм классов или методов ERM. Поскольку стандартным методом моделирования данных в рамках бизнес-процессов в реальных проектах является ERM, следующие несколько глав мы посвятим рассмотрению структуры репозитория для этого метода. Объектно-ориентированные диаграммы классов описываются в главе A.III.2.1.1.1 в контексте объектно-ориентированных моделей, связывающих представления данных и функций.

Элементы ERM изображаются как диаграммы классов UML. Сначала мы рассмотрим простую модель ERM, а затем добавим несколько графических операторов.

Простые модели ERM для структурирования данных бизнес-приложений состоят из типов сущностей и отношений, связанных друг с другом ребрами. Расширенные модели ERM дополнены указани-

ем мощности связей, операторами конкретизации/обобщения и реинтерпретацией типов отношений в типы сущностей. Мощности различных связей моделей UML и ERM иллюстрируются на рис. 3.

A.2.3.1.2.1. Простая модель ERM

Для начала рассмотрим фрагмент структуры данных, связанных с продажами, который приведен на рис. 61.

КЛИЕНТ, ИЗДЕЛИЕ и ВРЕМЯ представляют собой типы сущностей, связанные друг с другом типами отношений ПОКУПКА и ЗАКАЗ. Заказы можно идентифицировать как объекты данных «транзакция» по их связи с типом сущности ВРЕМЯ. Остальные элементы представляют эталонные данные.

Элементам присваиваются соответственно ключевые и описательные атрибуты. Мощность связи указывает на количество экземпляров типов отношений, допустимых для данного типа сущности.

На рис. 62 изображены классы ТИП СУЩНОСТИ и ТИП ОТНОШЕНИЯ, представляющие тип сущности и тип отношения в терминологии бизнеса. КЛИЕНТЫ, ВРЕМЯ, ИЗДЕЛИЯ, ПОКУПКА и ЗАКАЗ являются экземплярами этих классов.

В терминах бизнеса типы сущностей и типы отношений связаны ребром, поэтому на рис. 62 мы вводим класс РЕБРО. Количество допустимых экземпляров направления отношений определяется атрибутами РЕБРА. Поскольку в структуре данных тип сущности может быть связан с типом отношения несколькими ребрами, мощность этой связи равна (0..*). Когда в отношении участвует по меньшей мере две сущности, мощность равна (2..*).

Между типом сущности и типом отношения возможны несколько ребер с раз-

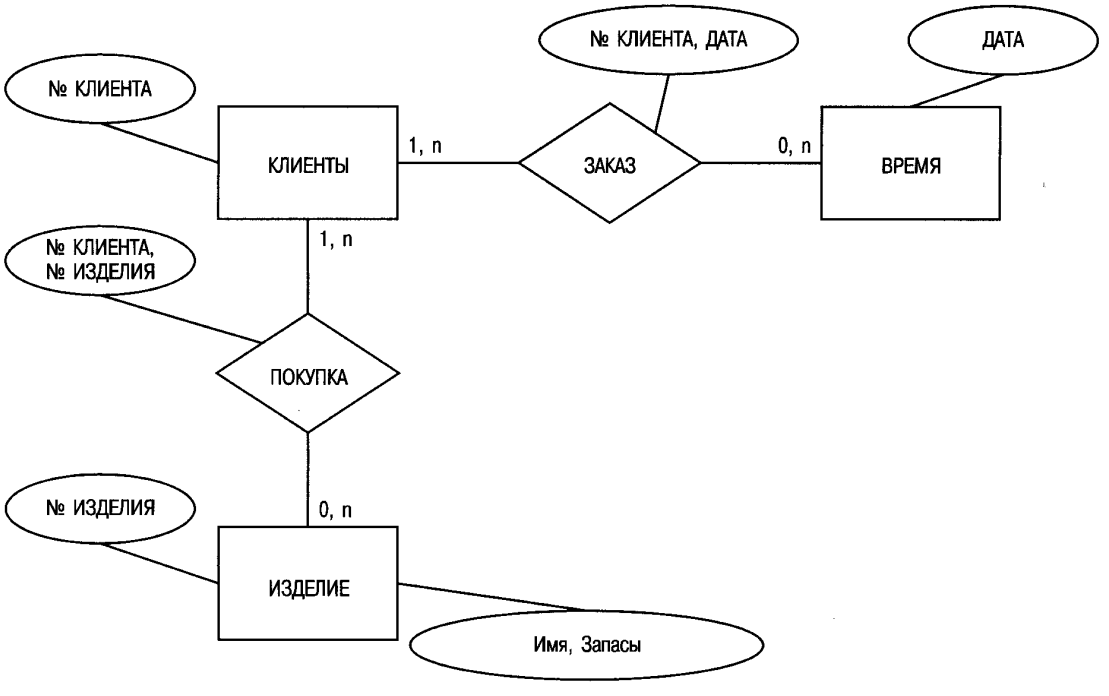


Рис. 61. Фрагмент модели ERM, описывающий структуру данных, связанных с продажами

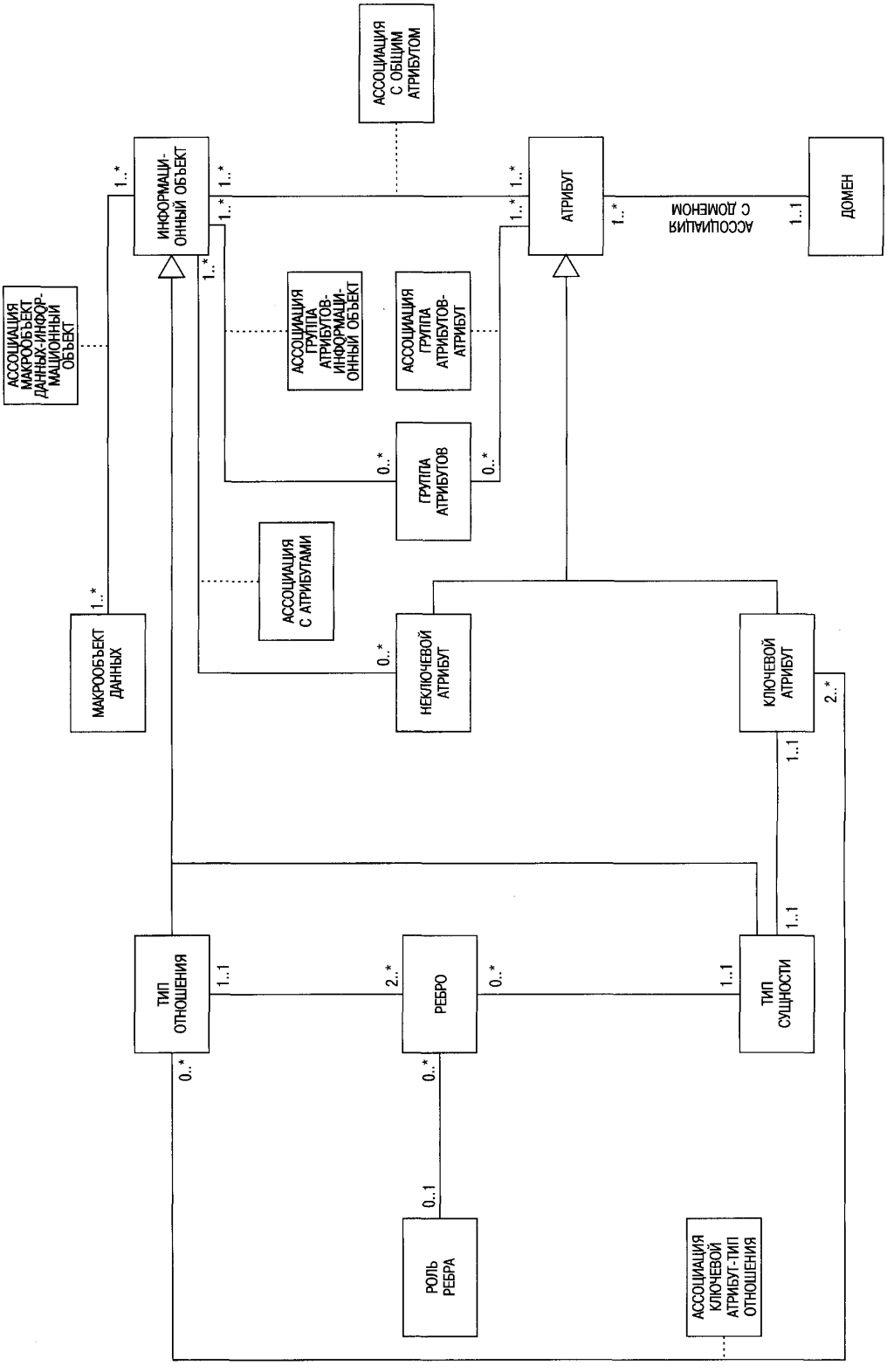


Рис. 62. Метамодель простого описания методом ERM

ными значениями, например, ребра, характеризующие доминирующую и подчиненную части в структуре прейскуранта материалов. В таких случаях мы можем однозначно идентифицировать то или иное ребро, введя класс РОЛЬ РЕБРА.

Каждому типу сущности присваивается идентифицирующий ключевой атрибут. Совокупность ключевых атрибутов образует класс КЛЮЧЕВОЙ АТРИБУТ. Между классом ТИП СУЩНОСТИ и классом КЛЮЧЕВОЙ АТРИБУТ существует связь 1:1. Следовательно, как минимальная, так и максимальная мощность этой связи равна 1. Например, типу сущности КЛИЕНТ в качестве идентификатора присваивается уникальный ключевой атрибут ь_КЛИЕНТА, а типу сущности ИЗДЕЛИЕ — уникальный ключевой атрибут ь_ИЗДЕЛИЯ.

Типы отношений идентифицируются по ключевым атрибутам связанных с ними типов сущностей. При этом отпадает необходимость вводить связь между классами ТИП ОТНОШЕНИЯ и КЛЮЧЕВОЙ АТРИБУТ, поскольку ключевые атрибуты уже соотносятся, хотя и в неявной форме, с типами отношений через связь между классами КЛЮЧЕВОЙ АТРИБУТ и ТИП СУЩНОСТИ. Однако для наглядности введем здесь избыточную связь АССОЦИАЦИЯ КЛЮЧЕВОЙ АТРИБУТ–ТИП ОТНОШЕНИЯ.

С классами ТИП СУЩНОСТИ и ТИП ОТНОШЕНИЯ связываются атрибуты, поэтому эти классы объединяются в общий класс ИНФОРМАЦИОННЫЙ ОБЪЕКТ, который создает связь с МАКРООБЪЕКТАМИ ДАННЫХ. Все типы сущностей и типы отношений, включая ребра, привязываются к макрообъекту данных (например, к модели данных маркетингового отдела). Это аналогично привязке описательных колонтитулов или фрагментов к документам или видеоданным.

Теперь, когда типы сущностей и типы отношений сконструированы, перейдем ко второму этапу: описанию и присвоению неключевых атрибутов. Введенный нами ранее класс КЛЮЧЕВОЙ АТРИБУТ является частным случаем общего класса АТРИБУТ и может быть разбит на классы КЛЮЧЕВОЙ АТРИБУТ и НЕКЛЮЧЕВОЙ АТРИБУТ.

Неключевые атрибуты связываются с типом ИНФОРМАЦИОННЫЙ ОБЪЕКТ отношением (1..*):(0..*). Это означает, что информационный объект может иметь множество неключевых атрибутов, как обычно и бывает в реальных ситуациях. Кроме того, один и тот же атрибут может быть связан с рядом информационных объектов. Например, атрибут <имя> можно присвоить как информационному объекту <клиент>, так и информационному объекту <поставщик>. Избыточное звено АССОЦИАЦИЯ С ОБЩИМ АТРИБУТОМ между классами АТРИБУТ и ИНФОРМАЦИОННЫЙ ОБЪЕКТ охватывает как ключевые, так и неключевые признаки.

Атрибуты, связанные по содержанию, можно объединить в группы. Например, группа атрибутов <адрес> может содержать такие атрибуты, как название улицы, номер дома, почтовый индекс и город. Можно создавать частично совпадающие группы атрибутов. При этом между классами АТРИБУТ и ГРУППА АТРИБУТОВ формируется связь (1..*):(0..*). Таким образом, в любой группе атрибутов должен присутствовать как минимум один атрибут, но совсем не обязательно, чтобы каждый атрибут входил в какую-то группу. Информационные объекты могут связываться с группами атрибутов и напрямую.

Набор значений атрибута характеризуется классом ДОМЕН. С каждым атрибутом можно связать только один домен. Например, точно так же, как в словаре, атрибут <имя> может охватывать всю совокупность имеющихся в домене имен и задавать диапазоны числовых значений.

Связь (1..*):(0..*) между АТРИБУТОМ и ИНФОРМАЦИОННЫМ ОБЪЕКТОМ по-

Ри

зволяет практически устранить избыточность в администрировании атрибутов и доменов.

A.2.3.1.2.2. Расширенная модель ERM

До сих пор мы говорили о простой модели ERM. Теперь перейдем к расширенному варианту, включающему следующие дополнения:

- реинтерпретацию типов отношений в типы сущностей;
- конкретизацию / обобщение сущностей;
- создание сложных объектов из типов сущностей и типов отношений.

Более детальная спецификация мощностей путем описания диапазонов значений формирует атрибуты ассоциативного класса РЕБРО и ничего не дает для улучшения информационной модели.

На метауровне, изображенном на рис. 64, реинтерпретация типа отношения в тип сущности требует введения не только общего класса ТИП СУЩНОСТИ, но и конкретизированных классов, представляющих исходный и реинтерпретированный типы сущностей. Таким образом, реинтерпретированные типы прослеживаются дважды. С одной стороны, они являются элементом конкретизации общего класса ТИП СУЩНОСТИ, а с другой — конкретизированной версией ТИПА ОТНОШЕНИЯ.

Введение в модель ERM операции обобщения и конкретизации приводит к созданию класса ОБЩ./КОНКР. ПРЕДСТАВЛЕНИЕ. В примере на рис. 63 модель регионального рынка включает тип сущности КЛИЕНТ, разбитый на классы МЕЖДУНАРОДНЫЕ КЛИЕНТЫ и ОТЕЧЕСТВЕННЫЕ КЛИЕНТЫ. Таким образом, региональный рынок является экземпляром класса ОБЩ./КОНКР. ПРЕДСТАВЛЕНИЕ. Одно представление может охватывать множество типов сущностей (МЕЖДУНАРОДНЫЕ КЛИЕНТЫ и ОТЕ-

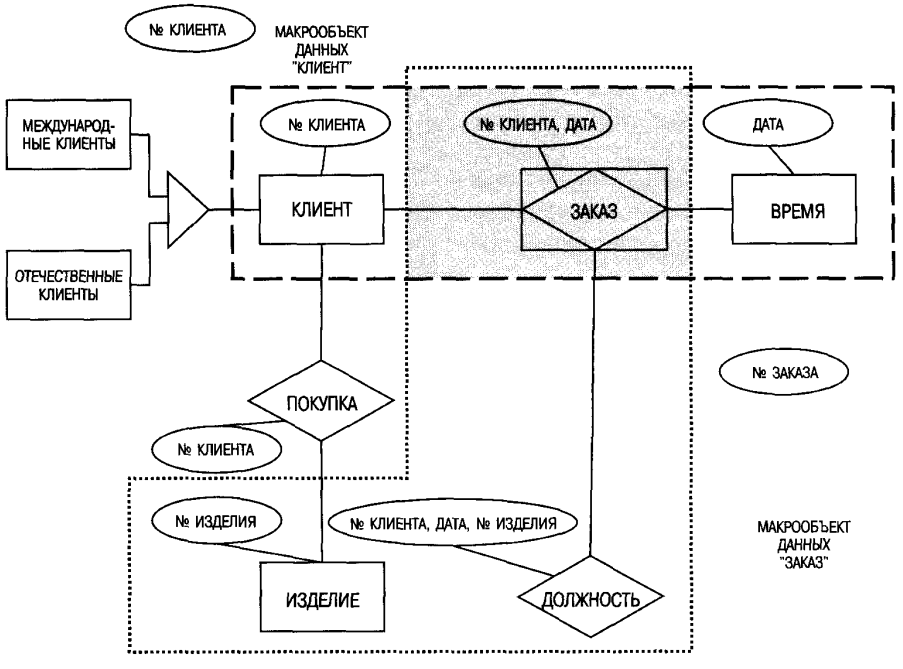


Рис. 63. Пример расширенной модели ERM

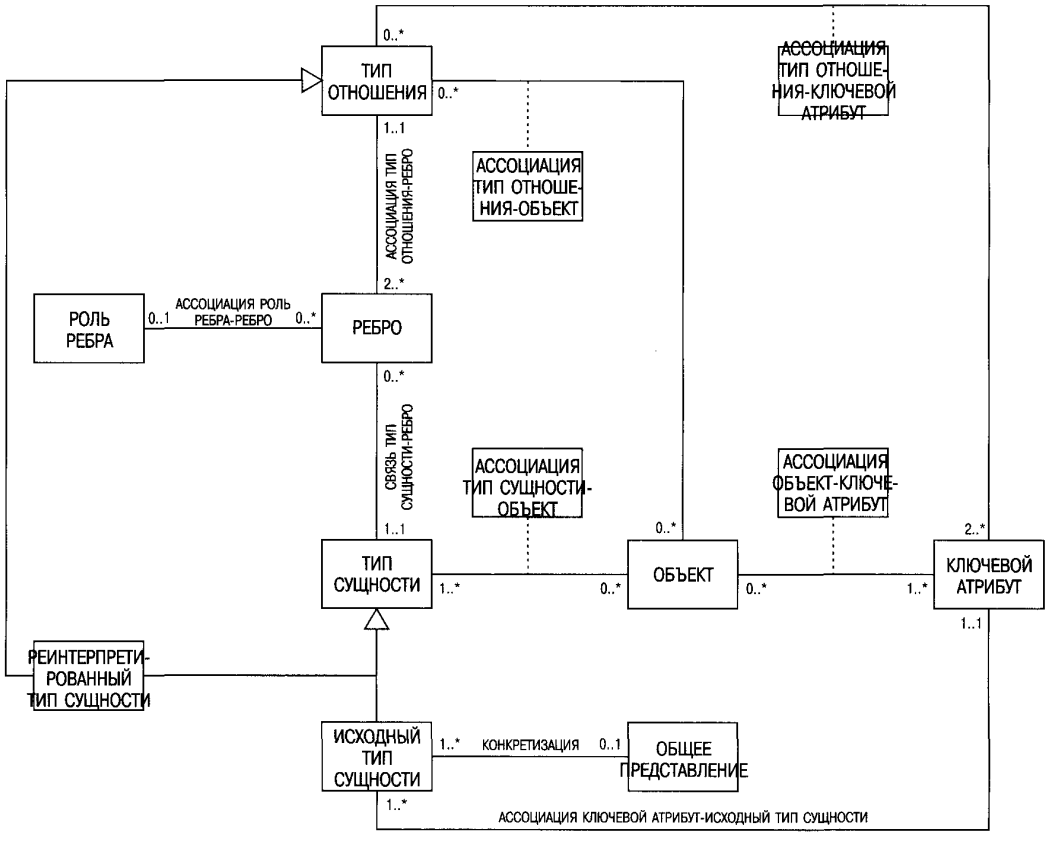


Рис. 64. Метамодель расширенной ERM

ЧЕСТВЕННЫЕ КЛИЕНТЫ), тогда как один тип сущности должен однозначно принадлежать одному конкретизированному представлению.

Поскольку на конкретизации переносятся и ключевые атрибуты доминирующей сущности, один ключевой термин оказывается действителен для нескольких типов сущностей (например, один номер клиента действительным как для общего типа сущности КЛИЕНТ, так и для конкретизаций МЕЖДУНАРОДНЫЕ КЛИЕНТЫ и ОТЕЧЕСТВЕННЫЕ КЛИЕНТЫ). Таким образом, мощность отношения между КЛЮЧЕВЫМ АТРИБУТОМ и классом ИСХОДНЫЙ ТИП СУЩНОСТИ равна (1..*):(1..1).

Ключевые атрибуты для типов отношений можно формировать путем привязки ребер ключевых атрибутов соответствующих типов сущностей. Однако для наглядности создается отдельный ассоциативный класс АССОЦИАЦИЯ ТИП ОТНОШЕНИЯ–КЛЮЧЕВОЙ АТРИБУТ.

В моделях ERM сложные сценарии разбиваются на простые и понятные структуры, хотя их связь с целой структурой не всегда остается наглядной. Поэтому мы вводим понятие <сложный объект> (ОБЪЕКТ), объединяющее множество типов сущностей и отношений, принадлежащих данному объекту *(Dittrich. Nachrelationale Datenbanktechnologie. 1990;*

Harder. Relationale Dantenbanksysteme. 1989; Lockemann. Weiterentwicklung relationaler Datenbanken. 1991; Kilger. C.: Objektbanksysteme. 1996).

Сложные объекты включают множество типов сущностей и отношений. Рассмотрим, например, чертеж, отражающий целую геометрическую структуру сборного узла, содержащую множество типов сущностей и отношений (ФОРМА, СЕКТОРЫ, РЕБРА, ТОЧКИ и т.д.), или деловой контракт со сложной структурой данных, содержащей множество вложений.

Заказы, изделия и данные, относящиеся к определенному клиенту, также можно рассматривать как сложные объекты (см. рис. 63). То, что в терминах бизнеса является индивидуальным объектом (ЧЕРТЕЖ, КОНТРАКТ, ЗАКАЗ), на метауровне становится экземпляром класса ОБЪЕКТ. Поскольку объекты могут налагаться друг на друга, мощность связи будет равна (0..*) или (1..*).

В данном определении понятия <объект> подчеркивается переход к понятию МАКРООБЪЕКТ ДАННЫХ. Этой темы мы коснемся при обсуждении спецификации проекта, когда будем рассматривать объектно-ориентированные модели данных.

Таким образом, классы ТИП СУЩНОСТИ, ИСХОДНЫЙ ТИП СУЩНОСТИ, ТИП ОТНОШЕНИЯ и ОБЪЕКТ, которые сами становятся информационными объектами, являются источниками для формирования описательных атрибутов. Атрибуты РЕИНТЕРПРЕТИРОВАННОГО ТИПА СУЩНОСТИ совпадают с атрибутами исходного класса ТИП ОТНОШЕНИЯ и поэтому вторично не присваиваются. В расширенной версии атрибуты следует присваивать таким же образом, как и в простой модели ERM.

A.2.3.2. Конфигурирование данных

Модель данных привязывает типы и параметры затрат, необходимые для расчета стоимости процессов, к системе пооперационного исчисления стоимости.

Аналогичным образом модель данных привязывает объекты данных, необходимые для описания существующих и целевых факторов и позволяющие рассчитывать профили составления графиков и управления мощностями, к системе планирования мощностей.

В системах workflow главной задачей является управление процессами. Тем не менее они тоже связаны с моделью данных. Поэтому нам следует разграничить данные, физически транспортируемые системой workflow, и содержимое баз данных, к которому workflow только обращается (см. рис. 65).

Первая группа преимущественно содержит нетрадиционные данные, такие как мультимедийные документы, доставляемые клиенту системой workflow. Во второй группе выполняются лишь обращения к данным, хранящимся в информационных объектах. Управление доставкой данных осуществляется логически путем передачи привилегий доступа в рамках процесса.

Рассматривавшиеся до сих пор объекты метамодели могут служить для описания потребностей системы workflow в данных с точки зрения бизнеса *(Galler. Vom Geschaftsprozeßmodell zum Workflow–Modell. 1997, с. 67).* Иногда при моделировании целесообразно использовать отдельные экземпляры данных (конкретные документы), если они присутствуют в каждом экземпляре типа бизнес-процесса. В этом случае модель данных необходимо дополнить функциональными возможностями для администрирования экземпляров.

С точки зрения бизнеса, структуры данных в бизнес-приложениях все чаще

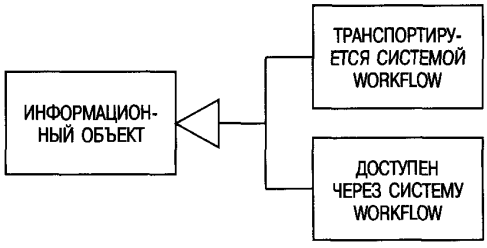


Рис. 65. Группы данных в системах workflow

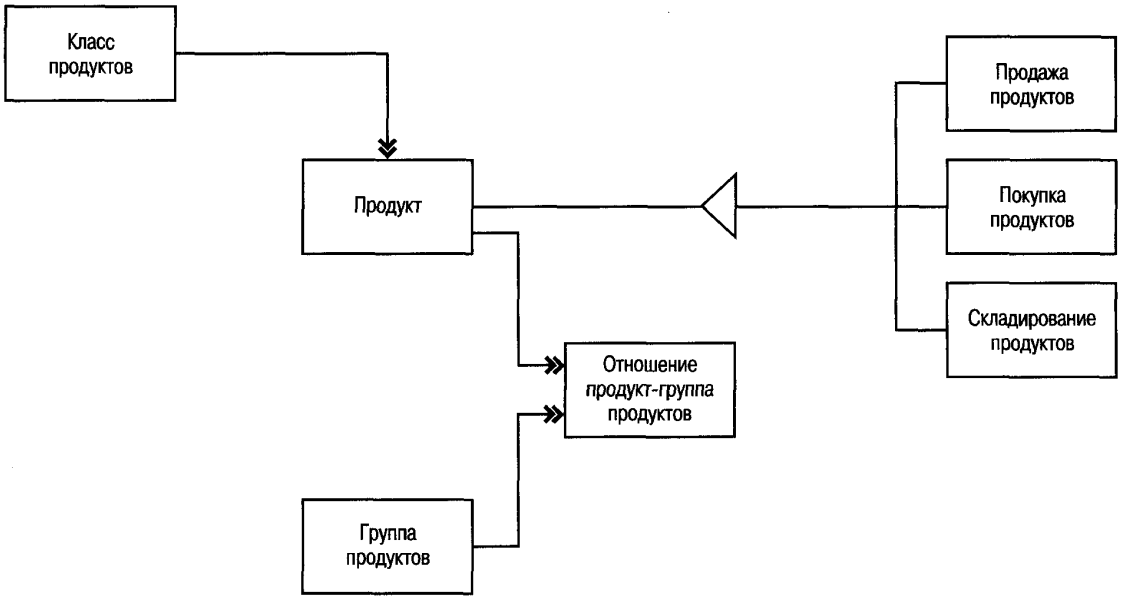


Рис. 66а. Модель данных SAP

описываются при помощи моделей данных. На рис. 66а и 66б приведены фрагменты моделей данных SAP R/3 и приложения ARIS. Хотя оба типа представления основаны на моделях ERM, их методология различна. Представление SAP–ERM объединяет модели ERM Чена *(Chen. Entity–Relational model. 1976)*, структурированные модели ERM Зинца *(Sinz. Datenmodellierung im SERM. 1993)* и систему обозначений Бахмана *(Bachman. The Programmer as Navigator. 1973)*.

Диаграмму ERM и выбранные здесь обозначения SAP–ERM можно объединить *(Scheer. Business Process Engineering. 1994; Nüttgens. Koordiniert–dezentrales Informationsmanagement. 1995, с. 104)*. С моделями данных для стандартного приложения обычно можно выполнять следующие манипуляции:

- удаление информационных объектов;
- удаление атрибутов;
- обновление числа разрядов в атрибутах;

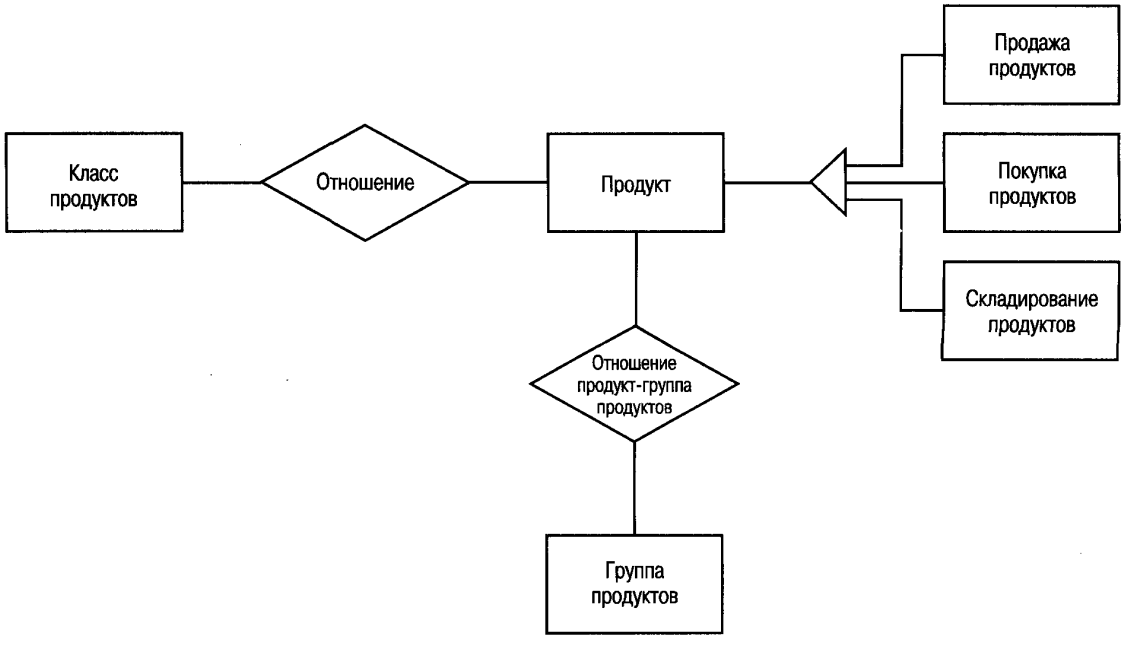


Рис. 66б. Модель данных приложения ARIS

- добавление атрибутов;
- добавление объектов данных.

При выполнении первых трех манипуляций условия целостности должны обеспечивать и адаптацию необходимых приложений, использующих эти данные. На рис. 67 приведен пример, показывающий, каким образом манипулирование числом разрядов в атрибуте автоматически обеспечивает адаптацию пользовательского интерфейса в бизнес-приложениях.

Если новые атрибуты переносятся исключительно в информационных целях и фактически не связаны с прикладной системой извне, то к существующему пользовательскому интерфейсу добавляются только новые поля. Это можно сделать вручную. При добавлении структур данных, сопровождаемых извне, возникает необходимость в новых пользовательских интерфейсах (создание, изменение, удаление), которые обеспечивали бы функции администрирования на уровне экземпляров. Поэтому новые интерфейсы должны генерироваться автоматически. Приложения, которые будут манипулировать вновь добавленными структурами данных, должны быть соответствующим образом дополнены.

A.2.3.3. Спецификация проекта в рамках модели данных

В процессе спецификации проекта языки интерфейсов для баз данных генерируются на основе семантической модели данных. Хотя эти интерфейсы соответствуют определенным моделям данных, мы должны отличать их от понятия «семантическая модель данных». Широко известны такие модели данных, как иерархические, сетевые, реляционные, объектно-ориентированные. Иерархические модели данных представляют интерес лишь с исторической точки зрения, а сетевые постепенно утрачивают свое значение, поэтому мы со-

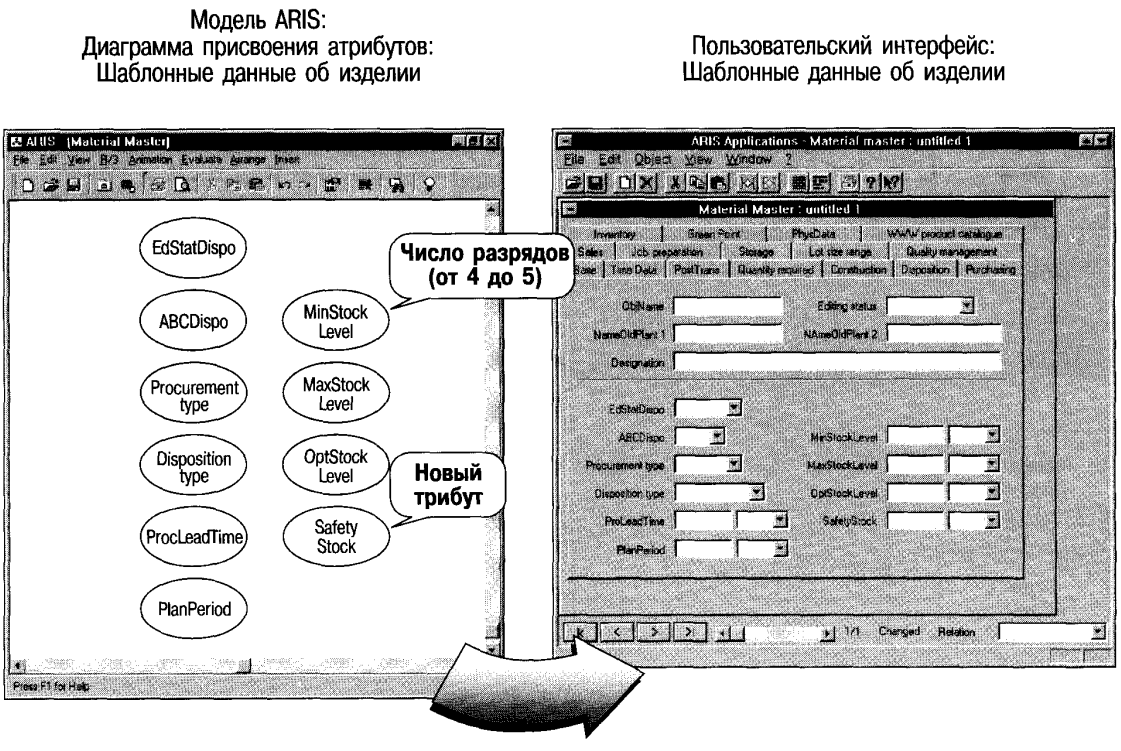


Рис. 67. Конфигурирование пользовательского интерфейса на базе модели данных

средоточим внимание на реляционных моделях данных и лишь вскользь затронем объектно-ориентированные.

На первом этапе информационные объекты уровня определения требований трансформируются в отношения. Здесь необходимо соблюдать определенные правила.

На втором этапе отношения оптимизируются посредством нормализации. При этом удаляются любые аномалии, возникающие при использовании функций «вставка», «изменение» и «удаление». Теперь предварительные отношения, перенесенные с уровня определения требований, разбиваются на более мелкие элементы. Если чрезмерная детализация приведет к снижению производительности, можно применить прямо противоположный метод, который называется денормализацией.

На третьем этапе описываются условия целостности, которые могут быть либо перенесены с уровня определения требований и преобразованы в соответствии с условиями реляционной модели, либо добавлены с уровня спецификации проекта. Новое описание условий целостности на основе определения требований является следствием ограниченного языкового диапазона реляционной модели. Это диктует необходимость описания условий целостности на языке манипулирования данными.

На четвертом этапе реляционная схема трансформируется в язык описания данных соответствующей системы управления базами данных с добавлением логических путей доступа для поддержки обработки, ориентированной на записи. Это последний этап, который непосредственно подводит к фазе реализации.

Реляционная схема переносится на используемый базой данных язык описания данных (ЯОД) и адаптируется к техническим требованиям ЯОД. Саму реализацию это никак не затрагивает. В то же время ЯОД представляет собой язык описания, наиболее «близкий» к реализации, где с моделью данных взаимодействуют другие модели ARIS. Приложения должны взаимодействовать только через схему базы данных, описанную посредством ЯОД.

A.2.3.3.1. Создание отношений

Отношения (Ri) описываются путем перечисления имен атрибутов Aij (см. (1) на рис. 68). Удобно представлять отношения в виде таблиц. С математической точки зрения, отношение есть подмножество декартова произведения доменов, связанных с атрибутами.

Соблюдая сравнительно простые правила, отношения можно вывести на основе модели ERM, описывающей требования на уровне данных. При этом каждый тип сущности и каждый тип отношения n:n преобразуется в отношение. Тип отношения n:n означает, что максимальное значение мощностей связей по крайней мере двух смежных типов сущностей равно n.

С другой стороны, связи типа 1:n не имеют собственного отношения. В этом случае отношения адаптируются путем введения ключевого атрибута в тип сущности, в результате чего максимальное значение мощности оказывается равно 1 (см. примеры на рис. 69). Такой перенесенный ключевой атрибут называется внешним ключом.

Метамодель, представленная на рис. 70, вводит класс ОТНОШЕНИЕ. Его отношение с классом ИНФОРМАЦИОННЫЙ ОБЪЕКТ, созданным на стадии определения требований, устанавливается при помощи связи СОЗДАНИЕ ОТНОШЕНИЯ. В соответствии с формированием отношений информационный объект может иметь либо 0, либо (максимум) 1 отношение, тогда как одно отношение может быть связано с одним или множеством информационных объектов. Атрибутом класса ОТНОШЕНИЕ является его собственное имя, которое также может совпадать с именем исходного информационного объекта ERM.

Имена атрибутов, принадлежащих тому или иному отношению, также можно взять из определения требований, хотя их можно и изменить. Если изменения не вносятся, атрибуты создаются путем связывания классов ОТНОШЕНИЕ и ИНФОР-

(1) $R_i (A_{i1}, A_{i2}, \ldots, A_{iz})$ A_{ij} = Attribute j in relation I

Деталь (Номер детали, Имя, Запас)

Деталь	Номер детали	Имя	Запас
	4717	Отверстие	526
	4728	Болт	768

(2) $R_i (D_{i1} \times D_{i2} \times \ldots \times D_{iz})$
whereby D is the domain of A

Рис. 68. Описание отношений

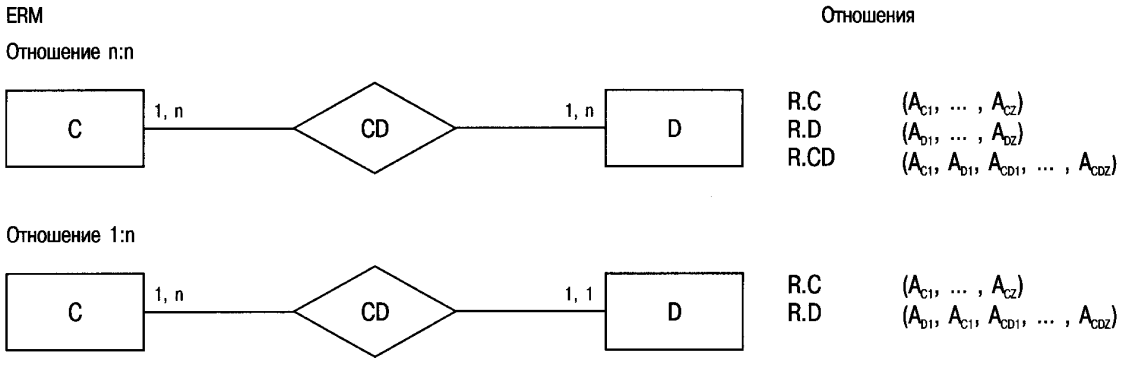


Рис. 69. Выведение отношений на основе ERM

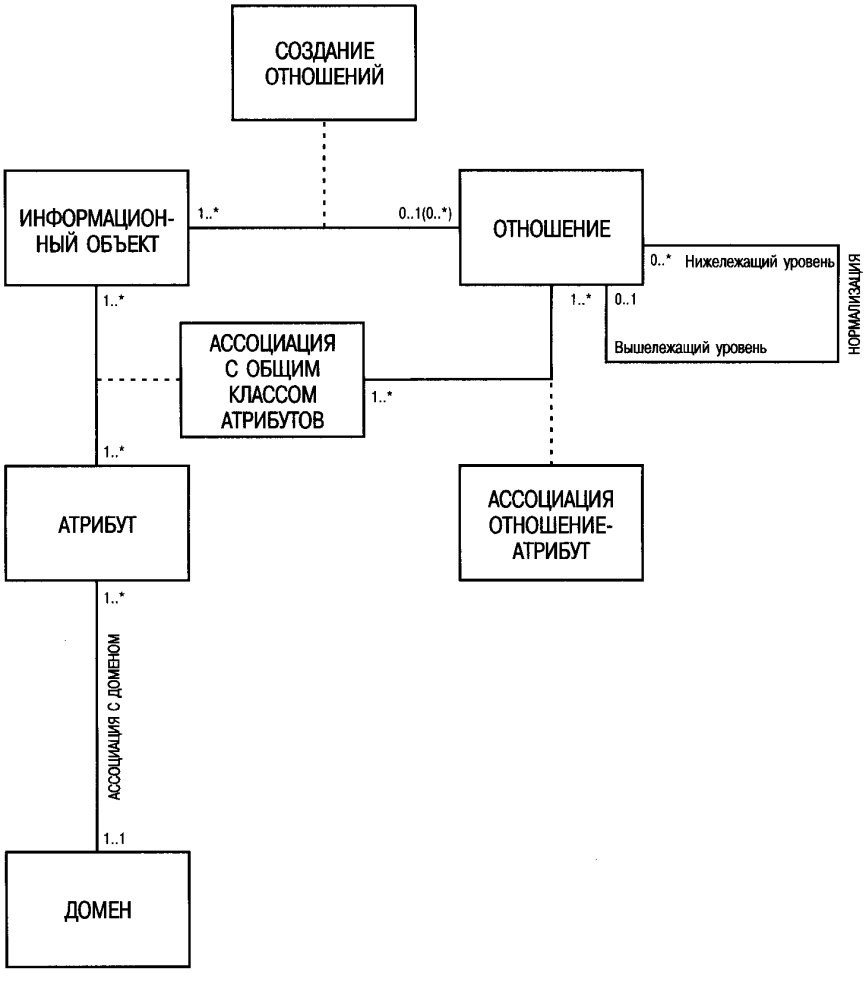


Рис. 70. Метамодель выведения отношений

МАЦИОННЫЙ ОБЪЕКТ. Однако для того чтобы подчеркнуть автономность спецификации проекта на уровне разработки, присваиваемые отношению атрибуты связываются с общим описанием атрибутов на уровне определения требований при помощи АССОЦИАЦИИ ОТНОШЕНИЕ–АТРИБУТ. Если при переносе с уровня определения требований имена не меняются, отношения можно формировать в соответствии с описанными требованиями. Многие коммерческие инструменты типа CASE обеспечивают такой автоматический переход от модели ERM.

Доступ доменов к имеющимся описаниям доменов, полученным на этапе определения требований, осуществляется через присвоение атрибутов. Для иллюстрации мы рассмотрим класс ДОМЕН, когда будем обсуждать реляционную модель и условия целостности, относящиеся к доменам.

В то время как перенос типов сущностей и отношений в реляционную модель не составляет проблемы, перенести в нее сложные объекты гораздо труднее. Это обусловлено тем, что возникает необходимость в таких дополнительных операциях, как импортирование в реляционную модель процедур неструктурированных групп данных или даже расширение модели данных до уровня объектно-ориентированной.

A.2.3.3.2. Нормализация — денормализация

Предварительные отношения, взятые из моделей бизнес-процессов, иногда приводят к нежелательным побочным эффектам при использовании таких функций базы данных, как «вставка», «изменение» или «удаление». Такие побочные эффекты известны как аномалии. Эти аномалии можно устранить с помощью так называемой нормализации. Хотя процедура нормализации проектировалась применительно к реляционной модели, она может выполняться как общая процедура для совершенствования структур данных, а также применяться к другим моделям данных. В этой книге каждая ступень нормализации дается лишь в виде определения. Более подробная информация содержится в работах: *Schlageter,Stucky. Datenbanksysteme 1983. c. 183; Wedekind. Datenbanksysteme I. 1991, c. 200; Vossen. Datenbank–Management–Systeme. 1995, c. 249–270.*

Кроме того, мы рассмотрим только нормальные формы 1–3 (так называемые нормальные формы Бойса–Кодда). Форм 4 и выше ввиду их крайне редкого применения касаться не будем. Возьмем следующий пример *(Schlageter, Stucky. Datenbanksysteme. 1983, c. 162)*, относящийся к организации проекта:

1–я НОРМАЛЬНАЯ ФОРМА (1 НФ):

(1,1) R_EMPLOYEE	(EMP_NO, NAME, ADDRESS, PROFESSION, DEPT_NO)
(1,2) R_PROJECT	(P_NO, P_NAME, P_DESCR, P_MGR)
(1,3) R_EMP_PROJ	(P_NO, EMP_NO, PH_NO, PERCENT_WORK_HOURS)
(1,4) R_DEPT_NO	(DEPT_NO, DEPT_MGR, BUILDG_NO, JANITOR)

2–я НОРМАЛЬНАЯ ФОРМА (2 НФ):

(2,1) R_EMPLOYEE*	(EMP_NO, NAME, ADDRESS, PROFESSION, DEPT_NO)
(2,3) R_EMP_PROJ*	(P_NO, EMP_NO, PH_NO, PERCENT_WORK_HOURS)

3–я НОРМАЛЬНАЯ ФОРМА (3 НФ):

(3,4) R_BUILDG	(BUILDG_NO, JANITOR)

(3,5) R_DEPT* (DEPT_NO, DEPT_MGR, BUILDG_NO)

Определения:

- Отношение R соответствует 1-й нормальной форме (1 НФ), когда значение каждого атрибута является элементарным.
- Отношение соответствует 2-й нормальной форме (2 НФ), когда оно соответствует 1 НФ и каждый неключевой атрибут функционально зависит от каждого ключевого кандидата.
- Отношение соответствует 3-й нормальной форме (3 НФ), когда оно соответствует 2 НФ и ни один из неключевых атрибутов транзитивно не зависит от ключевого кандидата.

Теперь рассмотрим на примере аномалии, возникающие в 1 НФ, которые нужно удалить посредством нормализации.

- Аномалия вставки возникает, например, в том случае, когда в базу данных вводится новый сотрудник, еще не связанный с каким-либо проектом. Из-за отсутствия такой связи ему нельзя присвоить номер телефона (PH_NO), поскольку этот атрибут присутствует только в отношении сотрудник-проект (R_EMP_PROJ).
- Когда проект завершен и отношение (1,3) нужно удалить, возникает аномалия удаления. Она выражается в том, что номер телефон сотрудника также удаляется, даже если он по-прежнему действителен.
- Аномалия обновления возникает при изменении номера телефона сотрудника, когда требуется разыскать все кортежи отношения (R_EMP_PROJ). При этом придется обновить каждый номер телефона данного сотрудника, который может участвовать в нескольких проектах, хотя изначально обновлен лишь один-единственный элемент данных.

Эти аномалии исчезают при преобразовании отношений (1,1) и (1,3) во 2-ю нормальную форму. Поскольку номер телефона (PH_NO) идентифицируется только по ключу EMP_NO отношения (1,1), он вводится здесь как атрибут. В отношении (1,3) номер телефона удаляется. Отношения же (1,2) и (1,4) соответствуют 2 НФ.

При принятии на работу смотрителя необходимо обновить отношения отдела (1,4) применительно к зданиям, где он будет работать. Таким образом, смотритель непосредственно связан не с отделом (DEPT), а со зданием (BUILDG). В 3-й нормальной форме разбиение отношения (1,4) на два отношения устраняет транзитивную зависимость, порождающую кортежи. В данном примере отношения (2,1), (1,2) и (2,3) уже соответствуют 3-й нормальной форме.

Технически процесс нормализации приводит к дальнейшей детализации исходных отношений. Степень фактической детализации зависит от начальной ситуации. Если за отправную точку взять так называемое универсальное отношение, где определение требований хранится без какой-либо особой сортировки, процесс нормализации ведет к полной реструктуризации. Если же схема бизнеса уже спроектирована, например, с помощью ERM, информационные объекты обычно характеризуются высоким уровнем нормализации.

Тем не менее, даже если информационные объекты тщательно спроектированы, ключевые атрибуты определены, а неключевые предполагается добавить на более позднем этапе, иногда имеет смысл прибегнуть к нормализации системы в целях проверки.

Разбиение класса ОТНОШЕНИЕ посредством нормализации означает выведение дополнительных отношений на основе адаптированных предварительных отношений. Поскольку один информационный объект может порождать множе-

ство отношений, мощность класса ОТНОШЕНИЕ принимает значение (0..*), как показано в скобках рядом с соответствующим ребром на рис. 70.

На происхождение отношения из другого отношения, существующего на предыдущем уровне нормализации, указывает связь НОРМАЛИЗАЦИЯ, благодаря чему становится очевидным, из какого отношения лежащего ниже уровня нормализации выведено данное отношение рассматриваемого (лежащего выше) уровня.

Когда в процессе нормализации создаются новые отношения, описание требований и предварительные отношения, связанные с присвоением атрибутов, обновляются. Однако это ведет не к расширению диаграммы классов, показанной на рис. 70, а к созданию новых экземпляров ассоциативного класса АССОЦИАЦИЯ ОТНОШЕНИЕ–АТРИБУТ.

A.2.3.3.3. Условия целостности

Условия целостности обеспечивают, чадекватное моделирование реальности с помощью базы данных *(Blaser, Jarke, Lehmann. Datenbanksprachen und Datenbankbenutzung. 1987, с. 586)*.

Поскольку таблицы в реляционных моделях не позволяют достаточно хорошо описывать семантические элементы, условия целостности описываются языком манипулирования данными. Можно задавать условия целостности и в рамках прикладной программы. Учитывая принцип локальности и преимущества централизованного управления целостностью данных, эти условия целесообразно включить в представление данных. Современные СУБД рассчитаны на хранение максимума функциональных модулей (которые раньше хранились в программах) в непосредственной близости от баз данных *(Dittrich, Gatziu. Aktive Datenbanksysteme. 1996)*.

Условия целостности относятся к хранению семантического содержания и к базовой реализации модели данных. Они тесно связаны с базой данных и, следовательно, не зависят от особенностей проектирования на уровне конечного пользователя. Именно поэтому мы уделяем здесь особое внимание условиям семантической целостности.

Условия непротиворечивости относятся к атрибутам, экземплярам отношений (кортежам) и отношениям, вытекающим из типов отношений *(Blaser, Jarke, Lehmann. Datenbanksprachen und Datenbankbenutzung. 1987, с. 588)*, которые также известны как условия целостности на уровне ссылок.

Стандартным языком манипулирования данными в реляционных моделях является SQL. В языке SQL условия целостности задаются при помощи команд утверждения (ASSERT) и описания событий, которые должны активизировать выполнение действий.

Например, если при удалении номера сотрудника (EMP_NO) в отношении EMPLOYEE (СОТРУДНИК) должна удаляться и связь с квалификацией сотрудника в отношении OWNS (ОБЛАДАЕТ), связанном с отношением SKILLS (КВАЛИФИКАЦИЯ), то активизатор описывается следующим образом *(Blaser, Jarke, Lehmann. Datenbanksprachen und Datenbankbenutzung. 1987, с. 592)*:

```
DEFINE TRIGGER T1
ON DELETE OF EMPLOYEE (EMP_NO):
DELETE OWNS
WHERE
OWNS.EMP_NO=EMPLOYEE.EMP_NO.
```

На рис. 71 приведено несколько примеров описания утверждений.

На рис. 72 показаны ключевые отношения, вытекающие из декомпозиции условий целостности. Отправной точкой для рассмотрения проблем целостности явля-

ется левая часть рис. 72, содержащая классы ОТНОШЕНИЕ, АТРИБУТ и ДОМЕН. Класс ТИП ЦЕЛОСТНОСТИ описывает различные типы условий целостности (Reuter. Sicherheits-und Integritatsbedingungen. 1987, с. 380). Их можно дифференцировать по диапазону (тип и число объектов, которым адресовано условие целостности; см. примеры на рис. 71), по времени проверки (всегда ли проверяются условия целостности или только после выполнения определенного количества операций), по способу проверки (условия состояния или условия перехода) или по возможности активизации действий в зависимости от условия целостности.

На рис. 72 каждое конкретное условие целостности связано с одним типом целостности. Условие целостности может относиться к одному или нескольким отношениям и к ассоциации атрибутов одного или нескольких отношений. Пределы значения атрибута контролируются их связью с классом АССОЦИАЦИЯ С ДОМЕНОМ.

A.2.3.3.4. Логические пути доступа

Выполнение вложенных запросов SQL может повлечь за собой серьезные проблемы с производительностью. Чтобы повысить эффективность базы данных следует создать структуры утилит для поддержки доступа к отдельным кортежам или их группам. В частности, следует устранить необходимость последовательного поиска в таблицах.

Типичными средствами поддержки являются функции доступа к кортежам по их ключам и доступа к группе кортежей в определенной последовательности (сортировка). Это называется логическим путем доступа. Логические пути доступа для первичных ключей можно классифицировать в соответствии с последовательными организационными формами, древовидными индексами или рассредоточенными организационными формами. Пути доступа для вторичных ключей — особенно

	Пояснение	Операторы SQL
1)	Условие относится к атрибуту. Пример: экземпляры EMP_NO должны иметь четыре разряда	ASSERT IB1 ON EMPLOYEE: EMP_NO BETWEEN 0001 AND 9999
2)	Условие относится к множеству атрибутов экземпляр записи. Пример: сумма зарплаты (SALARY_SUM) по отделу должна быть меньше его годового бюджет (ANNUAL_BUDGET).	ASSERT IB2 DEPARTMENT: SALARY_SUM < ANNUAL_BUDGET
3)	Условие относится к множеству экземпляров одного и того же типа записи (отношения). Пример: зарплата одного сотрудника не может превышать среднюю зарплату по отделу более чем на 20%	ASSERT IB3 ON EMPLOYER X: SALARY = 1,2 J (SELECT AVG(SALARY) FROM EMPLOYEE WHERE DEPART = X.DEPART)
4)	Условие относится к множеству экземпляров в различных отношениях. Пример: значение SALARY_SUM в отделе всегда должно равняться сумме полей SALARY его сотрудников.	ASSERT IB4 ON EMPLOYER X: SALARY_SUM = (SELECT SUM(SALARY) FROM EMPLOYEE WHERE DEPART = X.DEPARTNO)

Рис. 71. Условия целостности *(Reuter. Sicherheits und Integritätsbedingungen. 1987, с. 381, 385)*

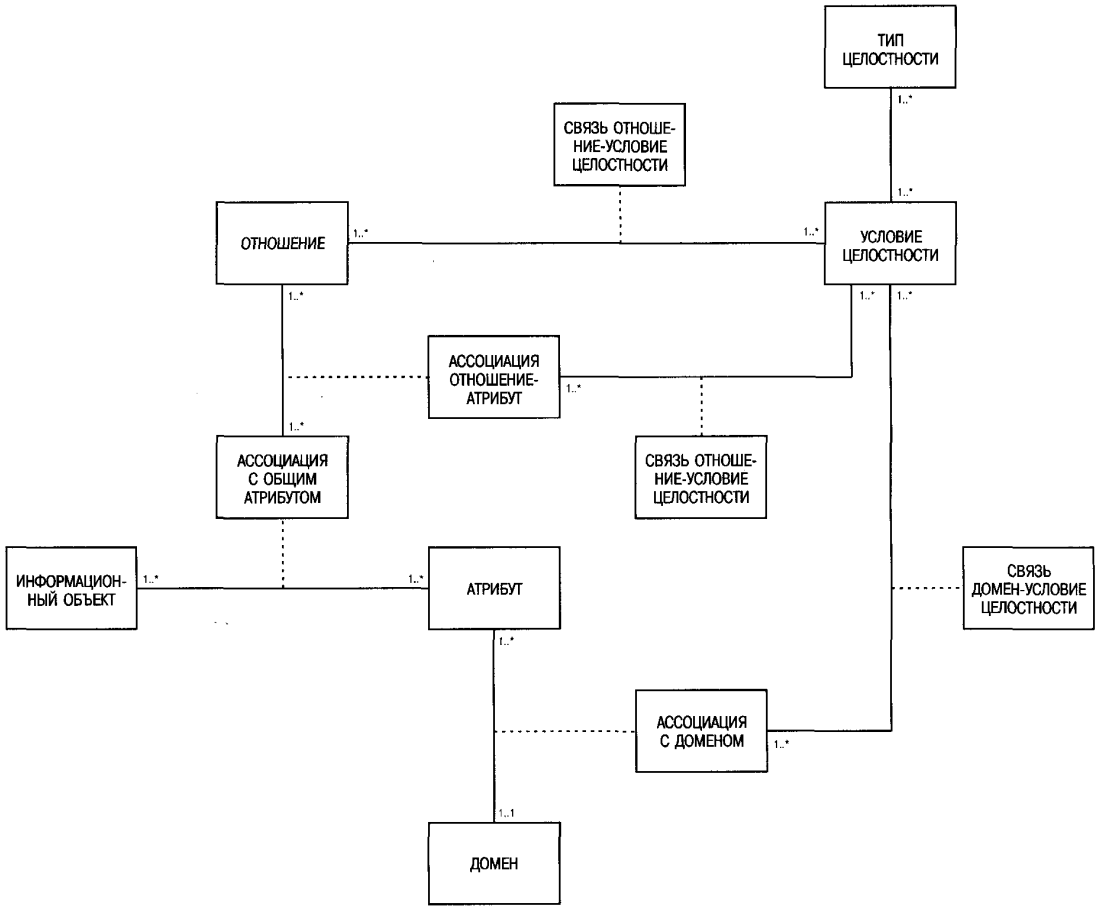


Рис. 72. Метамодель условий целостности

важные для реляционных баз данных — создаются при помощи инвертированных списков (индексных таблиц).

Спецификация проекта определяет, какие типы поддержки следует обеспечить для определенных атрибутов. Для описания различных типов поддержки создается класс ТИП ЛОГИЧЕСКОГО ПУТИ ДОСТУПА. Логический путь доступа характеризуется связью между атрибутом отношения и типом пути доступа. Это позволяет описать множество путей доступа для атрибута в определенном отношении, т.е. задать класс АССОЦИАЦИЯ ОТНОШЕНИЕ–АТРИБУТ (см. рис. 73).

В спецификации проекта пути доступа первоначально описываются исходя из общих предположений относительно числа кортежей в таблице и числа предполагаемых приложений таблицы. Однако после того как база данных создана и стали известны характеристики производительности и число операций базы данных, возможна дальнейшая дифференциация утилит доступа.

A.2.3.3.5. Схема базы данных

На заключительном этапе спецификации проекта структуры данных перево-

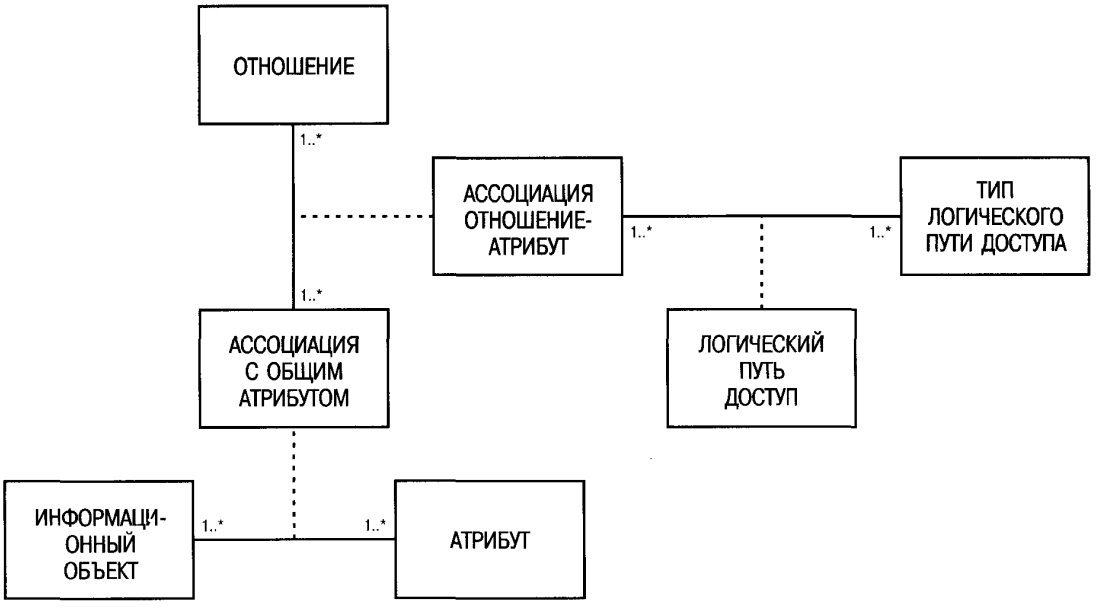


Рис. 73. Логические пути доступа

дятся на язык описания данных (ЯОД) конкретной системы баз данных (ORACLE, INFORMIX, CA-INGRES), которая будет использоваться. Благодаря математическому описанию реляционных моделей и применению стандартных SQL-операторов для описания условий целостности процесс перевода схемы реляционной базы данных на ЯОД системы баз данных — всего лишь вопрос техники.

Если предприятие одновременно внедряет несколько разных систем баз данных, нейтральное описание реляционной схемы можно перенести на несколько конкретных схем СУБД (система управления базами данных). Для поставщиков программного обеспечения, предлагающих продукты для различных баз данных, эта ситуация довольно типична. Схемы содержат конкретные описания баз данных, включая условия целостности и пути доступа. Этот сценарий представлен на рис. 74, где описывается тип сложного объекта СХЕМА.

A.2.3.4. Реализация на уровне модели данных

Мы начнем со спецификации проекта в рамках модели данных, т.е. с традиционной схемы базы данных с ее отношениями, атрибутами и условиями целостности, соответственно. Будут описаны логические пути доступа к определенным ассоциациям атрибутов с помощью предварительных концепций относительно частоты приложений и запросов.

На стадии реализации концептуальные схемы воплощаются во внутренние схемы, описывающие тот же фрагмент (<окно>) действительности, что и концептуальная схема. Семантика на этом этапе не добавляется. Напротив, внутреннюю схему можно вывести на базе концептуальной схемы, не зная семантического контекста.

Задача администраторов баз данных состоит в том, чтобы структурировать внутреннюю схему, создавая эффективные структуры базы данных с помощью

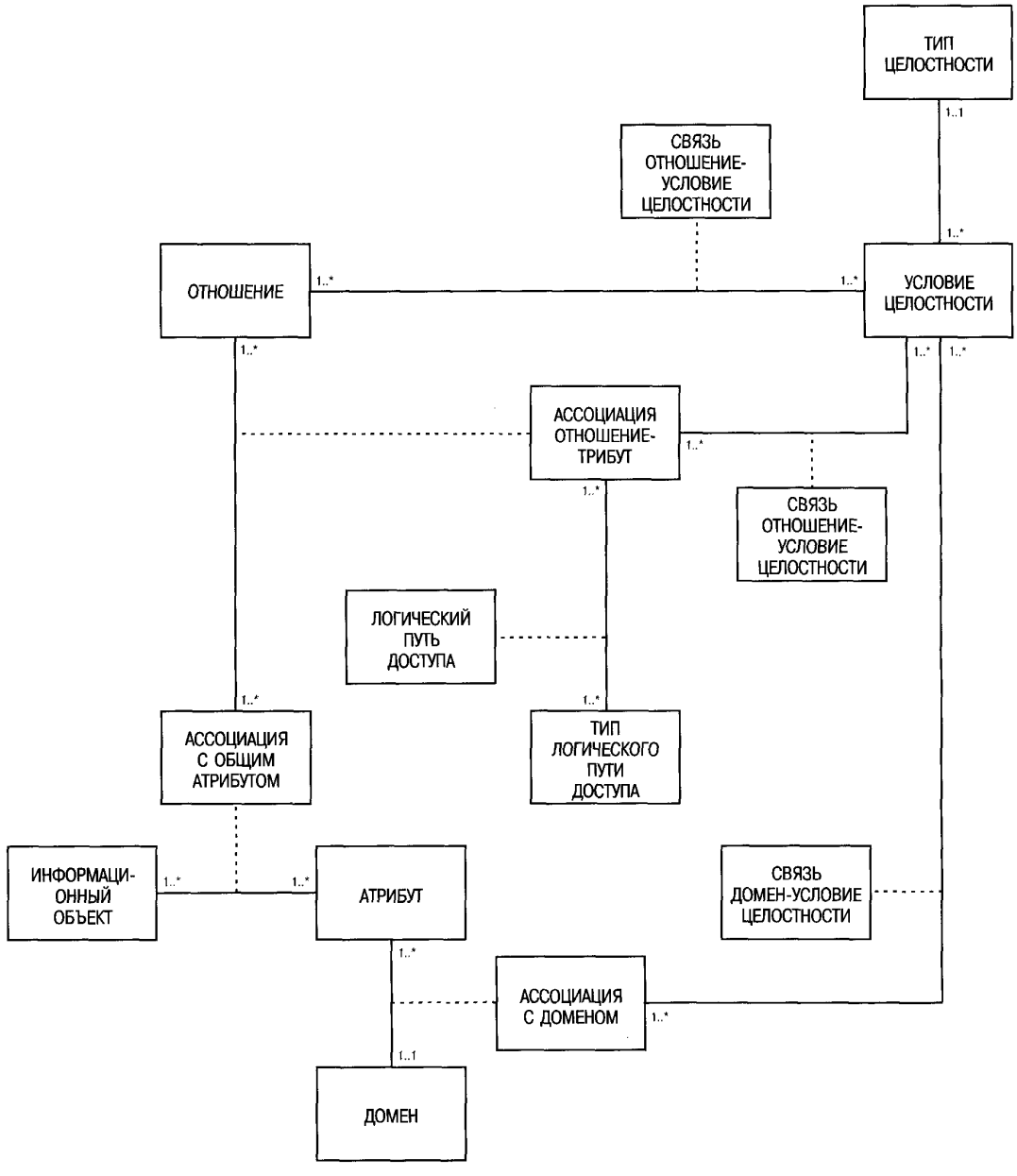


Рис. 74. Описание СХЕМЫ

имеющихся в их распоряжении информационных технологий. Администраторам баз данных необходимо следить за профилем использования и контролировать данные, объем и время отклика различных приложений. Однако для того, чтобы оперировать этой информацией, им необязательно знать содержание приложений.

Автономность уровня реализации поддерживается также употреблением спе-

цифических для предприятия терминов, которые, в свою очередь, моделируются на логическом уровне спецификации проекта (см. рис. 75). Например, обозначения ОТНОШЕНИЕ и АТРИБУТ связываются с обозначениями ЗАПИСЬ и ПОЛЕ на уровне реализации. Термин ЗАПИСЬ представляет тип записи, характеризующийся определенной комбинацией атрибутов. СТРАНИЦА может содержать различные типы записей. Работа администратора базы данных заключается в том, чтобы оптимизировать базу данных путем размещения часто требующихся типов записей в непосредственной близости друг от друга.

Уровни спецификации проекта можно отличить от уровней реализации по таким признакам (отличным от атрибутов реляционной схемы), как возможность изменения последовательности полей, их переименования и уплотнения данных, а также по наличию специфических форматов полей. Кроме того, можно создавать виртуальные поля, т.е. для полей можно описать правила преобразования, если их содержание состоит из других полей (например, поля итоговых величин).

Понятия «отношение» и «запись» могут не совпадать, если отношения разбиваются на несколько записей или если несколько записей объединяются в одну физическую запись.

Условия целостности и непротиворечивости, описанные на уровне спецификации проекта, теперь конкретизируются на физическом уровне в виде процедурных объектов. Дополнительные уточнения включают привязку конкретных методов физического доступа к логическим путям доступа или описание дополнительных физических путей доступа.

Помимо обозначений ЗАПИСЬ и ПОЛЕ, которые уже имеют базовые аналоги в спецификации проекта, введем теперь категории СТРАНИЦА и ОБЛАСТЬ (см. рис. 75), составляющие дополнительные организационные элементы для оптимизации структур распределения памяти. Эти предварительные единицы являются важнейшими «кирпичиками» для организации доступа к данным во внешних устройствах хранения и для распределения физических блоков хранения.

Потенциальные возможности оптимизации страниц и областей очевидны уже при рассмотрении осуществимости. Так, доступ к нескольким записям, размещенным на одной и той же странице, эффективнее, чем доступ к записям, разбросанным по разным страницам. Аналогично доступ к страницам, имеющим последовательную нумерацию, эффективнее, чем доступ к разрозненным страницам.

Язык описания хранения данных (DSDL) основан на ссылках между внутренними и внешними моделями и попутно реализует структуры распределения памяти. Физические пути доступа моделируются при помощи конкретных индексных таблиц, цепочек или кэш-функций.

Физические пути доступа описываются на уровне АССОЦИАЦИИ ЗАПИСЬ–ПОЛЕ. Они либо конкретизируют логические пути доступа, определенные на стадии спецификации проекта, либо создаются, что называется, с нуля на стадии реализации при наличии детальных сведений о параметрах производительности.

Описания на уровне физических структур данных обусловлены в первую очередь заложенной в проекте целью обеспечить независимость данных. Изменения в устройствах или системном программном обеспечении должны затрагивать только уровень реализации, но не уровень концептуальной схемы базы данных. По этой причине одна концептуальная схема базы данных со временем может вобрать в себя несколько внутренних схем баз данных. И наоборот, концептуальную схему базы данных можно обновлять, не внося изменений в ее физическую схему.

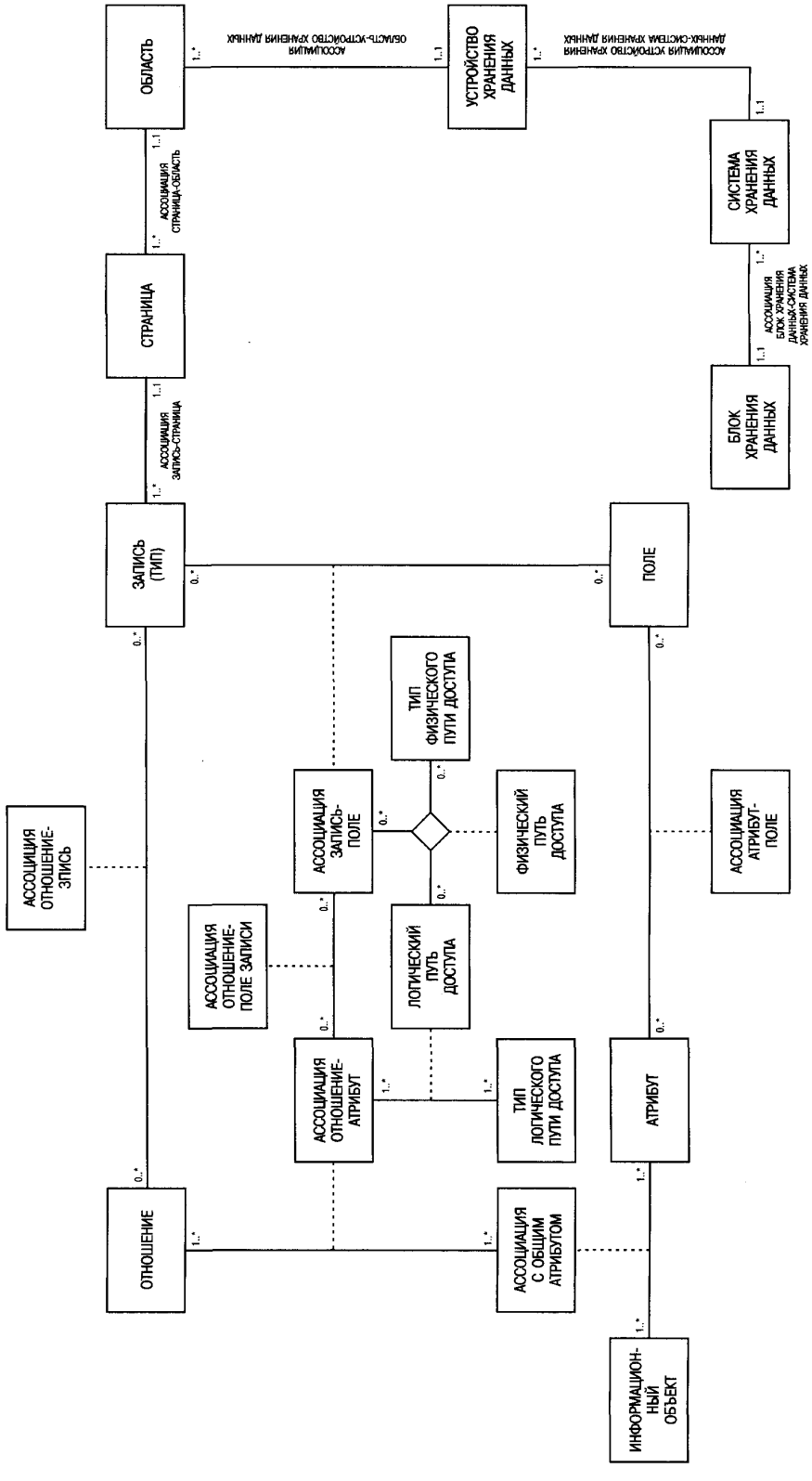


Рис. 75. Реализация модели данных

A.2.4. Моделирование на уровне выходов

На рис. 76 показано место модели выходов в концепции ARIS.

Выход — это результат выполнения процессов, инициируемых входом. Вследствие такой тесной связи степень структурирования описания выхода влияет на степень структурирования процессов, необходимых для его получения.

Таким образом, описание выхода является одним из ключевых аспектов описания бизнес-процессов. Это относится как к описанию областей деятельности при стратегическом планировании, так и к представлению выхода при моделировании бизнес-процессов.

Понятие «выход» отличается неоднородностью, охватывая самые разнообразные результаты деятельности (в частности, материальный выход и услуги), и может применяться на различных уровнях детализации. Мы будем использовать термин «выход» в том же контексте, что и «продукт», хотя нельзя не признать, что применительно к услугам он звучит несколько своеобразно.

Спецификация проекта и описание реализации здесь не рассматриваются, поскольку реализация модели выходов не отличается чем-то особенным. В создании материального выхода в виде автомобилей, станков и даже потребительских товаров (например, стиральных машин) все шире участвуют компьютерные технологии, а компьютерные системы уже содержат собственные информационные системы и по сути дела отвечают описаниям ARIS. Поэтому к ним применимы спецификация проекта и описание реализации на базе ARIS.

Информационные услуги моделируются как объекты данных, хотя следует иметь в виду, что для описания состояния выхода необходимо точно описать состояние объекта данных. Спецификация проекта и описание реализации информационных услуг принципиально не отличаются от аналогичных этапов, изложенных в разделе, где мы рассматривали модели данных.

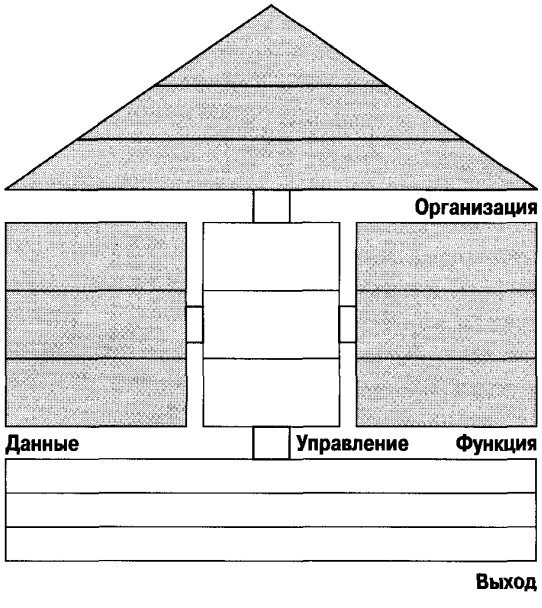


Рис. 76. Классификация Модели выходов в ARIS

A.2.4.1. Определение требований на уровне модели выходов

Моделирование материального выхода (с использованием моделей продуктов) в терминах бизнеса широко применяется для научных и практических целей. В качестве модели-прототипа для описания геометрических, физических, химических, функциональных и административных свойств реальных материальных и функционирующих продуктов во всей их полноте служит модель STEP *(Grabowski et al. Integriertes Produktmodell. 1993)*. Моделирование данных является прекрасным методом для описания продуктов, представляя продуктовые модели как модели данных.

В отличие от описаний материальных товаров модели услуг, как правило, пока еще находятся в стадии разработки.

Сейчас даже государственный сектор, стремясь упрочить свои позиции в сфере услуг, начинает прибегать к описанию своей продукции. В силу широкого спектра услуг, предоставляемых государственным сектором, предлагаемые им определения продукта можно использовать как обобщенные описания выходов или продуктов.

Вот некоторые примеры таких определений (с небольшими изменениями):

- Продукт есть выход (или группа различных выходов), востребованный секторами, лежащими вне корпоративного подразделения, где он был создан (внутри организации или за ее пределами; см. *KGSt. Das Neue Steuerungsmodell. 1994, с. 11*).
- Продукт есть то, что продуктовые центры поставляют субъектам за пределами своей организации для удовлетворения спроса этих субъектов независимо от того, предъявляется такой спрос на добровольной основе или же в силу юридических либо иных требований, и независимо от того, платят эти субъекты за данный продукт или они избавлены от такой необходимости *(KGSt. Wege zum Dienstleistungsunternehmen. 1992)*.

Эти определения относятся исключительно к внутрифирменным продуктам, которыми обмениваются организационные единицы.

Документирование продукта включает рассчет затрат, которые появляются при описании, например, при управлении складом или расчете стоимости. Можно ли отнести понятие «продукт» к определенному состоянию выхода, зависит от того, для чего этот выход предназначен. Скажем, в сфере материального выхода продукт получает наименование не после каждой операции, а лишь по достижении определенного состояния (например, на уровне управления складом), когда продукт требуется описать (см. рис. 77, а также работу: *Scheer. Business Process Engineering. 1994, с. 154*). Так же обстоит дело, когда каждый процесс операции связан с передачей в другое подразделение. Промежуточные этапы характеризуются информацией, относящейся к завершенной операции. В административных процессах новые имена продуктам также не присваиваются.

В моделировании процессов, если существует отношение клиент-поставщик со следующей обрабатывающей функцией или выполняющей ее организационной единицей, каждая модификация состояния после выполнения функции сама по себе является своего рода продуктом. На рис. 78 показано, как общие понятия «выход/вход» и «продукт» можно дифференцировать, выделив «материальный выход (вход)» и «услуги». Последние, в свою очередь, подразделяются на информационные и прочие услуги. При современной общей тенденции развития общества, ориентированного на услуги и информацию, центр тяжести все больше переносится на нефизические свойства. Таким образом, материальные продукты оказы-

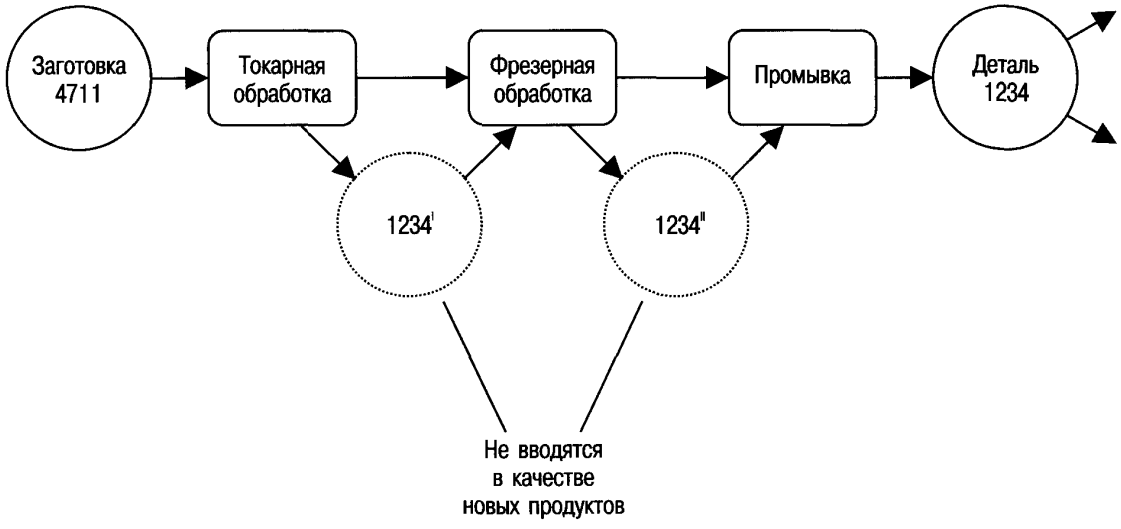


Рис. 77. Описания продуктов

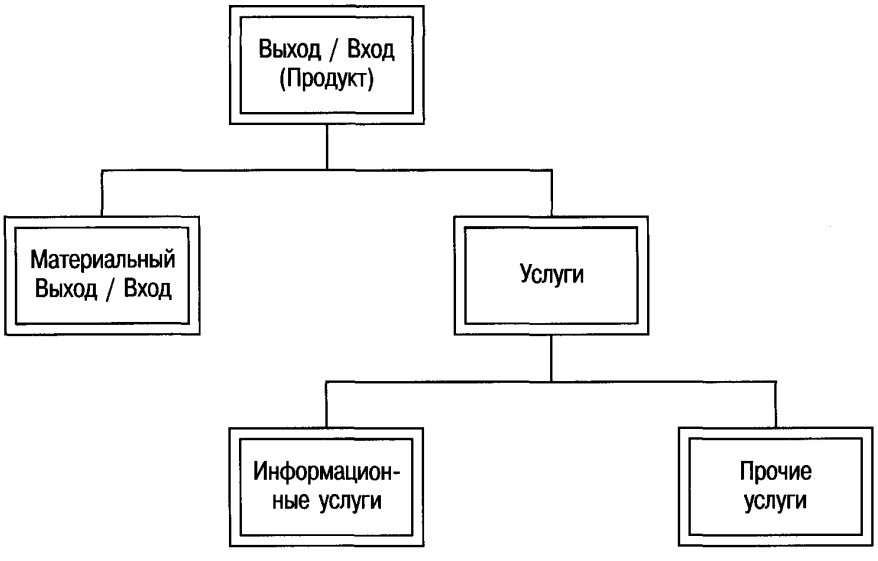


Рис. 78. Типы выходов и продуктов

ваются все в большей степени связанными с услугами.

Описание на уровне выходов моделируется при помощи соответствующих структур продуктов. Для моделирования выходов используются иерархические структуры продуктов (или сети продуктов) и логические связи, соответствующие отношениям «часть целого».

В отличие от продуктовых моделей данных, куда включается гораздо более сложная и подробная информация (произ-

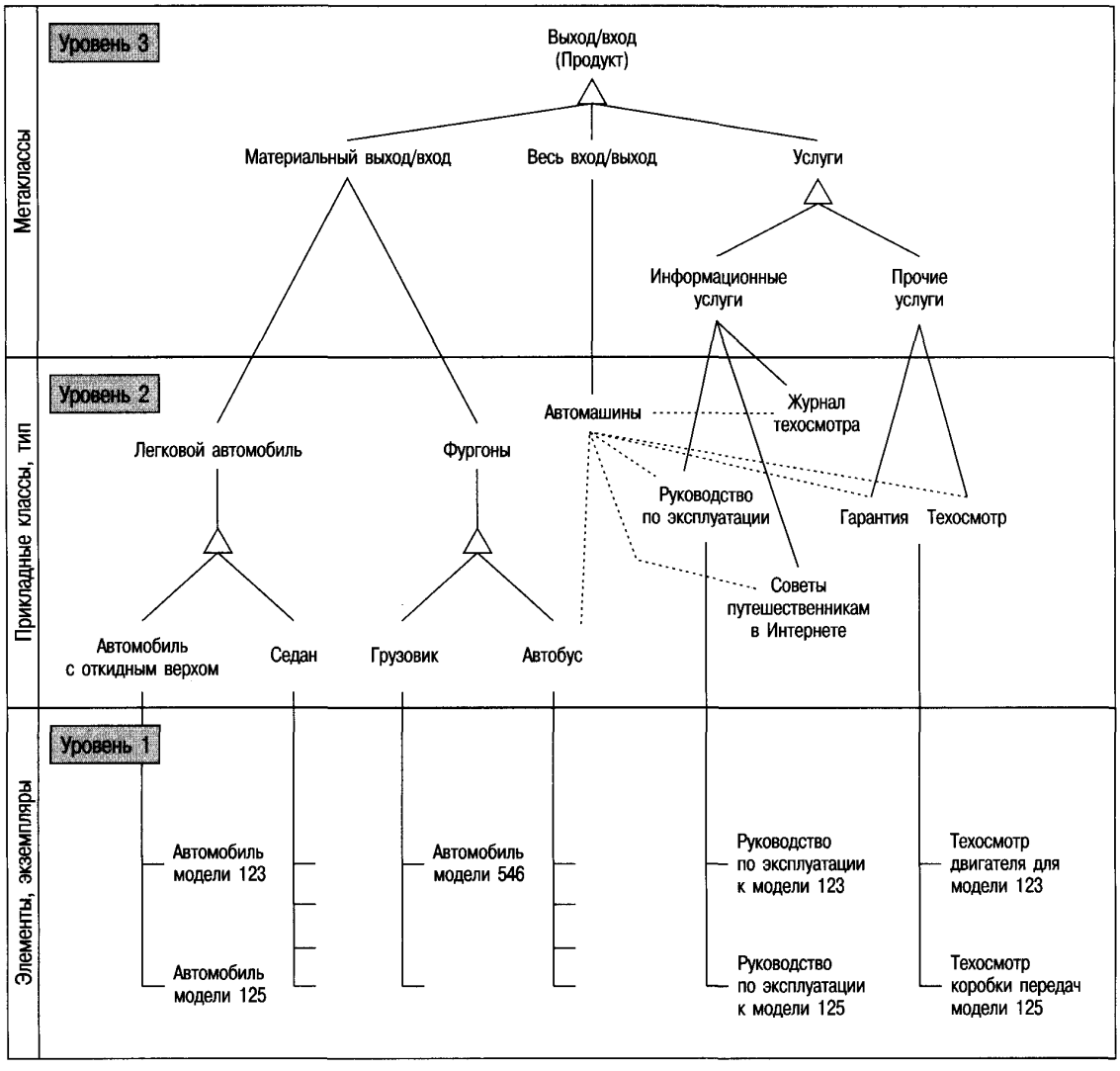


Рис. 79. Моделирование объектов на уровне выходов

водственные нормативы, геометрические характеристики и т.д.), структуры продуктов имеют ограниченный семантический диапазон. На рис. 79 показаны различные уровни моделирования примерами типичных моделируемых объектов.

Эти примеры иллюстрируют различные подклассы на 2–м уровне моделирования, иерархические структуры продуктов, а также связь между материальным выходом/входом и услугами. Связь «часть целого» представлена на 2-м уровне пунктирными линиями.

На рис. 80 мы вводим на метауровне общие классы ВЫХОД/ВХОД (или ПРОДУКТ) наряду с их подклассами МАТЕРИАЛЬНЫЙ ВЫХОД/ВХОД и УСЛУГИ. Объекты 2-го уровня моделирования являются экземплярами этих классов. Состав выхода выражается связью СТРУК-

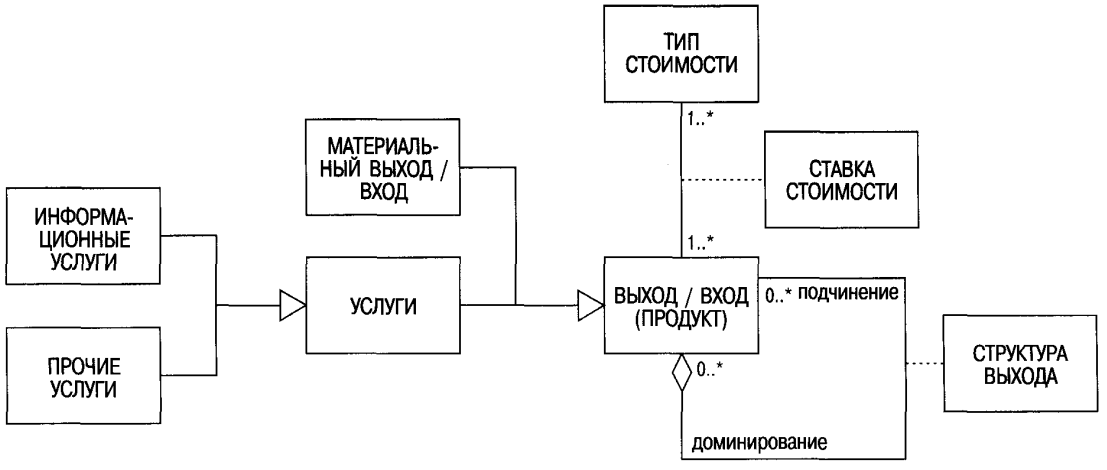


Рис. 80. Метамодель на уровне выходов

ТУРА ВЫХОДА, содержащей в качестве экземпляров связи «часть целого» 2-го уровня. Для иллюстрации на рис. 81а и б приводится несколько примеров продуктовых моделей на 2-м уровне моделирования.

В то время как в рамках других типов представлений ARIS моделирование бизнес-процессов на уровне экземпляров является исключением, с представлением выходов для промышленных предприятий, где существуют подробные каталоги продуктов, *дело обстоит иначе*. Экземпляры готовых, промежуточных и исходных продуктов подробно описываются в мастер-файлах изделий и в структурных файлах.

На рис. 82 приведен сокращенный вариант описания седана на уровне экземпляров. Администрирование описаний выходов осуществляется системами планирования и управления производством. В сфере услуг сегодня также наблюдается тенденция к описанию в продуктовых моделях не только классов выходов на 2-м уровне (в виде мастер-данных), но и экземпляров. Например, в одном исследовании, проводившемся на базе государственного сектора услуг в Берлине, было описано более 10 тыс. административных услуг (продуктов).

Проводится оценка выхода и затрат, необходимых для его создания. Если создание выхода сопряжено с чрезмерными затратами, в потоках выходов описываются также потоки стоимости. В метамодели на рис. 80 класс ТИП СТОИМОСТИ описывает категории затрат (например, стоимость материалов, людских ресурсов , коммунальных услуг), связанных с данным выходом. Отношение КОЭФФИЦИЕНТЫ СТОИМОСТИ содержит конкретные средние значения определенного типа стоимости для определенного типа выхода или пропорциональные ставки стоимости, вычисленные на базе общей стоимости выхода.

Объекты информационных услуг (сертификаты, руководства по эксплуатации и т.д.) представляются объектами данных и, таким образом, становятся частью модели данных ARIS. В модели выходов они идентифицируются как вход или выход функции. В модели данных требования, изложенные в этих описаниях, должны удовлетворяться описаниями соответствующих объектов данных.

Как и другие типы представлений, модель выходов ARIS частично перекрещивается с моделями данных. Материальный выход описывается в базе данных при по-

Рис. 81а. Продуктовая модель услуг авиакомпании

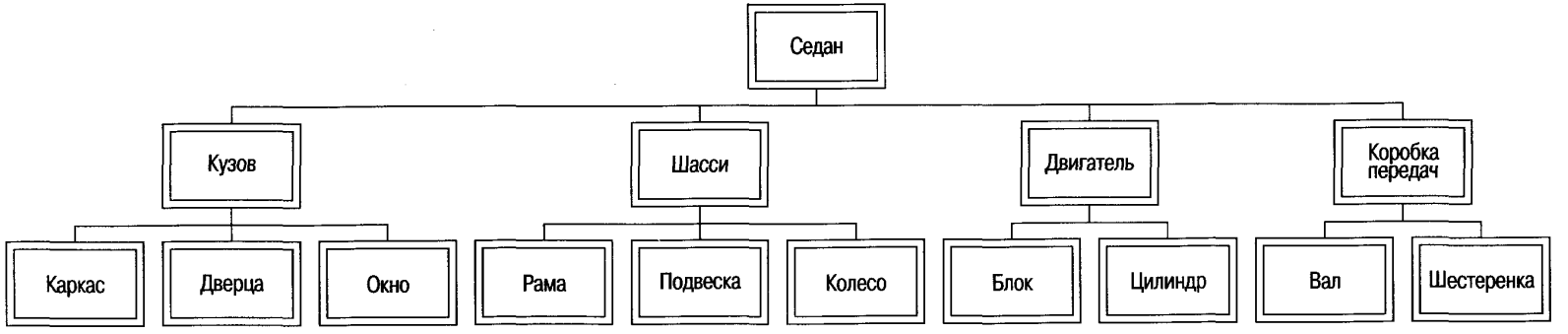


Рис. 81б. Продуктовая модель материального выхода «Седан»

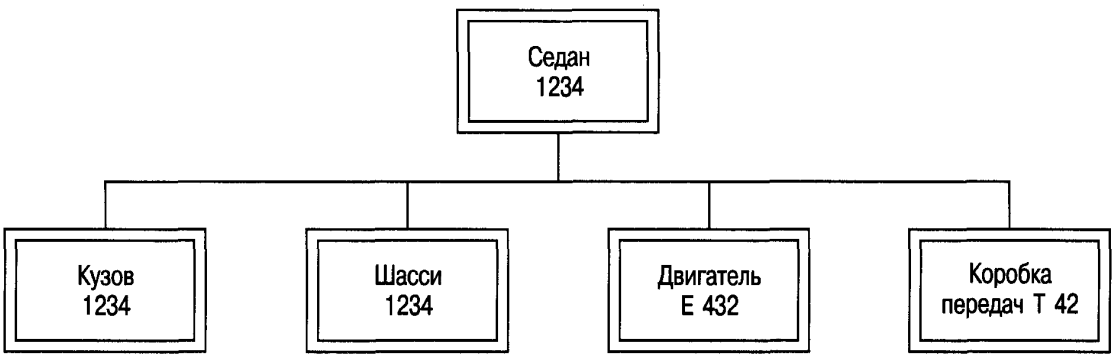


Рис. 82. Модель продукта «Седан 1234» (уровень экземпляров)

мощи продуктовых структур и классификаций производственных процессов обработки и управления. С другой стороны, прикладные системы не содержат столь же подробной информации об услугах.

Для описания выходов, в отличие от систем ПиУП, требуется только информация, необходимая для организационной оптимизации процессов или поддержки workflow. Однако в системах ПиУП можно использовать гораздо более детальные атрибуты и типы данных.

В то время как модель выходов представляет продукты с логической точки зрения, модель данных дает физическое описание. Тем не менее следует отметить, что между моделью выходов ARIS и описаниями материальных выходов в системах ПиУП и САПР существуют тесные связи. Например, в организациях, где отсутствует избыточность, экземпляры выходов системы ПиУП можно использовать для моделирования экземпляров на базе ARIS. И наоборот, ARIS Toolset можно применять для ввода и представления данных в системы, обслуживающие прейскуранты материалов .

A.2.4.2. Конфигурирование выходов

Модель выходов процесса должна фокусировать системы пооперационного исчисления стоимости на определенных стоимостных факторах. Применительно к управлению мощностями модели выходов используются для описания областей выхода.

В частности, системы управления потоками работ (workflow) требуют, чтобы транспортируемые папки были представлены в виде информационных услуг. Информация об экземплярах создается путем копирования описаний классов со 2-го уровня.

На промышленных предприятиях материальный выход представлен в виде прейскурантов материалов. Благодаря частичному пересечению моделей выходов и данных, если отсутствуют дополнительные уточнения, связанные с заказом, то при описании экземпляров продуктов в моделях workflow достаточно сослаться на экземпляры в системе ПиУП.

С другой стороны, если необходимость в уточнениях, связанных с заказом, все же возникает, в экземплярах workflow создаются и поддерживаются дополнительные атрибуты.

Модели выходов используются при конфигурировании бизнес-приложений с глобальным фокусированием на соответствующих процессах. Процессы могут быть различными в зависимости от того, является ли компания поставщиком услуг, предприятием розничной торговли или изготовителем продуктов. Программа SAP R/3 Business Engineer требует информации о типах выходов для первых двух уровней системной конфигурации — для описания отраслевых сценариев и определения вариантов необходимых процессов (см. рис. 83 и 84).

В бизнес-приложениях, которые не управляются workflow, модель выходов явным образом не документируется. Объекты информационных услуг содержатся в модели данных в виде объектов данных, но они не представляют собой результат явным образом описанной функции или процесса.

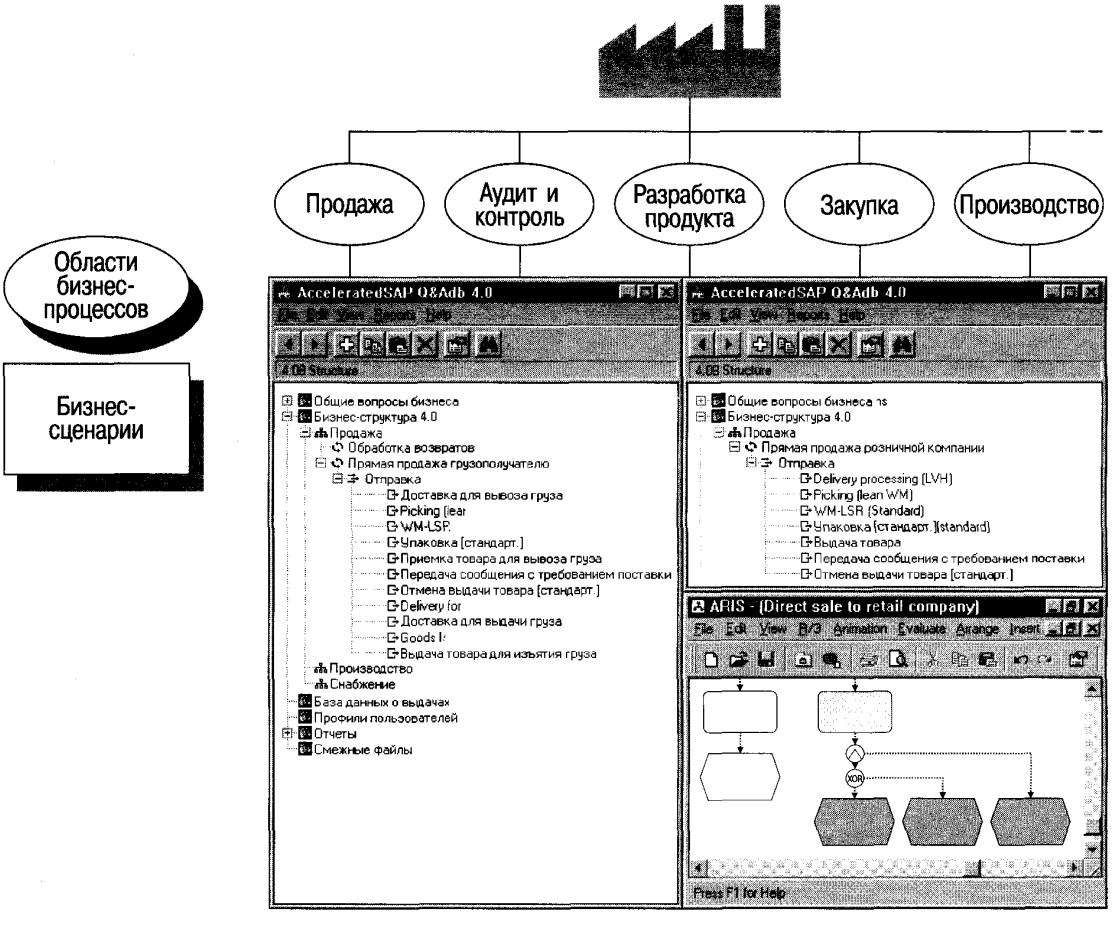


Рис. 83. Выбор бизнес-сценариев *(Schröder. Business Engineer. 1997)*

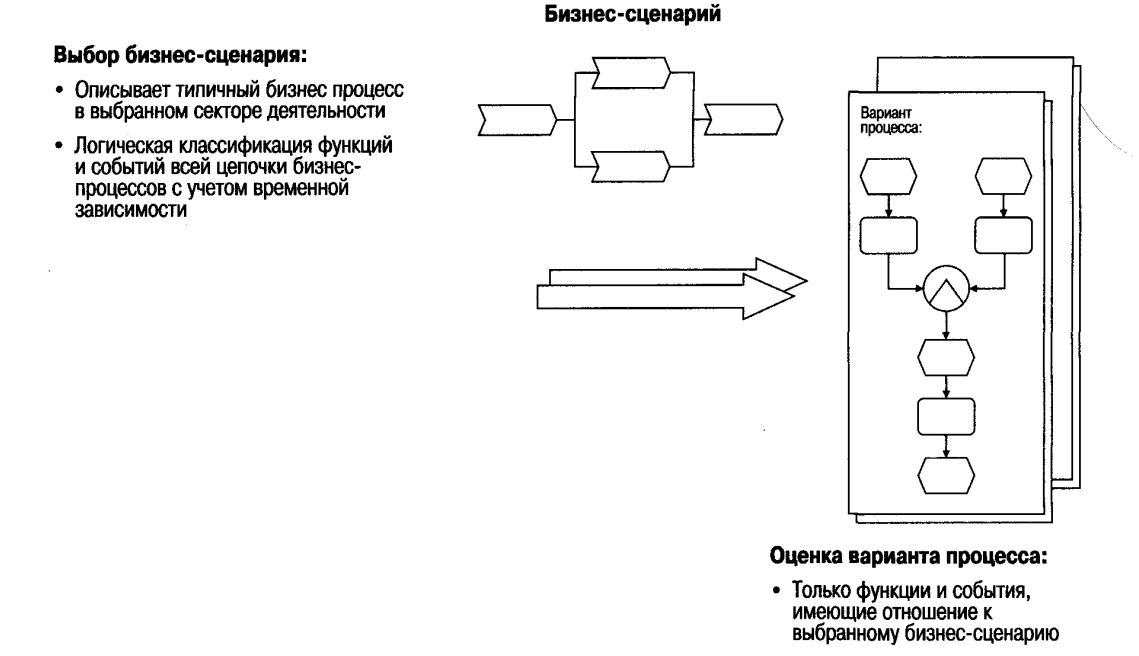


Рис. 84. Конфигурирование процессов *(Schröder. Business Engineer. 1997)*

A.3. Моделирование отношений между разными типами представлений (модель управления)

Задача моделирования управления заключается в том, чтобы объединить в одно целое разные типы представлений (функций, организации, данных и выходов), которые до сих пор рассматривались по отдельности.

На этом этапе моделируются структурные отношения и динамическое поведение системы, описываемое изменениями состояния. Сначала мы рассмотрим методы описания попарных отношений между типами представлений, а затем поговорим о том, как все эти представления можно связать воедино. В рамках этой классификации мы вновь применим модель жизненного цикла, включающую этапы определения требований, конфигурирования, спецификации проекта и описания реализации.

A.3.1. Отношения между функциями и организацией

На рис. 85 показаны отношения между функциональной и организационной моделями.

На рис. 86 приведено базовое отношение между бизнес-функцией и организационной единицей. Однако эта связь может иметь разные семантические ответвления. Кроме того, возможны различные исходные модели и разграничения, связанные с распределением функций.

A.3.1.1. Моделирование определения требований

A.3.1.1.1. Диаграммы связи функция–организация

При описании функций и организации возможны различные уровни детализации.

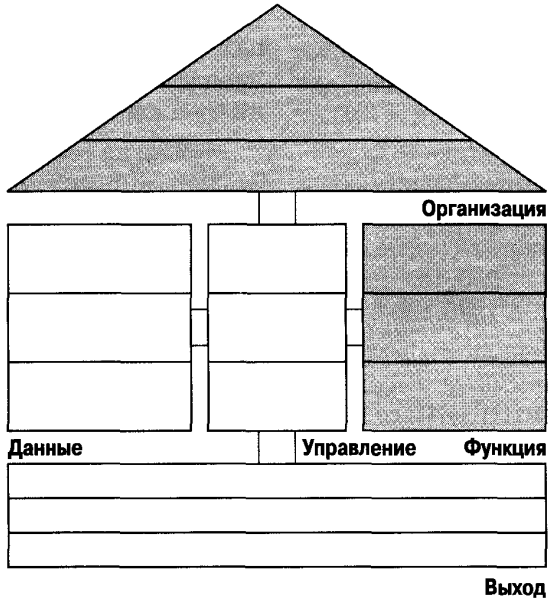


Рис. 85. Отношения между функциями и структурой организации

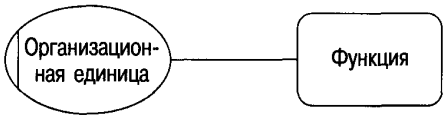


Рис. 86. Общее отношение между организационной единицей и функциями

В функциональных моделях ключевые функции процессов привязываются к соответствующим организационным единицам. Рис. 87 иллюстрирует привязку функций к уровням планирования на примере промышленного предприятия, приведенном на рис. 49. Для иллюстрации взяты функции материально-технического обеспечения, разработки продуктов и управления.

На рис. 88 распределение функций представлено в виде часто используемого матричного описания. Пример отражает фрагмент реальной процедуры обработки заказов в отрасли, производящей оборудование. В данном случае вся процедура обработки заказов состоит приблизительно из 150 функций без разграничения процессов, поддерживаемых ИТ, и процессов, выполняемых вручную.

Если в выполнении функции участвует несколько организационных единиц, можно уточнить тип участия каждой из них, дополнительно указав, какие организационные единицы несут ответственность за данную функцию, какие активно участвуют в ее реализации, а какие лишь оказывают содействие. Другие примеры из реальной практики содержатся в работе: *Martin. Information Engineering II. 1990, с. 58.*

Функциональную модель и матрицу распределения функций можно представить отношением АССОЦИАЦИЯ ФУНКЦИИ, как показано в метамодели на рис. 89. Эта ассоциация относится к основным функциям.

Каждую организационную единицу, участвующую в бизнес-процессе, можно определить путем связывания функций с соответствующими процессами. И наоборот, каждую функцию и цепочку процессов можно определить с точки зрения организационной единицы.

Что же касается мощностей отношений, то мы исходим из посылки, что каждая организационная единица участвует по крайней мере в одной функции и каждая функция присваивается по крайней мере одной организационной единице. С другой стороны, мощности отношений показывают, что каждая функция может выполняться несколькими организационными единицами и функции могут распределяться по нескольким уровням планирования в рамках процесса. Это описание отражает иерархическую структуру процессов и пути реализации идентичных функций (например, проверки наличия ресурсов) при различных графиках планирования, объектах-прототипах (заказах, операциях) и на разных уровнях (производство, корпоративные секторы, группы фондов и т.д.).

Можно не только детализировать связи основных функций с предварительными уровнями, но и привязывать пользовательские транзакции к исполнительским должностям в рамках организации или к конкретным пользователям (см. рис. 90). В этом контексте мы употребляем термин «пользовательская транзакция», введенный при обсуждении функциональных моделей. Этот термин, в свою очередь, связан с понятием «должность», введенным на уровне организационного моделирования. Это обеспечивает доступ к конкретным пользовательским транзакциям различным исполнителям, занимающим соответствующие должности. Можно также

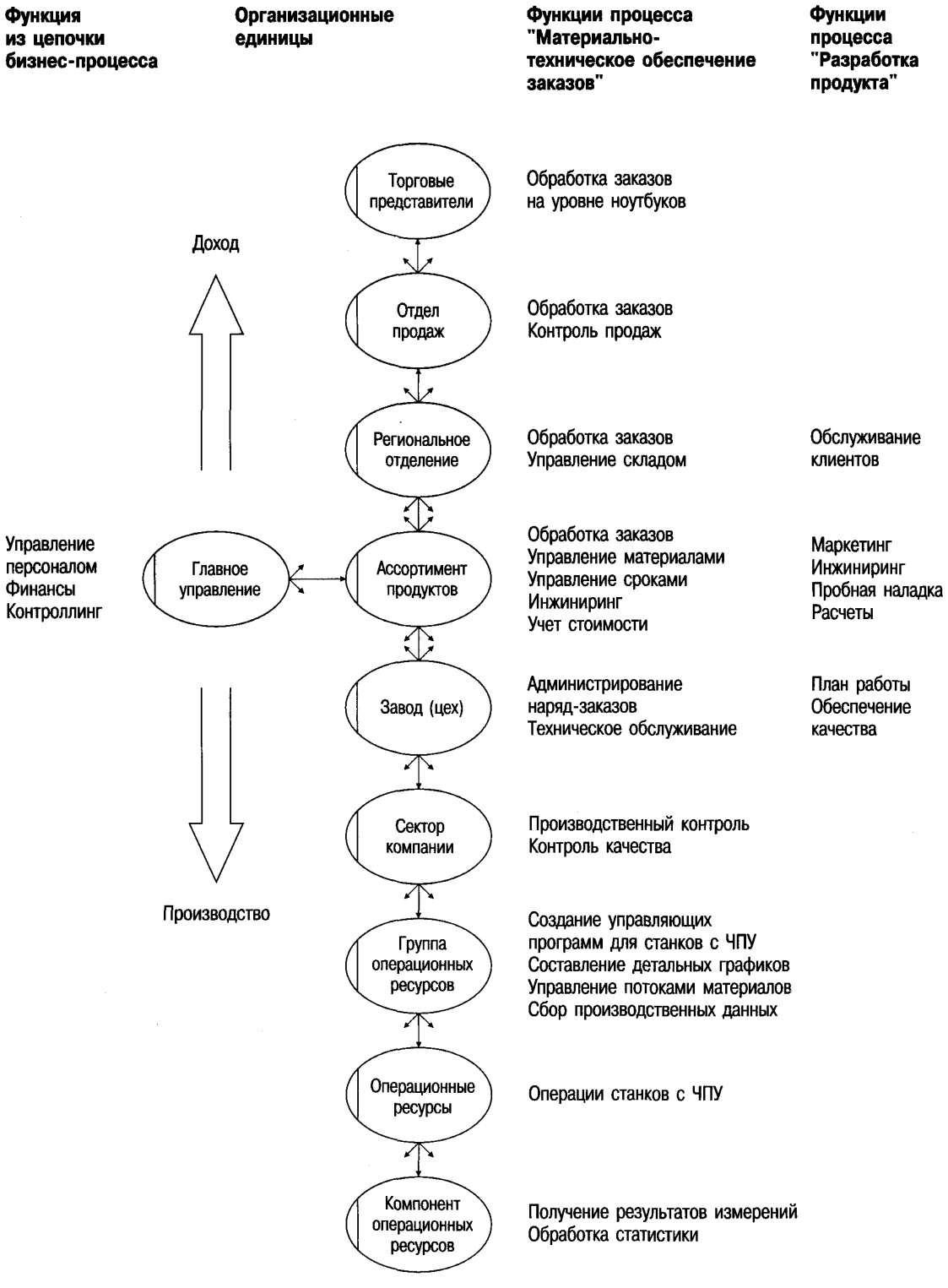


Рис. 87. Модель на уровне функций

Организационные единицы / Функции	Управление	Маркетинг	Управление и организация	Управление филиала	Людские ресурсы	Учет и контроль стоимости	Продажа	Продажа и дистрибуция	НИОКР	Производство	Закупки/ снабжение	Склад материалов
Анализ рынка	у	о	с	с			у					
Планирование производственной программы	о	у	у	с	с	с	у		с	с	с	
Обработка предложений							о					
Обработка заказов							о					
Разработка продуктов	у			с	с	у	у		о	у	у	
Производственное планирование			у		у	с	у		у	о	у	с
Закупка материалов											о	у
Управление складом								с				о
Производственное управление и контроль						с	с					
Обеспечение качества				с			с	с	у	о	у	о
Отправка				у			у	о				
Учет и контроль стоимости	с		с	у	с	о			с	с	с	
Финансовое и инвестиционное планирование	у		о	с	с	у				с	с	
Планирование и совершенствование людских ресурсов	у		у	с	о	с				с	с	
Инвентаризация и подведение баланса на конец года	у		с	у		о				с	с	

о = несет ответственность у = активно участвует с = оказывает содействие

Рис. 88. Матричное описание распределения функций

(хотя это и необязательно) присваивать должностям полномочия на исполнение нескольких пользовательских транзакций; минимальное значение мощности при этом равно 0.

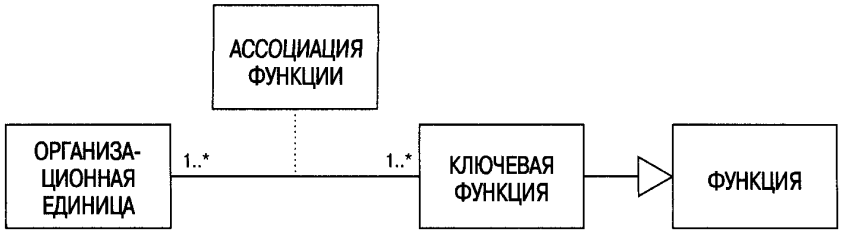


Рис. 89. Диаграмма классов для распределения функций

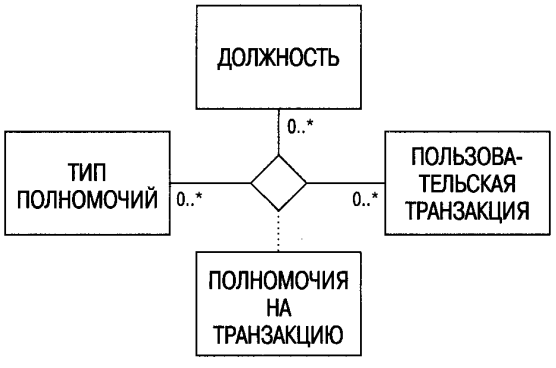


Рис. 90. Привязка транзакций к конкретным пользователям

A.3.1.1.2. Диаграмма взаимодействия

Диаграммы взаимодействия являются частью языка UML и в настоящее время получают развитие в расширенных версиях UML, ориентированных на конкретные процессы.

Диаграммы взаимодействия описывают, каким образом организационные единицы в качестве «субъектов действия» взаимодействуют с функциями. Понятие «диаграмма взаимодействия» несколько расплывчато и включает описание лишь части бизнес-процесса, выполняемой за одну операцию, т.е. без каких-либо существенных разрывов во времени или пространстве. Диаграммы взаимодействия позволяют «взять в свои руки» такие сложные вопросы, как бизнес-процессы. Пример диаграммы взаимодействия приведен на рис. 91 (дополнительные примеры можно найти в работе: *Oestereich. Objektorientierte Softwareentwicklung. 1997, с. 215).*

Диаграммы взаимодействия «обрамляют» конкретную ситуацию и привязывают к ней другие ситуации. Каждая функция взаимодействия, обозначаемая овалом, соответствует описанию элементарной функции. Субъекты действия и функции связаны линиями «коммуникации». Каждое обращение к приложению пронумеровано. Связи между приложениями, предполагающие, например, что одно приложение может включать (использовать) другое, представлены пунктирными линиями. На рис. 91 это показано стрелкой между размещением заказа и проверкой состояния.

С точки зрения ARIS, диаграммы взимодействия являются связующим звеном

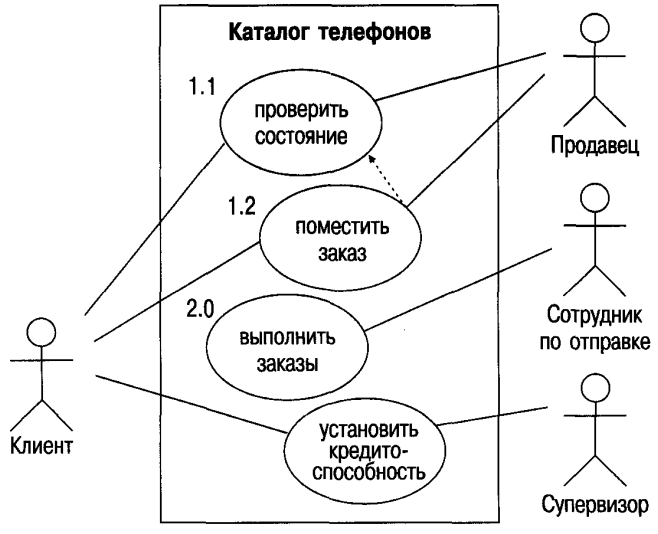


Рис. 91. Диаграмма взаимодействия *(UML Notation Guide. 1997, рис. 33)*

между организационной моделью (описывающей субъектов действия) и функциональной моделью. В соответствии с этим они включены в метамодель на рис. 89. На рис. 92 диаграмма взаимодействия усовершенствована в результате группировки отдельных взаимодействий. Такие диаграммы часто дополняются текстовым описанием и детализируются с помощью диаграмм последовательностей (см. ниже).

A.3.1.2. Конфигурирование

Распределение функций между организационными единицами играет ключевую роль в конфигурировании систем.

При анализе функций в процессе пооперационного исчисления стоимости необходимо привязать функции к организационным единицам как к стоимостным центрам. Это имеет важнейшее значение для вычисления ставок стоимости функций в стоимостных центрах.

В планировании мощностей распределение функций определяет базовый контекст, показывая, с какими единицами мощностей связаны те или иные функции бизнес-процесса.

Управление workflow опирается на связи функций с организационными единицами, определяя, в какой «входной почтовый ящик» рабочий поток помещает ту или иную функцию. Назначения функций («информировать», «участвует», «уполномочен подписывать», «обработано» и т.д.) активизируются соответствующей функцией.

В макроконфигурациях связи организация-функция определяют степень децентрализации стандартной прикладной системы. Нередко это делается с помощью программ, которые определяют, будет ли конкретное требование системы ПиУП удовлетворяться централизованно, на уровне предприятия, или децентрализованно, на уровне подразделения. Чтобы обеспечить более высокую степень организационной гибкости, необходимо свободное конфигурирование связей организация-функция.

В микроконфигурациях связи организация-функция управляют способом представления системных функций пользователям, определяя степень интег-

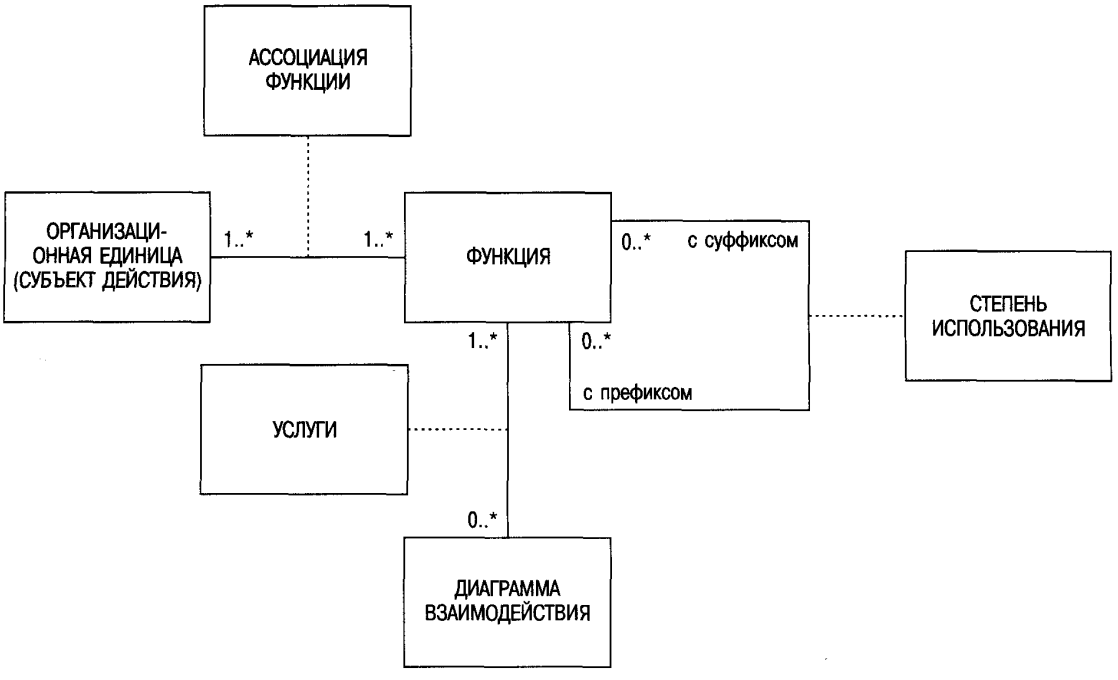


Рис. 92. Метамодель диаграммы взаимодействия

рации процесса с каждым рабочим местом. На рис. 93а-г приведен пример управления материалами, показывающий организационные структуры и соответствующие диаграммы EPC.

В организации, описанной на рис. 93а и б, отдел приемки товаров и склад отделены друг от друга, а прием товаров осуществляется разными людьми. В результате перед складом, где приемщик отбирает поступающие товары, находится «зона неразобранного товара». Это обусловливает необходимость двух разных пользовательских интерфейсов для приемки товаров, каждый из которых предоставляет соответствующему сотруднику функции выбора товара для обработки. В организации же, представленной на рис. 93в и г, отдел приемки товаров и склад объединены, вследствие чего необходим лишь один пользовательский интерфейс для приемки товаров, а второй процесс выбора становится избыточным. Функциональный процесс в обоих случаях идентичен, за исключением того, что распределение функций в рамках каждой из этих организаций влечет за собой разные процессы, что поразному проявляется и в оформлении интерфейса, и в управлении выбором.

Конфигурации системы, предназначенные для стандартного программного обеспечения, должны объединять пользовательские транзакции и экраны в соответствии с распределением функций.

Это требует от них большой гибкости организационных функций, чтобы поддерживать продуктивность конкретных пользователей, обеспечивая индивидуальную настройку управления транзакциями и пользовательского интерфейса. По этой причине одним из ключевых аспектов вычислительной системы, ориентированной на пользователя, является возможность индивидуальной настройки

Рис. 93а. Пример управления материалами для функций, разделенных в рамках организации

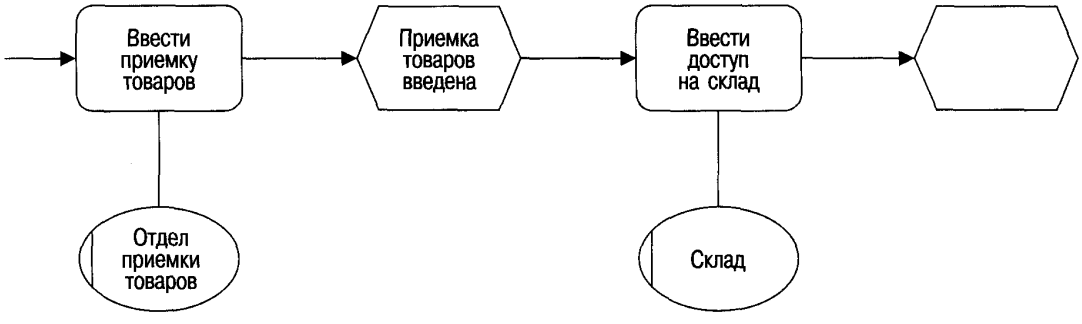


Рис. 93б. Диаграмма EPC для примера, приведенного на рис. 93а

связи функция-организация. Такие функциональные возможности уже предоставляются стандартным программным обеспечением на базе модели *(IDS. ARIS–Applications. 1997)*.

Полномочия, описанные на уровне определения требований, помогают сконфигурировать прикладные системы. Некоторые типы полномочий разрешают пользователям запускать определенные модули, вызывать входные или выходные экраны, подписываться на списки рассылки.

На рис. 94 показано два способа конструирования ПОЛНОМОЧИЙ. Различные типы полномочий называются ПРОГРАММНЫМИ ОБЪЕКТАМИ и представляют собой общие версии МОДУЛЕЙ, ЭКРАНОВ и СПИСКОВ. Если функции полномочий связываются с каждым пользователем через матрицу полномочий, ПОЛНОМОЧИЯ образуют ассоциацию между классами ТИП ПОЛНОМОЧИЙ, ОРГАНИЗАЦИОННАЯ ЕДИНИЦА (ПОЛЬЗОВАТЕЛЬ) и ПРОГРАММНЫЙ ОБЪЕКТ. Таким образом, пользователи с идентичными профилями полномочий описываются индивидуально, что делает администрирование громоздким и избыточным.

Другой способ, показанный на рисунке, позволяет избежать избыточности. Сначала некоторые профили полномочий привязываются к ПАРОЛЮ через ПОЛНОМОЧИЯ ПАРОЛЯ, который, в свою очередь, связывается с пользовательскими группами или отдельными пользователями при помощи ассоциативного класса АССОЦИАЦИЯ ПОЛЬЗОВАТЕЛЬ–ПАРОЛЬ. Та-

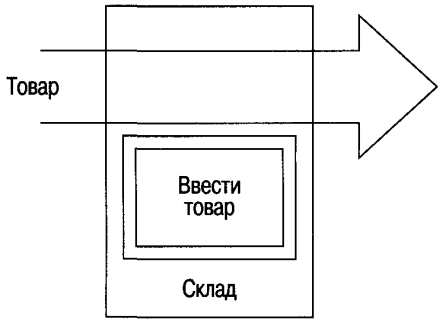


Рис. 93в. Пример управления материалами для функций, объединенных в рамках организации

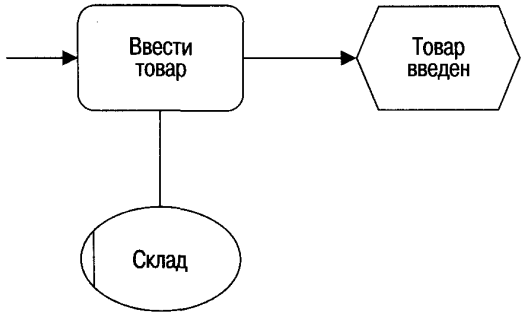


Рис. 93г. Диаграмма EPC для примера, приведенного на рис. 93в

кое косвенное описание существенно снижает избыточность информации.

A.3.1.3. Спецификация проекта

На стадии спецификации проекта ПРОГРАММНЫЕ ОБЪЕКТЫ (модули, бизнес-объекты, пользовательские транзакции) привязываются к конкретным УЗЛАМ компьютерной сети. Они могут влиять на физические системы хранения или физический доступ, используя стандартные методы (удаленные вызовы процедур, CORBA, DCOM и т.д.) или оригинальные методы доступа, такие как, например, удаленный вызов функций (RFC) или ALE в приложениях SAP. На рис. 95 некоторые из этих вариантов представлены классом ТИП АССОЦИАЦИИ.

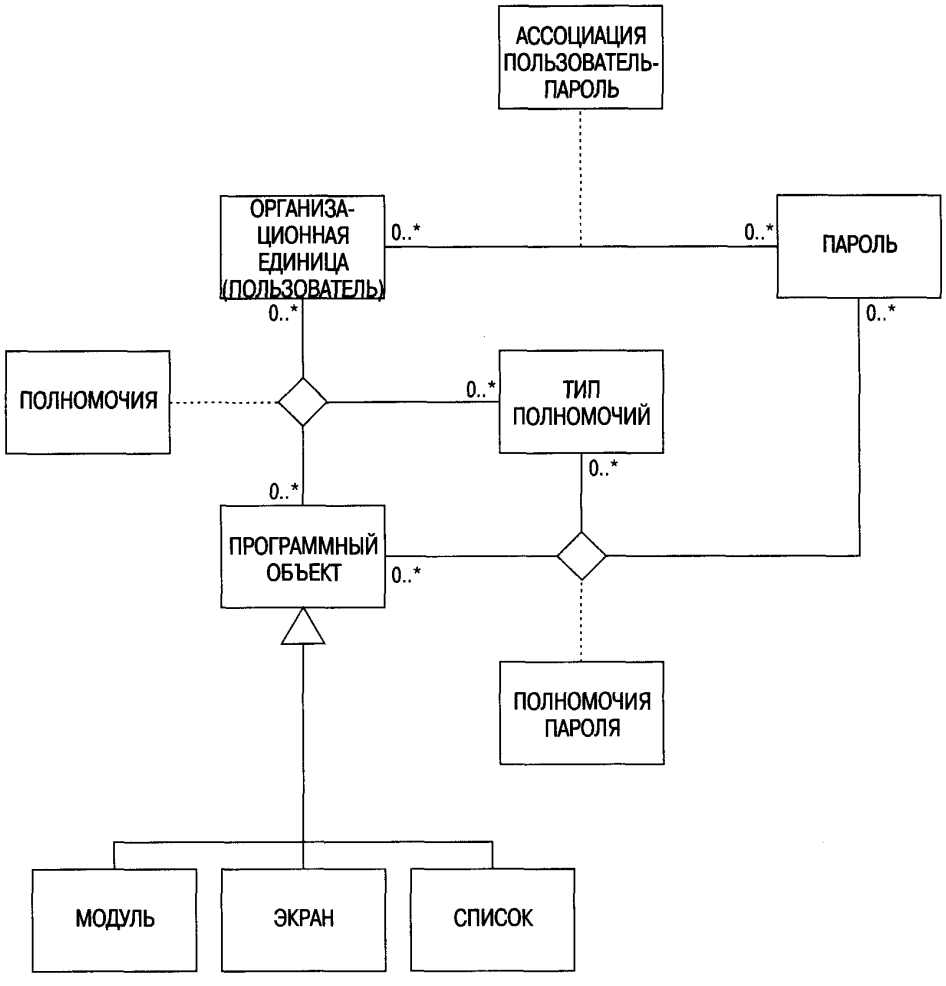


Рис. 94. Конфигурирование полномочий

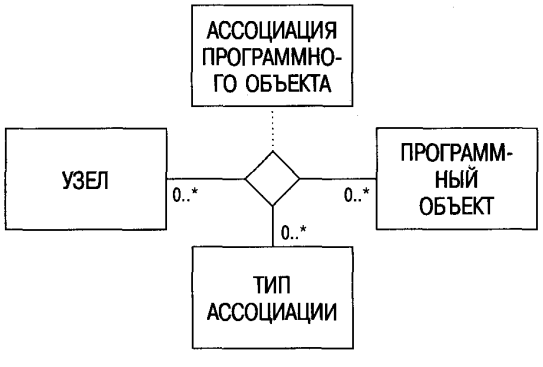


Рис. 95. Описание хранения и доступа

A.3.2. Отношения между функциями и данными

На рис. 96 показано место функций и данных в здании ARIS.

Формируя представление данных на основе общей ARIS-модели бизнес-процессов, мы можем выделить два вида отношений между функциями и данными:

- функции обрабатывают данные, преобразуя входные данные в выходные;
- события представляют собой модификации состояния (данных), порождаемые функциями. Сообщения указывают, что обнаружены модификации состояния, передают их последующим функциям, а затем активизируют их;

Первый вид отношений данные-функции положил начало многим методам проектирования, где данные и функции тесно взаимосвязаны. Это относится к таким традиционным методам, структурированный анализ и проектирование (SADT) или диаграммы ДеМарко, а также к объектно-ориентированным диаграммам классов.

Ключевым моментом в описании поведения системы является событийное управление функциями.

A.3.2.1. Моделирование определения требований

A.3.2.1.1. Установление связей между функциями и данными

A.3.2.1.1.1. Объектно-ориентированные диаграммы классов

Диаграммы классов описывают структуру системы при объектно-ориентированном проектировании бизнес-процессов.

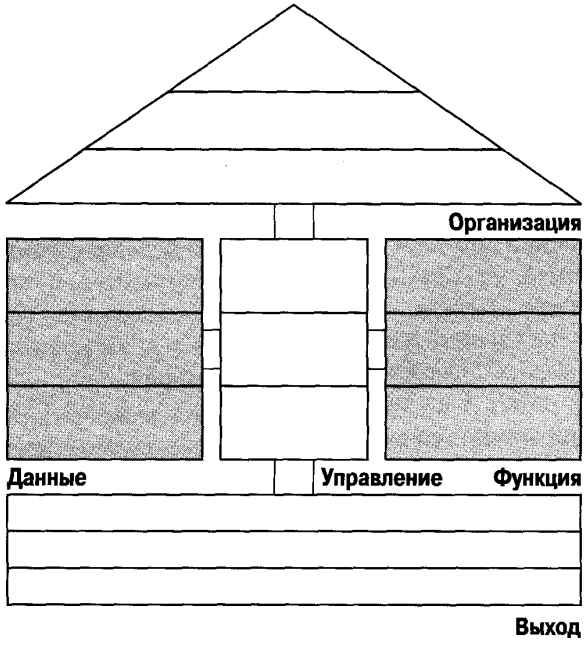


Рис. 96. Отношения между функциями и данными

Классы описываются посредством их определения, атрибутов и применяемых методов. Мы можем приравнять понятие «метод», используемое в объектно-ориентированном анализе, к понятию «функция». Поскольку речь часто идет о классах данных (КЛИЕНТЫ, ПОСТАВЩИКИ, ЗАКАЗЫ и т.д.), классы являются связующим звеном между моделью данных и моделью функций. Диаграммы классов описывают также ассоциации между классами и их взаимные ограничения.

Поведение системы описывается динамическими моделями, содержащими обмен сообщениями между объектами или взаимодействия между методами в рамках одного объекта. Эту тему мы рассмотрим в разделе, посвященном управлению посредством событий.

Мы уже обсуждали некоторые свойства объектно-ориентированного проектирования классов при описании отдельно взятой модели данных, а поскольку проектирование бизнес-процессов во многом имеет родственный характер, здесь мы можем ограничиться беглым рассмотрением свойств, отличающих создание объектно-ориентированных классов.

Объектно-ориентированный анализ (OOA) пока не относится к числу стандартизированных методов. Просто есть ряд авторов, которые разработали похожие или взаимно дополняющие подходы *(Coad, Yourdon. Object-Oriented Analysis. 1991; Booch. Object-Oriented Design. 1991; Rumbaugh et al. Object-Oriented Modeling and Design. 1991; Jacobsen. Object-Oriented Software Engineering. 1996)*. Они пользуются разными графическими символами, что затрудняет сравнение их методов.

Унифицированный язык моделирования (UML), введенный в работах Рамбо, Буча и Джекобсена, позволяет упростить методы OOA, поэтому мы тоже воспользуемся этой системой обозначений.

Объекты, каждый из которых индивидуален и имеет свой идентификатор, описываются при помощи свойств (атрибутов). Их поведение определяется применимыми к объекту функциями (методами). Объекты представляют экземпляры и обозначаются прямоугольниками (см. рис. 97а).

Объекты с идентичными атрибутами, функциональностью и семантикой группируются в классы объектов, или регулярные классы. Таким образом, совокупность клиентов образует класс КЛИЕНТ (см. рис. 97б).

С помощью атрибутов и методов классы определяют свойства и поведение своих экземпляров, т.е. объектов. Поскольку атрибуты и методы образуют единый блок, классы реализуют принцип инкапсуляции. Помимо описания атрибутов и методов для объектов, мы можем также использовать атрибуты и методы, действительные только для самих классов, но не для составляющих их объектов. Примерами могут служить «число клиентов» и «создание нового клиента».

Одним из важных свойств объектно-ориентированного подхода является наследование, обеспечивающее классам доступ к свойствам (атрибутам) и поведению (методам) других классов (см. рис. 98). Унаследованные атрибуты и методы могут быть перезаписаны и переопределены наследующим классом.

Наследование реализуется в рамках иерархии, содержащей два типа классов: доминирующие и подчиненные. Наглядной иллюстрацией служат операции обобщения и конкретизации при моделировании данных.

Класс может также наследовать свойства нескольких доминирующих классов (множественное наследование). В результате построенная диаграмма классов образует сеть.

Помимо обобщающих отношений между классами, существуют отношения (ассоциации) между объектами одноранговых классов, а также между объектами одного и того же класса. Эти ассоциации при-

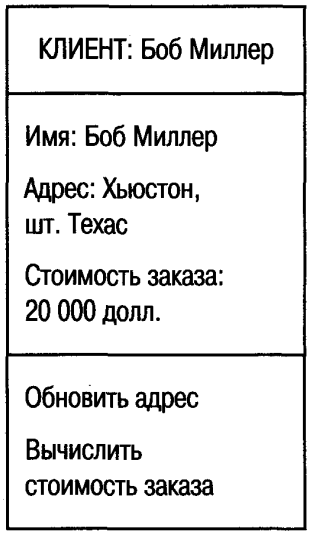


Рис. 97а. Описание объекта

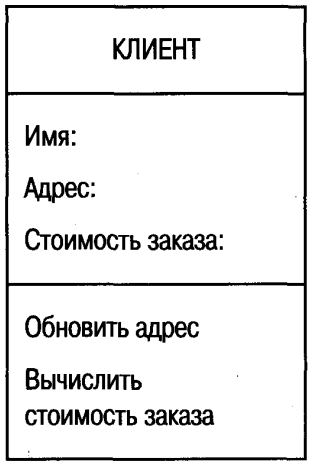


Рис. 97б. Класс (объектов)

равниваются к отношениям в моделях ERM, хотя здесь они изображаются только одной линией (см. рис. 99а).

Диаграммы следует читать слева направо. Ассоциации присваиваются соответствующие мощности.

Каждому концу ассоциации можно присвоить ролевое имя (покупатели, имеющиеся изделия, см. рис. 99а). Если ассоциации содержат атрибуты, то они изображаются в виде классов (см. класс ПРОЦЕСС ПОКУПКИ на рис. 99б). Для моделей ERM такая дифференциация необязательна, поскольку каждая ассоциация (отношение) может иметь собственные атрибуты.

Особым видом ассоциации является агрегация, описывающая отношение «часть

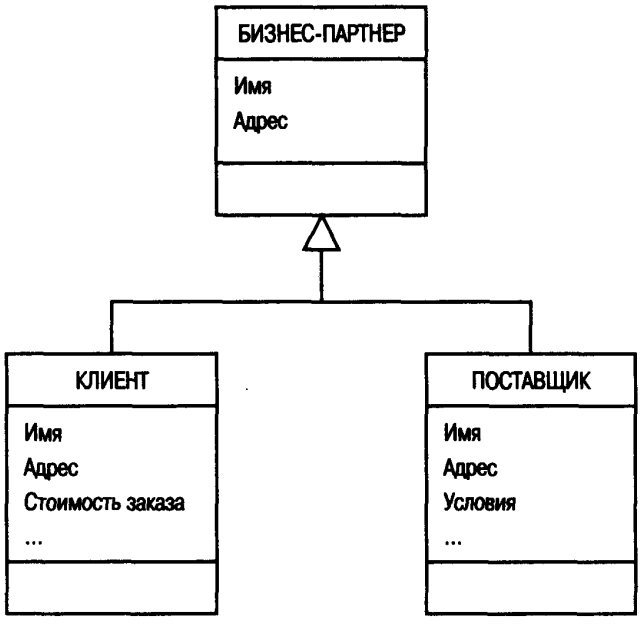


Рис. 98. Наследование

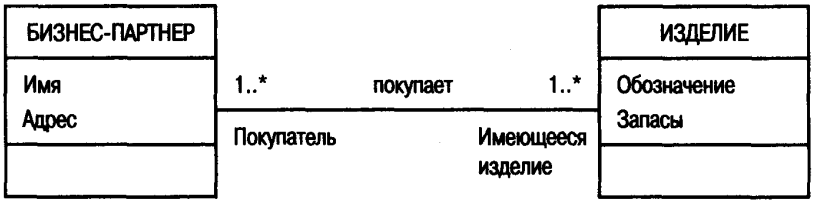


Рис. 99а. Ассоциация

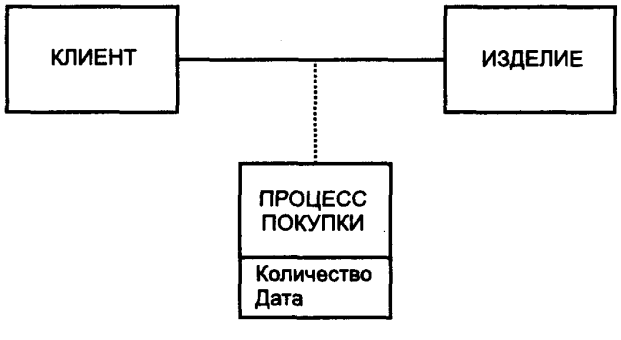


Рис. 99б. Ассоциация как класс

целого» между объектами двух разных классов. На рис. 100 она представлена классами ЗАКАЗ и ПОЗИЦИЯ ЗАКАЗА.

Агрегациям также можно присваивать ролевые имена. Если агрегации присваиваются атрибуты, мы получаем класс, примером которого может служить класс СТРУКТУРА на рис. 100, связанный отношением агрегации с прейскурантом материалов.

В отличие от ассоциаций, агрегациям присущи иерархии между классами, поэтому их также называют направленными ассоциациями.

Определение классов, наследование, ассоциации и агрегации - ключевые структурные аспекты объектно-ориентированного анализа.

На рис. 101 представлена предварительная метамодель структурной диаграммы классов, в центре которой - понятие КЛАСС. Поскольку классы обычно являются классами данных, они связаны с понятием ИНФОРМАЦИОННЫЙ ОБЪЕКТ модели данных ARIS отношением 1:1. Кроме того, классам присваиваются АТРИБУТЫ ИНФОРМАЦИОННЫХ ОБЪЕКТОВ. Методы, привязанные к классам, взяты из модели функций.

Эти связи относятся к высшему классу иерархии наследования. При этом наследование и образование подклассов обозначаются ассоциацией НАСЛЕДОВАНИЕ.

«Независимая ассоциация» и «направленная ассоциация» дифференцируются при помощи ТИПА АССОЦИАЦИИ, который соединяется непосредственно с АССОЦИАЦИЕЙ. Некоторые ассоциации имеют множество значений, поэтому максимальные диапазоны их мощностей обозначаются символом *. Ассоциативные классы квалифицируются как конкретизированные (специализированные) классы, поскольку ассоциации, представляющие собой формирующие факторы атрибутов, могут также выступать в качестве классов.

Предлагаемый метод UML позволяет детально описывать метамодели, используя фрагмент языка UML. В целом это касается диаграмм классов (обычно без указания методов), ассоциаций и пакетов, рассматриваемых в настоящей книге. Под

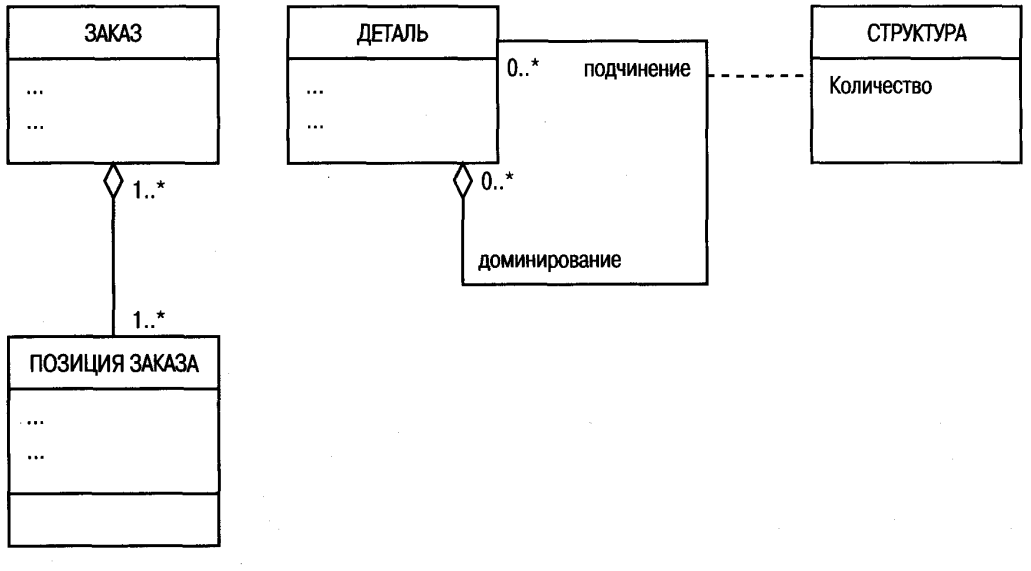


Рис. 100. Агрегации

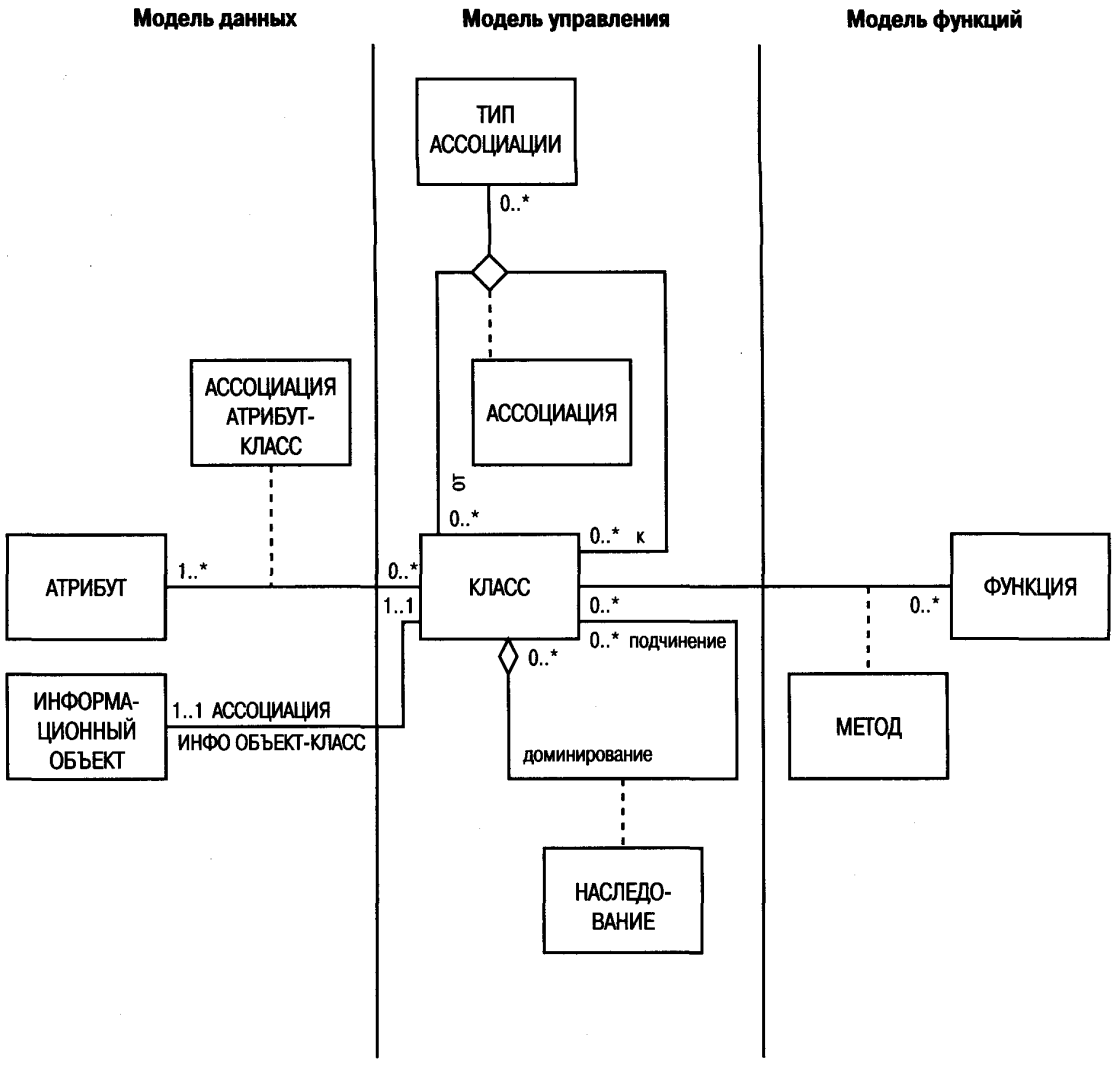


Рис. 101. Метамодель структуры объектно-ориентированного анализа

«пакетами» здесь понимаются сгруппированные объекты модели.

Обобщенные области (базовые понятия) метаописаний включают также информацию, которая в концепции ARIS отнесена к мета2-уровню (Scheer. ARIS — Business Process Frameworks. 1998, с. 120–125; в русском издании с. 109-115). Изображенная на рис. 102 метамодель диаграмм классов на языке UML наглядно показывает принцип описания и степень детализации.

Метамодели ARIS в отличие от метамоделей UML обладают следующими особенностями:

1. Метамодели ARIS повышают структурирования, так какони проектируются в соответствии с моделями и жизненным циклом ARIS.
2. Метамодели ARIS не сводятся лишь к объектно-ориентированным методам и обеспечивают более широкий подход.

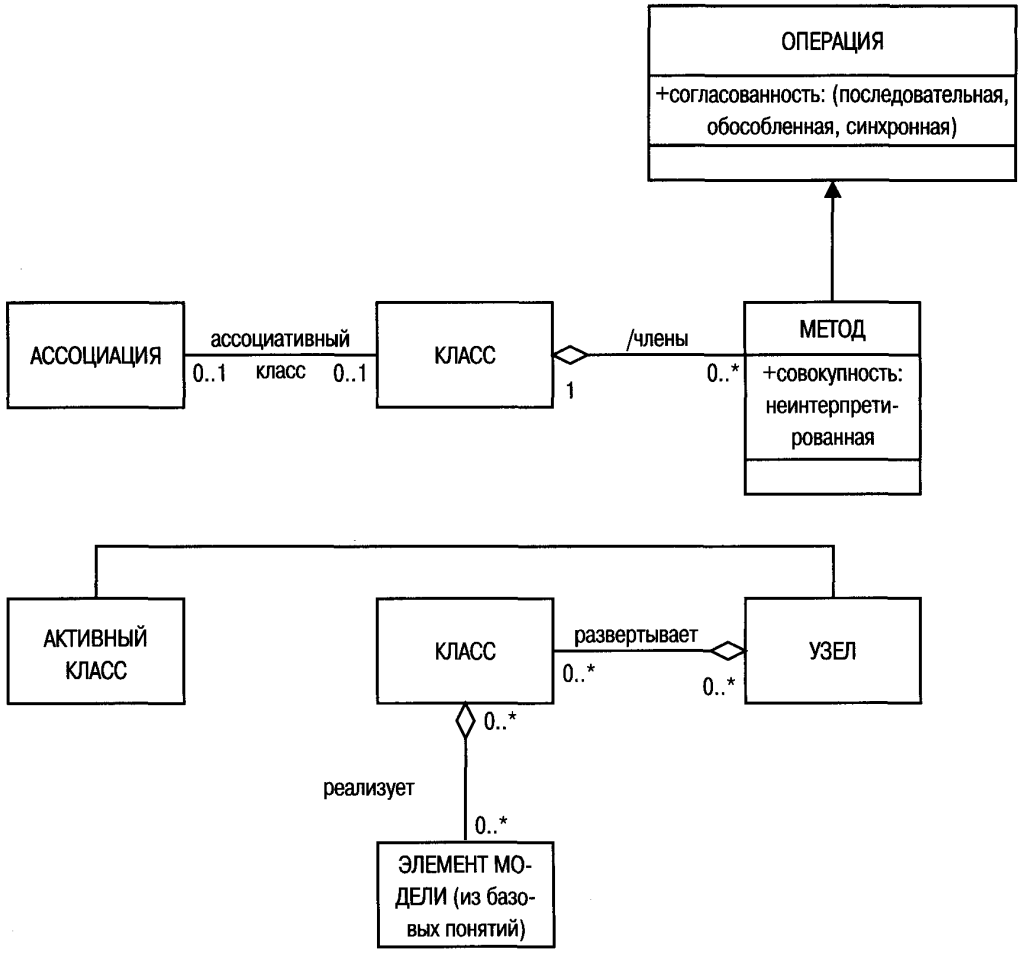


Рис. 102. Метамодель описания класса на языке UML (UML Semantics. 1997, с. 35, 44)

3. Метамодели ARIS расширяют возможности моделирования и диапазон бизнес-ориентированных приложений, включая в него, например, стратегическое планирование.

Таким образом, метамодели ARIS являются шагом вперед по сравнению с метамоделями UML, которые больше сфокусированы на процессе реализации.

A.3.2.1.1.2. Диаграммы привязки функций

В диаграммах классов классы данных четко привязываются к соответствующим функциям. Если необходимо моделировать функции независимо от данных, то устанавливается ассоциация *:*. Таким образом, функцию можно привязать к нескольким объектам данных. Этот вид администрирования позволяет создавать функциональные объекты, к которым также применимы принципы наследования. Проектирование безызбыточных, объектно-ориентированных описаний возможно в тех случаях, когда объекты данных и функциональные объекты создаются независимо друг от друга и допускают произвольный выбор связей. Кроме того, этот метод создает свя-

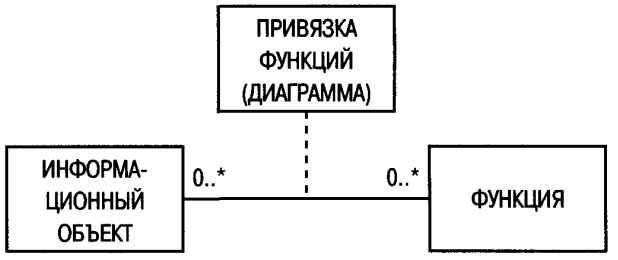


Рис. 103. Метамодель диаграммы привязки функций

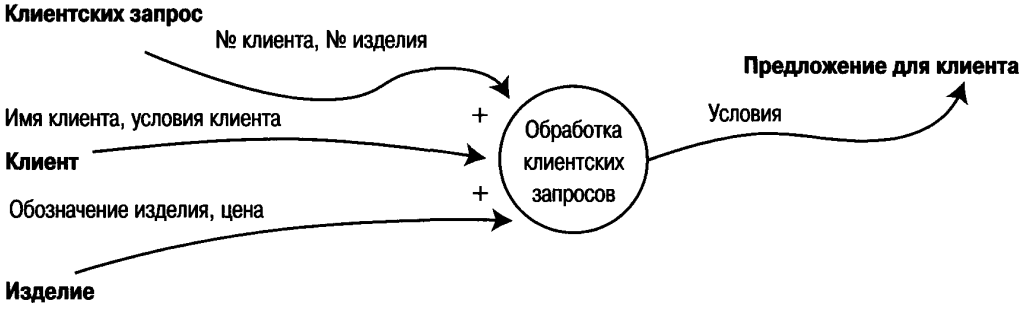


Рис. 104. Диаграмма потоков данных, предложенная ДеМарко для обработки клиентских запросов

зующее звено между моделированием, ориентированным на тип представления, и более традиционным объектно-ориентированным моделированием.

На рис. 103 показана метамодель диаграммы привязки функций.

А.3.2.1.1.3. Поток данных

На объектно-ориентированных диаграммах классов ключевыми элементами являются классы данных. К ним привязываются соответствующие методы.

Однако на диаграммах потоков данных ключевыми элементами являются функции, преобразующие входные данные в выходные. В традиционном функционально-ориентированном программировании диаграммы потоков данных с различной степенью детализации играют важнейшую роль в проектировании на уровне определения требований.

На рис. 104 приведена диаграмма потоков данных, предложенная ДеМарко, которая описывает функцию обработки клиентских запросов. Информационные объекты обозначены двумя горизонтальными чертами. Рядом со стрелками указаны ключевые атрибуты, необходимые для данной функции. Уорд и Меллор расширили этот подход, включив в него системы реального времени *(Ward, Mellor. Real-Time Systems. 1985)*.

В метамодели данных, изображенной на рис. 105, поток данных моделируется ассоциативным классом ОПЕРАЦИЯ, который располагается между ФУНКЦИЕЙ и АССОЦИАЦИЕЙ ОБЩЕГО АТРИБУТА. Кроме того, класс ОПЕРАЦИЯ позволяет более подробно описать операции, которые бизнес-функция может применить к атрибутам.

Эти операции включают:

- создание элементов данных;
- удаление элементов данных;
- обновление элементов данных;
- считывание элементов данных (только для чтения).

Для каждого элемента данных следует постараться выбрать максимально высокий уровень операций. При этом остальные элементы данных включаются автоматически. Такие привязки часто описываются с помощью таблиц *(Martin. Information Engineering II. 1990, с. 272).*

Для каждого типа операции создается отдельный класс — ТИП ОПЕРАЦИИ. При этом ОПЕРАЦИЯ становится связующим звеном между ТИПОМ ОПЕРАЦИИ, ФУНКЦИЕЙ и АССОЦИАЦИЕЙ ОБЩЕГО АТРИБУТА.

Операции, выполняемые той или иной функцией, можно логически связать друг с другом, как показано на диаграмме Де-Марко. Например, в функции могут быть необходимы несколько полей данных, в результате чего для функции считывания между ними устанавливается отношение «И». Эти варианты связей моделируются ассоциативным классом СВЯЗЬ между операциями. ТИПЫ ОТНОШЕНИЙ (где в данном случае возможны булевы операции) указывают на способ связи операций. Однако следует отметить, что установить все возможные логические отношения между входящими и исходящими элементами данных нереально.

Некоторые методы потоков данных, помимо функционально-ориентированного представления, включают представление, ориентированное на данные. Например, в методе SADT оба типа моделей описываются с помощью понятий «функциональный блок» и «блок данных». Функциональный блок устанавливает отношения с входящими, управляющими и исходящими данными, при этом правила преобразования, т.е. применяемые операции, описываются процессорами.

Блок данных рассматривается с точки зрения информационного объекта и показывает, с помощью каких функций создан этот информационный объект и в каких еще функциях он используется.

A.3.2.1.1.4. Ассоциация экранов

Функции представляются с помощью экранов, отображающих данные или задающих поля для их ввода. С такими бизнес-функциями, как, например, «создание заказа клиента», может быть связано несколько экранов. И наоборот, некоторые экраны могут активизироваться несколькими функциями. Следовательно, экраны связаны с классом ФУНКЦИЯ ассоциацией *:* (см. рис. 105).

Экраны, используемые для ввода, обновления и удаления свойств объектов данных, называются экранами данных и предоставляют в распоряжение пользователя стандартные функции создания, обновления и удаления.

Экраны приложений или функций отражают требования к входным данным или выходы бизнес-функций.

Экраны описываются соответствующими моделями. Привязывая модели к функциям, можно настраивать конфигурацию функций в соответствии с представленными на экране объектами данных. Например, функции ввода данных, связанные с типом сущности КЛИЕНТ, позволяют вводить данные о клиентах, а функции, связанные с типом сущности ПАЦИЕНТ, — данные о пациентах. Средства редактирования функций можно расширить путем изменения содержания экрана (например, атрибутов добавления или удаления).

На рис. 106а показана модель экрана, содержащая атрибуты типов сущностей КЛИЕНТ и АДРЕС. Соответствующие отношения для обработки мастер-данных формы о клиенте представлены на рис. 106в. В этом примере ФОРМА КЛИЕНТА представляет собой весьма сложный объект данных. В правой части рис. 106а указывается

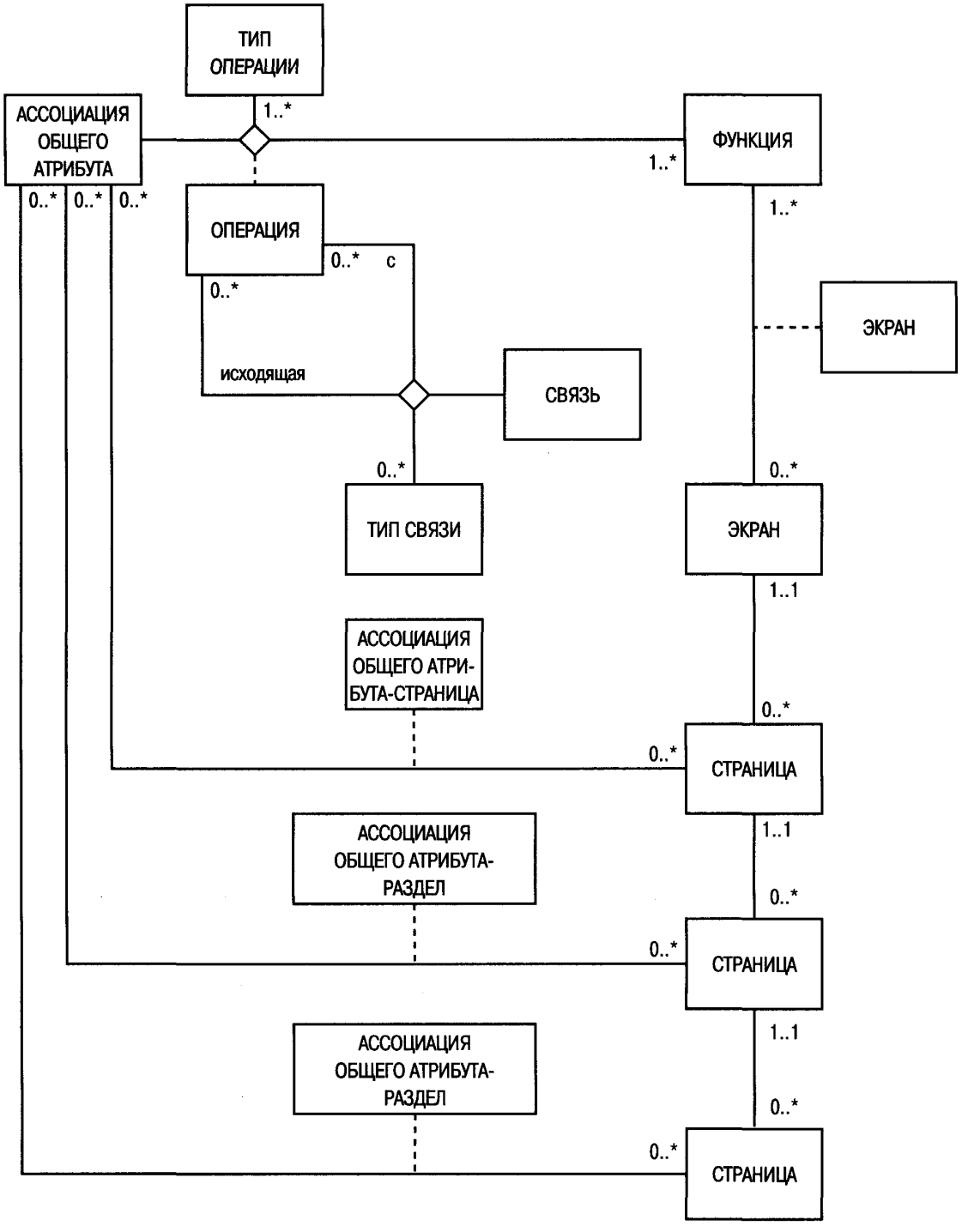


Рис. 105. Отображение потока данных с помощью ассоциации ОПЕРАЦИЯ

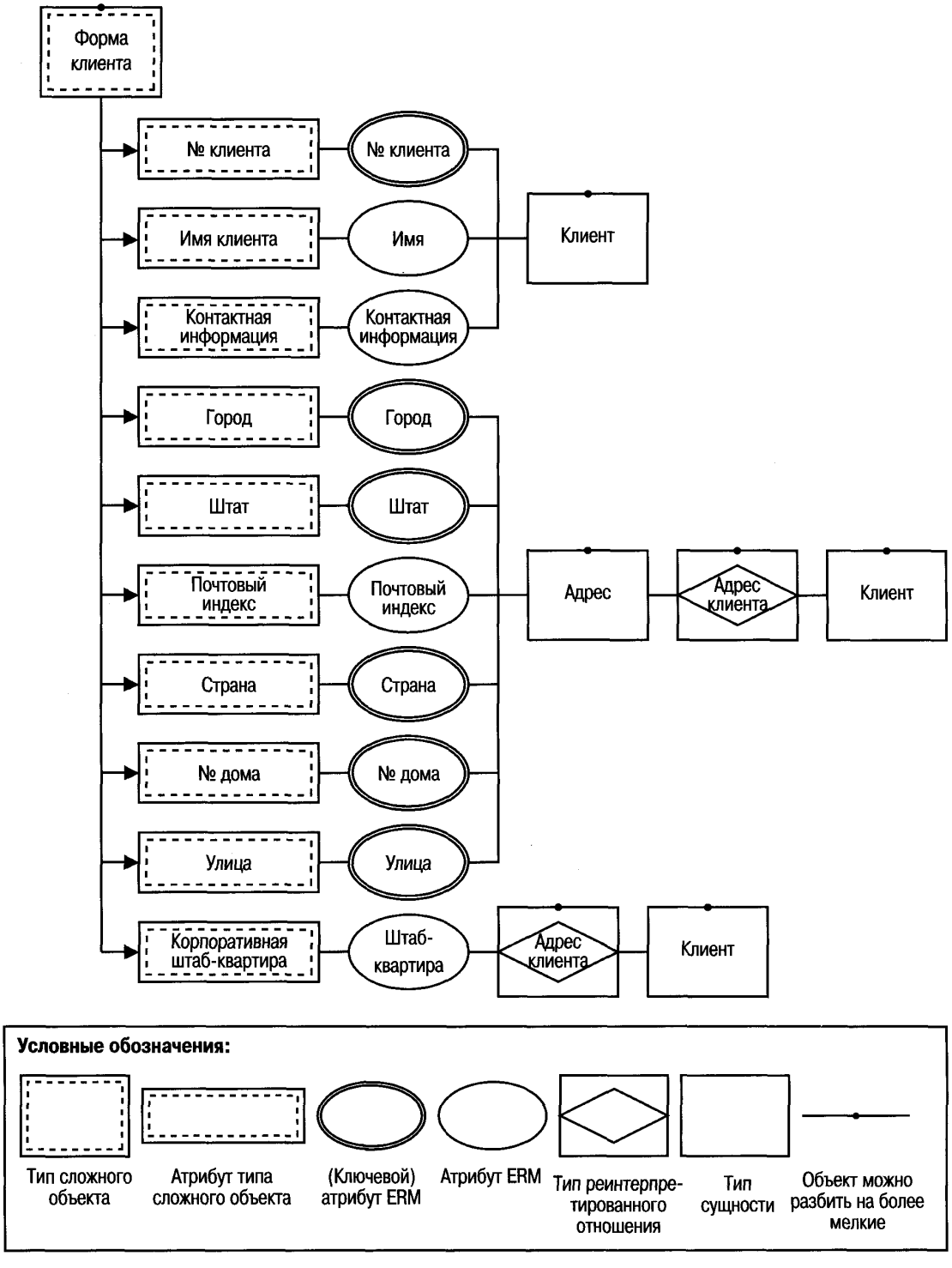


Рис. 106а. Структура и атрибуты сложного типа объекта «форма клиента»

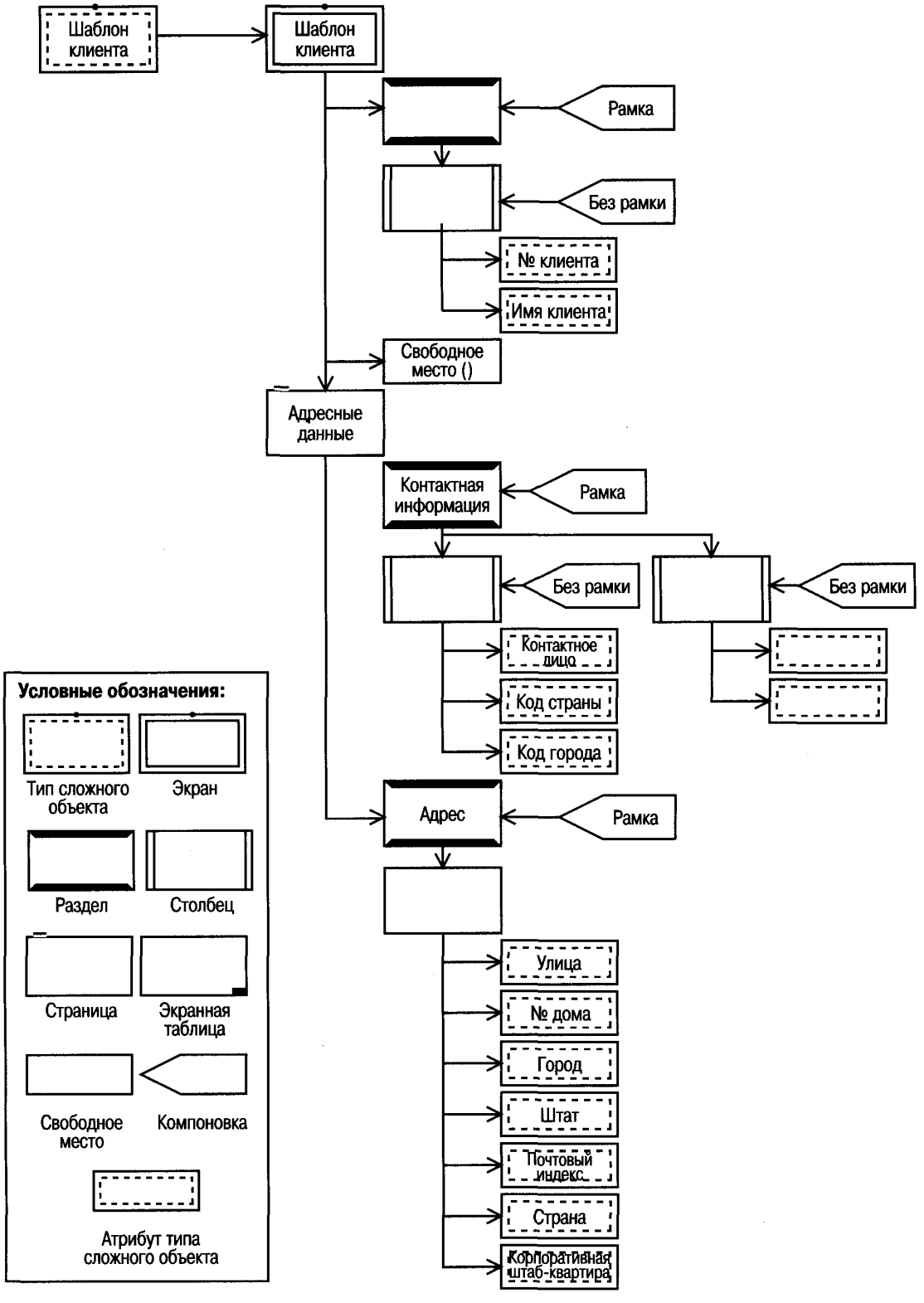


Рис. 10бб. Пример компоновки модели экрана

Рис. 106в. Модель данных для шаблона клиента

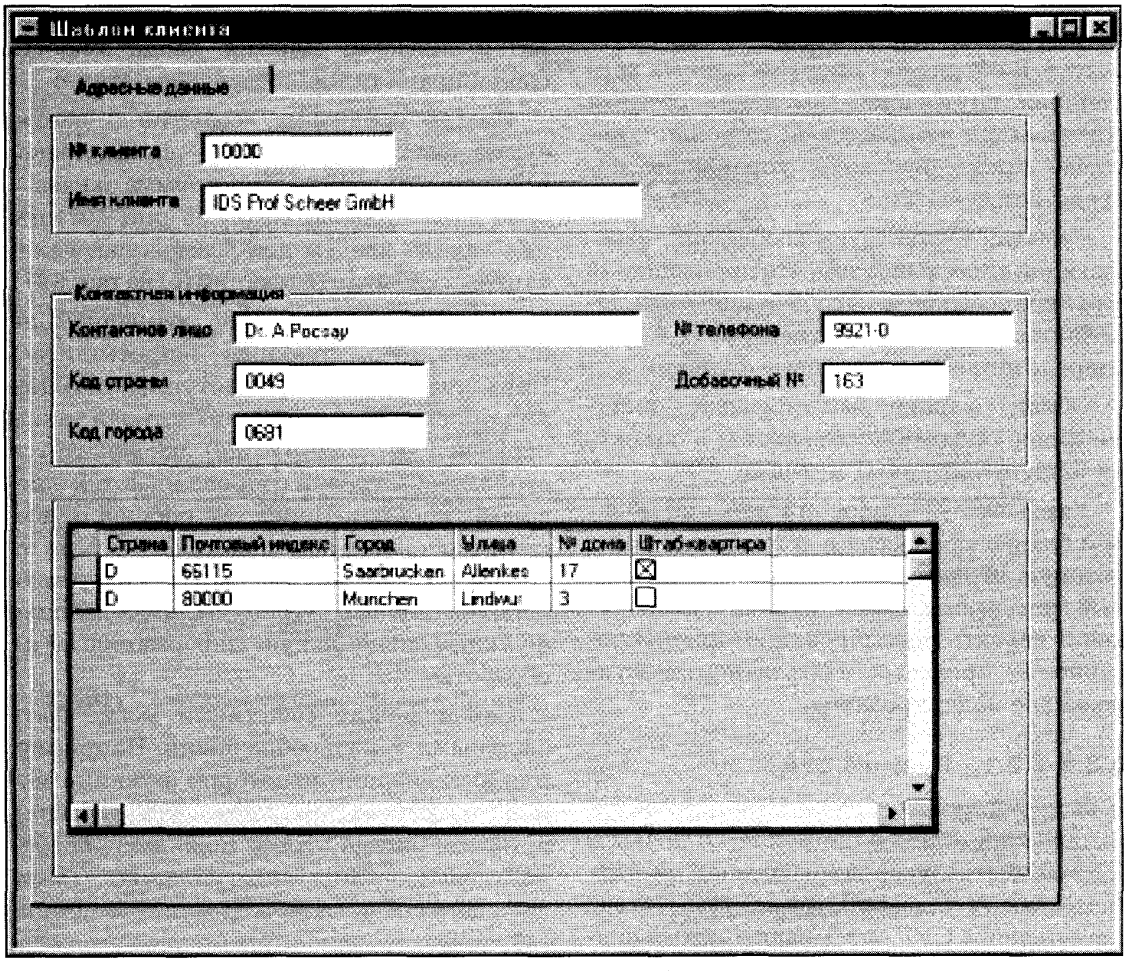


Рис. 106г. Экран с таблицей

источник происхождения данных, каковым служит модель данных. Эти отношения дополняют метамодель на рис. 105.

Экраны проектируются по иерархическому принципу и состоят из страниц, разделов и столбцов (см. рис. 106б). Дополнив модель экрана деталями компоновки, можно автоматически генерировать экран, представленный на рис. 106г (IDS. ARIS–Framework. 1997).

Кроме того, модель на рис. 106б по сравнению с моделью на рис. 106а дополнена контактным лицом. Группа данных, связанных с адресом, повторяется на экране в виде таблицы.

A.3.2.1.2. Управление посредством событий и сообщений

События как элемент управления являются одним из основных компонентов событийных цепочек процессов (EPC), позволяя описывать динамику бизнес- процессов. В объектно-ориентированных концепциях внутреннее поведение объекта описывается диаграммами состояний, которые также отражают управление посредством событий.

События инициируют сообщения, извещающие адресатов о том, что произошло то или иное событие. Эти сообщения могут дополняться другой информацией, и, в свою очередь инициировать новые функции.

Обмен сообщениями описывает динамическое поведение объектов и в объектно–ориентированных методах.

Учитывая успешное применение моделей бизнес-процессов и той важной роли, которую играют объектно-ориентированные методы, мы введем понятия, позволяющие связать моделирование, ориентированное на процессы, и объектно-ориентированное моделирование.

A.3.2.1.2.1. Правило СУД

В информатике для регулирования управляющих потоков применяется правило СУД (событие-условие-действие) *(Dittrich, Gatziu. Aktive Datenbanksysteme. 1996, с. 10)*, хотя правила СУД описывают также и бизнес-правила *(Herbst, Knolmayer. Geschä ftsregeln. 1995)*.

События характеризуют направленные действия, заключая в себе факты (что), происходящие в определенный момент времени (когда). В рамках временных событий «что» и «когда» совпадают (например, 6 часов вечера). Условия задают обстоятельства наступления соответствующего события. Действия определяют реакцию на возникновение той или иной ситуации.

В моделях бизнес–процессов ARIS события создаются обрабатывающими функциями или субъектами действия, находящимися за рамками модели. В процессе моделирования отбираются релевантные события, поэтому в модель включаются только те события, которые влияют на бизнес–процесс. Таким образом, условия входят в описание события, сводя правило СУД к правилу СД (событие-действие).

Вместо уравнения «событие = общая сумма заказа известна» с последующей проверкой условия «общая сумма заказа > 5000 долл.» (сценарий (а) на рис. 107) мы для начала введем два возможных релевантных события (сценарий (б) на рис. 107).

Действия описываются указанием на последующую функцию («преемницу»). Передача сообщений о наступлении события к следующей функции (которая таким образом активизируется) обозначается стрелками. Стрелки сопровождаются символом «письмо». Перед следующей функцией сообщения помещаются в очередь на обработку. Сообщения могут содержать дополнительные атрибуты, передающие функции специальную информацию об их обработке.

Связи между событиями могут быть сложными и разнообразными. Например, для активизации некоторой функции может потребоваться несколько событий, при этом иногда играет роль даже последовательность их наступления. Такие сложные события можно описывать «алгебраически», используя операторы дизъюнкции (логического сложения), последовательности, конъюнкции (логического умножения) и отрицания *(Dittrich, Gatziu. Aktive Datenbanksysteme. 1996, с. 26)*.

A.3.2.1.2.2. Событийные диаграммы процессов (СДП)

Метод СДП (в английской версии - EPC) разработан Институтом информационных систем (IWi) Университета Саарланда (Германия) в сотрудничестве с фирмой

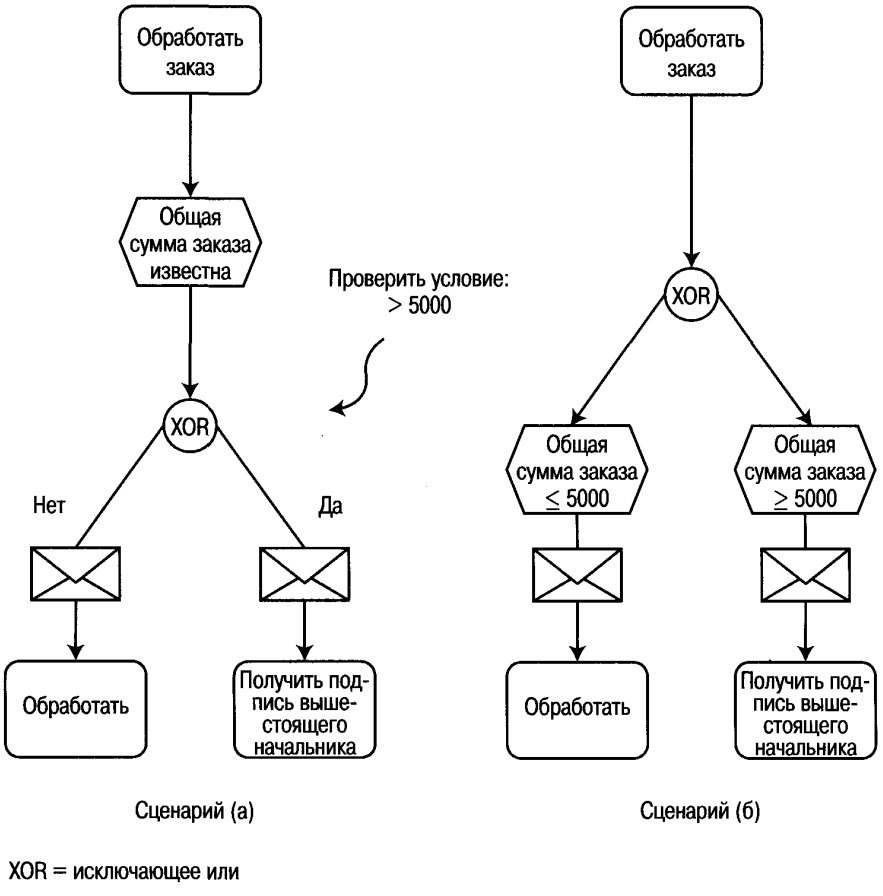


Рис. 107. Способы моделирования событий

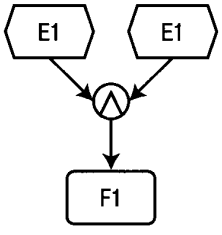
SAP AG *(Keller, Nüttgens, Scheer. Semantische Prozeßmodellierung. 1992)*. Этот метод является ключевым компонентом концепций моделирования SAP R/3 для бизнес-инжиниринга и индивидуальной настройки бизнес-приложений.

Метод основан на концепциях стохастических сетей и сетей Петри. В упрощенных версиях содержатся только описания событий и действий, а условия и сообщения опускаются.

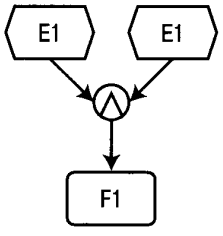
Одно событие может повлечь за собой выполнение нескольких функций. С другой стороны, для инициации события иногда предварительно требуется завершение нескольких функций. Логические отношения описываются символами «и» (/\), «включающее ИЛИ» (\/) и «исключающее ИЛИ» (XOR). На рис. 108 приведены типичные примеры отношений между событиями.

Когда между завершенными функциями и функциями, которые только что активизированы, существуют более сложные отношения (например, различные логические отношения между группами функций), в описании события могут храниться таблицы решений для входящих и исходящих функций.

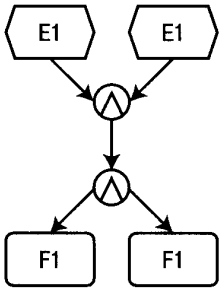
В информационных системах события представлены данными (обновлениями). Примерами типичных событий, которые предполагают изменение состояния (на-



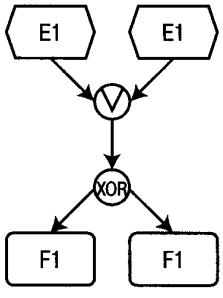
а) Когда происходят события E1 и E2, активизируется функция F1.



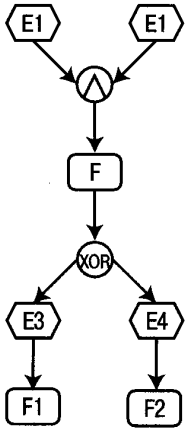
б) Когда происходят событие E1 или E2, активизируется функция F1.



в) Когда происходят события E1 и E2, активизируется функция F1 и F2.



г) Когда происходит событие E1 или E2, активизируется функция F1, или функция F2.



д) Когда происходит событие E1 или E2, активизируется функция решения F, пределяющая, какое из двух событий произойдет: или E3, или E4.

Рис. 108. Отношения между событиями в EPC

пример, для завершения какой-либо функции или как сигнал для дальнейшей обработки), являются создание заказа клиента и выполнение заказа на отправку.

В связи с изложенным на рис. 110 класс СОБЫТИЕ представлен как подкласс ИНФОРМАЦИОННОГО ОБЪЕКТА. Логические отношения между событиями моделируются ассоциацией между ЛОГИЧЕСКИМ ТИПОМ ОТНОШЕНИЯ (например, «и», «или») и СОБЫТИЕМ.

Многие поставщики бизнес-приложений предоставляют возможность конфигурирования программных решений под конкретные нужды предприятия при помощи моделей бизнес-процессов. В качестве примера на рис. 109 показан фрагмент модели бизнес-процесса, построенной средствами Baan Dynamic Process Modeling *(Baan. Dynamic Enterprise Modeling. 1996)*. Далее мы рассмотрим сеть Петри, основанную на модели, изображенной на рис. 110.

Прямые отношения с функцией, т.е. отношения, не включенные явным образом в управление сообщениями, представлены ассоциациями АКТИВИЗАЦИЯ ФУНКЦИИ СОБЫТИЕМ и ПОРОЖДЕНИЕ

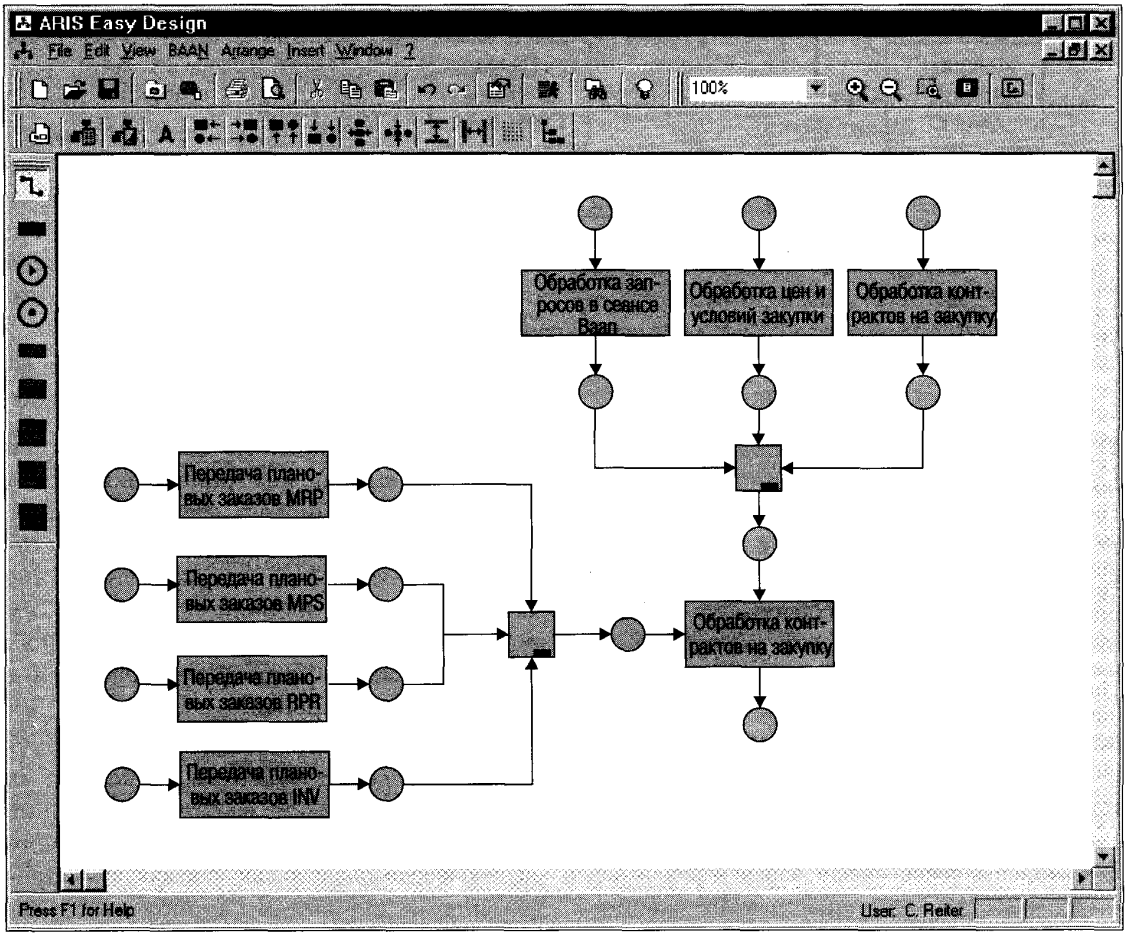


Рис. 109. Модель бизнес-процессов Baan *(источник: IDS Prof. Scheer GmbH)*

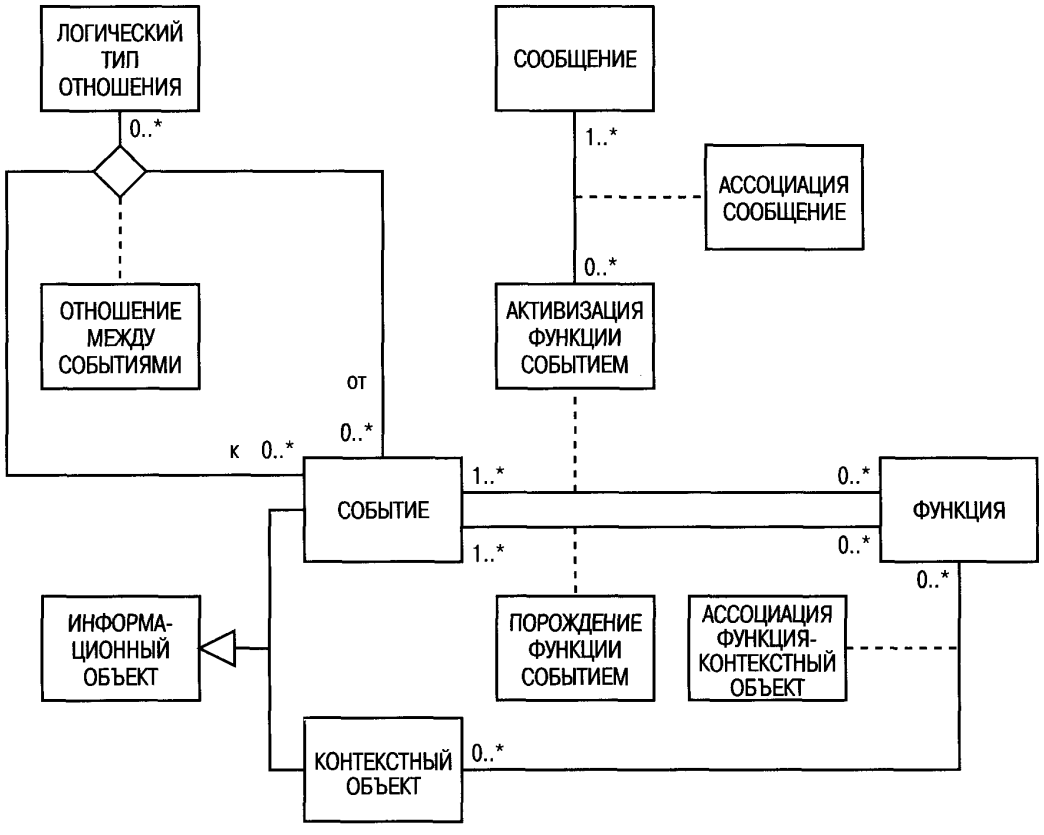


Рис. 110. Метамодель управления событиями и сообщениями (метод СДП)

ФУНКЦИИ СОБЫТИЕМ. События, не определенные как информационные объекты, и применяемые к ним функции, также связываются с этими ассоциациями. ФУНКЦИИ могут активизироваться одним или несколькими событиями. В то же время одна функция может порождать несколько событий. Событие может быть результатом выполнения нескольких функций. Например, окончание проекта иногда сопряжено с завершением одновременного выполнения ряда функций.

A.3.2.1.2.3. Диаграммы состояний

Метод СДП фокусируется на моделировании бизнес-процессов, осуществляемых аналитиком. В моделировании объектов аналогичную роль выполняют диаграммы состояний, хотя они в большей мере нацелены на внутреннее поведение объекта и описание его микроповедения. Диаграммы же СДП в значительной степени ориентированы на моделирование макроповедения цепочки процессов. Тем не менее оба подхода в целом схожи. Иногда диаграммы состояний применяются и при макромоделировании.

Диаграммы состояний описывают внутреннее поведение объектов, фиксируя их состояния и переходы из одного состояния в другое на протяжении всего жизненного цикла. Состояния характеризуются определенными значениями атрибутов объектов. Переходы из одного состояния в другое активизируются событиями. В описаниях широко применяется система обо-

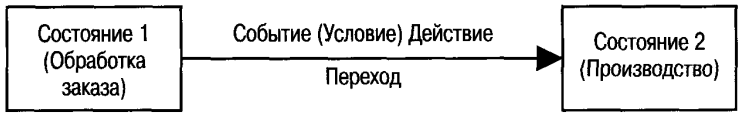


Рис. 111. Диаграмма состояний

значений Харела *(Harel. Statecharts. 1987, с. 231–274; Harel. On Visual Formalism. 1988, с. 514–530)*. Этой же системой пользуется Рамбо *(Rumbaugh et al. Object-Oriented Modeling and Design. 1991)*.

На рис. 111 показана базовая структура типичной диаграммы состояний, привязываемой к объекту.

В рамках определенного состояния — например, «обработка заказа» — могут выполняться те или иные операции. Изменение этого состояния на «завершение обработки заказа» является, следовательно, событием, активизирующим переход. С событием может быть связано некое условие, например, «Успешно ли завершена обработка заказа?» Такое условие указывается в скобках.

Действия приводят к возникновению новых состояний. Примером может служить действие «выдача заказа на производство», представленное на рис. 111. Это действие не обладает собственной функциональностью, в противном случае его нужно было бы моделировать как отдельный процесс. Возникшее новое состояние влечет за собой выполнение функции производства.

Диаграммы состояний строго следуют правилу СДП. Неукоснительное соблюдение этого правила позволяет выполнять теоретическую проверку для выявления каких-либо отклонений в процессе. В этом состоит одно из преимуществ диаграмм состояний. Аналогичные теоретические методы проверки (верификации) применимы и в моделях СДП, но они не вычисляются посредством алгоритмов. Однако подобных результатов можно достичь и при помощи эвристических имитационных исследований.

Диаграммы состояний позволяют создавать описания, напоминающие диаграммы СДП. Состояния описывают выполнение функций, управляемое событиями и переходами.

Следовательно, метамодель диаграмм состояний аналогична метамодели СДП, на рис. 110.

A.3.2.1.2.4. Управление посредством сообщений

Управления посредством сообщений позволяет определить, когда и как следует реагировать на изменения состояния объектов. Сообщения передаются субъектам действия, которые осуществляют эти изменения. Сообщение представляет собой требование отправителя к получателю создать выходной результат.

В диаграммах состояний это указывается действием, предшествующим переходу в новое состояние.

При моделировании простых событийных цепочек (СДП) имена отдельным сообщениям не присваиваются. Предполагается, что сообщения «входят» в стрелки, соединяющие события с последующими функциями.

Если сообщения можно описать при помощи их собственных свойств (например, атрибутов или инструкций), целесообразно построить их точную модель. Иллюстрацией может служить пример на рис. 112, с фрагментом диаграммы СДП, описывающей обработку заказа *(Scheer. ARIS — Business Process Frameworks. 1998)*. Теперь к сообщениям, которые обозначены

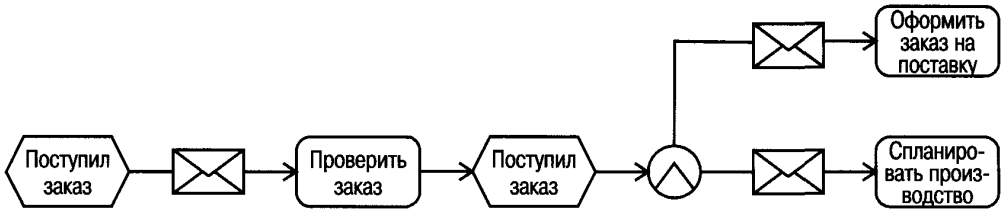


Рис. 111. Пример моделирования сообщений

символом «письмо», можно привязать различные свойства. В соответствии с этим следует расширить метамодель, показанную на рис. 110. Класс СООБЩЕНИЕ связывается с ассоциативной структурой АКТИВИЗАЦИЯ ФУНКЦИИ СОБЫТИЕМ. Одно и то же сообщение можно направлять различным функциям, хотя к «активизирующей» стрелке привязано только одно сообщение.

В объектно-ориентированном моделировании управление посредством сообщений имеет особое значение, поскольку поток сообщений между объектами управляет поведением системы. Потоки сообщений инициируют обработку задач. Помимо объектов «отправитель» или «получатель», сообщения включают функцию и необходимые параметры, которые нужно передать. Отправитель требует от получателя выполнить данную функцию и вернуть результат(ы). Этот процесс показан на рис. 113.

Объект «клиент Джоунс» посылает объекту «изделие 1234» сообщение с требованием вычислить стоимость заказа на 10 штук определенного изделия. Ответ направляется объекту «клиент», хотя, строго говоря, это сообщение активизируется функцией «вычислить стоимость заказа». Соответствующий объект проверяется с целью определить, реализована ли в нем требуемая функция. Если нет, то исследуется иерархия наследования объекта до тех пор, пока искомая функция не будет найдена.

Одним из важных свойств объектно-ориентированного подхода является полиморфизм. Это означает, что некоторые сообщения могут направляться к объектам различных классов, активизируя различные процессы в зависимости от способа реализации необходимой функции.

Маршруты сообщений могут изображаться в виде диаграмм взаимодействий, причем здесь возможны различные подформы. На рис. 114 представлена подробная диаграмма, показывающая последовательность обмена сообщениями и их распределение во времени. Здесь конкретный процесс можно представить как задачу, обрабатываемую двумя объектами.

В упрощенном виде диаграммы взаимодействий могут отражать только обмен базовыми сообщениями между объектами без детализации времени или последовательностей.

В диаграммах объектно-ориентированных классов маршруты сообщений уже определены ассоциациями. Именно поэтому моделирование ассоциаций играет такую важную роль в объектно-ориентированном анализе. Помимо маршрутов сообщений, моделируемых ассоциациями, бывают также специальные (ad hoc) сообщения, где пользователи адресуются к определенным объектам. Однако эти маршруты сообщений не включаются в определение требований, а описываются непосредственно на стадии исполнения.

На рис. 115 представлена метамодель диаграммы классов, дополненная управлением посредством сообщений (см. рис. 101).

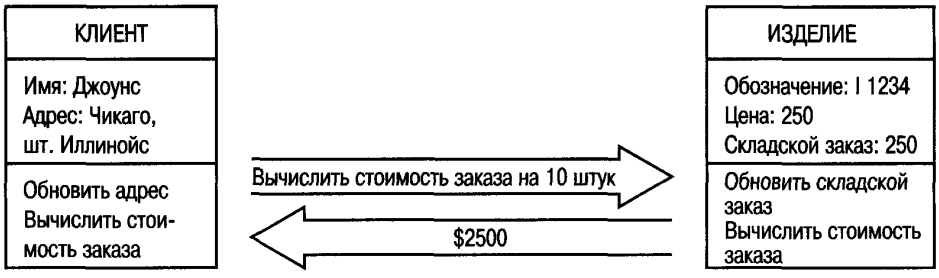


Рис. 113. Обмен сообщениями между объектами

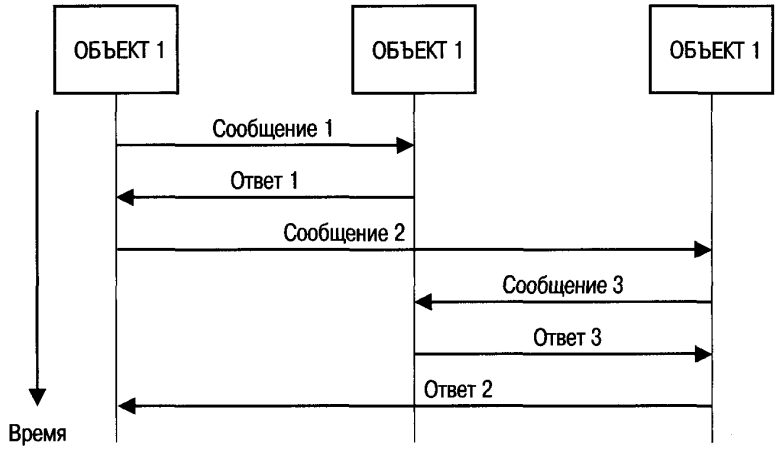


Рис. 114. Диаграмма взаимодействий

СООБЩЕНИЯ направляются от ФУНКЦИЙ ОБЪЕКТОВ к ФУНКЦИЯМ других объектов, при этом отношения определяются ассоциациями. Изделия, участвующие в процессе, обозначаются последовательными номерами или отметками времени, а параметры передачи привязываются к сообщениям.

Определенное состояние может повлечь за собой применение (выполнение) того или иного метода, но может включать и конкретное описание, например, состояние ожидания. В метамоделях такая возможность обеспечивается мощностью связи (0..1) между СОСТОЯНИЕМ и МЕТОДОМ. Состояние инициируется СОБЫТИЕМ, которое, в свою очередь, активизируется другим состоянием.

Если понятие «состояние» приравнять к понятию «функция», то метаструктура диаграммы состояний будет сопоставима с метаструктурой упрощенной диаграммы СДП, хотя и без ее логических отношений, связывающих события.

A.3.2.1.2.5. Связывание объектно-ориентированного моделирования и СДП

Немотря на то, что моделирование бизнес-процессов и объектно-ориентированное моделирование имеют разные парадигмы, неоднократно предпринимались попытки объединить то и другое.

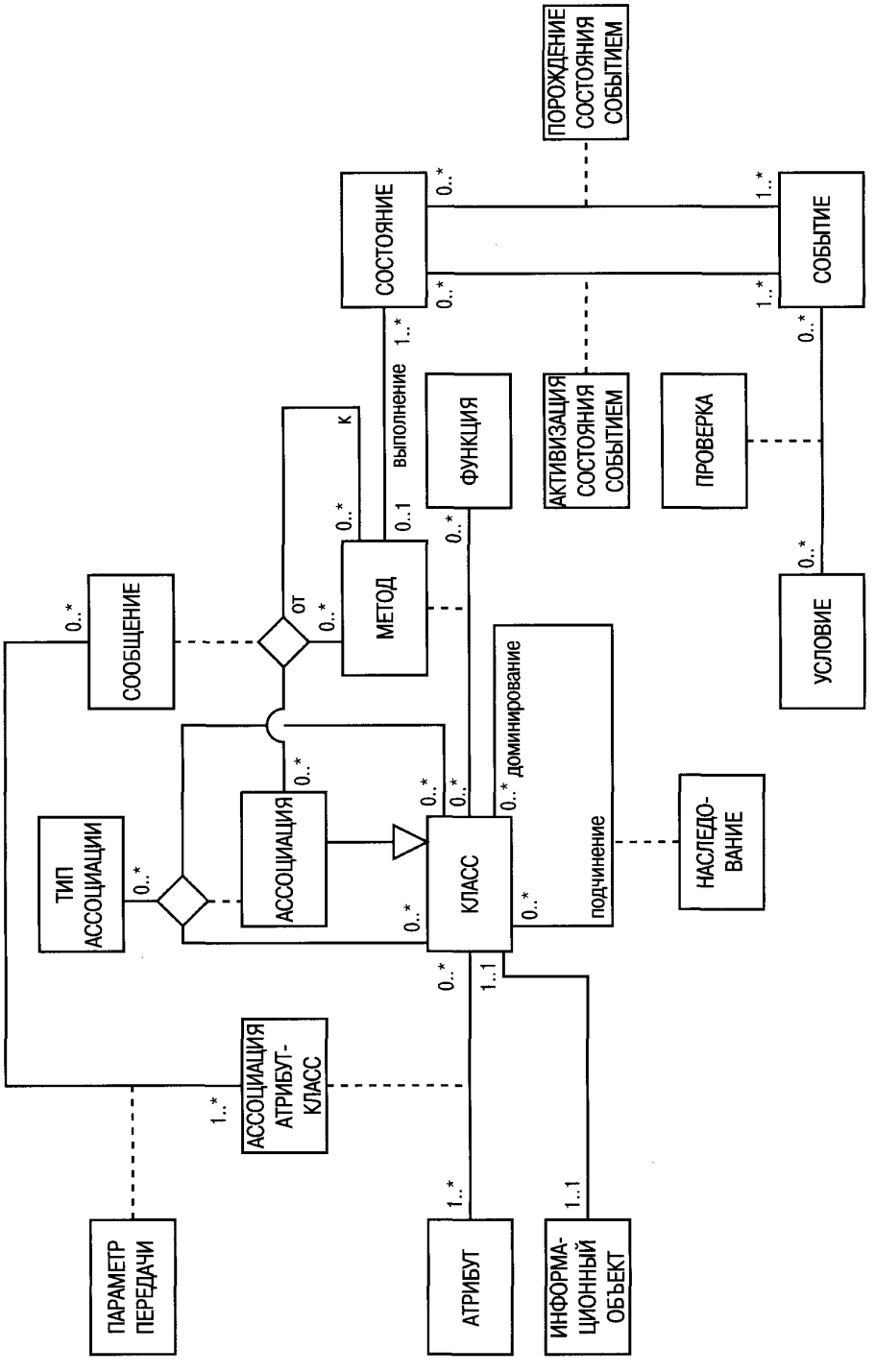


Рис. 115. Метамодель управления сообщениями при объектно-ориентированном подходе

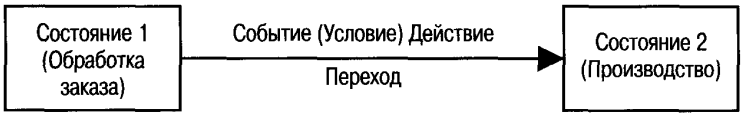


Рис. 111. Диаграмма состояний

значений Харела *(Harel. Statecharts. 1987, с. 231–274; Harel. On Visual Formalism. 1988, с. 514–530)*. Этой же системой пользуется Рамбо *(Rumbaugh et al. Object-Oriented Modeling and Design. 1991)*.

На рис. 111 показана базовая структура типичной диаграммы состояний, привязываемой к объекту.

В рамках определенного состояния — например, «обработка заказа» — могут выполняться те или иные операции. Изменение этого состояния на «завершение обработки заказа» является, следовательно, событием, активизирующим переход. С событием может быть связано некое условие, например, «Успешно ли завершена обработка заказа?» Такое условие указывается в скобках.

Действия приводят к возникновению новых состояний. Примером может служить действие «выдача заказа на производство», представленное на рис. 111. Это действие не обладает собственной функциональностью, в противном случае его нужно было бы моделировать как отдельный процесс. Возникшее новое состояние влечет за собой выполнение функции производства.

Диаграммы состояний строго следуют правилу СДП. Неукоснительное соблюдение этого правила позволяет выполнять теоретическую проверку для выявления каких-либо отклонений в процессе. В этом состоит одно из преимуществ диаграмм состояний. Аналогичные теоретические методы проверки (верификации) применимы и в моделях СДП, но они не вычисляются посредством алгоритмов. Однако подобных результатов можно достичь и при помощи эвристических имитационных исследований.

Диаграммы состояний позволяют создавать описания, напоминающие диаграммы СДП. Состояния описывают выполнение функций, управляемое событиями и переходами.

Следовательно, метамодель диаграмм состояний аналогична метамодели СДП, на рис. 110.

A.3.2.1.2.4. Управление посредством сообщений

Управления посредством сообщений позволяет определить, когда и как следует реагировать на изменения состояния объектов. Сообщения передаются субъектам действия, которые осуществляют эти изменения. Сообщение представляет собой требование отправителя к получателю создать выходной результат.

В диаграммах состояний это указывается действием, предшествующим переходу в новое состояние.

При моделировании простых событийных цепочек (СДП) имена отдельным сообщениям не присваиваются. Предполагается, что сообщения «входят» в стрелки, соединяющие события с последующими функциями.

Если сообщения можно описать при помощи их собственных свойств (например, атрибутов или инструкций), целесообразно построить их точную модель. Иллюстрацией может служить пример на рис. 112, с фрагментом диаграммы СДП, описывающей обработку заказа *(Scheer. ARIS — Business Process Frameworks. 1998)*. Теперь к сообщениям, которые обозначены

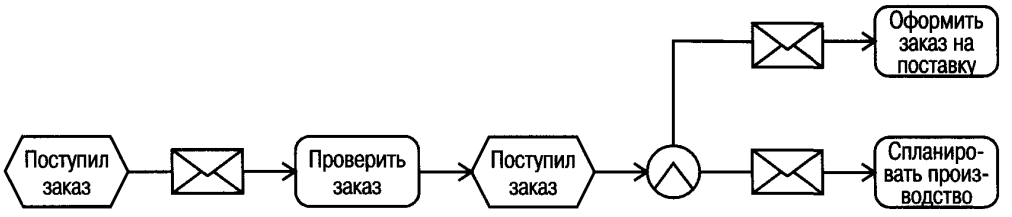


Рис. 111. Пример моделирования сообщений

символом «письмо», можно привязать различные свойства. В соответствии с этим следует расширить метамодель, показанную на рис. 110. Класс СООБЩЕНИЕ связывается с ассоциативной структурой АКТИВИЗАЦИЯ ФУНКЦИИ СОБЫТИЕМ. Одно и то же сообщение можно направлять различным функциям, хотя к «активизирующей» стрелке привязано только одно сообщение.

В объектно-ориентированном моделировании управление посредством сообщений имеет особое значение, поскольку поток сообщений между объектами управляет поведением системы. Потоки сообщений инициируют обработку задач. Помимо объектов «отправитель» или «получатель», сообщения включают функцию и необходимые параметры, которые нужно передать. Отправитель требует от получателя выполнить данную функцию и вернуть результат(ы). Этот процесс показан на рис. 113.

Объект «клиент Джоунс» посылает объекту «изделие 1234» сообщение с требованием вычислить стоимость заказа на 10 штук определенного изделия. Ответ направляется объекту «клиент», хотя, строго говоря, это сообщение активизируется функцией «вычислить стоимость заказа». Соответствующий объект проверяется с целью определить, реализована ли в нем требуемая функция. Если нет, то исследуется иерархия наследования объекта до тех пор, пока искомая функция не будет найдена.

Одним из важных свойств объектно-ориентированного подхода является полиморфизм. Это означает, что некоторые сообщения могут направляться к объектам различных классов, активизируя различные процессы в зависимости от способа реализации необходимой функции.

Маршруты сообщений могут изображаться в виде диаграмм взаимодействий, причем здесь возможны различные подформы. На рис. 114 представлена подробная диаграмма, показывающая последовательность обмена сообщениями и их распределение во времени. Здесь конкретный процесс можно представить как задачу, обрабатываемую двумя объектами.

В упрощенном виде диаграммы взаимодействий могут отражать только обмен базовыми сообщениями между объектами без детализации времени или последовательностей.

В диаграммах объектно-ориентированных классов маршруты сообщений уже определены ассоциациями. Именно поэтому моделирование ассоциаций играет такую важную роль в объектно-ориентированном анализе. Помимо маршрутов сообщений, моделируемых ассоциациями, бывают также специальные (ad hoc) сообщения, где пользователи адресуются к определенным объектам. Однако эти маршруты сообщений не включаются в определение требований, а описываются непосредственно на стадии исполнения.

На рис. 115 представлена метамодель диаграммы классов, дополненная управлением посредством сообщений (см. рис. 101).

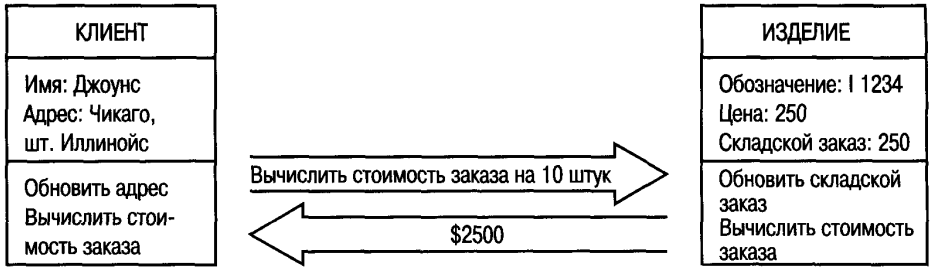


Рис. 113. Обмен сообщениями между объектами

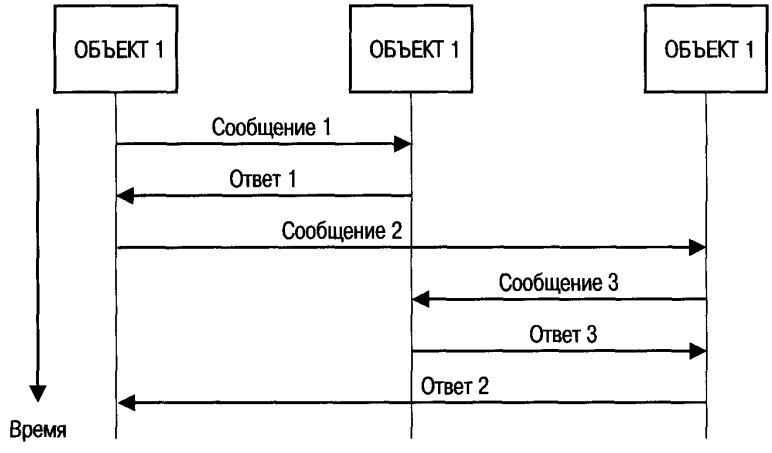


Рис. 114. Диаграмма взаимодействий

СООБЩЕНИЯ направляются от ФУНКЦИЙ ОБЪЕКТОВ к ФУНКЦИЯМ других объектов, при этом отношения определяются ассоциациями. Изделия, участвующие в процессе, обозначаются последовательными номерами или отметками времени, а параметры передачи привязываются к сообщениям.

Определенное состояние может повлечь за собой применение (выполнение) того или иного метода, но может включать и конкретное описание, например, состояние ожидания. В метамоделях такая возможность обеспечивается мощностью связи (0..1) между СОСТОЯНИЕМ и МЕТОДОМ. Состояние инициируется СОБЫТИЕМ, которое, в свою очередь, активизируется *другим состоянием*.

Если понятие «состояние» приравнять к понятию «функция», то метаструктура диаграммы состояний будет сопоставима с метаструктурой упрощенной диаграммы СДП, хотя и без ее логических отношений, связывающих события.

A.3.2.1.2.5. Связывание объектно-ориентированного моделирования и СДП

Немотря на то, что моделирование бизнес-процессов и объектно-ориентированное моделирование имеют разные парадигмы, неоднократно предпринимались попытки объединить то и другое.

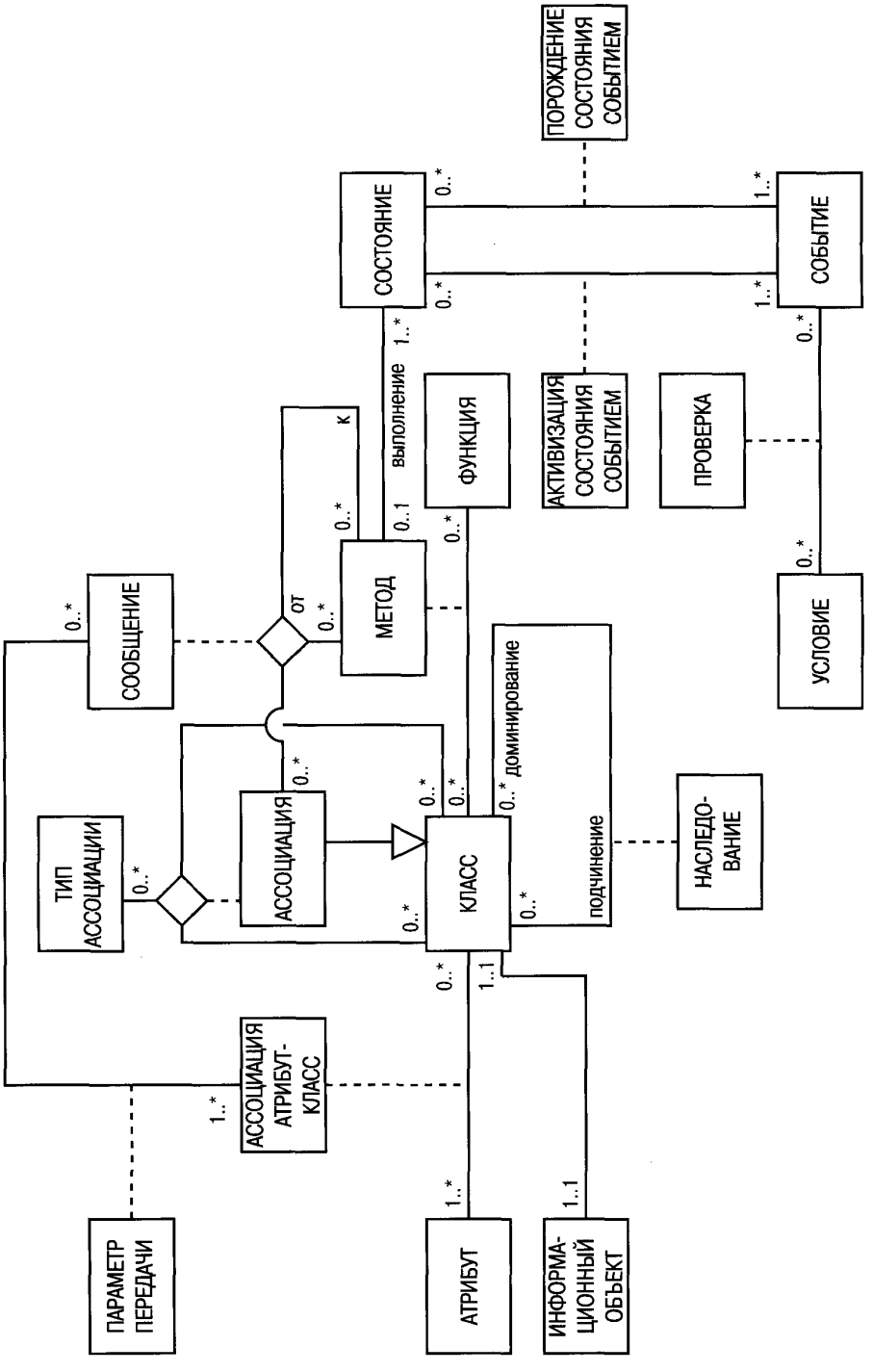


Рис. 115. Метамодель управления сообщениями при объектно-ориентированном подходе

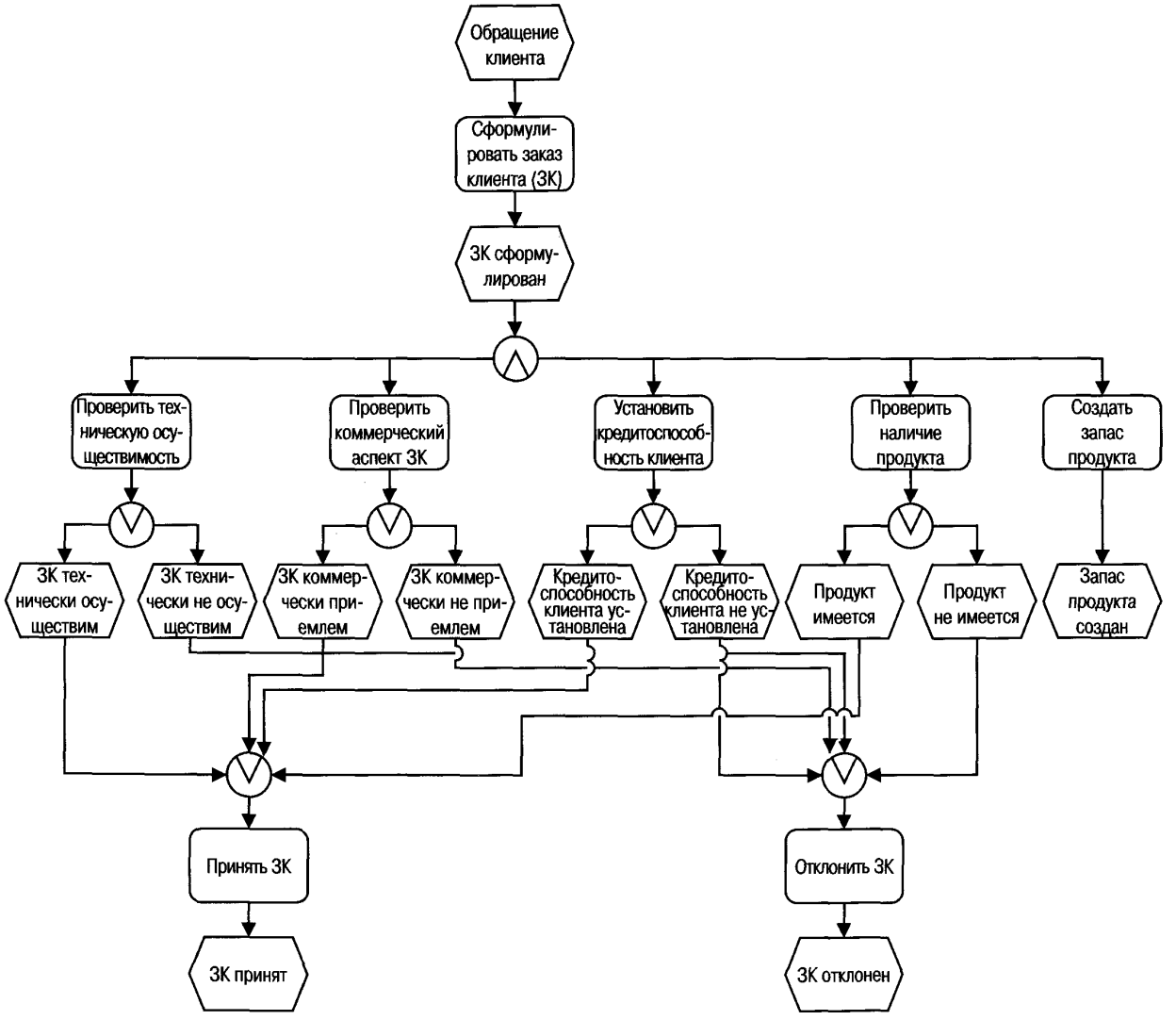


Рис. 116а. Описание полного процесса в виде диаграммы СДП *(Bungert, Heß. Objektorientierte Geschäftsprozeßmodellierung. 1995, с. 62)*

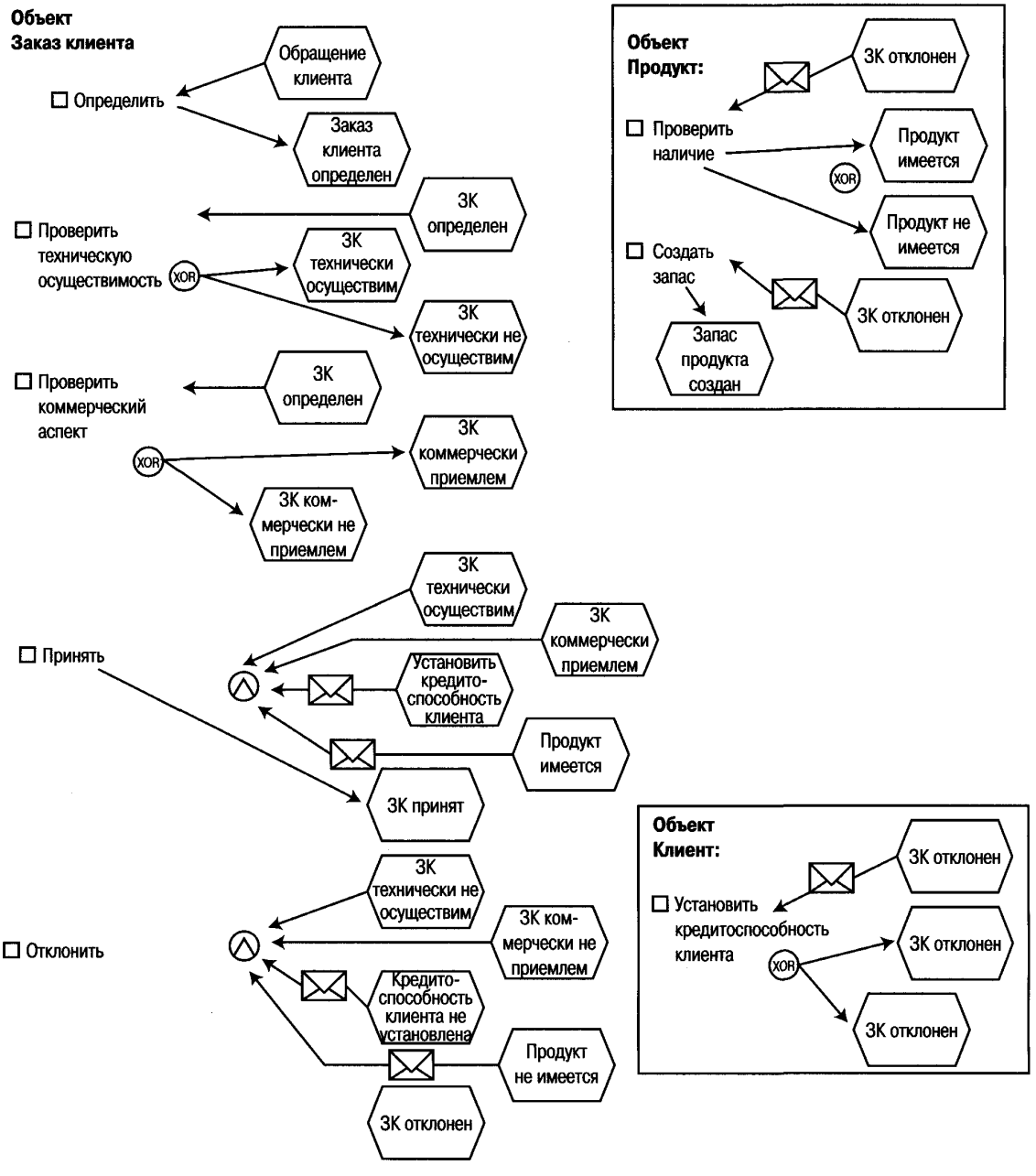


Рис. 116б. Описание событий, являющихся результатом выполнения функций *(Bungert, Heß. Objektorientierte Geschäftsprozeßmodellierung. 1995, с. 61)*

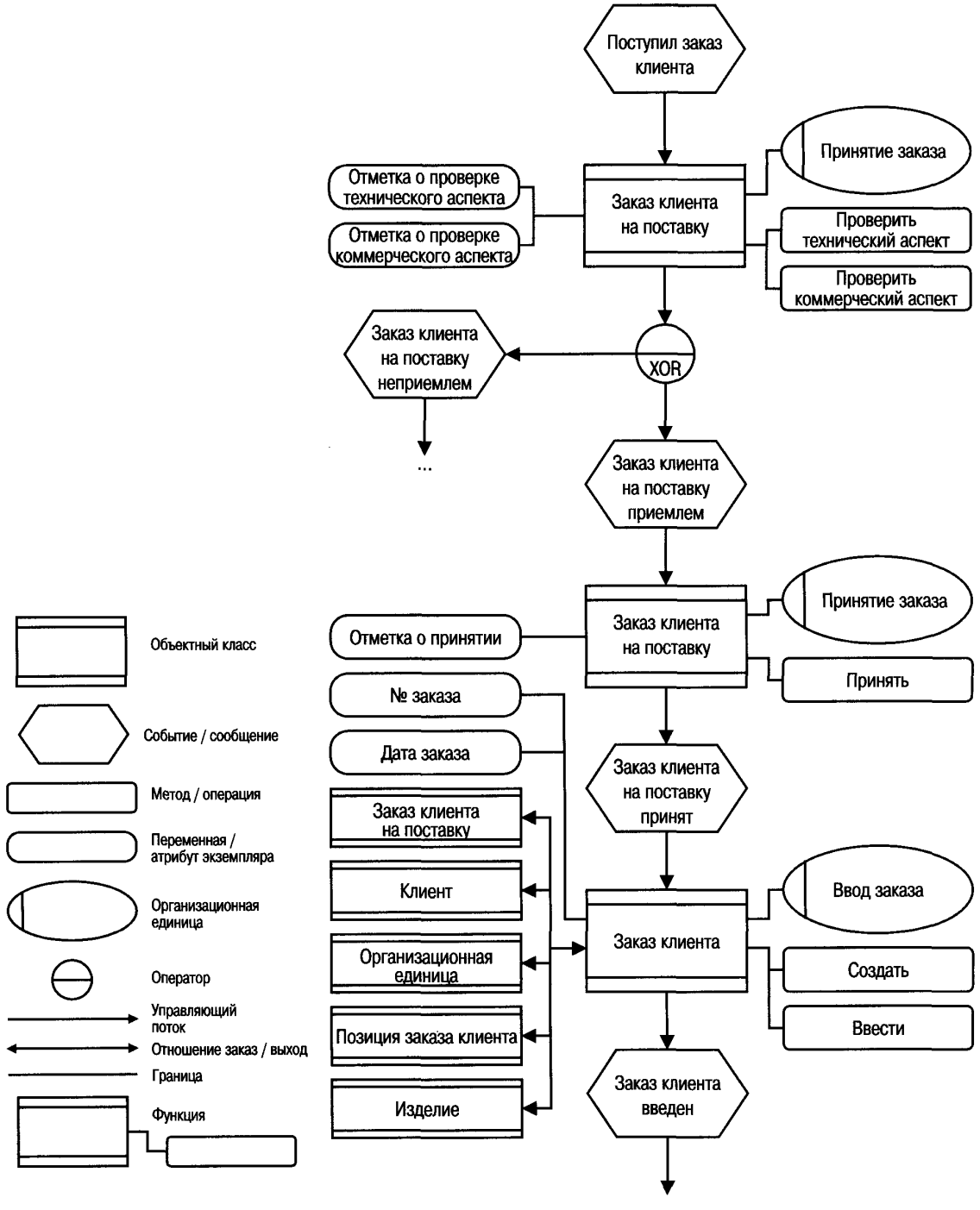


Рис. 117. Объектно-ориентированная диаграмма СДП для ввода заказа

Существует два способа связи диаграмм СДП с объектно-ориентированным моделированием.

В концепции, предложенной Бунгертом и Хессом *(Bungert, Heß. Objektorientierte Geschäftsprozeßmodellierung. 1995)*, диаграммы СДП можно трансформировать в объектные модели и наоборот. Примеры (слегка видоизмененные и расширенные по сравнению с оригиналом) приведены на рис. 116а и 116б.

Мы исходим из посылки, что информационные объекты, используемые в СДП, можно описать как объектно-ориентированные классы. К ним привязываются функции цепочки процессов, а активизирующие события и события, активизируемые классами, в свою очередь, привязываются к информационным объектам. Активизирующие объекты принимаются в виде сообщений и могут пересылаться событиями, активизированными другими функциями. Внутренние и внешние события можно описывать по отдельности. Описанная метамодель соответствует метамодели СДП (см. рис. 110).

В концепции Нюттгенса и Циммерманна управление событиями СДП переносится на поток объектов *(Scheer, Nüttgens, Zimmermann. oEPK. 1997; Nüttgens, Feld, Zimmermann. Business Process Modeling. 1998).*

Объектно-ориентированные событийные диаграммы процессов (оСДП) связывают управление событиями, ориентированное на процессы, с элементами объектно-ориентированного моделирования. Объекты группируются в соответствии с потоком управления процессами, а к ним при помощи соответствующей процедуры привязываются обрабатывающие функции. По сути, это одна из возможных реверсий описания процессного контекста, как показано в общей модели бизнес-процессов на базе ARIS *(Scheer. ARIS — Business Process Frameworks. 1998, с. 31; русское издание - с. 28).*

Описание методом оСДП весьма актуально, когда функция бизнес-процесса охватывает несколько объектов. Такая ситуация возникает, если нужно определить ключевые объекты, однако другие необходимые объекты включаются только в поток сообщений. При обработке объекта в несколько этапов в его описание вводятся указания на различные используемые функции. Описания объектов в этой ситуации тоже имеют важнейшее значение.

Данный подход вновь акцентирует внимание на различии между парадигмами проектирования с ориентацией на процессы и проектирования с ориентацией на объекты. На рис. 117 приведен фрагмент оСДП, наглядно иллюстрирующий проблемы, о которых мы говорили в связи с процессом создания объекта.

A.3.2.2. Конфигурирование

В зависимости от контекста, относящегося к функциям и данным модель может быть наполнена самым разным содержанием.

Используя здесь моделируемые события как «вехи», можно определить ответственные моменты для обновления данных, касающихся стоимости или времени выполнения конкретных бизнес-процессов. Наступление этих событий инициирует проведение оценки и сообщение ее результатов руководству.

Управление рабочими потоками (workflow) основывается на потоке управления, описываемом диаграммой СДП. Поток управления копируется из соответствующей модели для обработки в рамках экземпляров процессов.

Применительно к управлению потоком работ управляющие структуры можно моделировать более детально, включая, например, задержки перед началом работы, ситуации отказа от порученных задач или учет предельных мощностей при их распределении. *(Jablonski. Workflow-Management-Systeme. 1995, с. 35).*

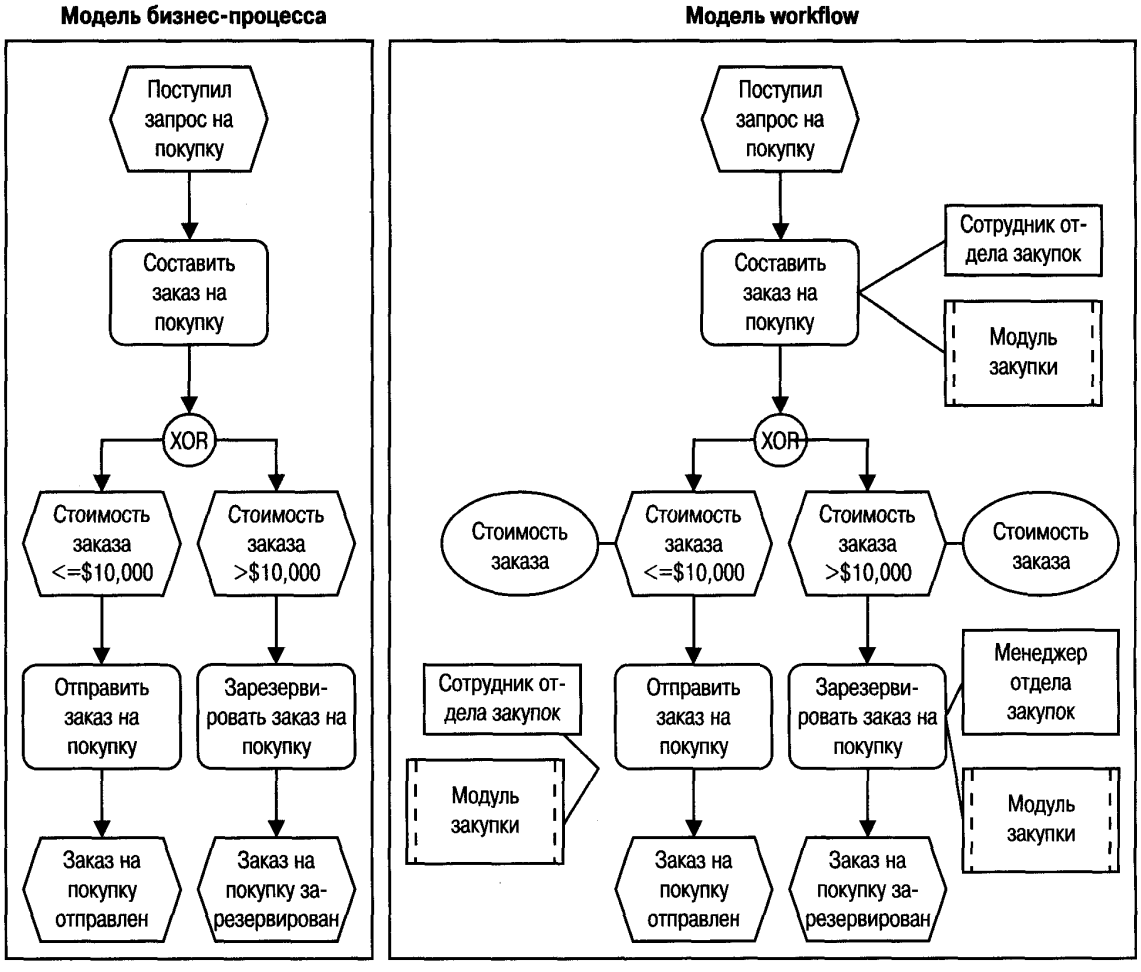


Рис. 118. Использование СДП в рамках модели workflow

Кроме того, используется привязка данных к функциям и описаниям экранов, чтобы установить для экранов системы workflow значения по умолчанию.

Бизнес-функции определяют, какие приложения должна запускать система workflow, включая присвоенные им программные имена. На рис. 118 показано, как использовать СДП в модели workflow.

Как и при управлении потоками работ, потоки управления стандартными приложениями можно представлять в виде СДП. Однако в отличие от workflow, эта процедура не допускает произвольного конфигурирования, а требует соблюдения параметров, предписываемых стандартным приложением. Это означает, что можно выбирать функции (путем редактирования) или задавать определенные последовательности вызовов функций.

Возможности конфигурирования становятся сейчас все разнообразнее. В со-

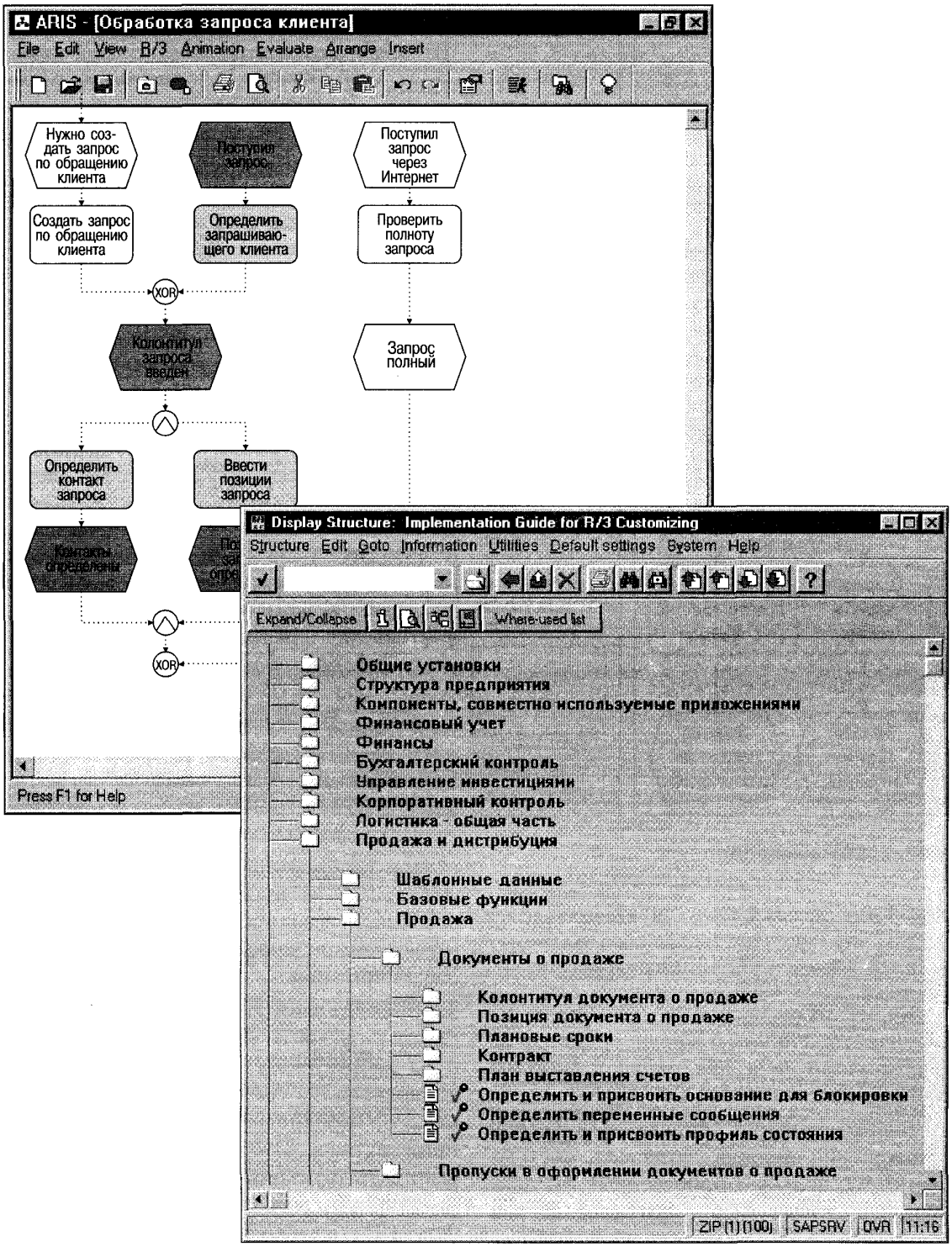


Рис. 119. Конфигурирование R/3 с помощью СДП (источник: SAP AG)

временных стандартных приложениях используются новые объектные технологии, так как теперь процессы в рамках бизнес-объектов и управляющий поток между ними можно проектировать, опираясь на их модели.

На рис. 119 приведен пример конфигурирования SAP R/3 при помощи модификации СДП (верхний экран). Незаштрихованная область диаграммы процесса в этом реальном сценарии деактивизирована, поэтому ее программные функции не задействованы. На нижнем экране показано руководство по настройке, открывающее прямой доступ к соответствующим транзакциям настройки, проектной информации и документам. Это отображается на экране символами «галочка», «карандаш» и «документ».

Функции можно также модифицировать путем изменения моделей экранов. Например, если в моделях экранов, показанных на рис. 106а и 106б, удалить данные об адресах, то функции создания и обновления применительно к этим данным становятся недоступны. Поскольку вводить адреса при этом тоже нельзя, модифицируется и связь между объектом данных и функцией.

A.3.2.3. Спецификация проекта

A.3.2.3.1. Связывание модулей с базами данных

В спецификации проекта на уровне функционального представления модули первоначально создаются исключительно на основе информации в виде данных без знания конкретной схемы базы данных. Затем эти модули и миниспецификации связываются с базами данных.

A.3.2.3.1.1. Привязка схемы

Прикладным модулям не нужно общаться со всей концептуальной схемой базы данных: достаточно лишь некоторых фрагментов.

В концептуальной схеме для отдельных приложений имеет смысл изменить имена, поскольку иногда — при автономной разработке приложения — к спецификации проекта добавляются описания, которые не совпадают с независимо разработанной моделью данных.

Внешние схемы баз данных, связывающие пользователей с концептуальной схемой, описывают логические представления баз данных с точки зрения конкретного приложения или отдельных пользователей. На основе реляционной схемы можно вывести новые отношения, опустив атрибуты или некоторые кортежи базисных отношений, скомбинировав базисные отношения в соответствии с определенными критериями или, наоборот, разложив их на ряд более детальных отношений. Один из важнейших методов состоит в описании так называемых представлений. В общем виде это выглядит следующим образом *(Mayr, Dittrich, Lockemann. Datenbankentwurf. 1987, с. 537)*:

- ОПИСАТЬ ПРЕДСТАВЛЕНИЕ [имя представления],
- ВЫБРАТЬ [выражение].

Метаструктура таких представлений показана на рис. 120. Вся концептуальная схема, состоящая из отношений, атрибутов и условий целостности, представлена сложным объектом КОНЦЕПТУАЛЬНАЯ СХЕМА. Ассоциации связывают ВНЕШНИЕ СХЕМЫ с КОНЦЕПТУАЛЬНЫМИ СХЕМАМИ. К одному МОДУЛЮ можно привязать несколько внешних схем и, наоборот, внешние схемы можно привязать к нескольким модулям.

A.3.2.3.1.2. Выведение структур управления

Как правило, модули состоят из трех частей: описание данных, управляющая логика и инструкции. Для целей управления в структурном программировании до-

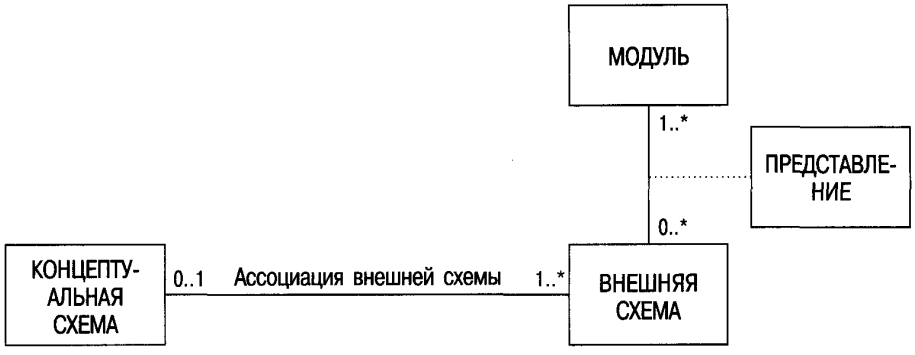


Рис. 120. Связывание модулей со схемой базы данных

пускаются следующие структуры: последовательность, итерация и выбор.

Структуры управления можно связывать со структурами данных. Ассоциации 1:1 между классами соответствуют последовательности; ассоциации 1:* соответствуют итерации, а операции конкретизации (разбивающие информационные объекты на подчлены) соответствуют выбору.

К информационному объекту КЛИЕНТ однозначным образом привязывается счет (ассоциация 1:1). Один клиент может порождать множество бизнес-событий, однако любое бизнес-событие всегда связано только с одним клиентом. БИЗНЕС-СОБЫТИЯ можно конкретизировать, подразделив их на ЗАКАЗ и ОТМЕНУ. (см. рис. 121).

Разные бизнес-события инициируют разные события бухгалтерской проводки. Результатом является управляющая процедура, представленная на рис. 122 в виде структурограммы. Сначала считывается запись, содержащая данные о конкретном клиенте. Затем считывается соответствующий счет. Эти два действия образуют последовательность, поскольку мощность (со стороны клиента) равна 1.

Различные бизнес-события с мощностью * (со стороны клиента), обрабатываемые для данного клиента, представлены как итерация.

В зависимости от типа бизнес-события выполняются различные бухгалтерские проводки в соответствии с их конкретным значением.

Метаописания опускаются. Такое проектирование программы, ориентированное на структуру данных, аналогично построению обмена сообщениями на основе ассоциаций диаграммы классов в контексте объектно-ориентированных методов.

A.3.2.3.1.3. Транзакции баз данных

Обновление баз данных основано на концепции транзакций, свойства которых описываются термином ACID (A - атомарность, C - согласованность, I - изолированность, D - долговечность). Транзакции включают серию операций базы данных, которая — с точки зрения приложения — не должна прерываться. Это означает, что с точки зрения приложений согласование базы данных производится лишь в том случае, если транзакция доведена до конца. В случае же ошибки выполняется «откат», т.е. данные возвращаются в состояние, предшествовавшее транзакции. Это свойство транзакций называется атомарностью. База данных обновляется лишь при условии успешного завершения транзакции.

Помимо атомарности, означающей, что транзакции не могут прерываться, адми-

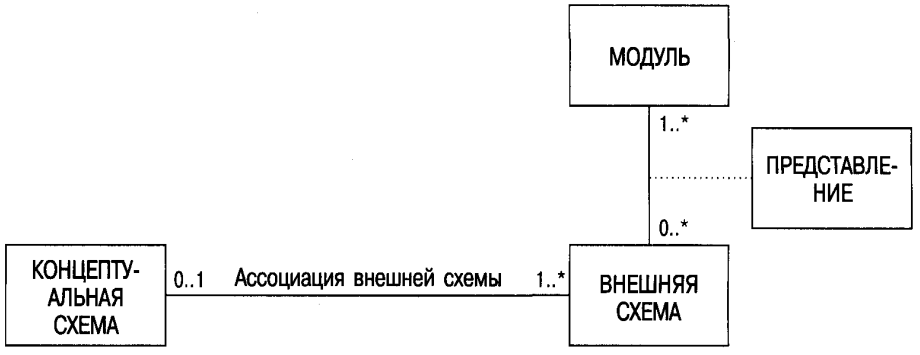


Рис. 121. Отношения между структурами управления и структурами данных

нистрирование их должно обеспечивать согласованность, т.е. транзакции должны переводить базу данных из одного согласованного состояния в другое согласованное состояние.

Еще одним фактором является изолированность, предполагающая, что частичные результаты могут не передаваться другим приложениям в процессе транзакции. Долговечность (или персистентность) означает, что результат успешно выполненной транзакции сохраняется в целости и может быть модифицирован или обновлен только новыми транзакциями.

Транзакции характеризуются понятиями «начало транзакции» и «конец транзакции». В рамках этих двух этапов может заключаться любое число команд записи в файл или чтения. Транзакции также используются в качестве единиц для измерения числа этапов восстановления данных.

С точки зрения проектирования программ, транзакции могут интерпретироваться как модули, поэтому на рис. 123 мы вводим ТРАНЗАКЦИИ как конкретизацию понятия МОДУЛЬ. На стадии спецификации проекта мы говорили, что модули можно связывать друг с другом в сети. Та же ситуация и здесь.

Несколько операций базы данных группируются в одну транзакцию, в результате чего ОПЕРАЦИЯ БАЗЫ ДАННЫХ (БД) превращается в ассоциацию между ТИПОМ ОПЕРАЦИИ БД (как в процессе чтения или записи в файл), соответствующей ТРАНЗАКЦИЕЙ и упоминавшимся ранее ИНФОРМАЦИОННЫМ ОБЪЕКТОМ.

A.3.2.3.2. Управление посредством триггеров

Базы данных являются не только средством для пассивного хранения корпоративных данных. К ним также подсоединяются компоненты, реагирующие на определенные события и действия, связанные с приложениями. Эти компоненты инициируют обновление базы данных (активные системы баз данных) и называются триггерами. Понятие «триггер» мы ввели в разделе A.2.3.3.3, когда рассматривали условия целостности при спецификации проекта на уровне модели данных.

Триггеры могут активизировать функции приложений. Например, если в системе планирования складских материалов непрерывно контролируется наличие той или иной детали, то при наступлении события «минимальный уровень запасов не поддерживается» инициируется заказ на закупку.

Говоря кратко, триггеры состоят из описания инициирующих событий, контролируемых условий и действий, инициир-

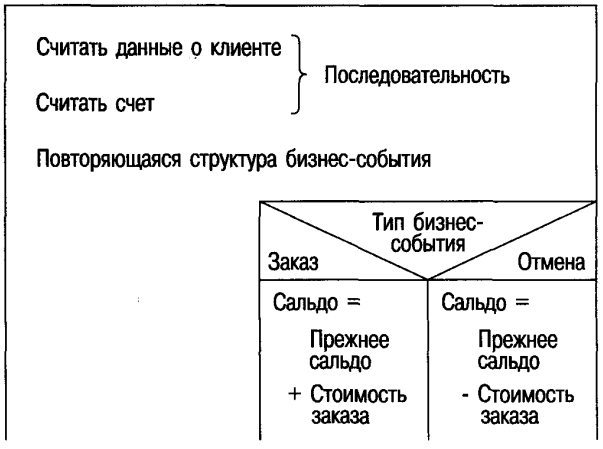


Рис. 122. Структура управления

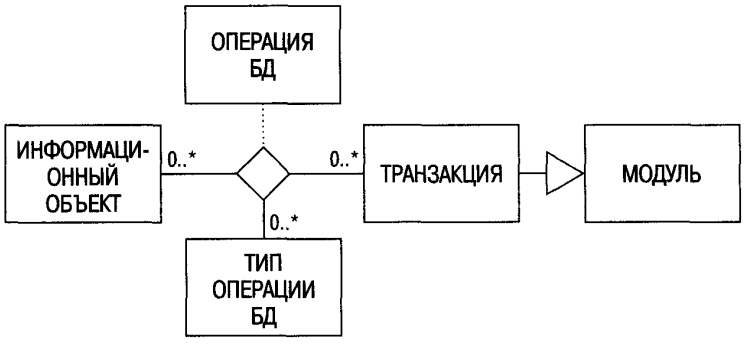


Рис. 123. Концепция транзакций

руемых в случае, если эти условия удовлетворяются. Действия представляют собой операции (т.е. транзакции) обновления данных. Более точно, в соответствии с механизмом событие/триггер (ETM) *(Kotz. Triggermechanismus in Datenbanksystemen. 1989, с. 54)* тригге-ром называется пара действий, состоящая из результата и собственно действия {T = (E,A)}. Таким образом, действие A выполняется, как только происходит событие типа E.

Например, если в процессе разработки продукта по завершении этапа «фаза проектирования 1» запускается процедура проверки результата, то триггер будет выглядеть следующим образом:

```
EVENT end_of_design_phase_1 (design
object: DB_ID);
ACTION verification_procedure_A
(verif_obj: DB_ID)
=<Verification of verif_obj>;
TRIGGER T1 =
ON end_of_design_phase_1
DO verification_procedure_A
(design_object);
```

(Kotz. Triggermechanismus in Datenbanksystemen. 1989, с. 64).

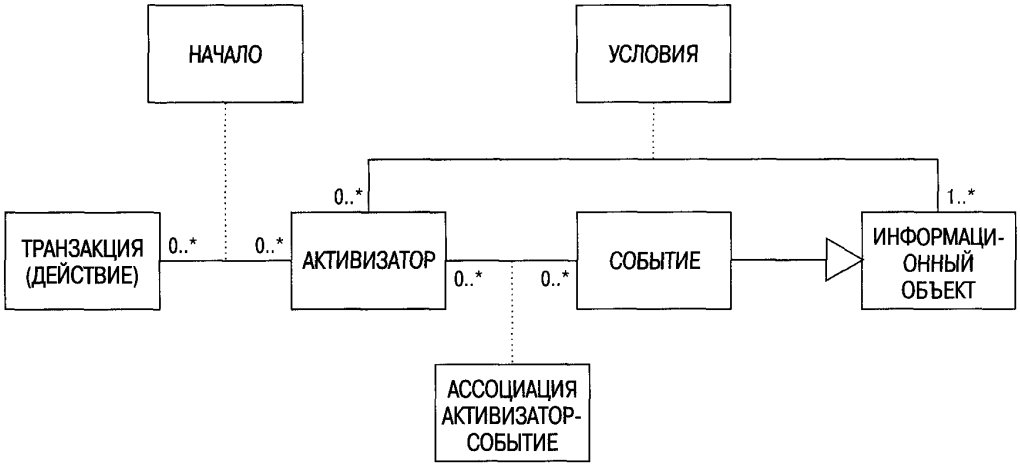


Рис. 124. Структура концепции триггеров

Здесь к событиям и действиям привязываются идентификаторы различных обрабатываемых объектов. В данном примере проектируемый объект именуется design_object (с идентификатором базы данных DB_ID), а объект, подлежащий проверке действием, именуется verification_object (с идентификатором базы данных DB_ID).

Раздельное описание событий, действий и триггеров обеспечивает различным триггерам доступ к одним и тем же событиям и описаниям действий. В то же время параметризация событий и действий позволяет реализовать различные процедуры управления, несмотря на идентичные базовые описания.

Поскольку приведенный здесь синтаксис относится к уровню спецификации проекта, он не зависит от особенностей триггеров, используемых в конкретных системах баз данных или средах программирования.

Механизм событие/триггер (ETM) можно непосредственно связать с СДП, так как события уже введены, а действия соответствуют функциональным модулям. Триггеры характеризуют контекст между E и A, совпадая с линиями соединений между событиями и функциями СДП.

Процессы, разработанные в виде СДП, могут быть трансформированы в нотации триггерного управления. Необходимо только дополнить диаграммы СДП символами, принятыми на уровне спецификации проекта. Сюда входит идентификация событий, триггеров и сообщений, соответствующих действиям.

На рис. 124 концепция триггеров представлена в виде метамодели, где класс СОБЫТИЕ является связующим звеном с уровнем определения требований.

Помимо внешних событий, существуют также внутренние события, обусловленные приложениями, например, подтверждение заказа. Конкретные моменты времени также могут описывать события, например, когда определенные действия выполняются по истечении каждого часа. Поскольку само время тоже можно описать как класс, сделав его информационным объектом, обозначение событий на уровне определения требований может быть использовано и для триггеров, описываемых здесь. Одно событие может инициироваться несколькими триггерами, а один триггер может инициироваться несколькими событиями.

После инициации того или иного триггера различные состояния базы данных могут проверяться в соответствии с правилами, описанными для триггеров. На это указывает ассоциация УСЛОВИЯ, связывающая ТРИГГЕР с ИНФОРМАЦИОННЫМ ОБЪЕКТОМ.

Если условия удовлетворяются, один триггер может инициировать одну или несколько транзакций. И наоборот, одна транзакция может быть инициирована различными триггерами. Скажем, при проверке складских запасов активизирующим событием может служить определенный момент времени (например, ежечасно проверяется наличие на складе некоего минимума запасов). Другим событием может быть бухгалтерская проводка складского запаса.

A.3.2.3.3. Объектно-ориентированная спецификация проекта

Одним из аспектов парадигмы объектно-ориентированного проектирования является наличие тесных взаимосвязей между фазами жизненного цикла. Проектируемые элементы на уровне определения требований следует в идеале реализовать с мощностью 1:1. Для этого существует ряд методов, один из которых — экспресс-создание прототипов. Тем не менее фазовые концепции характерны даже для объектно-ориентированного проектирования. Однако границы между определением требований, спецификацией проекта и описанием реализации не являются жесткими. Поскольку эти методы моделирования вышли из недр объектно-ориентированного программирования, они в значительной мере аналогичны описанию реализации. В свете этого в некоторых работах *(например: Oestereich. Objektorientierte Softwareentwicklung. 1997, с. 66)* предлагается использование диаграмм взаимодействия только на уровне определения требований (фаза анализа), тогда как за спецификацией проекта (фаза проектирования) уже закреплены диаграммы классов, последовательностей и состояний. Другие авторы применяют одни и те же методы и диаграммы на обоих уровнях с той лишь разницей, что на стадии спецификации проекта их описание подвергается дальнейшей детализации.

Мы склоняемся в пользу второй теории. При таком подходе отпадает необходимость в воссоздании метамоделей определения требований по умолчанию. Однако если вопросы производительности обусловливают необходимость в перегруппировке или детализации объектов, то между объектами на уровне определения требований и объектами на уровне спецификации проекта возможны отношения n:n. Применительно к метамодели это означает, что проектируемый объект на уровне спецификации проекта должен связываться с каждым проектируемым объектом на уровне определения требований посредством ассоциации n:n.

A.3.2.3.3.1. Общая детализация

Мы можем разграничить внутренние методы и методы, доступные извне.

Определяются типы атрибутов классов с техническими свойствами, такими как гарантии, начальные значения и параметры (см. рис. 125). Гарантиями называются условия, которым должны удовлетворять объекты. Параметры в операциях указывают, для каких критериев возможен перенос значений после запуска операций. В примере на рис. 125 очевидно представлено, что атрибут «радиус» не должен быть отрицательным (гарантия), центральная точка по умолчанию имеет начальное значение x = 10, y = 10, а окружностью можно манипулировать (если она не отцентрирована на экране) путем изменения координат (положения) центральной точки или ввода новых параметров для создания других радиуса.

Динамическое поведение объектов в течение их жизненного цикла можно диф-

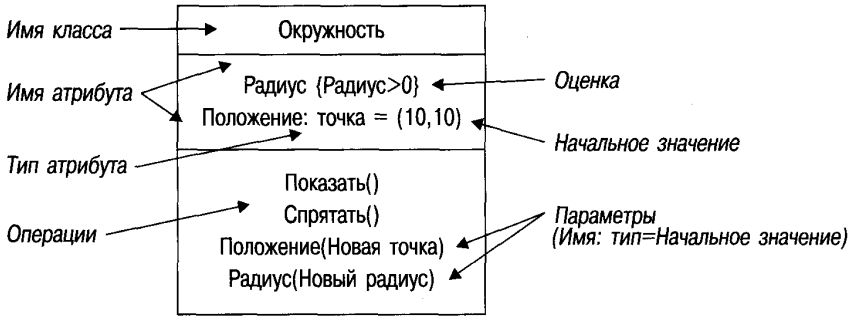


Рис. 125. Технические свойства атрибутов *(Oestereich. Objektorientierte Softwareentwicklung. 1997, с. 36)*

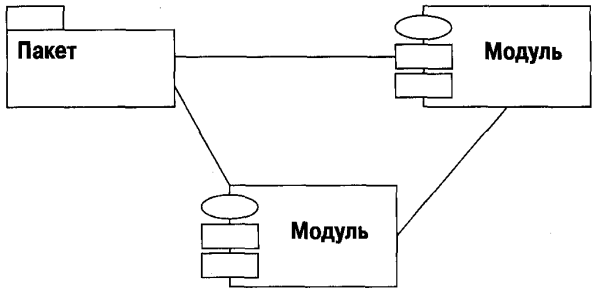


Рис. 126. Компонентная диаграмма UML

ференцировать при помощи диаграмм состояний, последовательностей и действий.

Термины «программы» и «модули» также используются в объектно-ориентированном системном проектировании; здесь модули содержат код классов, которые им присваиваются, или же связываются с файлами исходного кода и файлами трансляции.

Модули и их взаимосвязи отображаются компонентными диаграммами. Чаще всего их метаструктуры совпадают с метаструктурой отображения модулей. Управление версиями в значительной степени зависит от описания компонентов или модулей.

Термин «пакет» выполняет аналогичную функцию применительно к группировке элементов объектно-ориентированного описания. Пакеты не содержат физических кодов, а лишь объединяют элементы моделирования. Иногда термины «пакет» и «модуль» используются как синонимы.

Модули и пакеты являются компонентами UML и описываются с помощью компонентных диаграмм (см. рис. 126).

A.3.2.3.3.2. Связи с базами данных

Долговечные (персистентные) объекты можно хранить в объектно-ориентированных базах данных, поддерживающих также принципы наследования, чтобы объектные данные подчиненного класса можно было постоянно компоновать из унаследованных атрибутов доминирующих классов.

С другой стороны, если в объектно-ориентированном проектировании использу-

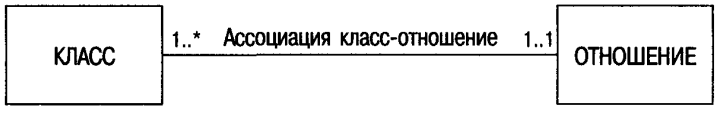


Рис. 127а. Каждый долговечный объект хранится в реляционной таблице

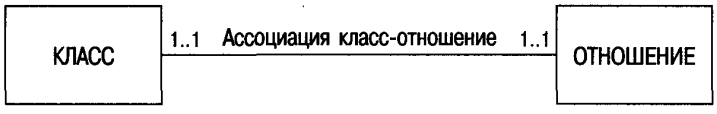


Рис. 127б. Каждому классу присваивается отношение

ются реляционные системы баз данных, могут возникать проблемы со связью двух парадигм проектирования. По умолчанию возможно применения двух противоположных методов для решения этого вопроса *(Oestereich. Objektorientierte Softwareentwicklung. 1997, с. 136).*

Метод 1. Каждый объект каждого класса хранится только в одном отношении, поэтому необходимо обновлять различные виды кортежей в таблице. Принцип метамодели требует, чтобы все КЛАССЫ были связаны только с одним ОТНОШЕНИЕМ (см. рис. 127а).

Метод 2. Для каждого класса создается отдельная таблица с мощностью ассоциации 1:1 (см. рис. 127б). При этом возникает необходимость в группировке данных из нескольких доминирующих классов для объектов подклассов.

Компромиссным решением было бы хранить данные на самом нижнем уровне подклассов (Oestereich. Objektorientierte Softwareentwicklung. 1997, с. 137). Это означало бы группировку всех взаимосвязанных данных в одном объекте. Однако в этом случае при обращении к доминирующим классам с множеством подклассов сначала пришлось бы сгруппировать данные из множества таблиц.

Подытоживая, можно сказать, что установление связей с базами данных до сих пор остается проблематичным, поскольку технология объектно-ориентированных баз данных пока еще в полной мере не разработана, а интерфейсы между языками объектно-ориентированного проектирования и реляционными системами баз данных далеки от идеальных.

A.3.2.4. Описание реализации

Взаимосвязи между функциями и данными реализуются системами баз данных и языками программирования. Опираясь на функциональные возможности языка описания данных (ЯОД), системы баз данных позволяют создавать конкретное описание внешних схем. В частности, активные системы баз данных предоставляют функциональные возможности для реализации механизмов активизации.

Объектно-ориентированные языки программирования (C++, Smalltalk, Java и т.д.) позволяют реализовать методы и диаграммы спецификации проекта в программном коде (см. рис. 128, где описание класса «окружность», приведенное на рис. 125, реализовано на языке C++).

```
class cirle
{
        int radius;
        point position;

    public:
        void radius (int newradius);
        void position (point newpoint);
        void display ();
        void hide ();
};
void cirle::radius (int newradius)
{
        if (newradius > 0)// evaluation
        {
         ...
        };
};
...
```

Конкретная реализация одиночных операций здесь опущена.

Рис. 128. Реализация объектного класса, приведенного на рис. 125 *(Oestereich. Objektorientierte Softwareentwicklung. 1997, с. 37)*

A.3.3. Отношения между функциями и выходом

Понятие «выход» охватывает материальный выход и услуги, в том числе информационные.

Отношения выхода с функциональной моделью состоят в том, что функции преобразуют вход в выход посредством обработки. Проще говоря, выход можно связать с тем или иным процессом (т.е. набором функций). Таким образом, создание входа инициирует процесс или, иными словами, выход есть также результат процесса.

На рис. 129 показаны отношения между функциями и выходом в здании ARIS.

A.3.3.1. Моделирование на уровне определения требований

На рис. 130 показана детальная модель выходов, включая количественные взаимосвязи функции, а на рис. 131 — соответствующая метамодель. Для упрощения результаты в метамодели опущены. На этом рисунке мы исходим из предположения, что после выполнения каждой функции создается выход, имеющий свое собственное обозначение.

На рис. 132 и 133 представлена ситуация, когда выход после обработки приобретает другое состояние, но не новое обозначение. Поэтому завершенный функциональный этап можно использовать для идентификации начального состояния для следующего процесса. Эта процедура применяется также для составления графиков операций в промышленном производстве.

Каждый результат бизнес-процесса моделируется путем привязки выходов к функциям. Выход последней функции бизнес-процесса является конечным выходом данного бизнес-процесса.

Привязка типов стоимости к выходу, как было показано при отдельном описа-

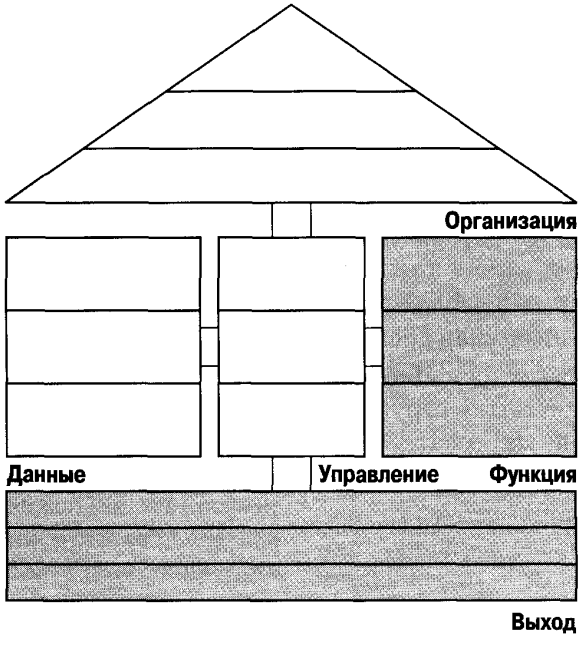


Рис. 129. Отношения между функциями и выходом

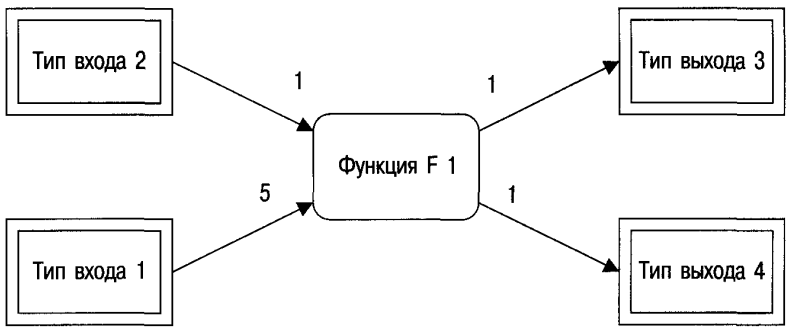


Рис. 130. Поток выходов функции

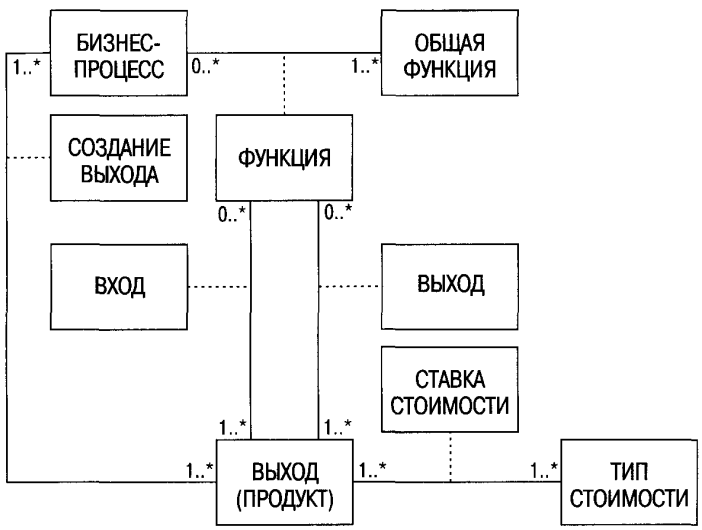


Рис. 131. Метамодель изменения типов выходов после выполнения функций

нии модели выходов, позволяет включить в модель и поток затрат в рамках процесса.

Поток выходов в рамках функционального процесса включает также структуру выхода, т.е. прейскурант материалов.

При описании потока управления жесткое разделение описания структур продуктов и процессов после создания продуктов приводит к отношениям подстановки (см. рис. 134). Связывание выходов с процессами позволяет описать процесс создания выхода из входа в той же мере, в какой выход моделируется в рамках структуры выхода.

На рис. 134а структура выхода характеризуется двумя промежуточными продуктами. Для создания выхода необходима одна функция на каждый продукт. В результате мы получаем четыре диаграммы EPC, каждая из которых описывает одну функцию, относящуюся к изготовлению соответствующего продукта.

На рис. 134б та же структура выхода представлена в виде двух выходов, т.е. без

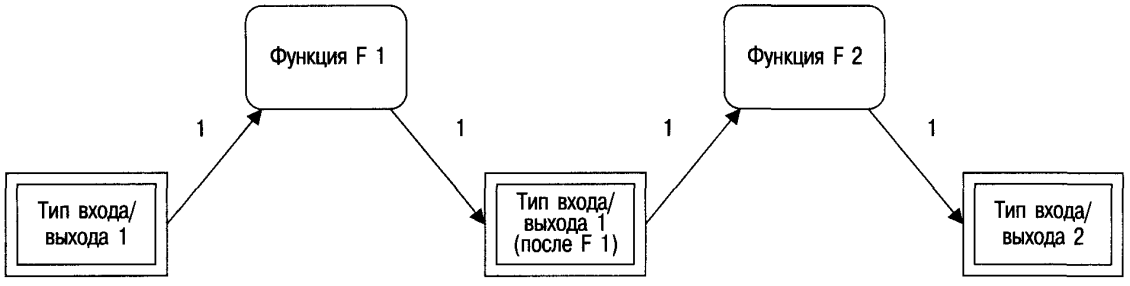


Рис. 132. Поток выходов после выполнения функций без изменения обозначения выхода

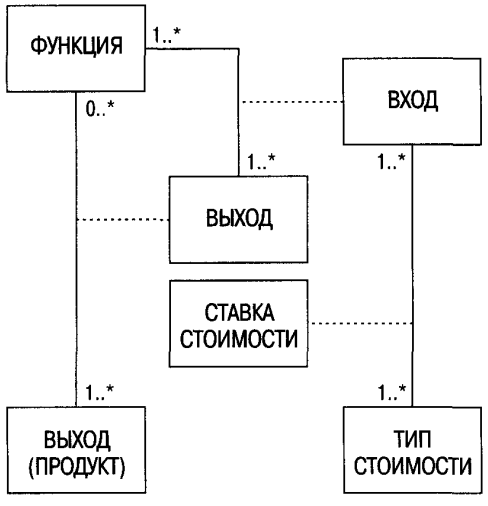


Рис. 133. Метамодель обновления состояния после выполнения функции

описания промежуточных продуктов. Здесь соответствующий бизнес-процесс для создания конечного выхода более сложный; параллельная обработка вводится теперь не в прейскурант материалов, а в процесс.

Отсюда понятно, почему графики работ в промышленном производстве, как првило включают лишь несколько (7–10) операций (функций). Это обусловлено тем, что наибольшая часть структуры процесса вводится в прейскурант материалов.

Однако в общем моделировании бизнес-процессов мы обычно имеем дело со всей структурой процесса. Лишь косвенно затрагивая промежуточные продукты, она включает также ряд функций и параллельных структур. Чем более обособлено администрирование выходов, процессов и соответствующих отношений, тем чаще возникают идентичные возможности. Это достигается тем, что структуры выходов моделируют структуру ключевых процессов, а процессы моделируют только непосредственные процессы входа/выхода для каждого выхода.

Метод UML описывает поток объектов, обрабатываемых функциями, посредством диаграмм операций или действий (см. рис. 135). Изменения состояний, вызванные

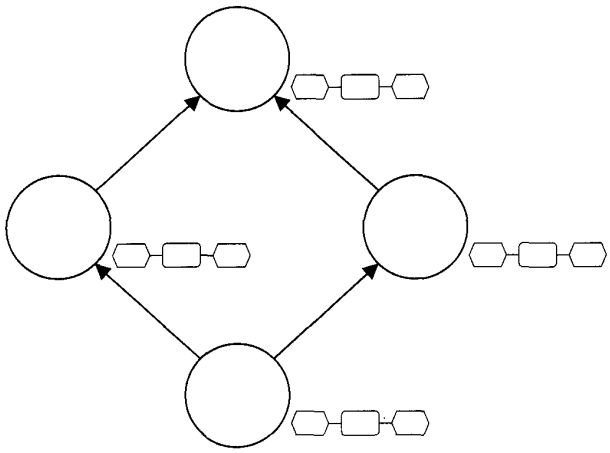


Рис. 134а. Структура процесса и структура выхода с промежуточными продуктами

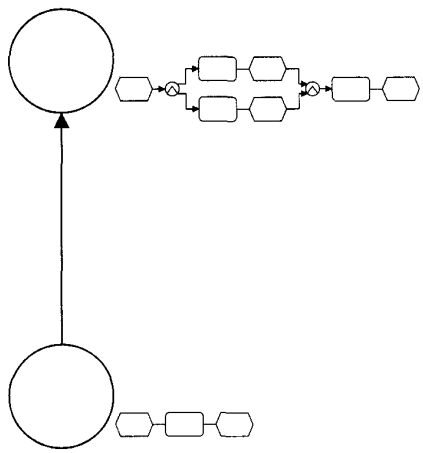


Рис. 134б. Структура процесса и структура выхода без промежуточных продуктов

функциями, можно интерпретировать как результаты, т.е. как информационные выходы. Метамодель (см. рис. 133) уже отражает этот поток объектов. В дополнение к этому, к операционным диаграммам привязаны организационные единицы.

A.3.3.2. Конфигурирование

После того как функция выполнена, изменения в состоянии выхода и соответствующих затратах передаются на уровень управления бизнес-процессом. В зависимости от типа выхода (материальный выход или услуги) эти данные доставляются системой сбора производственных данных или — в информационных системах — системами workflow.

В процессе конфигурирования типы выходов и типы стоимостей, которые необходимо отслеживать, передаются системе управления процессами.

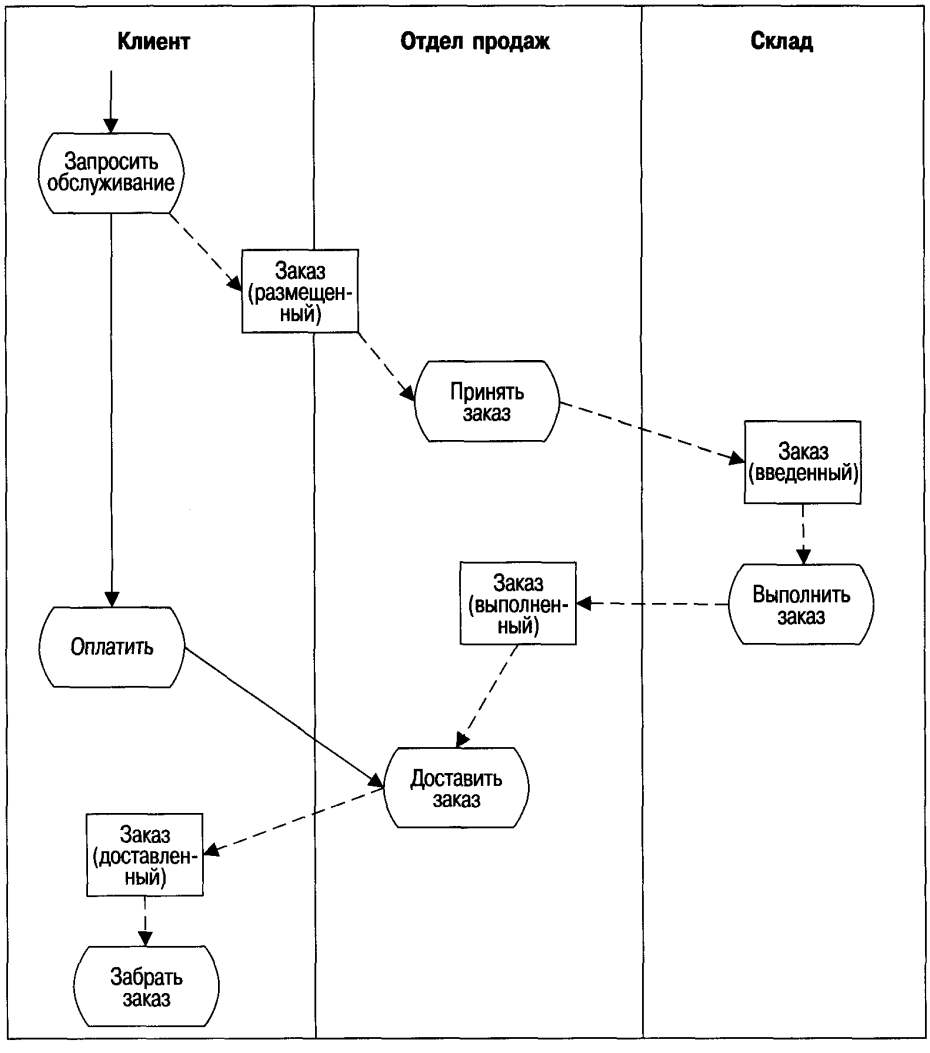


Рис. 135. Поток действий и объектов (UML Notation Guide. 1997, рис.56)

Информационные услуги, создаваемые и передаваемые для дальнейшей обработки, описываются как папки. Эта информация задает конфигурацию системы workflow и описывает электронные папки.

A.3.4. Отношения между организационной структурой и данными

На рис. 136 показано, как соотносятся организационная структура и данные в «здании» ARIS.

A.3.4.1. Моделирование определения требований

По аналогии с функциональной моделью, где к корпоративным уровням планирования привязываются ключевые функции (см. верхнюю часть рис. 87), на рис. 137 представлена модель данных, где к уровням планирования привязаны важнейшие объекты данных. Отношение между организационной структурой и данными можно квалифицировать более подробно с помощью таких терминов, как «используемый», «уполномоченный на чтение» или «ответственный за обновления». Соответствующая метамодель показана на рис. 138.

При более пристальном рассмотрении полномочия пользователей на уровне атрибутов описываются такими операциями, как «создать», «удалить», «обновить» или «читать». Эти полномочия можно представить в виде таблиц. Соответствующая метамодель приведена на рис. 139. Пользователи, связываемые с определенной организационной единицей, описываются через должности.

Мы должны определить, какие атрибуты разрешено обрабатывать каждому типу операции и каждому пользователю. Можно также задать дополнительные пределы для экземпляра значений атрибута. Например, если пользователю отдела управления персоналом разрешается читать сведения о заработной плате только ниже определенной суммы, это необходимо уточнить с помощью значений атри-

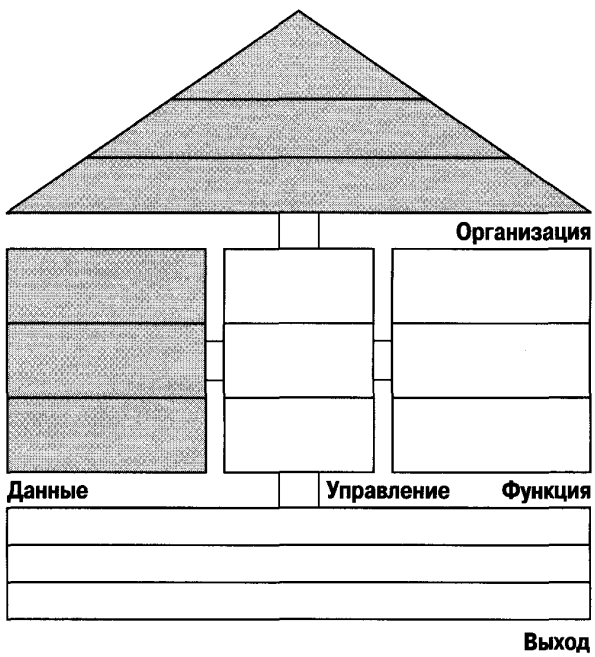


Рис. 136. Отношения между организацией и данными

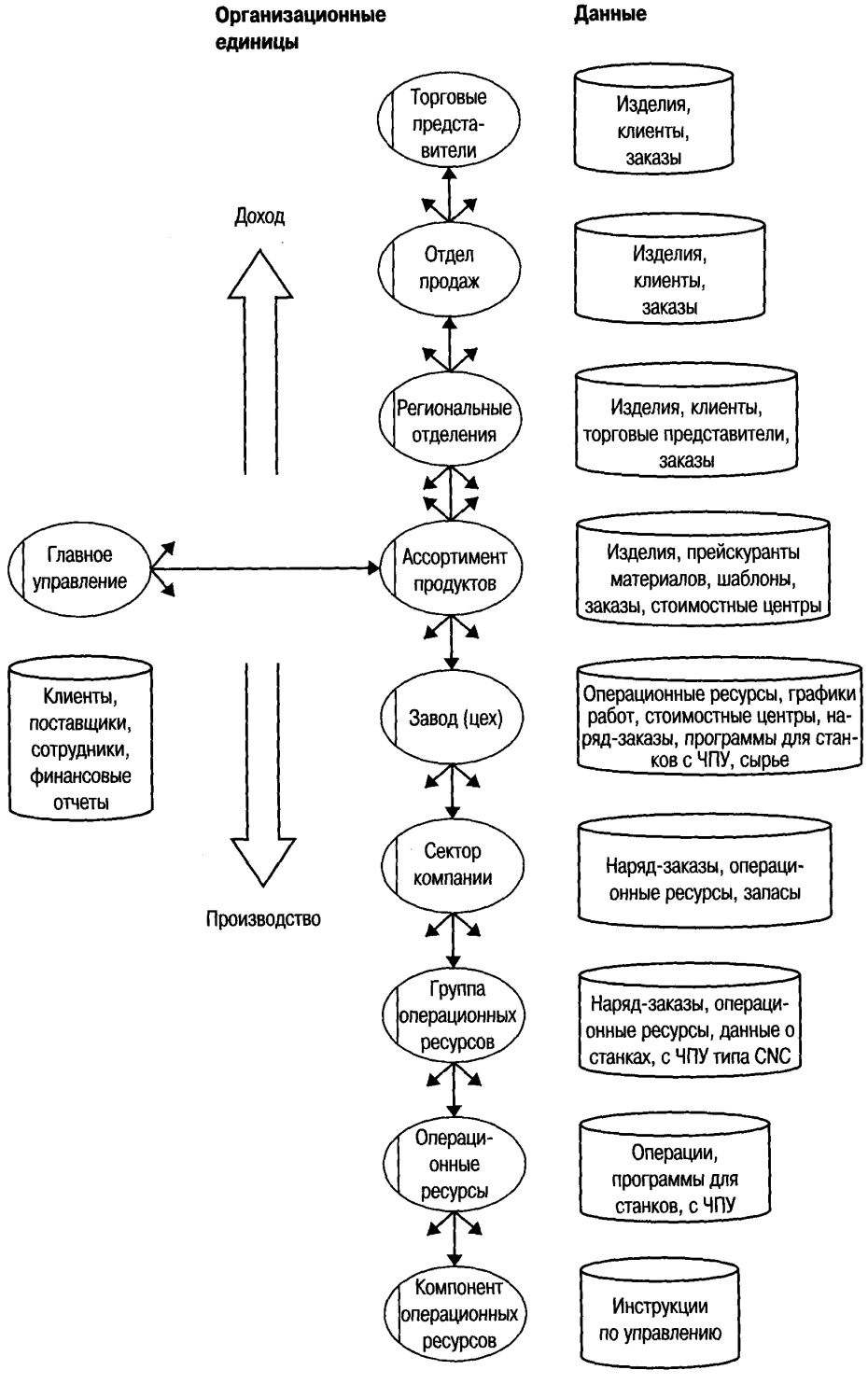


Рис. 137. Модель на уровне данных

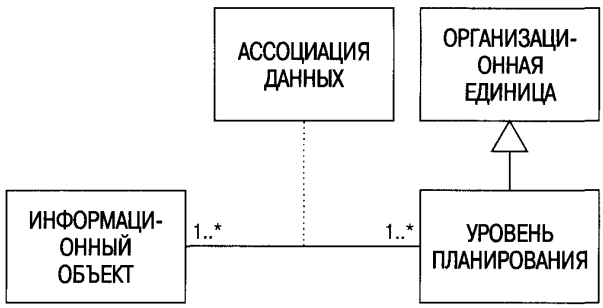


Рис. 138. Метамодель, связывающая организацию и данные

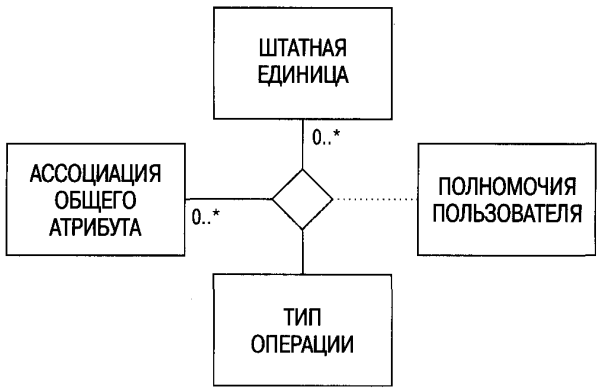


Рис. 139. Метамодель присвоения полномочий

бутов ассоциации ПОЛНОМОЧИЯ. При привязке должностей можно также задать различные полномочия для нескольких типов операций. Например, сотрудник отдела управления персоналом может обновлять данные о заработной плате до определенной суммы, а читать их - до некоторой другой суммы.

Как и управляющие функции, организационные единицы могут активизироваться событиями. Эта процедура называется «управлением процессами, ориентированными на действия (операции)». По аналогии с триггерами функций передача элементов данных в виде побуждающих сообщений инициирует выполнение действий организационными единицами, как показано на рис. 140. Это особенно важно в управлении потоками работ, где побуждающие сообщения рассылаются в виде электронных папок и представляются при помощи диаграмм EPC в соответствии с организационным распределением функций. В упрощенном виде обмен сообщениями между организационными единицами описывается диаграммами взаимодействий (см. рис. 114).

A.3.4.2. Конфигурирование

Привязка данных с привилегиями доступа к организационным единицам предоставляет сотрудникам, участвующим в управлении бизнес-процессами, необходимые инструменты для работы с хранилищами данных.

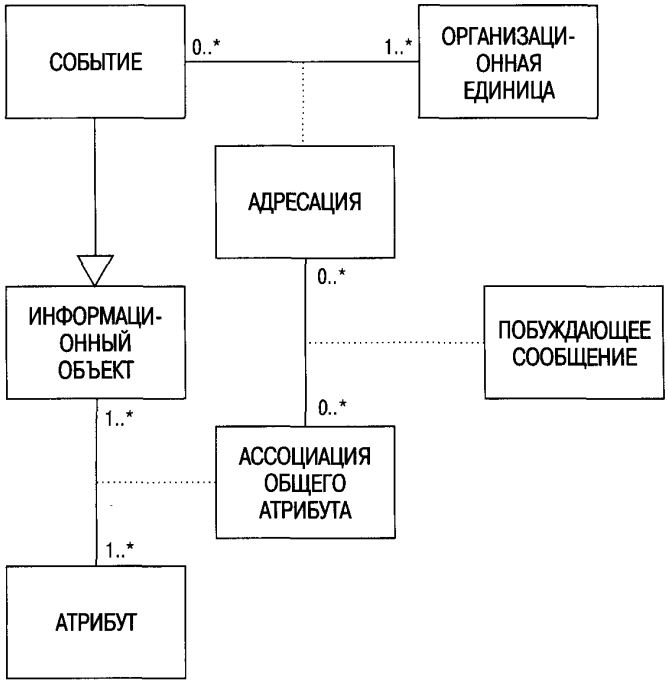


Рис. 140. Управление организационными единицами посредством событий

Присвоение полномочий позволяет системам workflow конфигурировать входные потоки документов в организационных единицах. Для этой цели диаграммы EPC дополняются связями организационных единиц с функциями.

Компоненты бизнес-приложений можно конфигурировать с помощью полномочий при привязке экранов и доступа к данным.

A.3.4.3. Спецификация проекта

A.3.4.3.1. Детализация полномочий

На стадии спецификации проекта выполняется дальнейшая детализация привилегий, присваиваемых пользователям для манипулирования данными. На рис. 141 вкратце представлены различные возможности определения привилегий доступа. К числу основных привилегий относятся чтение, создание, редактирование и удаление данных. Соответствующие объекты можно разбить на категории. Например, отношение можно рассматривать как объект, а объекты - как отдельные атрибуты, отдельные кортежи или комбинацию кортежей. Логические связи с содержанием данных можно использовать в качестве критериев выбора. Например, руководителям отделов можно разрешить читать личные дела сотрудников только своего отдела. Другой пример в том же прикладном контексте: можно предоставить сотруднику отдела управления персоналом право читать записи, относящиеся только к зарплате ниже определенного уровня.

Метаструктура такого детального представления большей частью согласуется с описанием на рис. 139.

Помимо прямой связи между ШТАТНОЙ ЕДИНИЦЕЙ и АССОЦИАЦИЕЙ ОБ-

Пользователь	Объекты базы данных				
	O_1	O_2	O_3		O_n
U_1	P_1,P_2		P_3		P_1
U_2		P_1	P_2,P_3		P_1
U_3		P_1,P_2	P_2		
U_m	P_1,P_2	P_1	P_1		P_1,P_n

Условные обозначения:

U_i : Пользователь
O_i : Объекты базы данных
P_i : Привилегии

Рис. 141. Таблица полномочий *(Reuter. Sicherheits- und Integritätsbedingungen. 1987, с. 352)*

ЩЕГО АТРИБУТА (см. рис. 139), можно установить косвенную связь, присвоив привилегии определенным паролям, а затем определенным пользователям. Это позволяет сделать описание менее избыточным, так как, например, два пользователя с одинаковым профилем привилегий получают одинаковые привилегии, обусловленные паролем. Эта метамодель показана на рис. 142.

Механизм защиты можно описывать с помощью таблиц, где проверяется каждый ключевой запрос к базе данных. Реляционные базы данных CA-Ingres отличаются особой реализацией, описывающей правила присвоения привилегий в виде так называемых диапазонных условий. При выполнении запроса диапазоны условий и описание запроса группируются в инструкцию ЯМД (язык манипулирования данными) и, таким образом, обрабатываются в рамках одного и того же синтаксиса. Рис. 143 в пояснениях не нуждается.

A.3.4.3.2. Распределенные базы данных

На уровне определения требований выполняется логическая привязка данных к организационным единицам независимо от физических систем хранения. На уровне спецификации проекта фрагменты схемы базы данных распределяются между узлами. Управление этими фрагментами может осуществляться разными системами управления базами данных (СУБД) при условии, что они могут взаимодействовать друг с другом. Таким образом, на системы баз данных возлагаются две задачи: управление локальными данными, хранящимися на узле, и их координация.

Распределенные базы данных повышают отказоустойчивость всей системы, обеспечивают более оперативное обновление данных, сокращают затраты и повышают гибкость *(Nerreter. Zur funktionalen Architektur von verteilten Dantenbanken.*

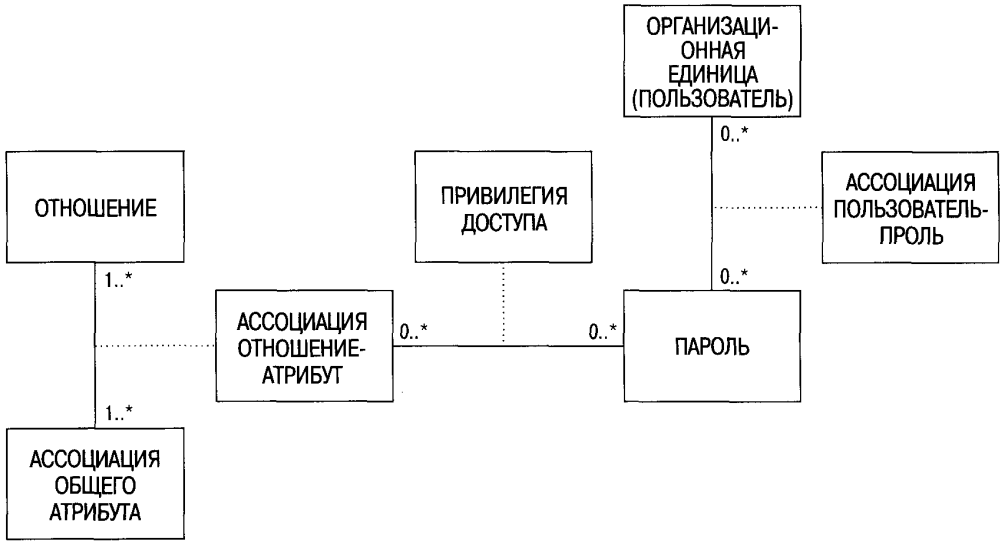


Рис. 142. Метамодель описания полномочий

1983, с. 2; Jablonski. Datenverwaltung in verteilten Systemen. 1990, с. 5). Недостаток, однако, заключается в том, что для их координации требуются значительные усилия.

Дейт объединил свойства распределенных СУБД в 12 правил, сформулировав фундаментальный принцип распределенных баз данных в виде правила 0 *(Date. Distributed Database Systems. 1987).*

Правило 0. Основополагающий принцип

Для пользователей распределенная система баз данных не должна представляться как распределенная; иными словами, для них она должна выглядеть централизованной базой данных. Это правило позволяет поддерживать единообразно описанные условия целостности и привилегии доступа, поскольку на них не влияет распределение данных в соответствующей схеме.

Правило 1. Локальная автономия

Каждый узел должен управлять вверенными ему данными на месте, обеспечивая возможность локального управления их защитой, целостностью и хранением. Это позволяет беспрепятственно выполнять чисто локальные операции в условиях распределенной системы.

Правило 2. Локальные компоненты не зависят от центрального компонента

Это означает, что каждый локальный участок имеет равные права. Если бы узел был подчинен централизованной системе, это негативно сказывалось бы на всей системе в случае аварии.

Правило 3. Безостановочная работа

При отключении отдельных узлов не должно возникать необходимости в прерывании работы системы.

Правило 4. Локальная прозрачность

Пользователям необязательно знать, где хранятся данные.

Устная формулировка	На ЯМД CA-Ingres QUEL
Устная формулировка	
Руководителю отдела Миллеру разрешается доступ к личным делам всех сотрудников его отдела D43.	RANGE OF X IS EMPLOYEE RESTRICT ACCESS FOR 'MILLER' TO EMPLOYEE WHERE X.DEPT.MEMB='D43'
Описание запроса	
Найти фамилии всех сотрудников с годовой зарплатой более 30 тыс. долл.	RANGE OF X IS EMPLOYEE RETRIEVE INTO LIST (X.NAME) WHERE SALARY > 30,000
Скомпилированный запрос	
	RANGE OF X IS EMPLOYEE RETRIEVE INTO LIST (X.NAME) WHERE SALARY > 30,000 AND X.DEPT.MEMB = 'D43'

Рис. 143. Присвоение полномочий в QUEL *(Reuter. Sicherheits- und Integritatsbedingungen. 1987, с. 361)*

Правило 5. Прозрачность фрагментации

Для повышения производительности системы отношения можно разбивать на фрагменты. При этом пользователи могут продолжать работать, как если бы эти наборы данных не были фрагментированы.

Правило 6. Прозрачность репликации

Копии идентичных данных можно связывать с различными локальными участками (микрорайонами). При этом пользователям необязательно знать, что существуют копии данных.

Правило 7. Распределенная обработка операций баз данных

Чтобы обеспечить возможность распределенной обработки запросов, можно внедрить оптимизаторы.

Правило 8. Управление распределенными транзакциями

В распределенной системе каждая отдельная транзакция может включать процессы обновления на других участках. Если обновление не доводится до конца, то происходит откат транзакции, т.е. отказ от всех изменений, сделанных до последней точки сохранения.

Правило 9. Аппаратная прозрачность

Система управления базами данных (СУБД) должна поддерживать различные аппаратные системы.

Правило 10. Прозрачность операционной системы

СУБД должна поддерживать различные операционные системы.

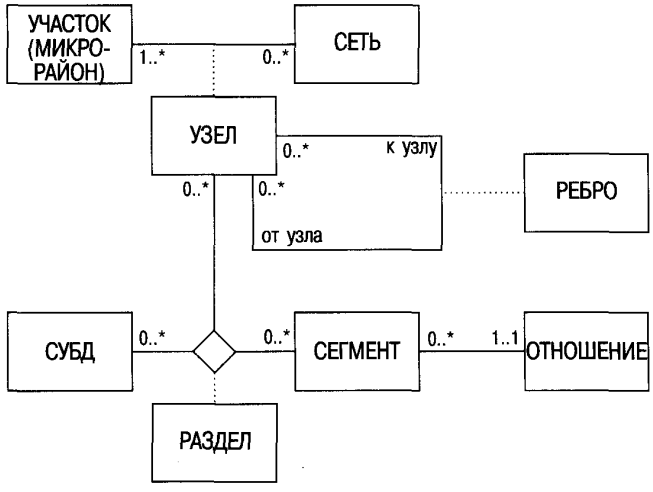


Рис. 144. Распределенные базы данных

Правило 11. Прозрачность сети

СУБД должна поддерживать различные сетевые системы.

Правило 12. Прозрачность баз данных

Распределенная реализация должна быть возможна и для различных локальных СУБД, включая использование шлюзов в качестве мостов между различными базами данных.

На рис. 144 показана информационная модель, построенная в соответствии с этими принципами, применительно к понятиям, взятым из спецификации проекта на уровне организационной модели (представлена топологией сети) и модели данных (представлена реляционной схемой) *(Jablonski. Datenverwaltung in verteilten Systemen. 1990, с. 198).*

Фрагментация означает, что отношения можно разбивать по горизонтали или по вертикали с передачей каждому фрагменту ключа базисного отношения. Фрагменты могут частично накладываться друг на друга (см. рис. 145). Такие фрагменты представлены на рис. 144 классом СЕГМЕНТ, при этом каждый сегмент однозначно связан с ОТНОШЕНИЕМ. Сегмент данных, привязанный к определенному узлу или системе управления базами данных, называется РАЗДЕЛОМ *(Jablonski. Datenverwaltung in verteilten Systemen. 1990, с. 193).*

Система клиент-сервер представляет собой упрощенный вариант распределенной системы (без условий прозрачности). Клиенты посылают запросы серверу базы данных, который обслуживает эти запросы, используя свои внутренние функции управления данными. Привязка отношений к серверу базы данных представляет собой поднабор метамодели. На стадии описания реализации физические данные привязываются к физическим компонентам систем хранения. При администрировании прозрачных распределенных систем возникают затруднения, связанные с проблемами координации. Решение этих вопросов в условиях сегодняшних систем баз данных лишь в самой начальной стадии.

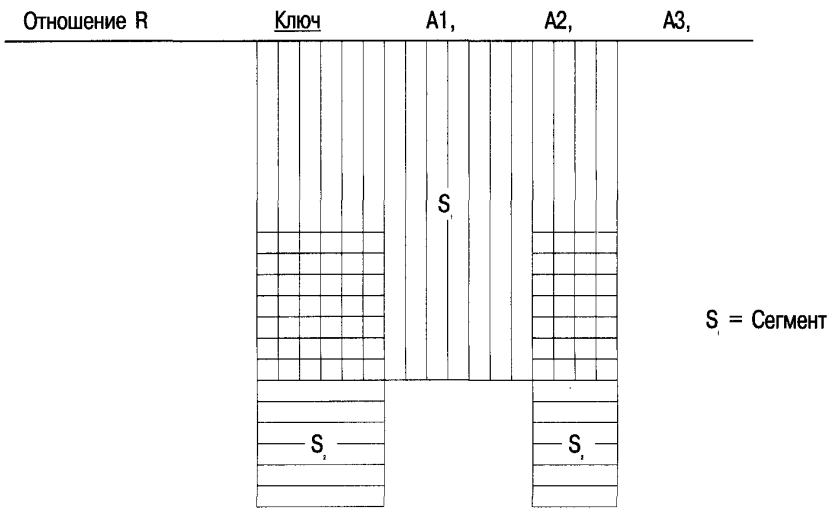


Рис. 145. Сегментация отношения

A.2.5. Отношения между организационной структурой и выходом

На рис. 146 приведены отношения между организационной структурой и выходом в здании ARIS.

A.3.5.1. Моделирование определения требований

В соответствии с организационным распределением функций и данных на рис. 147 показана привязка выхода к уровням планирования. Выход служит, например, источником базовых нормативов для управления, ориентированного на выход. Одни выходы (скажем, обращения торгового представителя в отдел продаж по поводу представленных предложений) обрабатываются той же организационной единицей, другие (например, предложения, пересылаемые в отдел продаж для дальнейшей обработки) передаются следующей организационной единице. Однако результатом выполнения той или иной функции в рамках организационной единицы могут быть также промежуточные продукты, необходимые для последующей функции, лежащей на той же организационной единице. В этом процессе участвуют различные организационные (под)единицы (рабочие группы, рабочие места) внутри соответствующих организационных единиц. Таким образом, число промежуточных продуктов, возникающих внутри организационной единицы, зависит от степени структурирования организационной модели.

Нередко описывается только поток выходов, получаемых организационной единицей от других организационных единиц.

Рис. 148 иллюстрирует детализированный поток выходов между организационными единицами «торговый представитель» и «отдел продаж», а также поток выходов к клиенту.

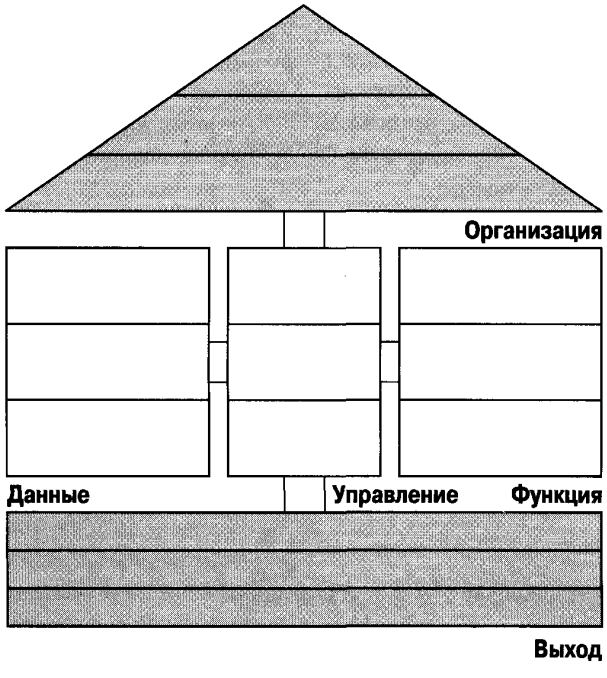


Рис. 146. Отношения между организацией и выходом

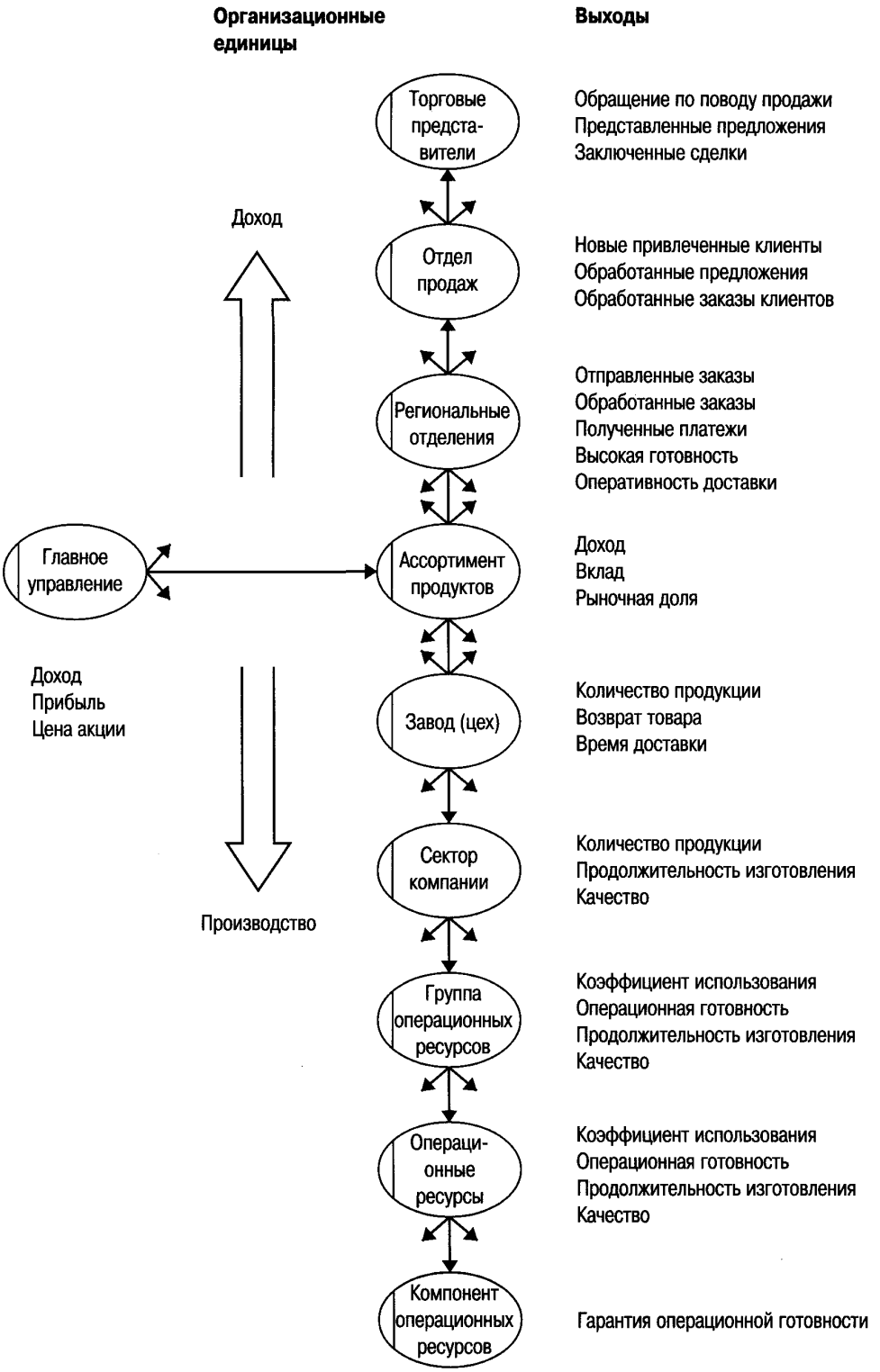


Рис. 147. Модель на уровне выходов

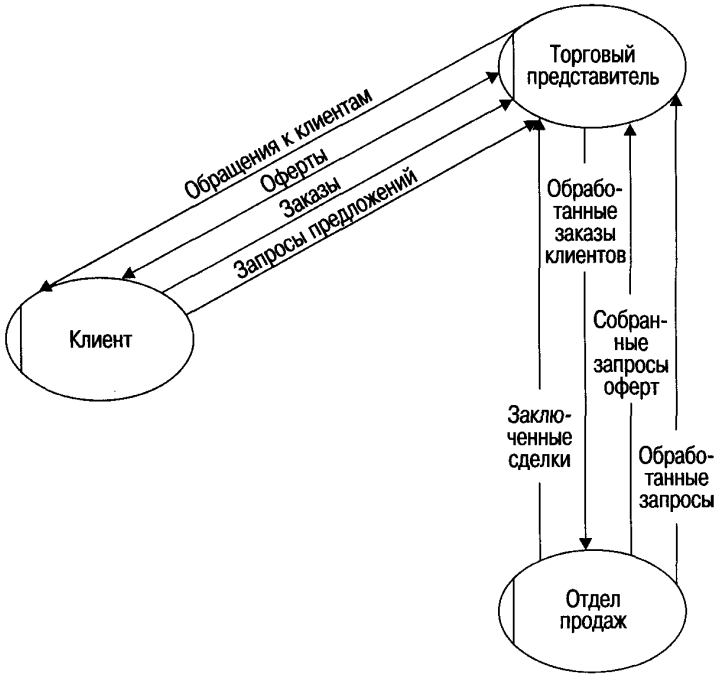


Рис. 148. Поток выходов между организационными единицами

Внешние субъекты действия, такие как клиенты, поставщики, правительственные органы и т.д., также могут быть описаны как организационные единицы. Для моделирования предварительных потоков выходов используются диаграммы взаимодействий.

Одно из свойств выхода заключается в том, что получатели должны быть готовы заплатить за него определенную цену, т.е. признать его ценность.

В теории бизнеса потоки выходов между организационными единицами имеют особую значимость при учете по стоимостным центрам. Для планирования затрат стоимостного центра определяются индикаторы выхода, известные также как эталонные значения. Элементарный пример приведен на рис. 149. Для эталонных значений определяются ставки стоимости, применяемые для расчета стоимости продуктов.

Если предполагается ввод затрат стоимостного центра, то необходимо оценить выходы, получаемые остальными стоимостными центрами. Это подводит нас к определению ставок стоимости выходов внутри компании, о чем мы уже говорили ранее.

На рис. 150 представлена метамодель отношений потоков выходов между организационными единицами.

Ассоциация ДОСТАВКА показывает, каким образом реализуется связь между выходами организационных единиц. АССОЦИАЦИЯ ДОСТАВКА–ВРЕМЯ, связанная с классом ВРЕМЯ, наряду с соответствующими обозначениями количества и стоимости содержит специфические отношения выходов в рамках временного процесса.

А.2.5.2. Конфигурирование

Ключевое значение для планирования и управления имеют типы выходов и количественные характеристики, создаваемые

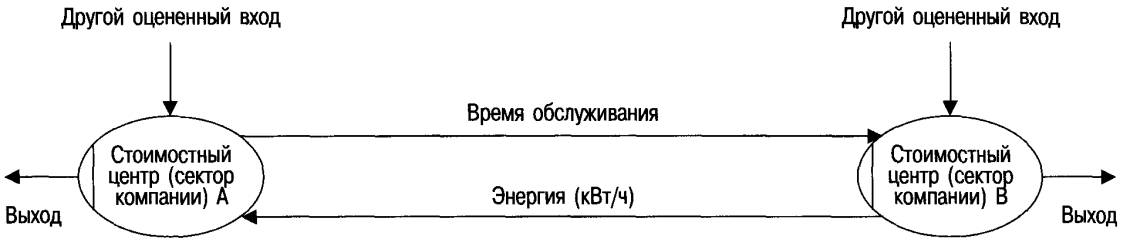


Рис. 149. Вычисление стоимости выхода внутри компании

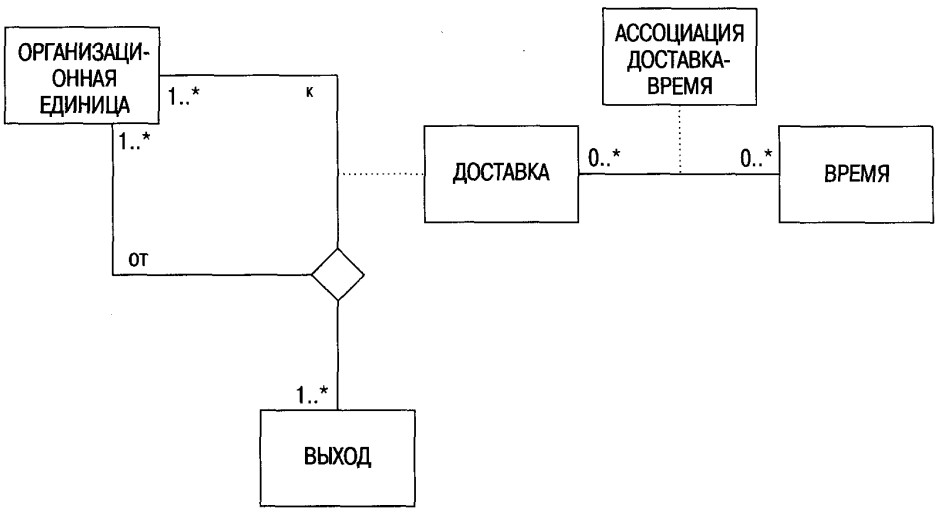


Рис. 150. Метамодель потока выходов

организационной единицей. Любые расхождения между существующими («как есть») и плановыми показателями можно использовать для корректировки плановых затрат в соответствии с целевыми значениями. Поэтому выход, связываемый с той или иной организационной единицей, определяет конфигурацию систем для расчета стоимости и выхода.

Выход, вводимый системой workflow в организационной единице, выводится непосредственно из контекста модели, позволяя детализировать описание организационных единиц вплоть до конкретных рабочих мест. Применительно к информационным услугам описание выхода соответствует описанию состояния передаваемых папок.

Если говорить проще, выход, связываемый с организационными единицами, определяет также необходимые функциональные возможности. Например, если с одной организационной единицей связаны только простые заказы, а с другой — еще и сложные, то их функциональные возможности будут разными.

Вместо привязки функций напрямую к организационным единицам их можно привязывать к выходу, если известен контекст функции в виде отношений между выходом и функциями.

A.3.6. Отношения между данными и выходом

На рис. 151 показано как соотносятся данными и выходом в здании ARIS.

A.3.6.1. Моделирование определения требований

Выход и данные уже связаны между собой в силу того, что информационные услуги описываются посредством данных. В примере на рис. 152. информационные услуги (результат выполнения функций) исходят из предпосылки, что данные о заказе еще не введены и не проверены. Выходом является состояние данных («введены и проверены»). При вводе данных их состояние обозначается самим фактом их существования, однако «проверенными» они становятся лишь после того, как соответствующий сотрудник рассмотрит их и «поставит галочку».

Для документирования обоих типов информации в объекте данных ЗАКАЗ создается атрибут состояния, фиксирующий состояние заказа и, таким образом, состояние выхода. Материальный выход и другие услуги представляют в виде их собственных физических или нефизических моделей, хотя в информационных системах их также описывают при помощи объектов данных. Например, материальный выход документируется в виде прейскурантов материалов или шаблонов САПР, тогда как услуги описываются с помощью функциональных спецификаций или других документов. Можно использовать и мультимедийные описания продуктов, например, видео.

Таким образом, материалы, обрабатываемые на производстве, существуют в виде материального выхода, но при этом они одновременно представлены и информационным объектом ДЕТАЛЬ (см. рис. 153). Соответствующее состояние вводит-

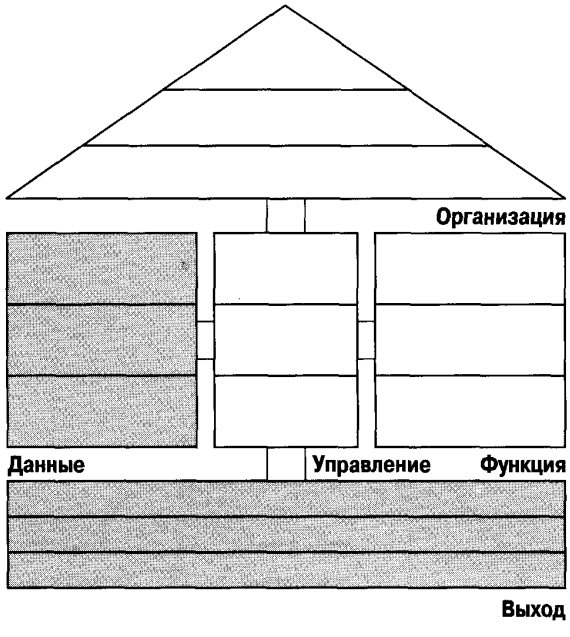


Рис. 151. Отношения между данными и выходом

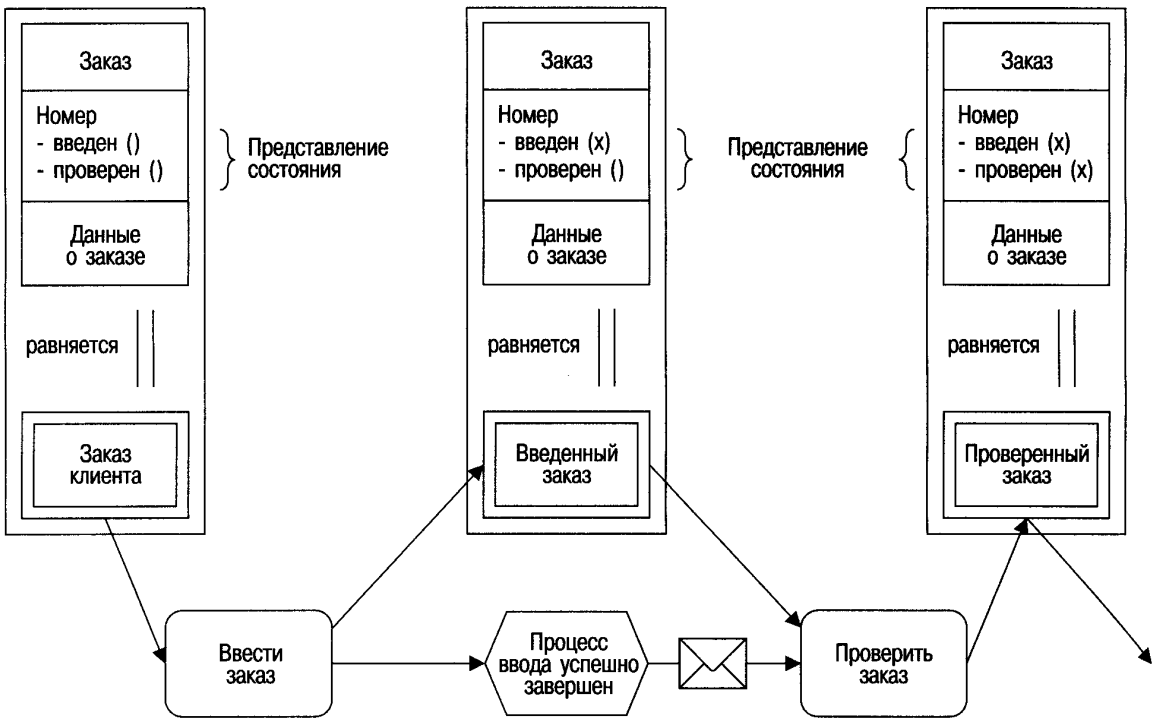


Рис. 152. Отношения между данными и информационными услугами

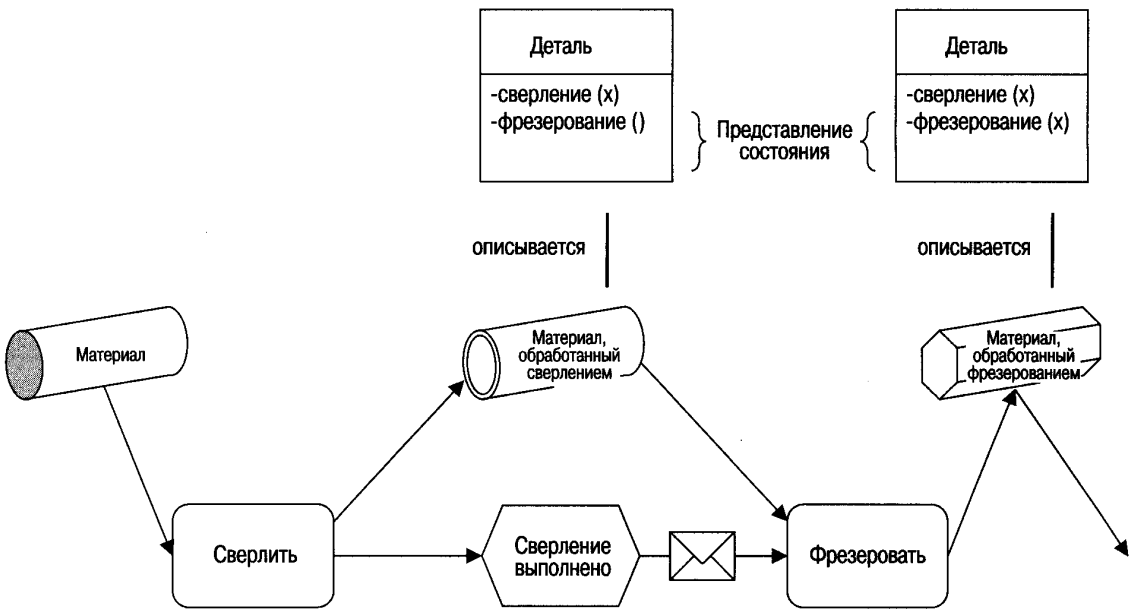


Рис. 153. Взаимодействие данных и материалов

ся в этот информационный объект, как если бы это был объект «информационная услуга». Единственное различие состоит в том, что в материальном выходе и услугах другого типа объекты данных описывают выход, тогда как в информационных услугах объекты выхода и объекты описания, соответственно, могут быть тождественны.

В метамоделях различие между материальным выходом и информационными услугами не имеет принципиального значения (см. рис. 154).

Поскольку выход описывается как самостоятельный класс, его состояние после обработки устанавливается при помощи связи ВЫХОДА с ФУНКЦИЕЙ, что позволяет присвоить выходу несколько состояний обработки вместо того, чтобы вводить новое описание выхода после каждой функции.

Состоянию обработки присваивают различные значения атрибутов, устанавливая связь с АССОЦИАЦИЕЙ ОБЩЕГО АТРИБУТА. Эта ассоциация может представлять собой оператор равенства или описание. И то и другое обозначается классом ПРЕДСТАВЛЕНИЕ СОСТОЯНИЯ.

Выход можно описать с помощью комбинации атрибутов от нескольких объектов данных.

На рис. 155 показаны две общие ассоциации: РАВНО и ХАРАКТЕРИЗАЦИЯ, где роли объектов данных являются переменными. Например, шаблоны, созданные системой САПР в процессе разработки продукта, фактически представляют собой выход, поскольку являются результатом отношения тождества. В производственных процессах, предназначенных для изготовления продукта, который описывает-

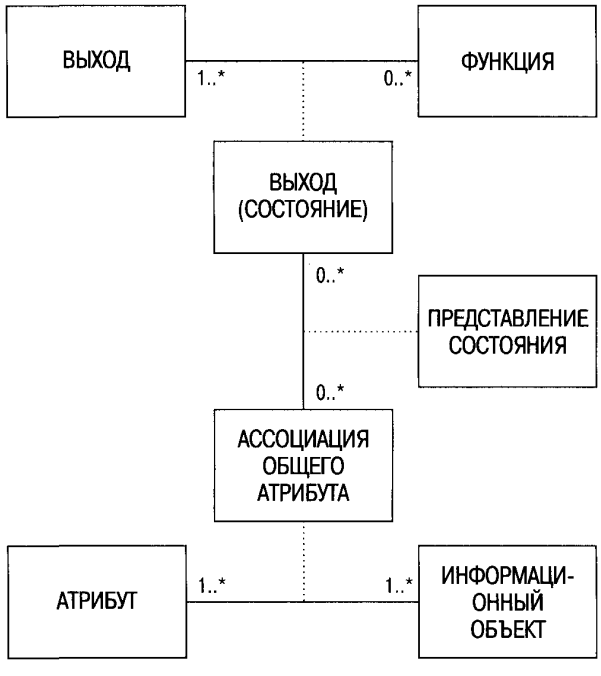


Рис. 154. Метамодель взаимодействия данных и выхода

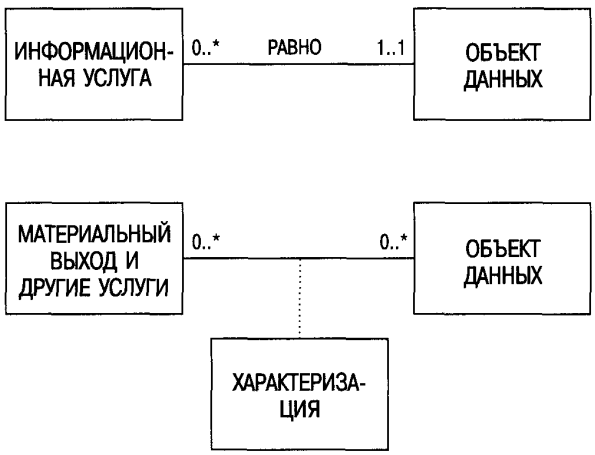


Рис. 155. Общая связь между данными и выходом

ся таким шаблоном, шаблон является описанием выхода продукта.

A.3.6.2. Конфигурирование

Поскольку описания выходов и данных иногда частично совпадают, мы лишь бегло перечислим различные возможности конфигурирования.

Для целей анализа и проверки выходов представления данных можно непосредственно привязать к соответствующему выходу, а затем передать их дальше — системам управления.

Обратная связь потока выходов с потоком работ (workflow) конфигурируется при помощи описаний данных.

Описав выход, который должен создаваться на предприятии, в рамках бизнес-приложений можно определить необходимые данные (например, прейскуранты материалов, результаты гарантии качества и т.д.), связав их с выходом.

A.3.7. Объединение всех представлений ARIS в полную модель

В ходе систематизированного описания двусторонних отношений в рамках концепции ARIS мы обсудили все ключевые вопросы моделирования, конфигурирования и реализации бизнес-процессов. В заключение мы объединим все представления ARIS в полную модель. Поскольку здесь хотелось бы заострить внимание на принципах объединения, мы не станем подробно вдаваться в вопросы реализации.

A.3.7.1. Моделирование определения требований

В качестве отправной точки для рассмотрения полных моделей может выступать, по сути, любое представление ARIS (на уровне функций, организации, данных, выхода или управления), вокруг которого группируются элементы других представлений. Здесь мы взяли за фокусные точки представление, ориентированное на процессы, и представление, ориентированное на объекты. Оба типа представления мы подробно рассмотрели выше.

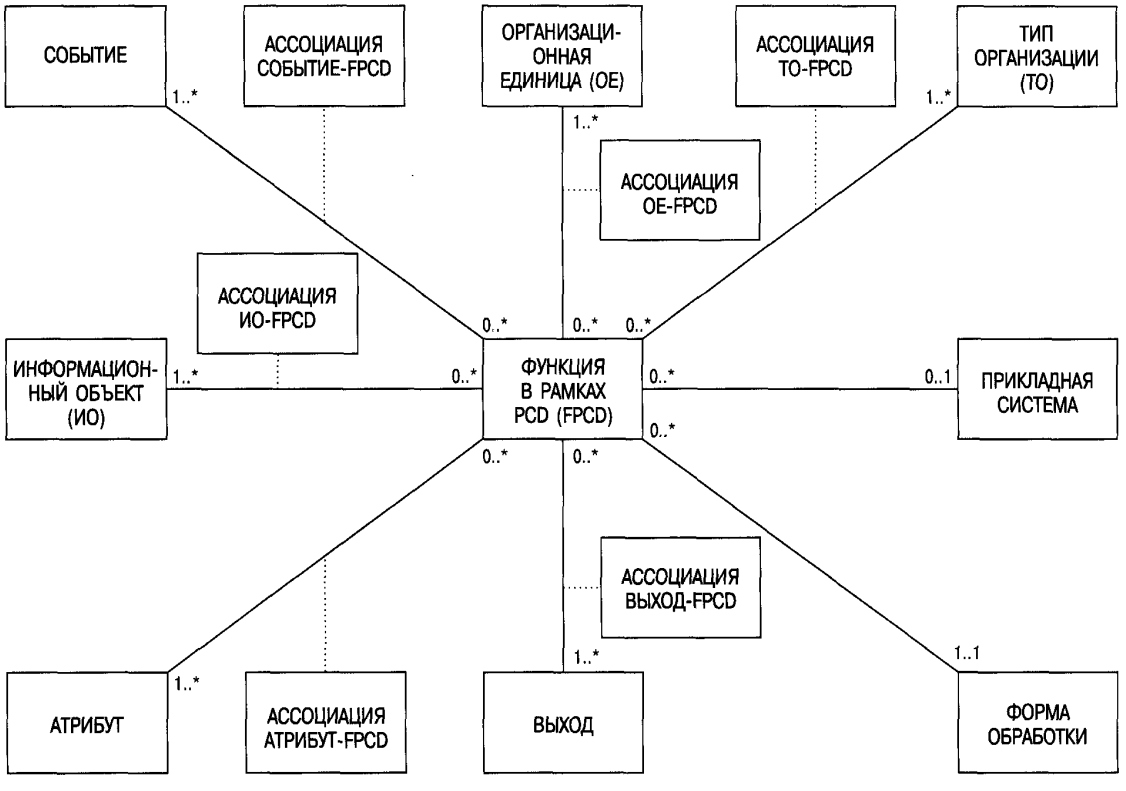


Рис. 156. Метамодель диаграммы цепочки процессов

A.3.7.1.1. Модели процессов

Диаграмма процесса (ДП) для стратегического планирования, приведенная на рис. 12а и 12б, отображает бизнес-процессы в виде таблицы. Каждая строка диаграммы показывает, каким образом данная функция в рамках процесса активизируется событием, какие объекты данных какими атрибутами обрабатываются, какое организационное подразделение в этом участвует и какой выход, соответственно, реализуется или создается. Кроме того, можно указать, какое бизнес-приложение используется для поддержки функций.

На рис. 156 показана предварительная метамодель этого контекста с функцией в качестве отправной точки.

Для описания интегрированных бизнес-процессов можно вместо таблиц использовать свободное моделирование, включив в каждое представление ARIS диаграмму СДП (событийная диаграмма процесса). Это соответствует общему описанию модели бизнес-процессов на базе ARIS (Scheer. ARIS — Business Process Frameworks. 1998, с. 31; русское издание с. 24), при этом метаструктура же соответствует диаграмме цепочки процессов (поскольку ситуация та же, за исключением способа ее представления). Поэтому одно описание можно вывести из другого.

A.3.7.1.2. Бизнес-объекты

Бизнес-объекты также согласуются с этим комплексным представлением. Изначально мы описывали их как сложные объекты данных или как связи между данными и функциями, однако бизнес-объекты содержат также поток выходов, организационный поток и поток управления.

Таким образом, бизнес-объекты описываются путем привязывания этих элементов (см. рис. 157). В соответствующий набор методов и функций входят методы, инициируемые извне посредством сообщений. В приложении SAP R/3 эти методы известны как BAPI (интерфейсы программирования бизнес-приложений). На рис. 158 изображена метамодель для бизнес-объектов.

Для того чтобы охватить внутреннее управление процессами, присвоенные бизнес-объекту данные, методы и экраны связываются с моделью процесса. Выход, требующийся или создаваемый для бизнес-объектов, представлен в модели выхода.

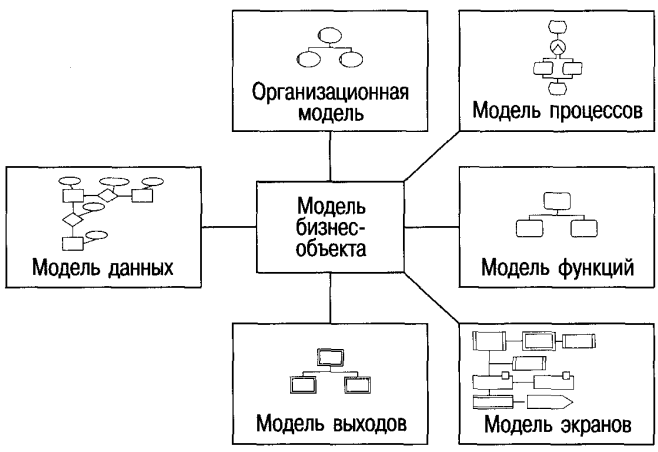


Рис. 157. Модель бизнес-объекта

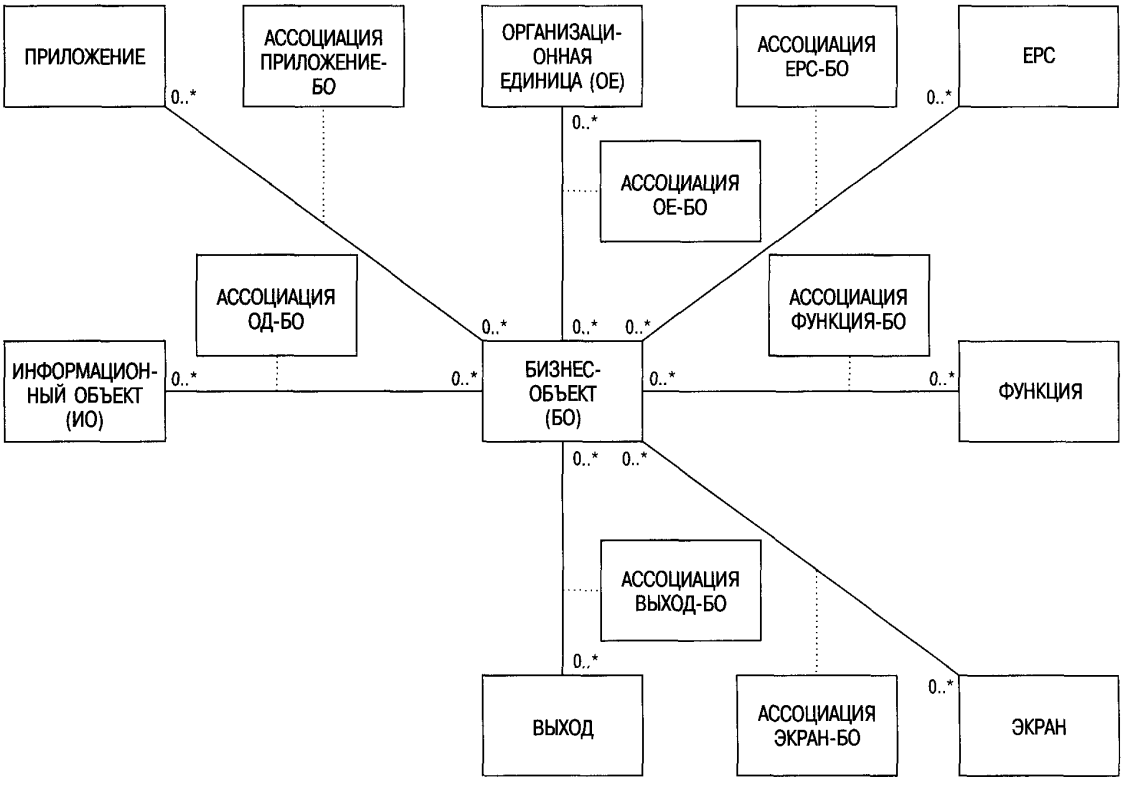


Рис. 158. Метамодель бизнес-объекта

Приложения состоят из множества бизнес-объектов, управляемых бизнес-процессом, описанным специально для данного приложения (см. рис. 159).

Иногда для выполнения одной бизнес-функции необходимо несколько бизнес-объектов или, наоборот, для нескольких бизнес-функций требуется один и тот же бизнес-объект. Соответствующий поток сообщений обозначен пунктирной линией, так как бизнес-объекты могут активизировать друг друга, не будучи привязаны к бизнес-функции.

A.3.7.2. Конфигурирование

Включив все представления ARIS, можно скомпилировать сконфигурированные приложения и уже скомпонованные модульные приложения.

A.3.7.2.1. Конфигурирование на базе моделей бизнес-процессов

На рис. 160 показаны конфигурации, возможные в рамках модели бизнес-процессов на базе ARIS. Инфраструктура ARIS Framework поддерживает возможности конфигурирования в реальных приложениях *(IDS. ARIS–Framework. 1997)*.

Часть (а) представляет фрагмент существующей цепочки процессов для приложения, связанного с закупками. При активизации определенным событием возможна реализация обоих вариантов

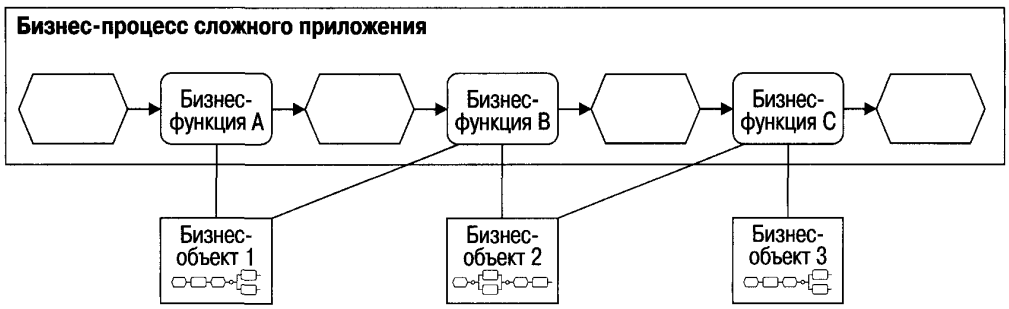


Рис. 159. Бизнес-объекты, встроенные в приложение

выполнения функции «обновить данные о закупках/продажах». ARIS Framework внедряет эту модель процесса в прототип действующей системы.

Часть (б) показывает, как пользователи могут манипулировать системой, изменяя ее конфигурацию путем обновления модели. При добавлении функции «проверить стоимость товара», которая выбирается из дерева функций, эта функция дополняется соответствующими функциональными модулями системы.

Используя атрибут «стоимость товара», организационное подразделение по приемке товара проверяет эти данные, после чего полученная информация интерпретируется системой workflow.

Если стоимость товара превышает заданный критерий, функция «обновить данные о закупках/продажах» выполняется руководителем подразделения.

Эта функция объединяет в себе две другие функции: «обновление данных о закупках» и «обновление данных о продажах». Эта новая ветвь включается в модель и интерпретируется системой workflow и инфраструктурой ARIS Framework. К функции «обновить данные о закупках/продажах» привязывается новый модуль от дерева модулей.

Компиляция функций обусловливает и выделение нового экрана. ARIS Framework и система workflow ARIS реализуют эти обновления модели в прототипах действующих систем.

Приведенный пример иллюстрирует такие возможности конфигурирования, как:

- модификация процессов,
- добавление функций,
- интеграция функций,
- обновление структуры организационной отчетности,
- обновление экранов.

В нем наглядно представлены также функциональные возможности для адаптации систем, открывающие широкие перспективы для непрерывного совершенствования процессов.

A.3.7.2.2. Конфигурирование бизнес-объектов

Если в качестве отправной точки конфигурирования выбраны бизнес-объекты, мы можем обратиться непосредственно к рис. 157.

На рис. 161 показано, как изменять элементы бизнес-объекта для настройки стандартного бизнес-объекта в соответствии с требованиями конкретного пользователя.

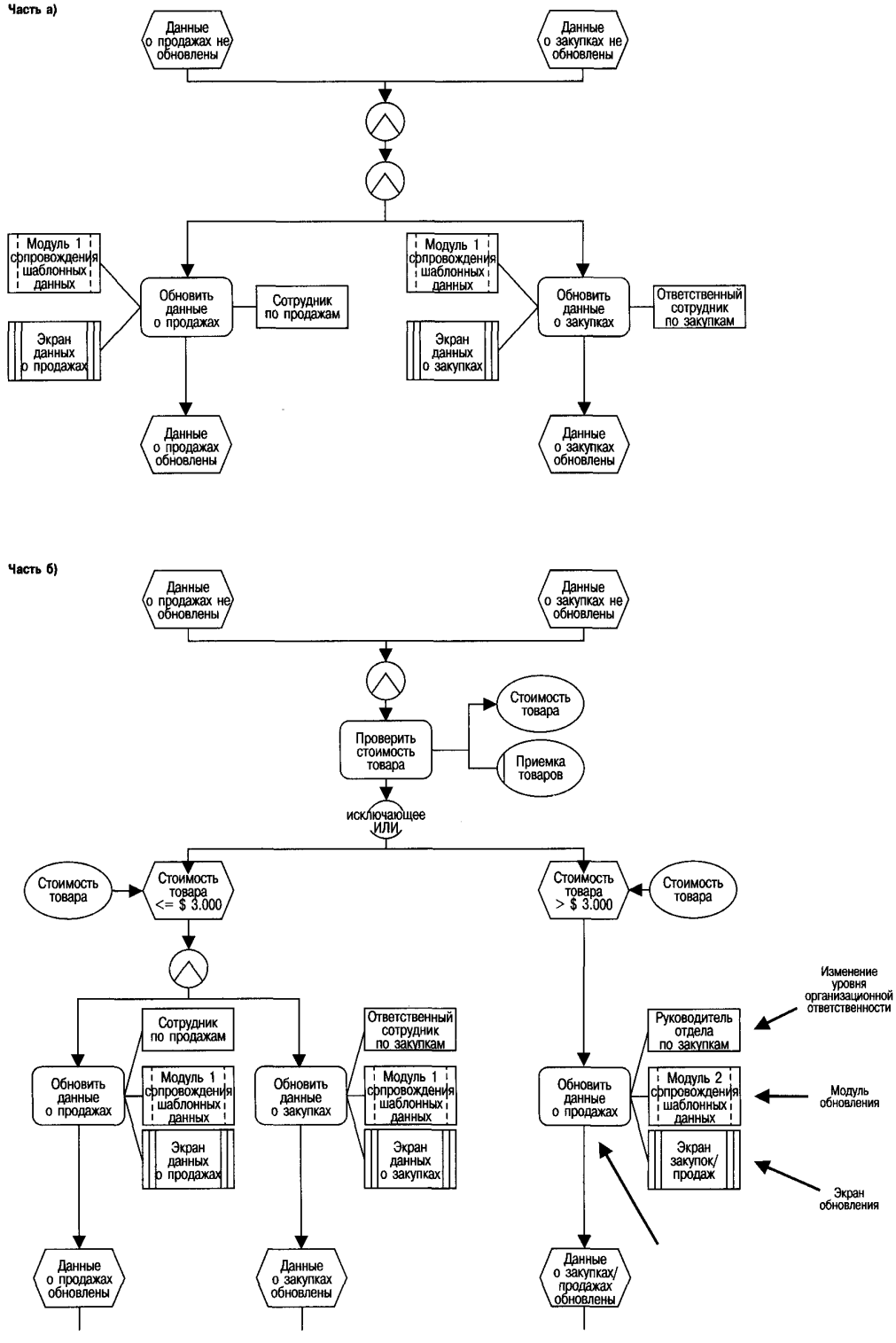


Рис. 160. Конфигурирование бизнес-процессов

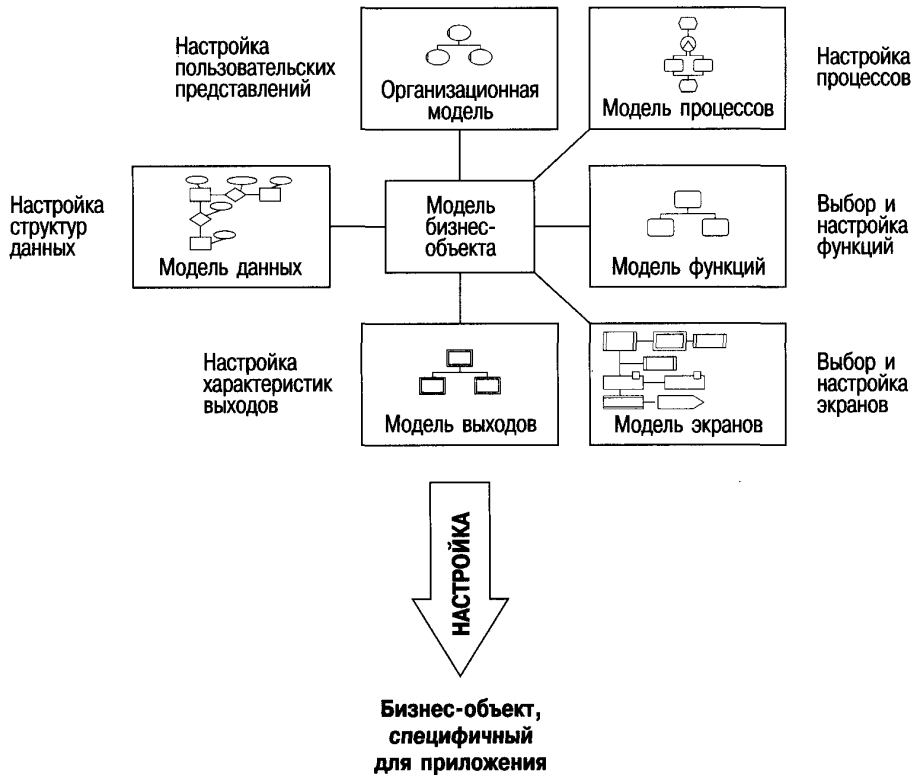


Рис. 161. Возможности индивидуальной настройки бизнес-объекта

A.3.7.3. Спецификация проекта

Если в описании на уровне определения требований модель процессов и модель объектов были «на равных» (модели процессов даже отдавалось предпочтение), то в последующих описаниях будут доминировать представления объектов. Это облегчает такие задачи реализации, как многократная применимость программ и повышение сопровождаемости системы.

Для настройки системы в соответствии со спецификацией проекта бизнес-элементы (функции, организационные единицы, объекты данных, выходы и т.д.) переносятся на уровень элементного модуля, узла, отношения и выходного объекта.

Кроме того, к бизнес-объекту привязываются межплатформные интерфейсы, являющиеся «посредниками» между прикладными программами и аппаратными средствами.

Особенно важны коммуникационные интерфейсы, предоставляемые определенными бизнес-объектами для доступа к другим бизнес-объектам, например, CORBA (общая архитектура брокера запросов к объектам), COM/DCOM (компонентная объектная модель/распределенная компонентная объектная модель) и вызов удаленных методов (RMI) для Java (см. рис. 162).

Затем различные элементы можно скомпоновать в бизнес-объект в соответствии со спецификацией проекта.

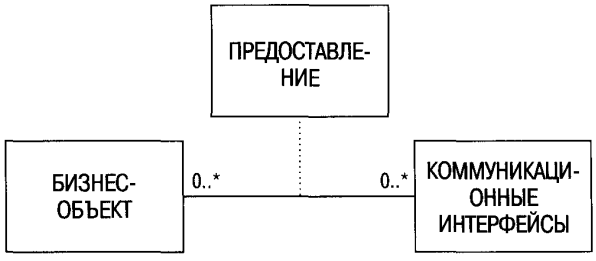


Рис. 162. Интерфейсы для бизнес-объектов

Б. Процедурные модели и приложения ARIS

Далее излагаются процедуры внедрения моделей ARIS для отдельных приложений в реальных условиях. Мы рассмотрим следующие темы:

- реализация бизнес-приложений (SAP R/3);
- реализация систем workflow;
- использование инфраструктур для разработки приложений;
- моделирование средствами UML.

Авторы, цитируемые в данном разделе, являются компетентными специалистами и обладают большим опытом в указанных областях.

Дополнительные сведения об успешной реализации концепции ARIS в таких сферах, как:

- реинжиниринг бизнес-процессов (BPR);
- сертификация качества в соответствии со стандартом ISO 9000;
- управление знаниями,

можно найти в книге *Scheer. ARIS – Business Process Frameworks. 1998.* (русское издание: А.-В. Шеер. Бизнес-процессы. Основные понятия. Теория. Методы. М., Весть-МетаТехнология, 1999).

Б.1. Реализация стандартного программного обеспечения с помощью моделей ARIS

Д-р Петер Маттхайс (Peter Mattheis), д-р Вольфрам Йост (Wolfram Jost); IDS Prof. Scheer GmbH, Саарбрюккен, Германия

Б.1.1. Разрешение критических вопросов при управлении стандартным проектом

Даже сегодня при реализации бизнес-приложений на предприятиях могут возникать затруднения, обусловленные следующими проблемами:

- чрезмерная длительность и стоимость реализации,
- ненадежные и непредсказуемые сроки и расходы,
- рискованность передачи данных,
- невозможность гарантировать качество реализации,
- неиспользование стандартов и рычагов управления,
- недостаточное использование опыта и знаний.

Для решения этих проблем необходима структурированная, методичная процедура. Ключ к успеху лежит в эффективном использовании имеющихся знаний в области соответствующих процедур, системных параметров и процессов, специфичных для вертикальных рынков.

Стандартные программы часто реализуются в виде специализированных решений под конкретного заказчика. При этом чрезмерное значение придается «уникальности» заказчика. В силу того, что результаты, к сожалению, остаются непрозрачными, опыт других, уже завершенных проектов оказывается невостребованным.

Между тем, трудности реализации бизнес-приложений можно успешно преодолеть, если придерживаться следующих принципов:

- фокусирование на вертикальных рынках с привлечением опыта и знаний, приобретенных в ходе реализации других, уже завершенных проектов,

- ориентация на бизнес-процессы с эффективным использованием имеющихся знаний о способах их описания в стандартном программном обеспечении,
- внедрение приложений, предварительно сконфигурированных для соответствующего вертикального рынка,
- использование в моделях-прототипах и контрольных списках инструментов и имеющегося опыта в отношении процедур, системных параметров и процессов, специфичных для отрасли.

Здесь мы опишем реализацию стандартного программного обеспечения и внедрение различных моделей ARIS, иллюстрируя преимущества использования в этих проектах пакета ARIS Toolset.

Б.1.2. ARIS Quickstep for R/3

Рассмотрим процедуру реализации бизнес-приложений на примере SAP R/3. Упомянутые проекты реализации опираются на следующую процедуру, отраженную в процедурной модели ARIS Quickstep for R/3 (см. 163). Эта модель состоит из трех следующих компонентов:

- процедурная модель,
- приложения R/3, предварительно сконфигурированные для соответствующего вертикального рынка,
- описание предварительно сконфигурированных приложений R/3, документально представленное в виде диаграмм EPC.

Методика ARIS Quickstep for R/3 разработана фирмой IDS Prof. Scheer и прошла испытание во многих проектах реализации R/3.

Модель, подразделяющаяся на четыре фазы, показана на рис. 164. Она позволяет сократить время, снизить стоимость реализации, улучшить защиту и повысить качество проекта.

Описание базируется на моделях ARIS. Фазы представлены в виде цепочки добавленного качества, при этом каждая фаза детально описывается диаграммой EPC. Диаграммы входа и выхода описывают входные и выходные данные как «материалы», поставляемые детализированным функциям. Этот метод облегчает планирование проекта и дает более ясную картину, обеспечивая, таким образом, высокую степень прозрачности процесса выполнения. Эффективное прослеживание хода проекта и заблаговременное выявление возможных задержек позволяют вносить коррективы, чтобы уложиться в срок.

Quickstep for R/3 соответствует требованиям процедурной модели SAP ASAP (Accelerated SAP), поскольку она преследует те же цели, что и ASAP, и предлагает аналогичный способ выполнения проекта. Кроме того, поскольку различные приложения R/3 настроены применительно к конкретным вертикальным рынкам, Quickstep for R/3 уже приспособлена для определенных отраслей. Это служит дополнительным катализатором.

Б.1.3. Quickstep for R/3: описание фаз реализации SAP

Предложение / Заказ

Обычно детальный анализ ситуации «как есть» опускается, хотя по желанию заказчика существующие бизнес-процессы могут быть проанализированы. Если заказчик хочет выяснить количественные характеристики сценария «как есть» (время производственного цикла, стоимость процессов, соблюдение сроков, установленных графиком, и т.д.), без анализа бизнес-процессов не обойтись. Для этого используются диаграммы EPC, которые обеспечивают наглядность и прозрачность даже при описании больших процессов.

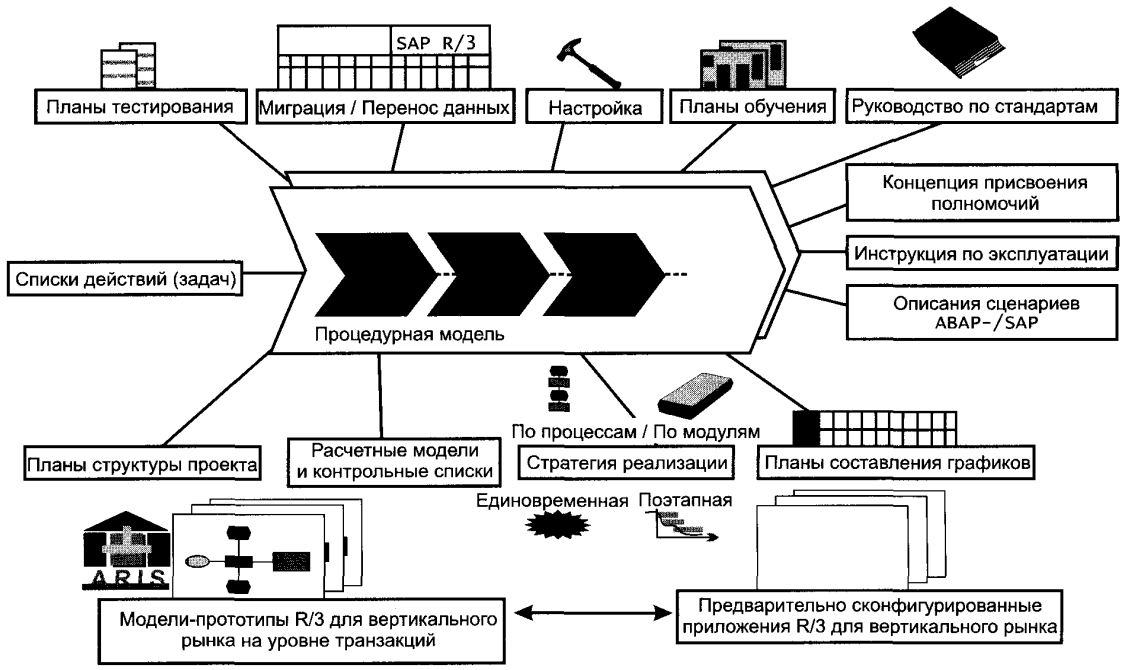


Рис. 163. Quickstep for R/3: компоненты

Рис. 164. Quickstep for R/3: процедура

Описание ситуации «как есть» предполагает предварительное создание общего эскиза процессов с крайне незначительной степенью детализации (один-два уровня).

На семинарах по выработке требований с ориентацией на процессы выясняются требования заказчика к реализации R/3. Это делается при помощи контрольных списков. В результате составляется официальное предложение для заказчика. В зависимости от стратегии реализации Quickstep for R/3 предоставляет также ориентированные на процессы расчетные модели, план структуры проекта, списки действий (задач) и календарные графики. Эти способы контроля значительно повышают эффективность и качество планирования проекта.

Описания процессов необходимы, чтобы оценить объем работ по реализации и получить четкое представление о том, какая поддержка или какие дополнительные работы по настройке требуются для внедрения нового прикладного пакета. Если такие модели отсутствуют или в связи с приложением требуется соответствующая целевая концепция, то рекомендуется следующая процедура.

Сформулируйте вместе с руководством общую концепцию бизнес-процессов. Решите, например, в какой мере следует на практике реализовать управление цепочками поставок и какие преимущества привнесут в ваш проект современные технологии (скажем, связи с корпоративными бизнес-партнерами через экстрасети).

Опираясь на эти концепции, спроектируйте продуктовую модель, описывающую услуги и продукты, предлагаемые вашим предприятием. Эта модель представляет основные характеристики потребительских продуктов и услуг, имеющие ключевое значение для решений о закупках исходных материалов.

Рыночная модель отображает маркетинговые мероприятия, каналы сбыта и ожидания рынка в отношении услуг и продуктов, предлагаемых компанией. Для правильного позиционирования выходной продукции (услуг) компании на рынке обе модели необходимо синхронизировать.

Продуктовая и рыночная модели строятся средствами ARIS Toolset, которые позволяют наглядно представить отношения между отдельными компонентами, сделать их прозрачными, а также предоставляют пользователям возможность дополнительно вводить пояснительную информацию.

Цепочки добавленного качества, требующиеся для проектирования и продажи продуктов, описываются в соответствии с этой продуктовой стратегией. Вспомогательные функции, такие как финансовый учет и контроль или управление людскими ресурсами, дифференцируются по фактическим процессам добавления качества. Чтобы обеспечить достаточную прозрачность, цепочки добавленного качества должны содержать от 30 до 50 элементов, представляющих стратегическое направление, выбранное руководством для реализации программного обеспечения, ориентированного на процессы.

Стратегический компонент проекта может быть необязательным. Однако практика показывает, что именно этот компонент позволяет руководству сориентировать проект в верном направлении. Будущие выгоды в огромной мере зависят как раз от стратегического компонента.

Концептуальный проект прототипа

В начале этого этапа на основании календарных графиков и списков действий (задач), включенных в Quickstep for R/3, определяются временные рамки проекта и план действий по его реализации. Следует проинформировать сотрудников о ходе и целях проекта и установить необходимое оборудование и программное обеспечение.

Еще одной из мер по подготовке к реализации программного обеспечения является определение организационной моде-

ли R/3. На уровне предприятия необходимо описать клиентов, коды компании, корпоративные отделения и правила учета стоимости. Следует также определить организационные структуры для материально-технического обеспечения (логистики), продаж и закупок.

Организационная структура хранится в ARIS Toolset в виде иерархической диаграммы наряду с текстовой документацией, где приводится обоснование для описания различных структур.

Масштабы прототипа определяются путем описания актуальных бизнес-процессов, которые должны быть отображены в R/3. Необходимые модули R/3 определяются исходя из бизнес-процессов.

Взяв в качестве отправной точки цепочку добавленного качества, проектируют варианты процессов, отражающие различные опции в R/3. Это делается для следующих бизнес-процессов:

- инжиниринг/мастер-данные,
- обработка предложений/заказов,
- закупки,
- производство,
- послепродажное обслуживание.

Скажем, для обработки предложений/заказов существует ряд вариантов, например, обработка стандартных продуктов, обработка сложных продуктов заданной конфигурации с помощью специальных конфигураторов или обработка сложных продуктов с помощью проектных инструментов. Эти варианты моделируются в ARIS Toolset как процессы. Для того чтобы сократить объем работ по проектированию и эффективно использовать возрастающие функциональные возможности R/3, проектирование вариантов процессов опирается на модель-прототип R/3 и модели вертикального рынка IDS. Модели вертикального рынка документально описывают процессы для приложений R/3, предварительно сконфигурированных IDS.

Настройка этих моделей должна выполняться децентрализованно будущими пользователями, работающими в соответствующих отделах. Для этого требуются инструменты, удобные для пользователя, например, ARIS Easy Design.

Для поэтапного ознакомления рабочей группы с приложением R/3 коды транзакций можно хранить в каждой функции модели процессов. Это дает пользователям возможность переходить к конкретным экранам непосредственно из модели процессов, позволяя им лучше освоиться с новым приложением и, таким образом, повышая их психологическую готовность к его принятию.

Прототипы проектируются коллективно, в тесном сотрудничестве с клиентом, на базе приложений, предварительно сконфигурированных для соответствующего вертикального рынка.

Иерархия моделей представлена на рис. 165.

Реализация

На этапе реализации производится расширение прототипов и перенос данных, описанных на предыдущем этапе. Профили полномочий, присваивающие пользователям права относительно конкретных функций, настраиваются в приложении генератором профиля.

Работоспособность (функциональность) приложения сначала тестируется в рамках процессов с использованием данных, введенных клиентом. После завершения тестирования и одобрения результатов проводится интеграционный тест. После успешного переноса данных необходим дополнительный тест с использованием производственных данных.

Одновременно с интеграционным тестированием проводятся учебные занятия для пользователей. По времени эти занятия должны быть максимально приближены к дате ввода в действие.

Рис. 165. Иерархия моделей

Модели процессов детализируются до уровня транзакций, чтобы обеспечить основу для пользовательской документации. На базе описания этих, обычно линейных, процессов создаются конкретные инструкции для пользователей, позволяющие им выполнять свои повседневные задачи.

Непрерывное совершенствование

За этапом фактической реализации приложения следует этап оптимизации, на котором заново проводится проверка приложений и процессов, а также выполняется дальнейшая оптимизация моделей, разработанных на предыдущих стадиях. Если на предыдущих этапах в моделях процессов были сохранены стоимостные или временные показатели, то совершенствование процессов можно выражать количественными характеристиками.

Б.1.4. Резюме

Отличительные особенности этого метода сжато можно сформулировать следующим образом.

- **Ориентация на стратегию.** Характеристики услуг и продуктов, предлагаемых предприятием, заложены в продуктовую модель. Рыночные требования находят отражение в рыночной модели, а продуктовые концепции и их реализации, соответственно, хранятся в цепочках добавленного качества. Реализация программного обеспечения основана на цепочках добавленного качества, которые связывают его с корпоративной стратегией.
- **Эффективность.** Используются знания в области вертикального рынка, хранящиеся в приложениях R/3, предварительно сконфигурированных для конкретной отрасли. Процессы, специфичные для вертикального рынка, документально описываются в ARIS Toolset.

- **Ориентация на методы и модели.** Внедрение ARIS Toolset обеспечивает наличие необходимых методов и моделей.
- **Стандартизированность, прозрачность и понятность** обеспечиваются благодаря процедурной модели Quickstep for R/3.
- **Ориентация на процессы.** Обучение пользователей, семинары по выработке требований и организация проекта ориентированы на процессы. Иерархия процессов полностью согласована — от цепочек добавленного качества до уровня кодов транзакций.
- **Ориентация на вертикальный рынок.** Используются специальные знания, относящиеся к конкретной отрасли.

Б.2. Реализация систем workflow с помощью моделей ARIS

Андреас Кронц (Andreas Kronz), дипл. по информатике; IDS Prof. Scheer GmbH, Саарбрюккен, Германия

Б.2.1. Факторы успеха при реализации систем workflow

Фирма IDS Prof. Scheer GmbH на основе концепции ARIS разработала свой подход к реализации систем workflow, который включает методы моделирования ARIS, соответствует архитектуре АБИ и интегрирует ARIS Toolset. Этот подход под названием ARIS Workflow используется в первую очередь для разработки прототипов в целях быстрого создания исполняемых, ориентированных на процессы приложений на базе моделей бизнес-процессов. ARIS Workflow взаимодействует с серийно выпускаемыми системами workflow, такими как WorkFlow (CSE), FlowMark (IBM), WorkParty (SNI), Staffware (Staffware) и Visual WorkFlo (FileNET). В будущем станет возможна бесшовная интеграция ARIS Workflow с концепцией ARIS и ARIS Workflow Prototyper.

Важнейшим условием успешной реализации системы управления потоками работ (workflow) является наличие структурированных и методичных процедур, позволяющих упростить процесс, сократить количество источников ошибок, а также повысить надежность планирования мощностей и затрат. Благодаря систематизированной и согласованной процедуре внедрения моделей ARIS процедурные модели ARIS для систем workflow дают возможность эффективно реализовать бизнес-процессы в рамках корпоративных приложений workflow. Модели ARIS являются необходимыми предпосылками непрерывного совершенствования процессов между уровнями I и II. Они применяются для описания вспомогательных процессов и процедурных моделей — от оптимизации бизнес-процесса до моделирования агрегированных данных в условиях реальной эксплуатации.

Преимущество систем workflow заключается в использовании хорошо структурированных процессов, связанных с операциями. К сожалению, существующие на сегодняшний день средства моделирования не дают возможности достаточно успешно реализовать слабо структурированные бизнес-процессы. Необходимы методы, которые позволили бы описывать их, поддерживая при этом принцип непрерывного совершенствования.

Б.2.2. Процедурная модель ARIS для реализации workflow

Процедурная модель, изображенная на рис. 166, разбита на восемь фаз и описывает процесс реализации системы управления потоками работ — от стратегического планирования до этапа производства. Для ускорения процесса реализации ряд задач можно выполнять одновременно. Кроме того, в связи с мерами по обеспечению качества (тестовые прогоны или контроль качества) может возникнуть необходимость в итерации отдельных фаз.

Процессы реализации должны сопровождаться комплексным обучением пользователей. Различные группы пользователей следует обучать поэтапно. Темы учебных программ могут охватывать широкий диапазон — от подготовки системных администраторов системы workflow до освоения новых бизнес-приложений (если это применимо) и оптимизации бизнес-процессов. Для успеха проекта необходимо, чтобы центры ответственности (организационные под-

разделения) и отдельные сотрудники понимали структуру и преимущества бизнес-процессов. Важную роль играют единая методология и инструментальная поддержка, поскольку их можно использовать при обучении или для моделирования и мониторинга текущих процессов в системе workflow. Использование диаграмм EPC в качестве языка описания бизнес-процессов облегчает понимание процессов, особенно когда в диаграммы привязки (распределения) функций включаются принятые в моделировании обозначения, детализирующие связи с приложениями.

Первым этапом этого процесса является инжиниринг бизнес-процессов, как часть стратегического корпоративного планирования.

На подготовительной фазе проекта проводится семинар, где вниманию будущих пользователей предлагаются система workflow и прототип бизнес-процесса. Цель семинара заключается в том, чтобы разъяснить принципы управления потоком работ, обсудить вопросы, связанные с осуществимостью проекта, и определить функциональные спецификации. К другим целям относятся выработка проектного предложения, определение требований, составление предварительного плана осуществления проекта, оценка потребностей в людских ресурсах, программы обучения и связанных с этим затрат. Здесь же следует рассмотреть общие вопросы, такие как количество пользователей, технические требования, необходимое оборудование и т.д. Следует подумать и о прототипе для интеграции соответствующего приложения.

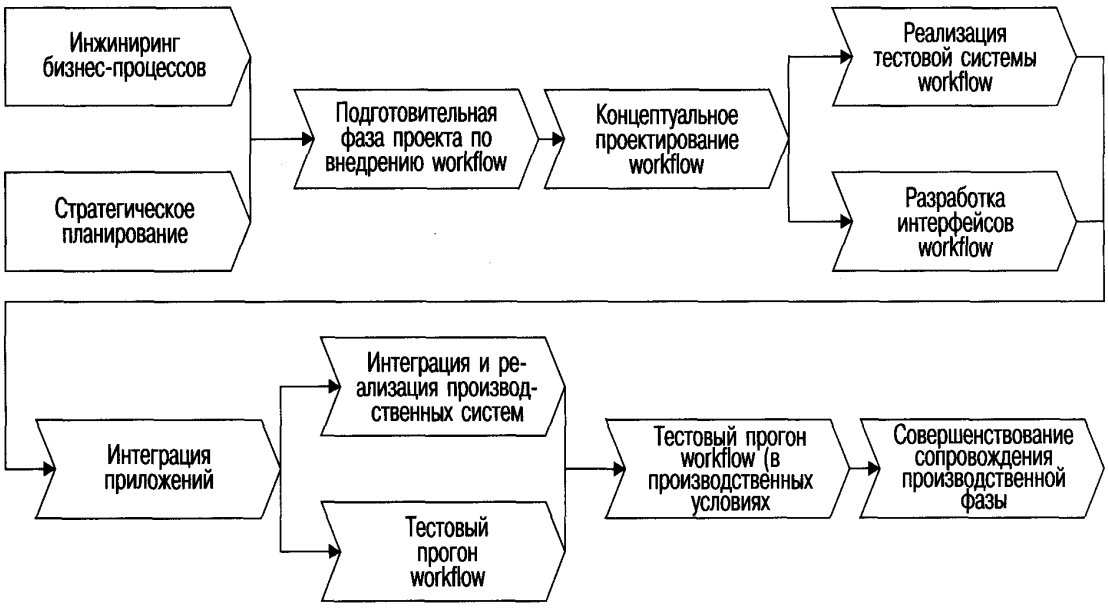


Рис. 166. Процедурная модель для реализации workflow на базе модели

Необходимо подготовить диаграмму СДП, описывающую вспомогательные процессы или хотя бы их предварительную версию.

На концептуальной стадии вспомогательные процессы расширяют, преобразуя их в исполняемые модели процессов, и дополняют данными, имеющими отношение к системе workflow (см. рис. 167). Необходимы также органиграммы, описывающие организационную структуру (см. рис. 168), описание данных, имеющих отношение к workflow (например, модели ERM), а также описания процессов в виде диаграмм СДП наряду с соответствующими диаграммами распределения функций (см. рис. 169). Отправной точкой для выполнения этих задач служат целевые модели, построенные на стадии инжиниринга бизнес-процессов.

Правдоподобие и согласованность этих моделей следует проанализировать с помощью инструментов моделирования.

Концептуальный этап дополнительно включает следующие задачи.

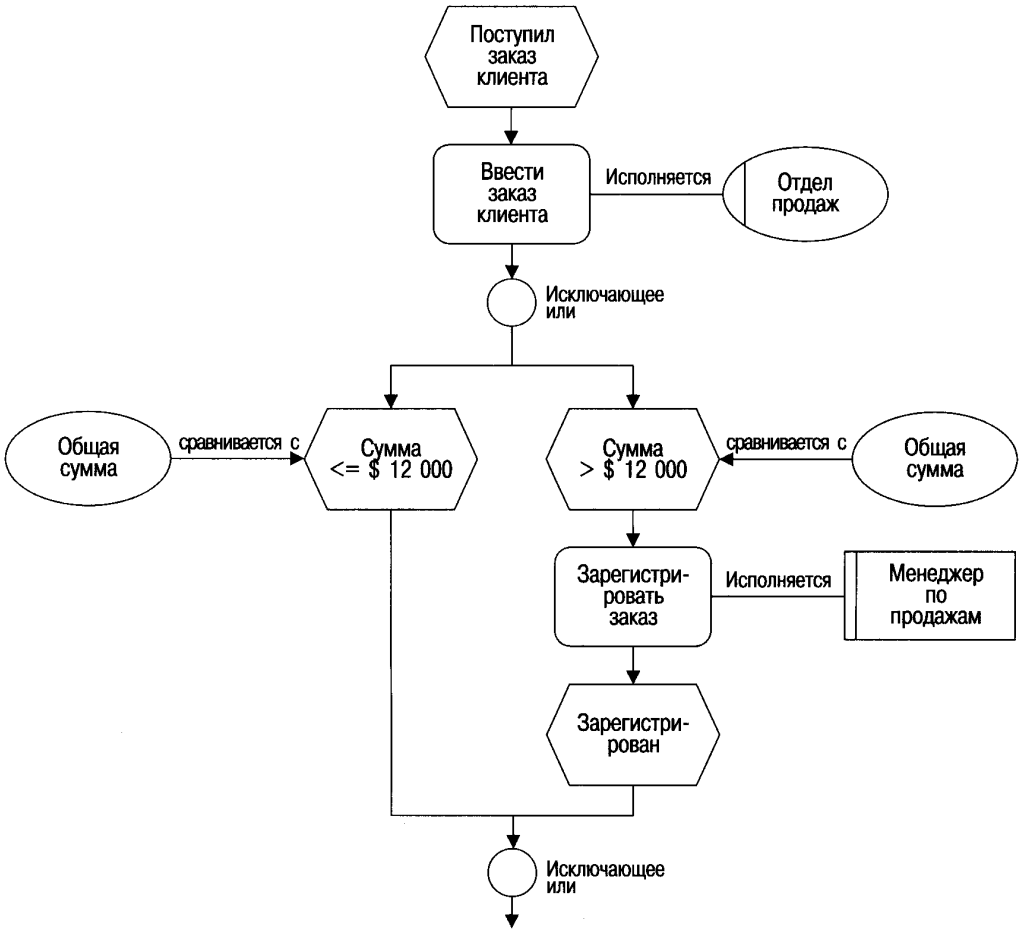


Рис. 167. Фрагмент исполняемой модели процесса.

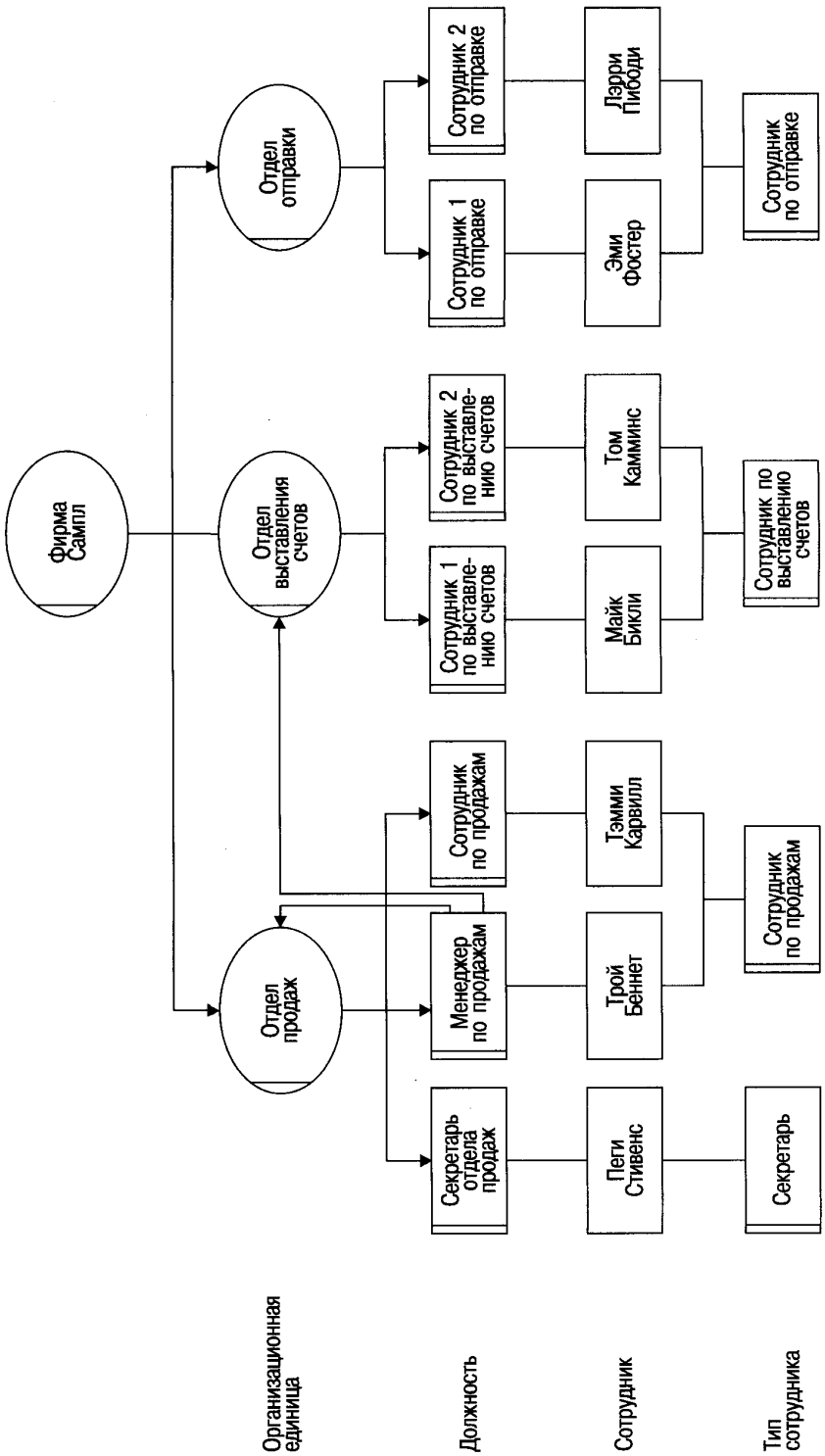


Рис. 168. Фрагмент органиграммы

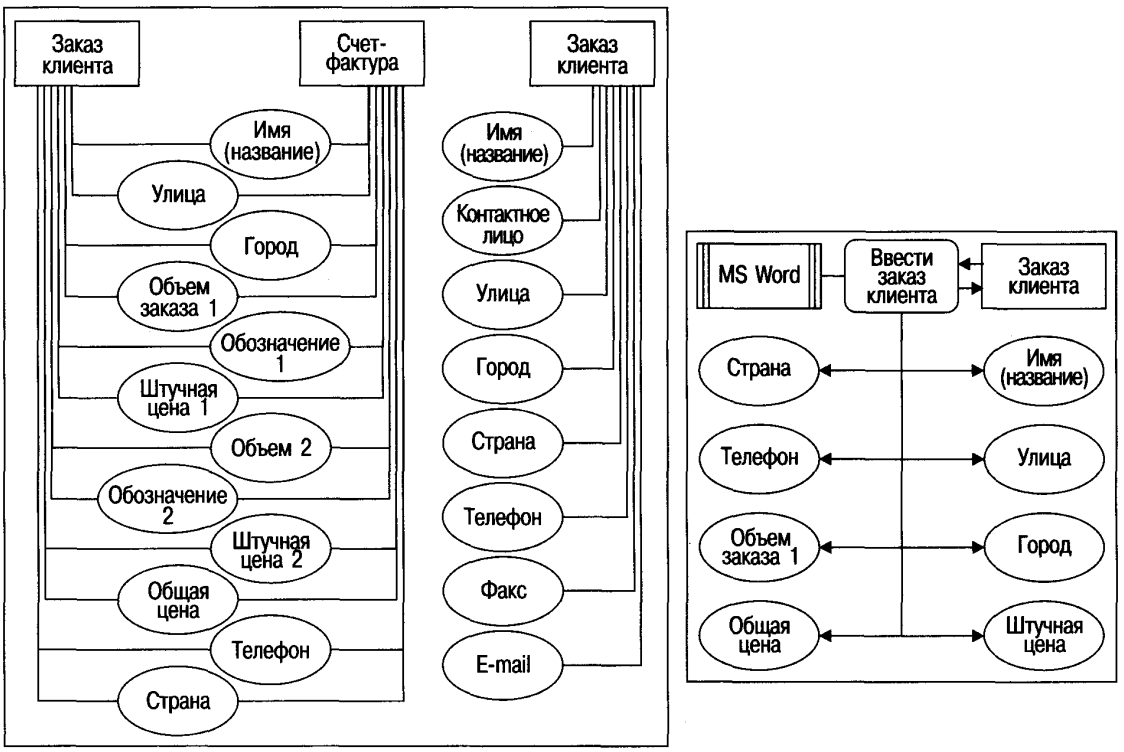


Рис. 169. Фрагмент модели данных и диаграммы распределения функций.

- Создание концепции ролей и полномочий в той мере, в какой эти привилегии могут быть присвоены в рамках системы workflow. Привилегии вводятся в органиграмму.
- Определение требований к инфраструктуре ИТ, начало мероприятий по закупке.
- Определение требований внедряемых приложений в отношении входов и выходов.
- Формирование и обучение группы сотрудников по тестированию. Несколько позже этой группе предстоит протестировать систему workflow. Необходимо также подготовить будущих администраторов системы workflow, чтобы они смогли помочь в реализации тестовой среды для workflow.

Реализация тестовой среды позволяет проверить корректность данных, организационных структур и бизнес-процедур, относящихся к соответствующим процессам, на концептуальном этапе. Будущих системных администраторов системы workflow следует привлекать к участию в процессе реализации или даже — при условии надлежащего руководства — предоставлять им возможность выполнить эту задачу самостоятельно. С учетом необходимых требований и спецификаций разрабатываются интерфейсы для взаимодействия с новыми бизнес-приложениями.

Приложения интегрируются с интерфейсами, после чего модели процессов дополняются специфичными для системы данными и проводится тестирование работоспособности (функциональности). Затем тестовая группа выполняет тестирование

совместимости системы workflow различных корпоративных приложений с требованиям и приложения. Любые проблемы, возникающие на этом этапе, могут влиять на моделирование или на интеграцию приложений, порождая итерации на всем пути вплоть до концептуальной фазы.

Фаза реализации производственной системы включает также создание концепции отработки отказов и технического сопровождения, в соответствии с которой моделируются профили конечных пользователей и их полномочий, а также планируется и проводится обучение пользователей работе с системой workflow. Последней ступенью этой фазы должно быть утверждение отделом обеспечения качества.

В ходе тестового прогона реальные сценарии тестируются с использованием реалистичных объемов данных. Для гарантии безотказной работы система workflow проверяется в условиях квазиреалистичной сетевой и серверной нагрузки. Надежность концепции защиты обеспечивается тестированием функций резервного копирования и восстановления. На этом этапе унаследованное приложение и новое приложение workflow обычно используются в производстве параллельно во избежание проблем, которые могли бы привести к итерациям в процедурной модели.

На этапе производственной эксплуатации системы workflow осуществляются управление и мониторинг бизнес-процессов. Мониторингом называется формирование запросов и моделирование текущих процессов *(Heß. Monitoring von Geschäftsprozessen. 1996)*. Очень важную роль играет согласованное представление текущих и новых бизнес-процессов в виде диаграмм СДП (см. верхнюю часть рис. 167). Можно провести дальнейшую оптимизацию бизнес-процессов путем документирования и оценки данных в условиях реальной эксплуатации, хотя в этом случае для того, чтобы получить значимую картину, требуется объединить море поступающей информации. По этой причине такие данные представляют в моделях бизнес-процессов с помощью диаграмм СДП, где агрегированные данные хранятся в виде атрибутов функций и событий (например, среднее время обработки или частота событий). На уровне II эти агрегированные модели можно проанализировать, чтобы выявить потенциальные возможности усовершенствования. Весь этот процесс известен как цикл непрерывного совершенствования процессов (CPI), включающий моделирование, управление, документирование и повторное моделирование процессов.

Для успешного внедрения системы workflow крайне важно, чтобы сотрудники четко представляли себе цикл оптимизации. Это обеспечит гибкость системы, что, в свою очередь, позволит постоянно адаптировать и совершенствовать бизнес-процессы.

Б.3. Разработка систем на базе модели с использованием инфраструктуры ARIS Framework

Саид Эмрани (Saeed Emrany), дипл. по информатике; Ричард Бок (Richard Bock), дипл. по информатике; IDS Prof. Scheer GmbH, Саарбрюккен, Германия

Было время, когда производительность и архитектура бизнес-приложений, поддерживающих бизнес-процессы, создавали серьезные трудности при оптимизации бизнес-процессов. Сегодня программное обеспечение, совместимое с инфраструктурой ARIS Framework, предоставляет гораздо больше степеней свободы.

Тесно связанная с ARIS Toolset и ARIS Workflow инфраструктура ARIS Framework идеально подходит для быстрого создания прототипов бизнес-приложений. Она базируется на современной архитектуре клиент-сервер и согласуется с концепцией АБИ. Корпоративные модели, структурированные в соответствии с методами ARIS, можно использовать для документирования бизнес-процессов, а также для проектирования и создания приложений, поддерживающих workflow. Это позволяет индивидуально настраивать приложения, опираясь на модель, что, в свою очередь, позволяет клиентам адаптировать свои программы к меняющимся условиям и директивным установкам.

Б.3.1. Общая процедурная модель

Инфраструктуру ARIS Framework можно использовать для реализации и настройки приложений, разработанных на ее основе.

Общая процедура представлена на рис. 170 в виде диаграммы СДП.

Специализированные приложения, отвечающие конкретным нуждам заказчика, можно реализовать либо путем адаптации и компоновки бизнес-объектов, ориентированных на процессы, либо путем разработки приложений «с нуля». Если в качестве отправной точки выбираются бизнес-объекты, то описывающие их модели настраиваются с учетом специфики заказчика, а полученные в результате такой настройки специализированные модели служат в качестве входных элементов для создания специальных приложений с помощью ARIS Framework. Процедура настройки позволяет модифицировать бизнес-объекты и соответствующие процессы с их последующим исполнением интегрированной в инфраструктуру системой workflow.

Если приложение разрабатывается с нуля, то целевая корпоративная концепция сначала моделируется средствами ARIS Toolset. Реализация существующих объектных методов в рамках ARIS Framework до того, как модель будет полностью готова к внедрению в приложение, необязательна.

Далее мы рассмотрим функцию «моделирование целевой концепции» более подробно и укажем ключевые аспекты создания приложений для моделей с различным содержанием. Описание функции вновь опирается на процедурную модель.

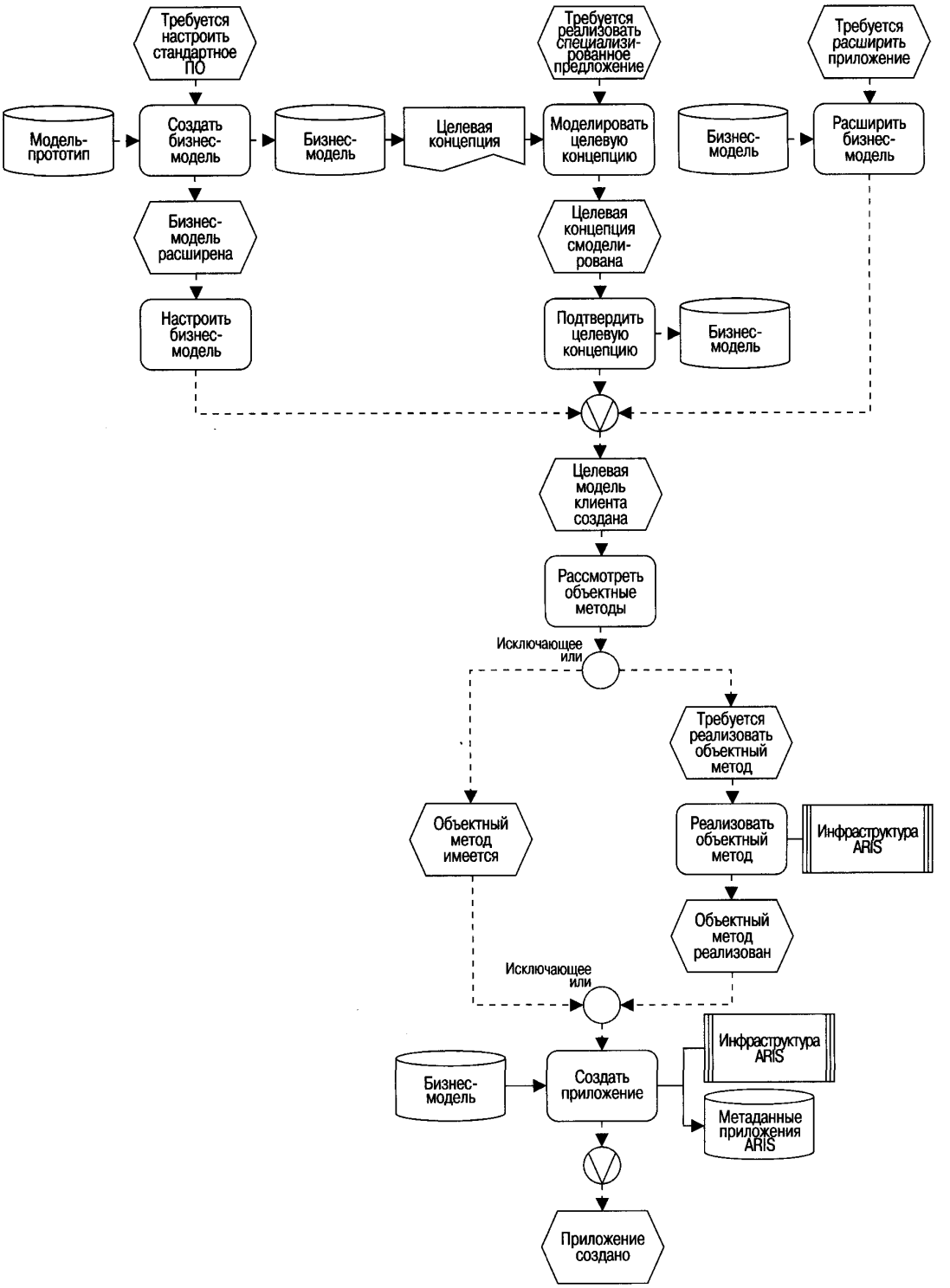


Рис. 170. Процедурная модель для разработки и настройки приложений

Б.3.2. Процедурная модель для моделирования целевых концепций

Модели ARIS Framework можно создавать в соответствии с процедурой, ориентированной на процессы или на объекты. В процедурах, ориентированных на процессы, модели опираются на бизнес-процессы, а в объектно-ориентированных процедурах основу составляют бизнес-объекты. На рис. 171 показаны необходимые этапы моделирования целевой концепции.

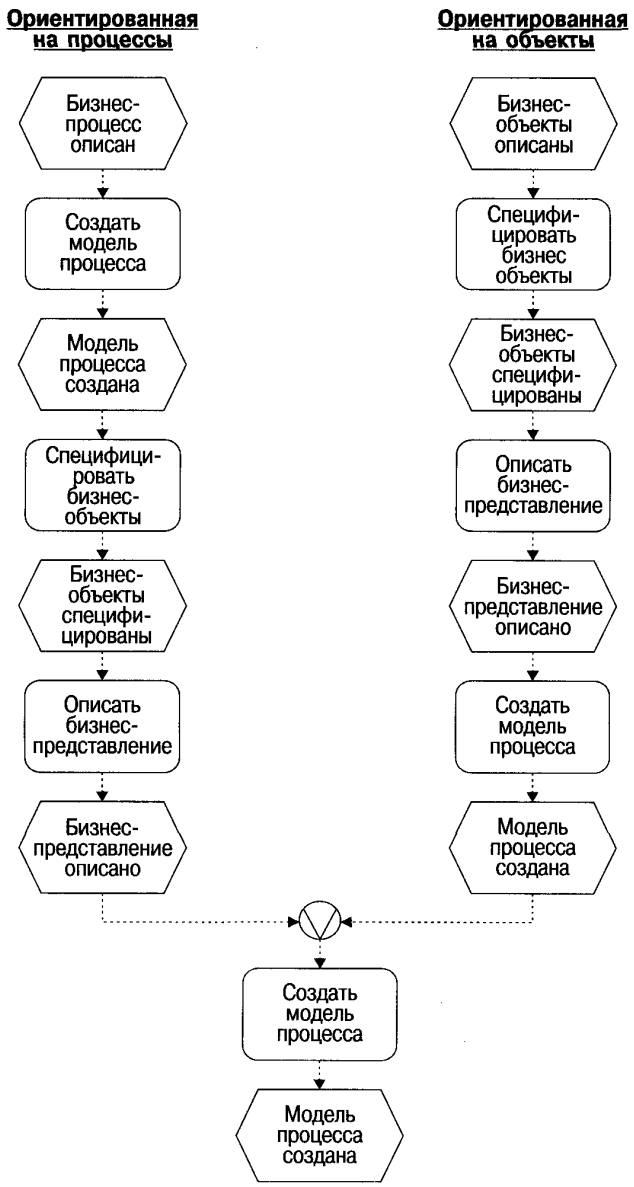


Рис. 171. Процедурная модель для создания целевых концепций

Создание моделей процессов

Помимо обеспечения прозрачности процессов и их оптимизации, составляющих общие цели моделирования, ARIS Framework позволяет специфицировать структуры приложений в соответствии с конкретными процессами. Ниже мы рассмотрим эту тему более обстоятельно. В процедурах, ориентированных на процессы, модели процессов, наряду с приложениями, специфицируют также требующиеся объекты. Эти модели процессов создаются в ARIS Toolset в виде диаграмм СДП. Возможны различные степени детализации, позволяющие специфицировать бизнес-процессы вплоть до уровня исполняемых функций.

Спецификация бизнес-объектов

Бизнес-объекты представляют элементы данных, необходимые для бизнес-приложений, а также методы, применимые к элементам данных. Элементы данных можно дифференцировать на объекты данных, их атрибуты, отношения и правила. Объекты данных и их отношения описываются моделью сущность-отношение (ERM), тогда как атрибуты описываются в ARIS Toolset в виде диаграмм присвоения атрибутов. Структуры ERM для тех или иных бизнес-объектов можно адаптировать к структуре бизнес-объектов, необходимой для конкретных процессов. Привязка методов к бизнес-объектам осуществляется с помощью соответствую-

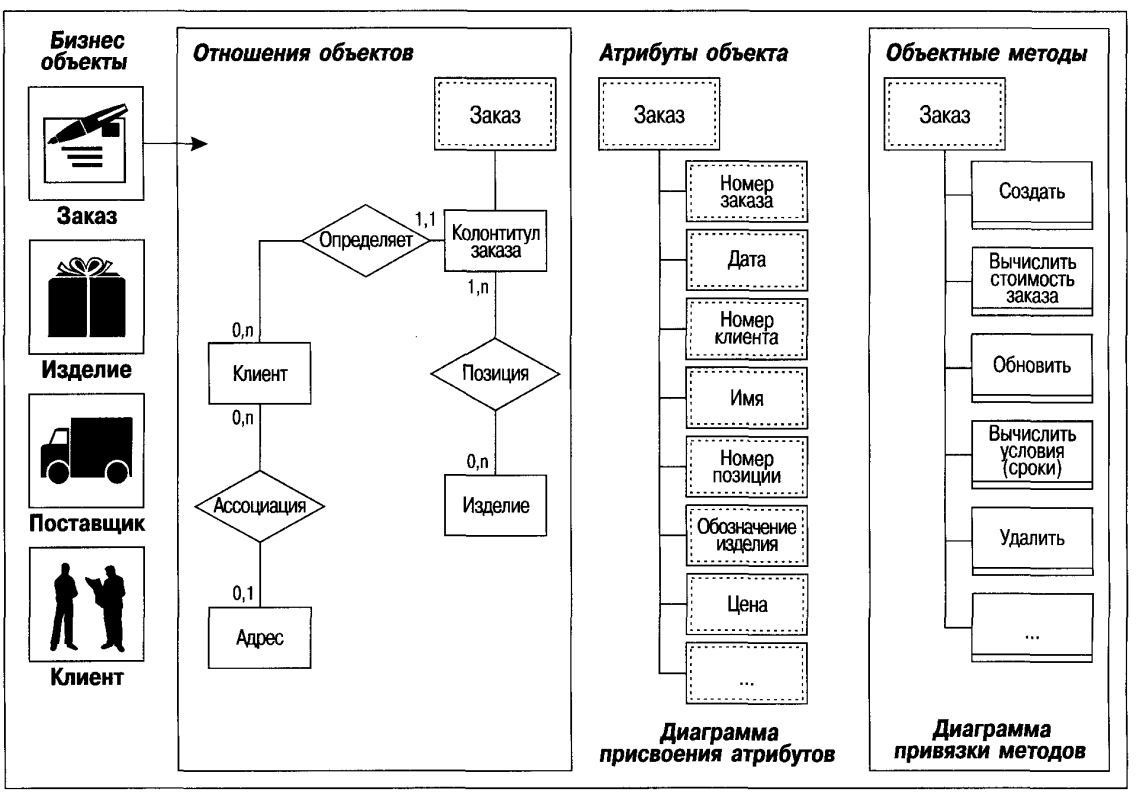


Рис. 172. Структура бизнес-объекта

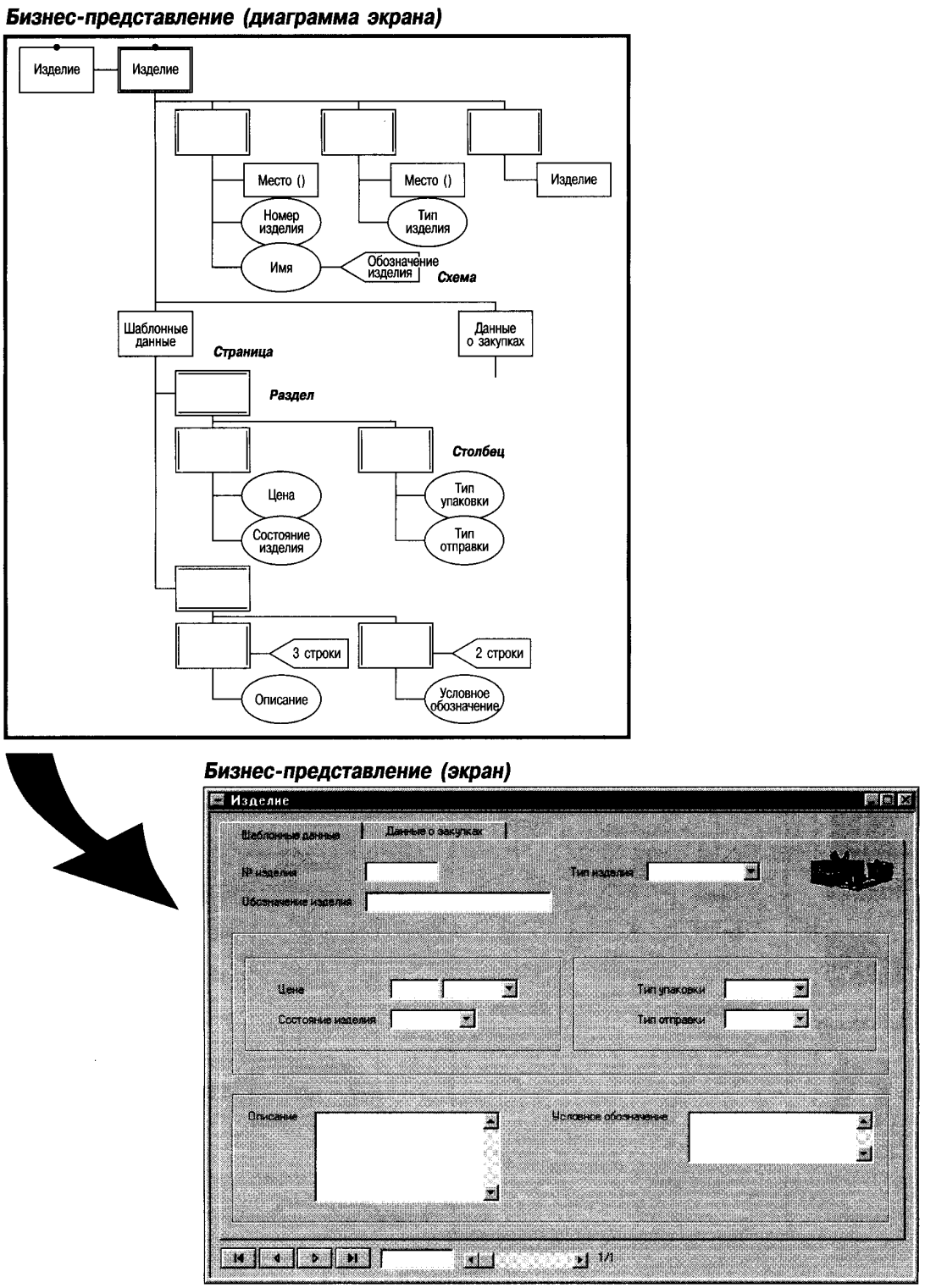


Рис. 173. Логическая схема и результирующий экран

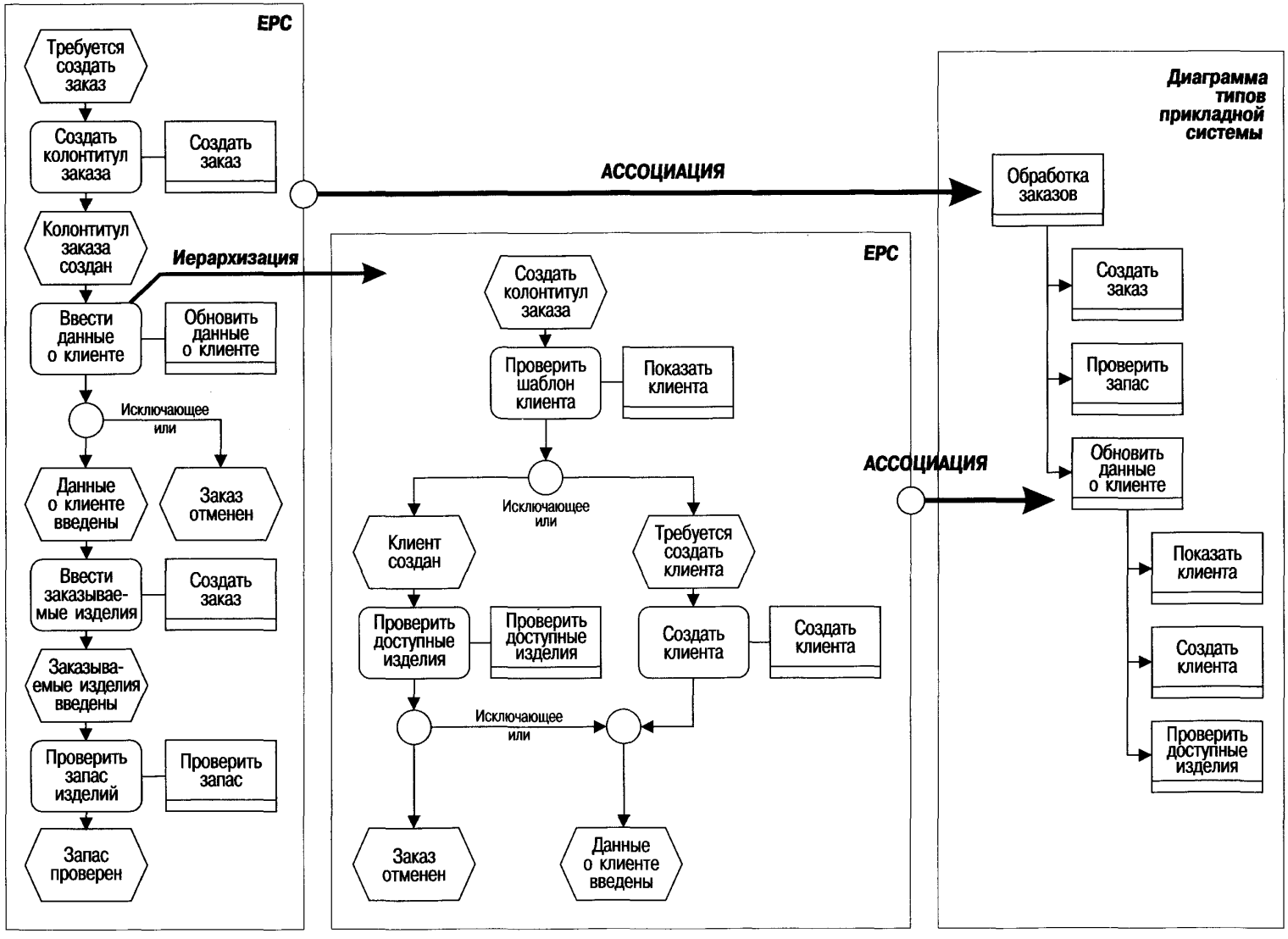


Рис. 174. Отношение между моделью процессов и диаграммой типов приложения

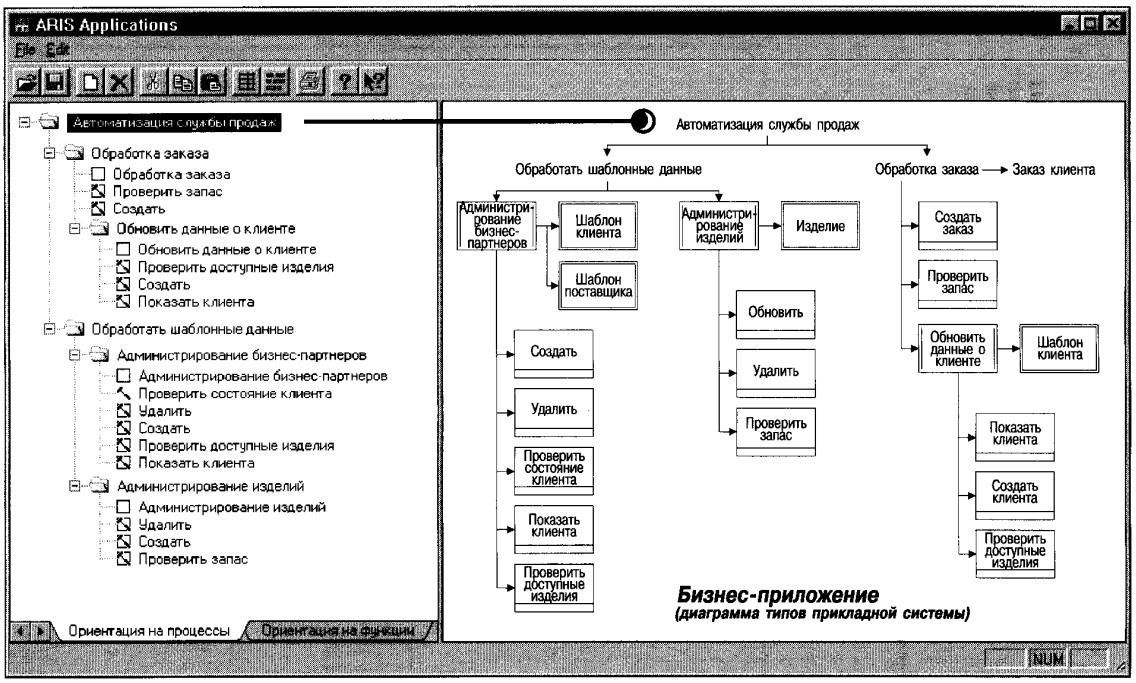


Рис. 175. Внедрение диаграммы типов приложения в рабочее пространство

щих диаграмм. На рис. 172 приведена структура бизнес-объекта «заказ».

При создании приложений бизнес-объекты используются для создания физической схемы для реляционной системы баз данных и для описания представлений. Создается также метаинформация, позволяющая установить соответствие между конкретными бизнес-объектами и общими базисными объектами в рамках инфраструктуры ARIS Framework.

Описание представления

На этом этапе описывается форма представления бизнес-объектов в рамках приложения.

Бизнес-объекты можно представлять различными способами. Конкретные типы представлений описываются с помощью диаграмм экранов. Цель моделирования экранов заключается в создании логической схемы экрана, согласующейся с моделями данных. В процессе создания приложения эта логическая структура переносится на Windows-совместимые экраны. На рис. 173 приведена логическая схема и результирующий экран.

Создание архитектуры приложения на основе диаграмм СДП

Модели процессов, созданные в ARIS Toolset, составляют фундамент для генерации структур приложений в ARIS Framework. Компоненты приложений, требующиеся в моделях процессов, определяются и моделируются в рамках диаграммы типов приложения с помощью типа приложения, типа модуля и типа функции ИТ для соответствующих типов объектов. На рис. 174 показано отношение между цепочкой процессов и диаграммой типов приложения.

С помощью диаграмм типов приложения генерирующая функция ARIS Framework создает для приложений так называемые «рабочие пространства». Эти

рабочие пространства можно использовать для обработки приложений, ориентированных на функции, объекты или процессы. На рис. 175 показана реализация диаграммы типов приложения в рамках рабочего пространства.

Кроме того, в зависимости от структуры приложения с его внедрением инициализируется интерфейс workflow. Таким образом, в распоряжении пользователя оказывается законченная, действующая прикладная программа.

Б.4. Объектно-ориентированная разработка систем с помощью унифицированного языка моделирования (UML)

Д-р Маркус Нюттгенс *(Markus Nüttgens)*; Майкл Хоффманн *(Michael Hoffmann)*, дипл. Hdl.; Томас Фельд *(Thomas Feld)*, дипл. по информатике; Институт информационных систем (IWi), Университет Саарланда, Германия.

Применительно к объектно-ориентированной разработке приложений в специальной литературе рассматривались преимущественно эволюционные процедурные модели *(Boehm. Spiral Model. 1988; Henderson-Sellers, Edwards. Object Oriented System Life Cycle. 1990, с. 152; Meyer. Object Oriented Design. 1989)*. В основе такой разработки лежат теоремы объектно-ориентированной парадигмы, где объекты представляют отдельные подсистемы «закрытой системы». В соответствии с определением внутренних и внешних объектных структур, можно разрабатывать масштабируемые системы. В эволюционной процедуре каждый цикл завершается созданием исполняемой программы. Это достигается за счет того, что результаты разработки вытекают непосредственно из целей проекта. Эти результаты можно реализовать и поодиночке, что позволяет заранее развернуть и протестировать каждую подсистему. Дополнительная разработка включает внесение

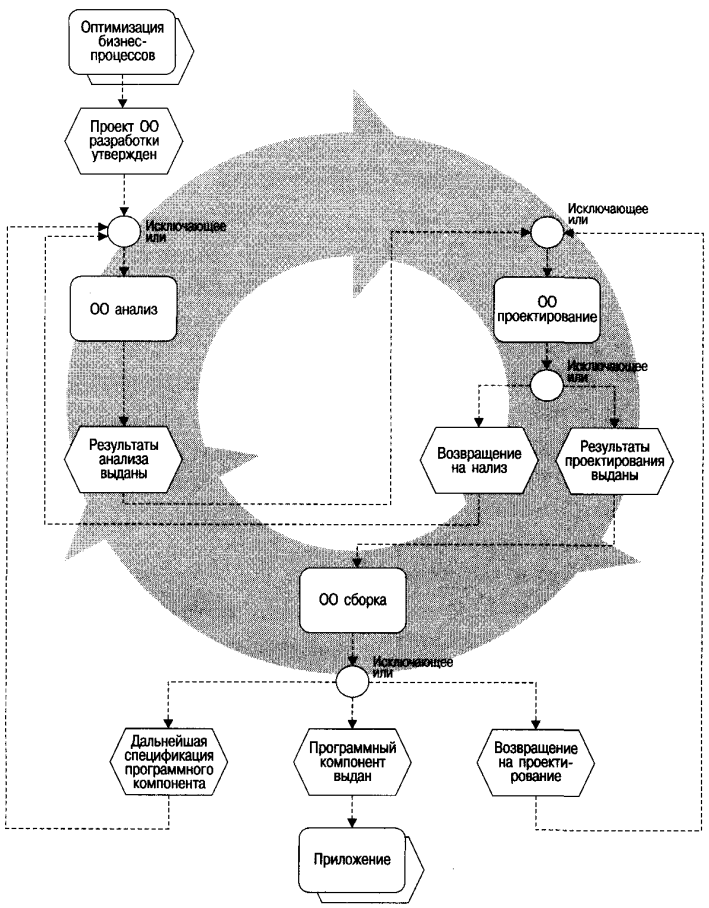


Рис. 176. Предварительная процедурная модель для объектно-ориентированной разработки приложений

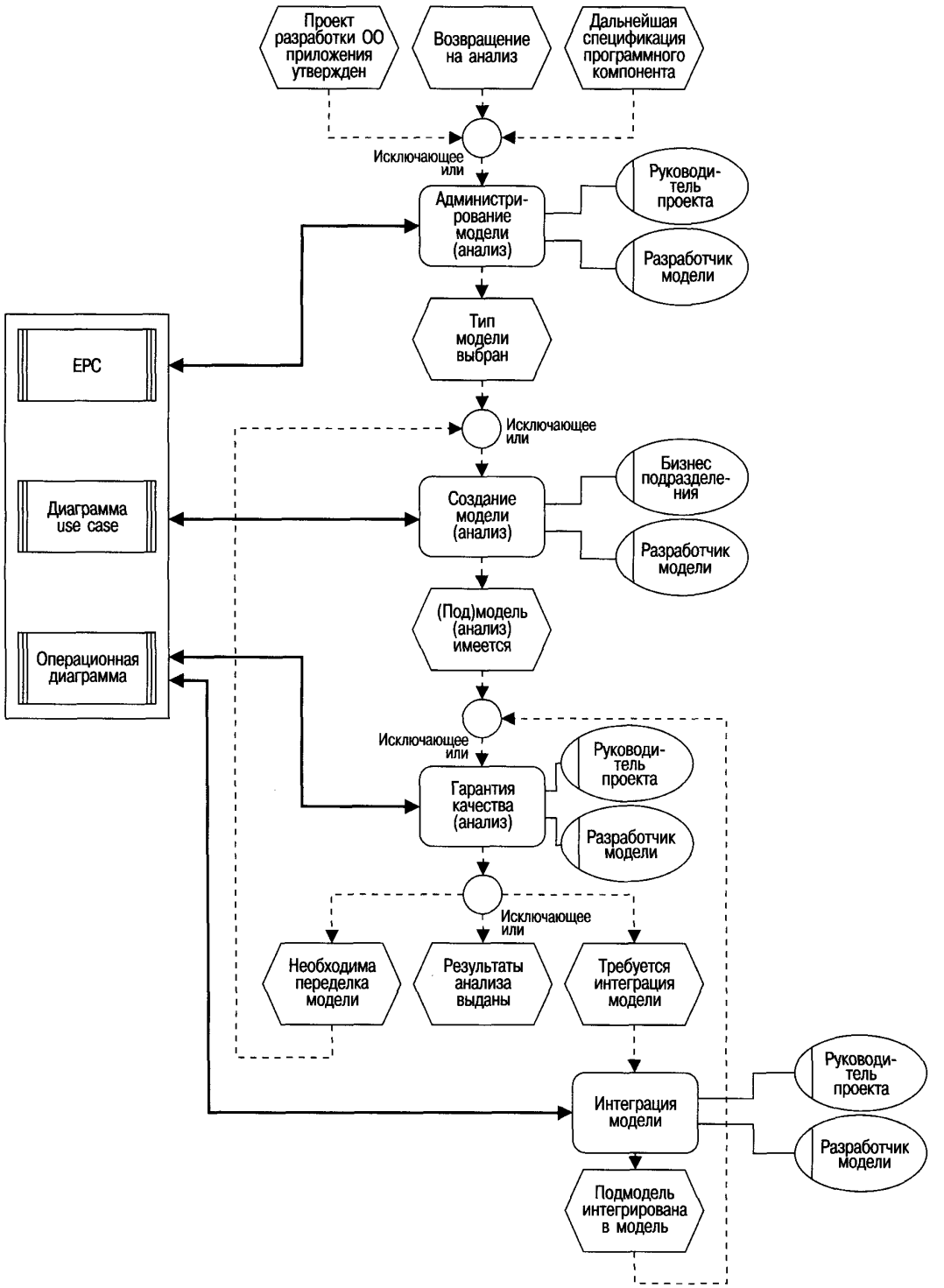


Рис. 177. Процедурная модель объектно-ориентированного анализа

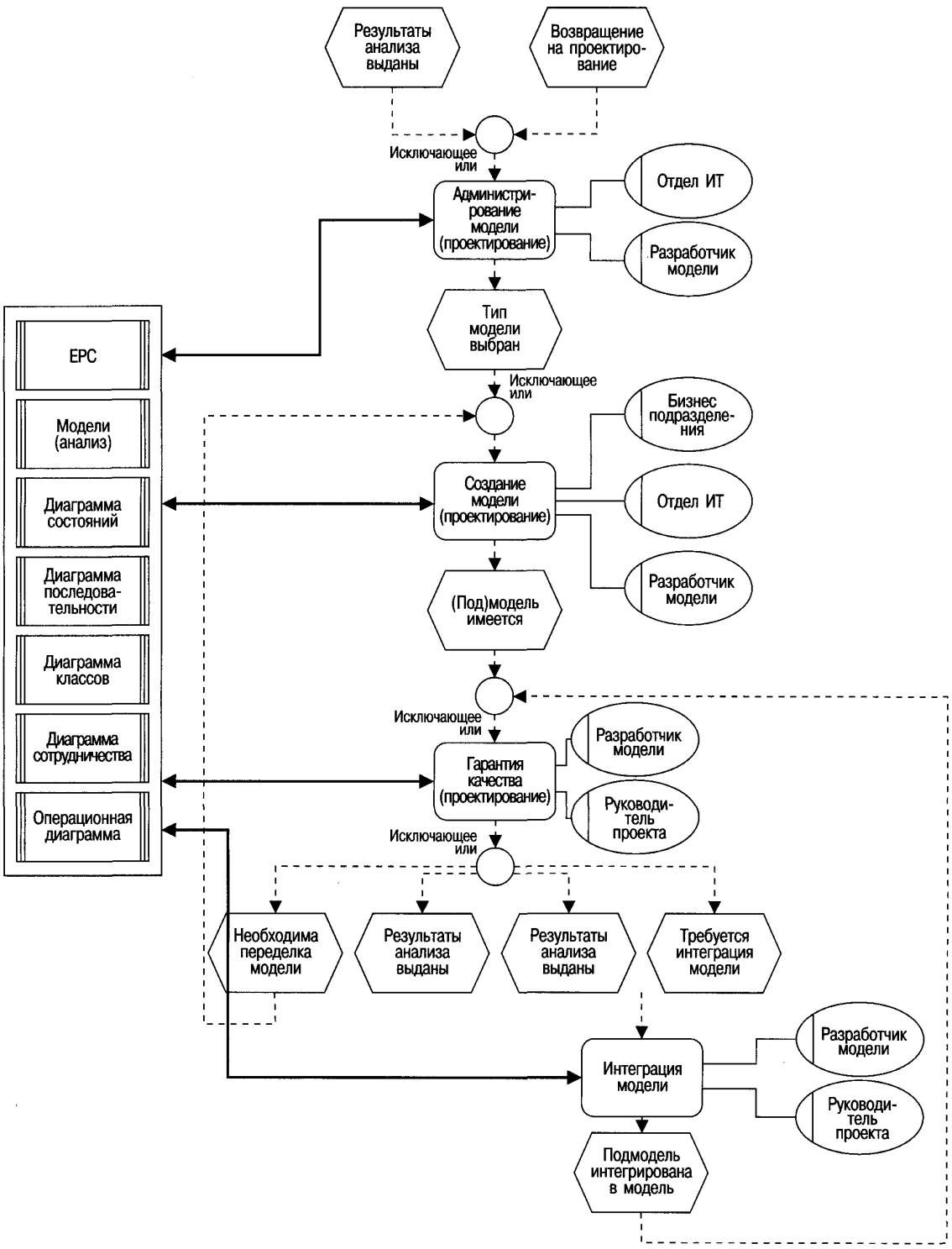


Рис. 178. Процедурная модель объектно-ориентированного проектирования

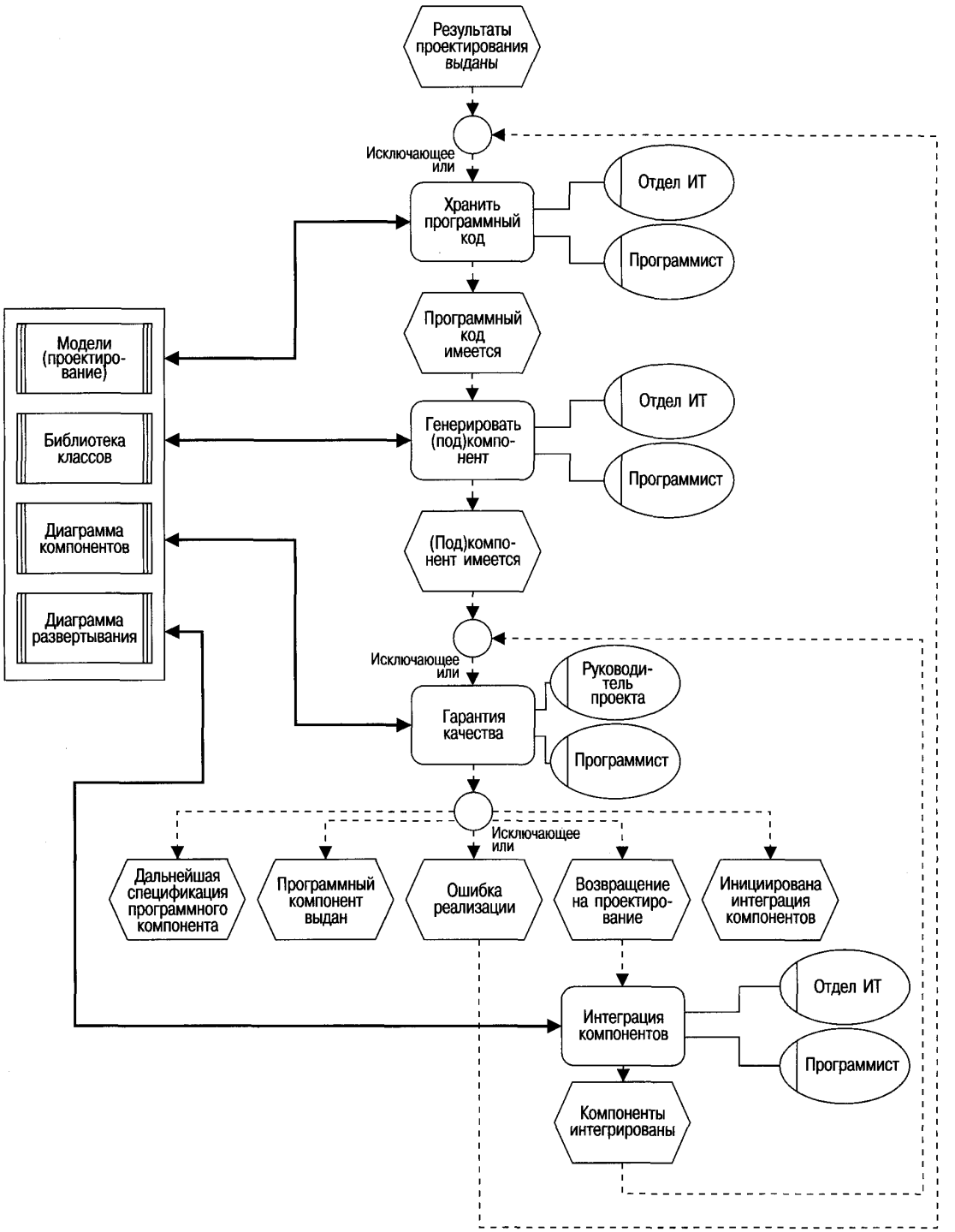


Рис. 179. Процедурная модель объектно-ориентированной сборки

усовершенствований на основе тестирования в реальных условиях и внедрение дополнительных подсистем. Это дает возможность представить результаты уже на ранней стадии и избежать «тупиков» в ходе разработки.

Б.4.1. Разработка и описание процедурных моделей

Рассмотрим процедурную модель объектно-ориентированной разработки приложений, используя в качестве примера унифицированный язык моделирования (UML) (UML Notation Guide. 1997). До сегодняшнего дня не существовало ни одного явного описания процедурной модели на языке UML. На рис. 176 представлена предварительная процедурная модель, описывающая возможную процедуру для объектно-ориентированной разработки приложений.

Объектно-ориентированная разработка приложений обычно базируется на оптимизированной модели бизнес-процессов *(Oestereich. Objektorientierte Softwareentwicklung. 1997, с. 85; Yourdon et al. Mainstream Objects. 1996, с. 71).*

Подпроцессы объектно-ориентированной разработки приложений (анализ, проектирование и сборка) описывают цикл, содержащий обратные связи с каждым элементом процедурной модели. После утверждения проекта объектно-ориентированной разработки приложения выполняется анализ и проектирование приложения объектно-ориентированным методом. Если на стадии объектно-ориентированного проектирования возникают какие-либо противоречия или вопросы, которые не могут быть разрешены на этом этапе, то приложение возвращается на стадию объектно-ориентированного анализа.

После того как результаты проектирования выданы, они становятся входным элементом для объектно-ориентированной сборки. Проектные модели реализуются более или менее автоматически с помощью генераторов кода. При выявлении ошибок, возникших на стадии генерации кода, выполняется возврат на предыдущую стадию. К развертыванию приложения можно приступать после выдачи соответствующего программного компонента. Это завершает переход от проектного цикла к реальной эксплуатации. Если возникает необходимость в переделке программного компонента, цикл начинается заново с объектно-ориентированного анализа.

Теперь перейдем к описанию подпроцессов процедурной модели на более детальном уровне транзакций.

Б.4.2. Фазы процедурной модели

Объектно-ориентированный анализ

Процесс объектно-ориентированного анализа представлен на рис. 177. В основе описания лежат стандартный блок фазовых процедурных моделей *(Nüttgens. Koordiniert-dezentrales Informationsmanagement. 1995, с. 223).*

На первом этапе администрирования модели руководитель проекта вместе с разработчиками моделей выбирает типы моделей для системного анализа. Подходящими типами моделей UML являются диаграммы use case, которые можно создавать на базе существующих моделей бизнес-процессов, например, СДП, а также операционные диаграммы.

Диаграммы use case реализуются на языке UML прежде всего для первоначальной оценки организационных сценариев. Их можно вывести, взяв за основу отдельные функциональные строительные блоки модели СДП, а затем описать с помощью соответствующей нотации UML. Отправной точкой для структурирования элементов use case могут служить компоненты приложения, поддерживающие определенные функции.

Операционные диаграммы можно вывести на основе информации потока управления, содержащейся в модели СДП, а затем расширить путем описания конкретных состояний объектов.

Операционные диаграммы служат также фундаментом для построения объектно-ориентированных моделей управления workflow.

После завершения этапа обеспечения качества (QA) (под)модели следует интегрировать в модель, а дефектные модели переделать. Полные же безупречные модели UML станут основой для объектно-ориентированного проектирования.

Объектно-ориентированное проектирование

На рис. 178 изображена процедурная модель для объектно-ориентированного проектирования, аналогичная процедурной модели для объектно-ориентированного анализа. В качестве типов моделей UML используются диаграммы состояний, диаграммы последовательности, диаграммы классов, диаграммы взаимодействия или детальные операционные диаграммы; при этом особое значение имеет проектирование диаграмм классов. Отправной точкой для проектирования диаграмм классов могут служить модели бизнес-процессов, соответствующие моделям объектно-ориентированного анализа. Например, функциональные строительные блоки и входные/выходные данные моделей СДП можно детализировать объектно-ориентированным методом, а затем привязывать к соответствующим классам бизнес-объектов.

Если структурированная модель данных уже имеется, например, в виде ERM, то ее можно использовать для выведения ключевых классов и их структурных отношений.

Объектно-ориентированная сборка

На рис. 179 приведена процедурная модель для объектно-ориентированной сборки. При реализации моделей объектно-ориентированной сборки задача состоит в том, чтобы достичь максимально возможной автоматизации за счет внедрения объектно-ориентированных генераторов кода. До определенной степени необходимо дополнительное программирование, например, для реализации методов. Можно использовать библиотеки классов, если они имеются. Остальная часть процесса аналогична процедурной модели для объектно-ориентированного проектирования или объектно-ориентированного анализа. После того как (под)компоненты успешно интегрированы, можно начать цикл процедурной модели для прототипов, которые еще не выпущены для данного приложения.

Б.4.3. Перспективы

Структурированная и объектно-ориентированная разработки приложений основываются на концепции первоначальной оптимизации целей организации бизнеса на определенных участках бизнес-процесса. Оптимизированные бизнес-процессы служат инфраструктурой для разработки приложений в виде сценариев организации процессов. Метод СДП доказал свою состоятельность применительно к описанию контекста бизнеса как в теории, так и на практике. Однако концепции, позволяющие расширить методику путем переноса организационных и функциональных моделей СДП в модели объектно-ориентированного анализа и проектирования, пока еще нуждаются в дальнейшем совершенствовании. Некоторые из них изложены в работах: *Hoffmann, Scheer, Hoffmann. Modellierungsmethoden. 1995; Bungert, Heß. Objektorientierte Geschäftsprozeßmodellierung. 1995; Scheer, Nüttgens, Zimmermann. oEPK. 1997.*

Раскрывая и документально подтверждая преимущества моделей комплексной разработки систем, процедурные модели могут внести серьезный вклад в интеграцию методов моделирования.

Шеер Август-Вильгельм
«Моделирование бизнес-процессов»

Лицензия № 064209 от 15 августа 1995 г.

ООО «Издательство «Серебряные нити»
129515, Москва, а/я 34

Подписано в печать 10.06.2000. Объем
Печать офсетная. Формат 1/8 60x84. Гарнитура «Journal»
Бумага офс. Тираж 1000 экз. Заказ № 207.

Изготовлено с готовых диапозитивов
ООО «Момент»
141 400 Моск. обл., г. Химки, ул. Нахимова д. 2