АУДИТОРУ О РАБОТЕ НА КОМПЬЮТЕРЕ

ПРОГРАММНЫЕ ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА АВТОМАТИЗАЦИИ ПРОЕКТИРОВАНИЯ СИСТЕМ

Маклаков С.В., к.ф.-м.н.

Фирма INTERFACE

Введение

Технология создания крупных информационных систем (далее - ИС) предъявляет особые требования к методикам реализации и программным инструментальным средствам, а именно:

Реализацию крупных проектов принято разбивать на стадии анализа (прежде чем создавать ИС необходимо понять и описать бизнес-логику предметной области), проектирования (необходимо определить модули и архитектуру будущей системы), непосредственного кодирования, тестирования и сопровождения. Известно, что исправление ошибок, допущенных на предыдущей стадии обходится примерно в десять раз дороже, чем на текущей, откуда следует, что наиболее критичными являются первые стадии проекта. *Поэтому крайне важно иметь эффективные средства автоматизации ранних этапов реализации проекта.*

Крупный проект невозможно реализовать в одиночку. Коллективная работа существенно отличается от индивидуальной, поэтому *при реализации крупных проектов необходимо иметь средства координации и управления коллективом разработчиков*.

Жизненный цикл создания сложной ИС сопоставим с ожидаемым временем ее эксплуатации. Другими словами, в современных условиях компании перестраивают свои бизнес - процессы примерно раз в два года, столько же требуется (если работать в традиционной технологии) для создания ИС. Может оказаться, что к моменту сдачи ИС она уже никому не нужна, поскольку компания, ее заказавшая, вынуждена перейти на новую технологию работы. Следовательно, *для создания крупной ИС жизненно необходим инструмент значительно (в несколько раз) уменьшающий время разработки ИС.*

Вследствие значительного жизненного цикла может оказаться, что в процессе создания системы внешние условия изменились. Обычно внесение изменений в проект на поздних этапах создании ИС - весьма трудоемкий и дорогостоящий процесс. Поэтому для успешной реализации крупного проекта необходимо, чтобы *инструментальные средства, на которых он реализуются, были достаточно гибкими к изменяющимся требованиям.*

На современном рынке средств разработки ИС достаточно много систем, в той или иной степени удовлетворяющих перечисленным требованиям. Ниже будет рассмотрена вполне конкретная технология разработки, основывающаяся на решениях фирм Logic Works и Rational Software, которая является одной из лучших на сегодняшний день по критерию стоимость/эффективность.

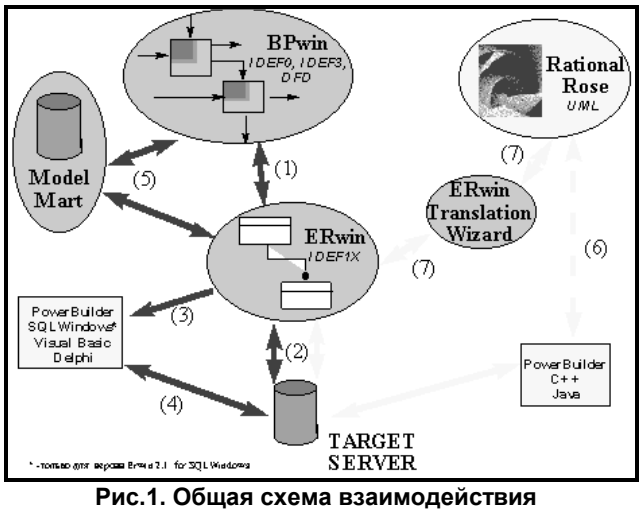


Рис.1. Общая схема взаимодействия инструментальных средств Logic Works и Rational Software

Для проведения анализа и реорганизации бизнес-процессов Logic Works предлагает CASE - средство верхнего уровня - BPwin, поддерживающий методологии IDEF0 (функциональная модель), IDEF3 (WorkFlow Diagram) и DFD (DataFlow Diagram). Функциональная модель предназначена для описания существующих бизнес-процессов на предприятии (так называемая модель AS-IS) и идеального положения вещей - того, к чему нужно стремиться (модель TO-BE). Методология IDEF0 предписывает построение иерархической системы диаграмм - единичных описаний фрагментов системы. Сначала проводится описание системы в целом и ее взаимодействия с окружающим миром (контекстная диаграмма), после чего проводится функциональная декомпозиция - система разбивается на подсистемы и каждая подсистема описывается отдельно (диаграммы декомпозиции). Затем каждая подсистема разбивается на более мелкие и так далее до достижения нужной степени подробности. После каждого сеанса декомпозиции проводится сеанс экспертизы, каждая диаграмма проверяется экспертами предметной области, представителями заказчика, людьми, непосредственно участвующими в бизнес - процессе. Такая технология создания модели позволяет построить модель адекватную предметной области на всех уровнях абстрагирования. Если в процессе моделирования нужно осветить специфические стороны технологии предприятия, BPwin позволяет переключиться на любой ветви модели на нотацию IDEF3 или DFD и создать смешанную модель. Нотация DFD включает такие понятия как внешняя ссылка и хранилище данных, что делает ее более удобной (по сравнению с IDEF0) для моделирования документооборота. Методология IDEF3 включает элемент “перекресток”, что позволяет описать логику взаимодействия компонентов системы.

На основе модели BPwin’а можно построить модель данных. Для построения модели данных Logic Works предлагает мощный и удобный инструмент - ERwin. Хотя процесс преобразования модели BPwin в модель данных плохо формализуется и поэтому полностью не автоматизирован, Logic Works предлагает удобный инструмент

для облегчения построения модели данных на основе функциональной модели - механизм двунаправленной связи BPwin - ERwin (1, рис.1). ERwin имеет два уровня представления модели - логический и физический. На логическом уровне данные представляются безотносительно конкретной СУБД, поэтому могут быть наглядно представлены даже для неспециалистов. Физический уровень данных - это, по - существу, отображение системного каталога, который зависит от конкретной реализации СУБД. ERwin позволяет проводить процессы прямого и обратного проектирования БД (2, рис.1). Это означает, что по модели данных можно сгенерировать схему БД или автоматически создать модель данных на основе информации системного каталога. Кроме того, ERwin позволяет выравнивать модель и содержимое системного каталога после редактирования того, либо другого. ERwin интегрируется с популярными средствами разработки клиентской части - PowerBuilder, SQLWindows, Visual Basic, Delphi (3, рис.1), что позволяет автоматически генерировать код приложения, который готов к компиляции и выполнению (4, рис.1).

Создание современных информационных систем, основанных на широком использовании распределенных вычислений, объединении традиционных и новейших информационных технологий, требует тесного взаимодействия всех участников проекта: менеджеров, бизнес и системных аналитиков, администраторов баз данных, разработчиков. Для этого использующиеся на разных этапах и разными специалистами средства моделирования и разработки должны быть объединены общей системой организации совместной работы. Фирма Logic Works разработала систему Model Mart - хранилище моделей, к которому открыт доступ для участников проекта создания информационной системы (5, рис.1). Model Mart удовлетворяет всем требованиям, предъявляемым к средствам разработки крупных информационных систем, а именно:

Совместное моделирование. Каждый участник проекта имеет инструмент поиска и доступа к интересующей его модели в любое время. При совместной работе используются три режима: незащищенный, защищенный и режим просмотра. В режиме просмотра запрещается любое изменение моделей. В защищенном режиме модель, с которой работает один пользователь не может быть изменена другими пользователями. В незащищенном режиме пользователи могут работать с общими моделями в реальном масштабе времени. Возникающие при этом конфликты разрешаются при помощи специального модуля - Intelligent Conflict Resolution (ICR). В дополнение к стандартным средствам организации совместной работы Model Mart позволяет сохранять множество версий, снабженных аннотациями, с последующим сравнением предыдущих и новых версий. При необходимости возможен возврат к предыдущим версиям.

Создание библиотек решений. Model Mart позволяет формировать библиотеки стандартных решений, включающие наиболее удачные фрагменты реализованных проектов, накапливать и использовать типовые модели, объединяя их при необходимости "сборки" больших систем. На основе существующих баз данных с помощью ERwin возможно восстановление моделей (обратное проектирование), которые в процессе анализа пригодности их для новой системы могут объединяться с типовыми моделями из библиотек моделей.

Управление доступом. Для каждого участника проекта определяются права доступа, в соответствии с которыми они получают возможность работать только с определенными моделями. Права доступа могут быть определены как для групп, так и для отдельных участников проекта. Роль специалистов, участвующих в различных проектах может меняться, поэтому в Model Mart можно определять и управлять правами доступа участников проекта к библиотекам, моделям и даже к специфическим областям модели.

Архитектура Model Mart. Model Mart реализована на архитектуре клиент - сервер. В качестве платформы реализации хранилища выбраны РСУБД Sybase, Microsoft SQL Server и Oracle. Клиентскими приложениями являются ERwin 3.x и BPwin 2.0x. В следующих версиях Model Mart предполагается открыть доступ к хранилищу моделей через API, что позволит постоянно наращивать возможности интегрированной среды путем включения новых инструментов моделирования и анализа.

Как было указано выше (см. пункт С), при разработке крупных проектов критичным становится время реализации проекта. Одним из решений проблемы может стать автоматическая генерация кода приложения (клиентской части) CASE - средствами на основе модели предметной области. Хотя ERwin решает эту задачу, код генерируется на основе модели IDEF1X, то есть фактически на основе реляционной модели данных, которая непосредственно не содержит информацию о бизнес - процессах. Как следствие этого, сгенерированный код не может полностью обеспечить функциональность приложения со сложной бизнес-логикой. Существует альтернативная технология кодогенерации, которая лишена этого недостатка - объектно-ориентированное проектирование, реализованное в Rational Rose (Rational Software). Rational Rose – позволяющее строить объектные модели в различных нотациях (OMT, UML, Буч) и генерировать на основе полученной модели приложения на языках программирования C++, Visual Basic, Power Builder, Java, Ada, Smalltalk и др. Поскольку генерация кода реализована на основе знаний предметной области, а не на основе реляционной структуры данных, полученный код более полно отражает бизнес-логику. Rational Rose поддерживает не только прямую генерацию кода, но и обратное проектирование, то есть создание объектной модели по исходному коду приложения (6, рис.1).

Rational Rose предназначен для генерации клиентской части приложения. Для генерации схемы БД объектную модель следует конвертировать в модель данных IDEF1X. Модуль ERwin Translation Wizard (Logic Works) позволяет перегружать объектную модель Rational Rose в модель данных ERwin (и обратно) и, с помощью ERwin, сгенерировать схему БД (7, рис.1). Таким образом, технологическая цепочка Rational Rose - ERwin Translation Wizard - ERwin позволяет реализовывать крупные проекты в технологии клиент - сервер.

1. МОДЕЛИ ПРОЦЕССОВ

Создание модели процессов в BPwin (IDEF0)

На начальных этапах создания ИС необходимо понять, как работает организация, которую мы собираемся автоматизировать. Никто в организации не знает,

как она работает в той мере подробности, которая необходима для создания ИС. Руководитель хорошо знает работу в целом, но не в состоянии вникнуть в детали работы каждого рядового сотрудника. Рядовой сотрудник хорошо знает, что творится на его рабочем месте, но плохо знает, как работают коллеги. Поэтому для описания работы предприятия необходимо построить модель. BPwin как раз и предназначен для построения такой модели - функциональной модели (или модели процессов).

Обычно сначала строится модель существующей организации работы -“AS-IS” (как есть). Анализ функциональной модели позволяет понять, где находятся наиболее слабые места, в чем будут состоять преимущества новых бизнес-процессов и насколько глубоким изменениям подвергнется существующая структура организации бизнеса. Найденные в модели “AS-IS” недостатки можно исправить при создании модели “TO-BE” (как должно быть) - модели новой организации бизнес-процессов.

Наиболее удобным языком моделирования бизнес-процессов является IDEF0, предложенный более 20 лет назад Дугласом Россом (Ранее назывался SADT - Structured Analysis and Design Technique).

Подробно методология SADT излагается в книге Дэвида А.Марка и Клемента МакГоуэна“ Методология структурного анализа и проектирования SADT”, издательство Метатехнология, 1993.

Под моделью в IDEF0 понимают описание системы (текстовое и графическое), которое должно дать ответ на некоторые заранее определенные вопросы. Процесс моделирования какой -либо системы в IDEF0 начинается с определения контекста, т.е. наиболее абстрактного уровня описания системы в целом. В контекст входит определение субъекта моделирования, цели и точки зрения на модель. Под субъектом понимается сама система, при этом необходимо точно установить, что входит в систему, а что лежит за ее пределами, другими словами, мы должны определить, что мы будем в дальнейшем рассматривать как компоненты системы, а что как внешнее воздействие. На определение субъекта системы будет существенно влиять позиция, с которой рассматривается система, и цель моделирования - вопросы, на которые построенная модель должна дать ответ. IDEF0-модель предполагает наличие четко сформулированной цели, единственного субъекта моделирования и одной точки зрения.

Основу методологии IDEF0 составляет графический язык описания бизнес- процессов. Модель в нотации IDEF0 представляет собой совокупность иерархически упорядоченных и взаимосвязанных диаграмм. Вершина этой древовидной структуры, представляющая собой самое общее описание системы и ее взаимодействия с внешней средой, называется контекстной диаграммой. После описания системы в целом проводится разбиение ее на крупные фрагменты. Этот процесс называется функциональной декомпозицией, а диаграммы, которые описывают каждый фрагмент и взаимодействие фрагментов, называются диаграммами декомпозиции. После декомпозиции контекстной диаграммы проводится декомпозиция каждого большого фрагмента системы на более мелкие и так далее до достижения нужного уровня подробности описания. После каждого сеанса декомпозиции проводятся сеансы экспертизы - эксперты предметной области указывают на соответствие реальных бизнес - процессов созданным диаграммам. Найденные несоответствия исправляются и только после прохождения экспертизы без замечаний можно приступать к следующему сеансу декомпозиции. Таким образом достигается соответствие модели реальным бизнес - процессам на любом и каждом уровне модели. Синтаксис описания системы в целом и каждого ее фрагмента одинаков во всей модели. Работы (Activity), которые означают некие поименованные процессы, функции или задачи, изображаются в виде прямоугольников. Именем работы должен быть глагол или глагольная форма (например “Изготовление детали”, “Прием заказа” и т.д.). Взаимодействие работ с внешним миром и между собой описывается в виде стрелок. Стрелки представляют собой некую информацию и именуются существительными (например, “Заготовка”, “Изделие”, “Заказ”). В IDEF0 различают пять типов стрелок:

Вход (Input) - материал или информация, которая используется или преобразовывается работой.

Управление (Control) - правила, стратегии, процедуры или стандарты, которыми руководствуется работа. Каждая работа должна иметь хотя бы одну стрелку управления.

Выход (Output) - материал или информация, которая производится работой. Каждая работа должна иметь хотя бы одну стрелку выхода.

Механизм (Mechanism)- ресурсы, которые выполняют работу, например, персонал предприятия станки, механизмы и т.д.

Вызов - специальная стрелка, указывающая на другую модель работы.

Каждый тип стрелок подходит или выходит к определенной стороне прямоугольника, изображающего работу. К левой стороне подходят стрелки входов, к верхней - стрелки управления, к нижней - механизмов реализации выполняемой функции, а из правой - выходят стрелки выходов. Такое соглашение предполагает, что используя управляющую информацию и реализующий ее механизм, функция преобразует свои входы в соответствующие выходы.

При создании новой модели (меню File / New) автоматически создается контекстная диаграмма с единственной работой, изображающей систему в целом. Для внесения имени работы следует кликнуть по работе правой кнопкой мыши, выбрать в меню Name Editor и в появившемся диалоге внести имя работы. Для описания других аспектов контекста служит диалог Model Definition Editor (вызывается из меню Edit/Model Definition).

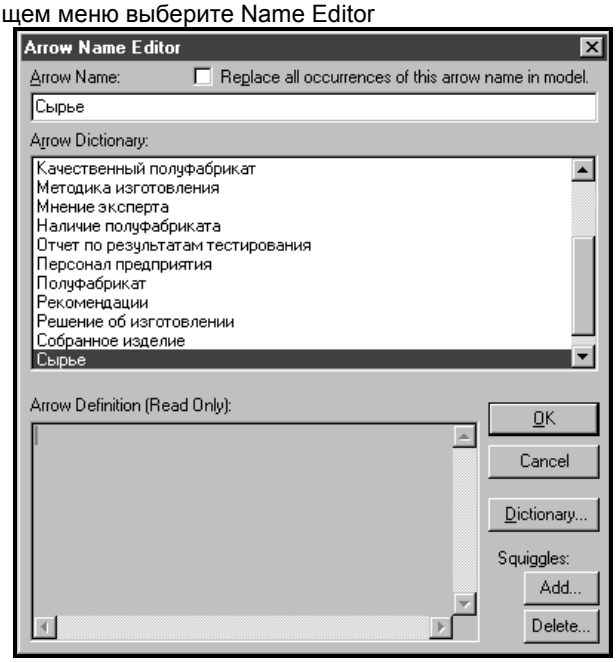
Стрелки на контекстной диаграмме служат для описания взаимодействия системы с окружающим миром. Они могут начинаться у границы диаграммы и заканчиваться у работы, или наоборот. Такие стрелки называются граничными. Для внесения граничной стрелки входа на контекстной диаграмме кликните на символе стрелки в палитре инструментов  ;

перенесите курсор к левой стороне экрана, пока не появится начальная штриховая полоска;

щелкните один раз по полоске (откуда выходит стрелка) и еще раз в левой части работы со стороны входа (где заканчивается стрелка);

вернитесь в палитру инструментов и выберите опцию редактирования стрелки  ;

дважды щелкните на линии стрелки , во всплывающем меню выберите Name Editor



и добавьте имя стрелки.

Стрелки управления, выхода, механизма и выхода изображаются аналогично.

Имена вновь внесенных стрелок автоматически заносятся в словарь (Arrow Dictionary). Словарь стрелок редактируется при помощи специального редактора Arrow Dictionary Editor, в котором определяется стрелка и вносится относящийся к ней комментарий. Словарь стрелок можно распечатать в виде отчета (меню Report / Arrow Report...) и получить тем самым толковый словарь терминов предметной области, использующихся в модели.

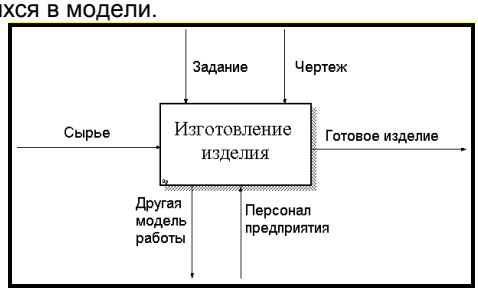


Рис.2. Контекстная диаграмма

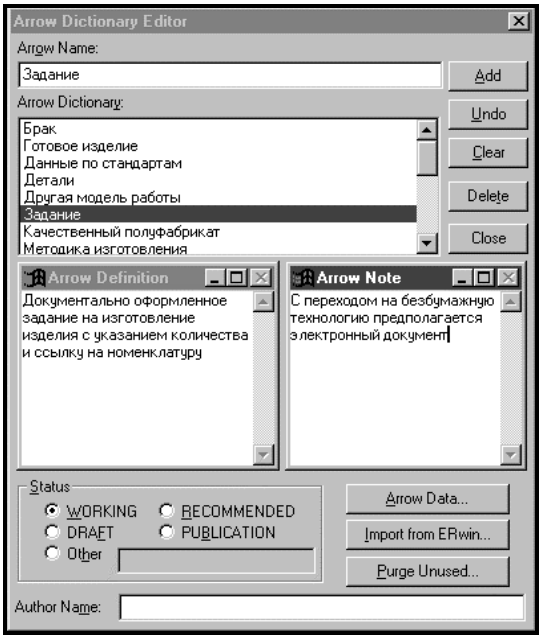


Рис.3. Словарь стрелок

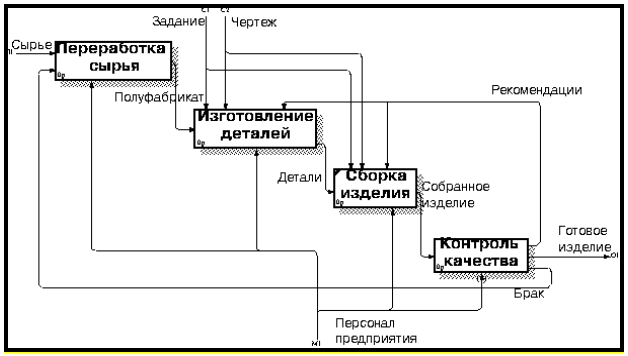


Рис.4. Диаграмма декомпозиции

После создания контекстной диаграммы можно приступить к декомпозиции. Для этого нужно кликнуть по кнопке

перехода на нижний уровень. Появляется диалог Activity Box Count, в котором необходимо указать количество работ на диаграмме декомпозиции (в дальнейшем можно будет добавить недостающие работы или удалить лишние) и нотацию диаграммы. BPwin позволяет создавать смешанные модели - в рамках одной модели могут сосуществовать и быть связанными модели IDEF0, DFD и IDEF3. Такой подход позволяет описать интересующие нас аспекты каждой подсистемы. Для обеспечения наглядности и лучшего понимания моделируемых процессов рекомендуется использовать от 3-х до 6-ти блоков на одной диаграмме. Остановимся пока на нотации IDEF0 и кликнем на OK. Появляется диаграмма декомпозиции. Работы расположены в так называемом порядке доминирования (по степени важности или в порядке очередности выполнения), начиная с левого верхнего угла и кончая нижним правым углом, что значительно облегчает в дальнейшем чтение диаграммы. Стрелки, которые были внесены на контекстной диаграмме, показываются и на диаграмме декомпозиции (миграция стрелок), но при этом не касаются работ. Такие стрелки называются несвязанными и воспринимаются, как синтаксическая ошибка. Для связывания стрелки необходимо перейти в

режим редактирования стрелок, кликнуть по стрелке и кликнуть по соответствующему сегменту работы. Для связи работ между собой используются внутренние стрелки, т.е. стрелки, которые не касаются границы диаграммы, начинаются у одной и кончаются у другой работы.

Для рисования внутренней стрелки необходимо в режиме рисования стрелок кликнуть по сегменту (например выхода) одной работы и затем по сегменту (например входа) другой. В IDEF0 различают пять типов связей работ:

прямая связь по входу, когда стрелка выхода вышестоящей (далее просто выход) работы направляется на вход нижестоящей (например, на рисунке стрелка “Детали” связывает работы “Изготовление деталей” и “Сборка деталей”);

прямая связь по управлению, когда выход вышестоящей работы направляется на управление нижестоящей;

обратная связь по входу, когда выход нижестоящей работы направляется на вход вышестоящей (стрелка “Брак”);

обратная связь по управлению, когда выход нижестоящей работы направляется на управление вышестоящей (стрелка “Рекомендации”);

связь выход-механизм, когда выход одной работы направляется на механизм другой.

Вновь внесенные граничные стрелки на диаграмме декомпозиции нижнего уровня изображаются в квадратных скобках и автоматически не появляются на диаграмме верхнего уровня. Для их “перетаскивания” наверх нужно сначала выбрать кнопку  на палитре инструментов и кликнуть по квадратным скобкам граничной стрелки. Появляется диалог Border Arrow Editor.

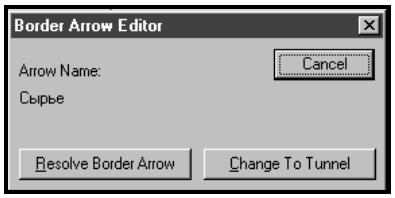


Рис.5. Диалог Border Arrow Editor

Если кликнуть по кнопке Resolve Border Arrow, стрелка мигрирует на диаграмму верхнего уровня, если по кнопке Change To Tunnel, стрелка будет затуннелирована и не попадет на другую диаграмму. Туннелирование может быть применено для изображения малозначимых стрелок.

Одна и та же информация может обрабатываться в нескольких работах, в то же время из нескольких работ могут выходить одинаковые данные, то есть стрелки могут разветвляться и сливаться. Для разветвления стрелки нужно в режиме редактирования стрелки кликнуть по фрагменту стрелки и по соответствующему сегменту работы.

Список синтаксических ошибок модели можно получить сгенерировав отчет об ошибках (Report / Model Consistency Report...).

Дополнение модели процессов диаграммами DFD и Workflow (IDEF3)

Общие принципы построения модели в методологиях DFD и IDEF3 сходны с IDEF0: модель представляет собой совокупность иерархически зависимых диаграмм, прямоугольники изображают работы или процессы, стрелки- это тоже некие данные, построение модели осуществляется сверху вниз путем проведения декомпозиции крупных работ на более мелкие.

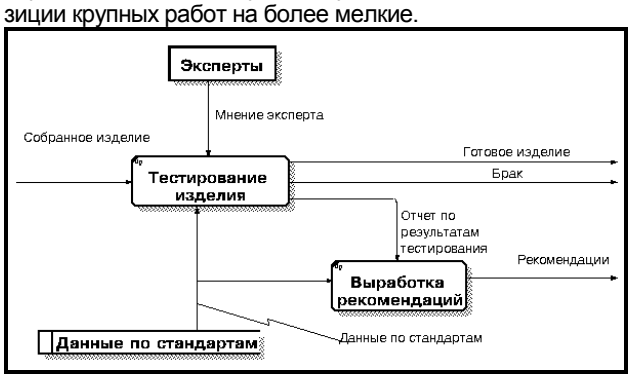
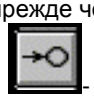


Рис.6. Диаграмма в нотации DFD

Диаграммы потоков данных (Data flow diagramming, DFD) используются для описания документооборота и обработки информации. Их можно использовать как дополнение к модели IDEF0 для более наглядного отображения текущих операций документооборота в корпоративных системах обработки информации. DFD описывают функции обработки информации (работы), документы (стрелки, arrow), объекты, сотрудников или отделы, которые участвуют в обработке информации (внешние ссылки, external references) и таблицы для хранения документов (хранилище данных, data store). В отличие от IDEF0 для стрелок нет понятия вход, выход, управление или механизм и неважно, в какую грань работы входит или из какой грани выходят стрелки. В BPwin для построения диаграмм потоков данных используется нотация Гейна-Сарсона. Для того, чтобы дополнить модель IDEF0 диаграммой DFD нужно в процессе декомпозиции в диалоге Activity Box Count кликнуть по радиокнопке DFD. В палитре инструментов на новой диаграмме DFD появляются новые кнопки:

 - добавить в диаграмму внешнюю ссылку (External Reference). Внешняя ссылка является источником или приемником данных извне модели ;

 - добавить в диаграмму хранилище данных (Data store). Хранилище данных позволяет описать данные, которые необходимо сохранить в памяти прежде чем использовать в работах.

 - ссылка на другую страницу. В отличие от IDEF0 инструмент off- page reference позволяет направить стрелку на любую диаграмму (а не только на верхний уровень).

Наличие в диаграммах DFD элементов для описания источников, приемников и хранилищ данных позволяет более эффективно и наглядно описать процесс документооборота. Однако, для описания логики взаимодействия

информационных потоков более подходит IDEF3, называемая также workflow diagramming, - методология моделирования, использующая графическое описание информационных потоков, взаимоотношений между процессами обработки информации и объектов, являющихся частью этих процессов. Диаграммы Workflow могут быть использованы в моделировании бизнес - процессов для анализа завершенности процедур обработки информации. С их помощью можно описывать сценарии действий сотрудников организации, например, последовательность обработки заказа или события, которые необходимо обработать за конечное время. Каждый сценарий сопровождается описанием процесса и может быть использован для документирования каждой функции.

Прямоугольники на диаграмме Workflow называются единицами работы (Unit of Work, UOW) и обозначают событие, процесс, решение или работу. Для редактирования диаграммы используются примерно те же диалоги, что и для IDEF0. В палитре инструментов на диаграмме Workflow имеются кнопки для новых элементов:


- добавить в диаграмму объект ссылки (Referent). Объект ссылки в IDEF3 выражает некую идею, концепцию или данные, которые нельзя связать со стрелкой, перекрестком или работой. Имя объекта ссылки задается в диалоге Referent (пункт всплывающего меню Name Editor), в качестве имени можно использовать имя какой-либо стрелки с других диаграмм или имя сущности из модели данных (о том, как использовать модель данных в BPwin будет рассказано в следующем разделе). Объекты ссылки должны быть связаны с единицами работ или перекрестками пунктирными линиями. Официальная спецификация IDEF3 различает три стиля объектов ссылок - безусловные (unconditional), синхронные (synchronous) и асинхронные (asynchronous). BPwin поддерживает только безусловные. Синхронные и асинхронные, используемые в диаграммах переходов состояний объектов не поддерживаются.


- добавить в диаграмму перекресток (Junction). Перекрестки используются для отображения логики взаимодействия стрелок при слиянии и разветвлении или для отображения множества событий, которые могут или должны быть завершены перед началом следующей работы. Различают перекрестки для слияния (Fan-in Junction) и разветвления (Fan-out Junction) стрелок. Перекресток не может использоваться одновременно для слияния и для разветвления. При внесении перекрестка в диаграмму в диалоге Junction Type Editor необходимо указать тип перекрестка. Смысл каждого типа приведен в табл. 1.

Все перекрестки на диаграмме нумеруются, каждый номер имеет префикс "J". Можно редактировать свойства перекрестка при помощи диалога Definition Editor. В отличие от IDEF0 и DFD, в IDEF3 стрелки могут сливаться и разветвляться только через перекрестки. Здесь различают три типа стрелок, стиль которых устанавливается через меню Edit / Arrow Style:

Старшая (Precedence)  - сплошная линия, связывающая единицы работ (UOW). Рисуется слева направо или сверху вниз.

Таблица 1

Тип перекрестка

Обозначение	Наименование	Смысл в случае слияния стрелок (Fan-in Junction)	Смысл в случае разветвления стрелок (Fan-out Junction)
	Asynchronous AND	Все предшествующие процессы должны быть завершены	Все следующие процессы должны быть запущены
	Synchronous AND	Все предшествующие процессы завершены одновременно	Все следующие процессы запускаются одновременно
	Asynchronous OR	Один или несколько предшествующих процессов должны быть завершены	Один или несколько следующих процессов должны быть запущены
	Synchronous OR	Один или несколько предшествующих процессов завершены одновременно	Один или несколько следующих процессов запускаются одновременно
	XOR (Exclusive OR)	Только один предшествующий процесс завершен	Только один следующий процесс запускается

Отношения (Relational Link)  - пунктирная линия, использующаяся для изображения связей между единицами работ (UOW) и между единицами работ и объектами ссылок.

Потоки объектов (Object Flow)  - стрелка с двумя наконечниками используется для описания того факта, что объект используется в двух или более единицах работы, например, когда объект порождается в одной работе и используется в другой.

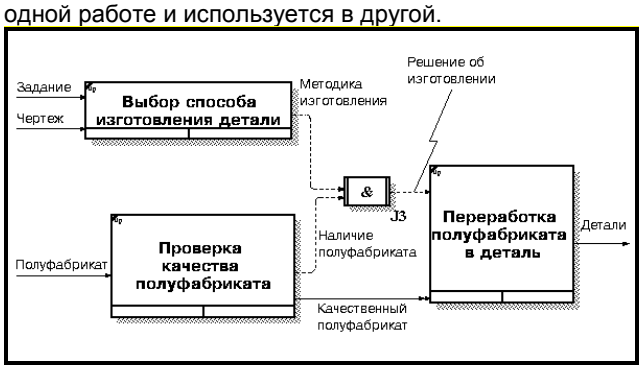


Рис.7. Диаграмма в нотации IDEF3

В результате дополнения диаграмм IDEF0 диаграммами DFD и IDEF3 может быть создана смешанная модель, которая наилучшим образом описывает все стороны деятельности предприятия. Иерархию работ в смешанной модели можно увидеть в окне Model

Explorer Работы в нотации IDEF0 изображаются зеленым цветом, IDEF3 - желтым, DFD- синим.

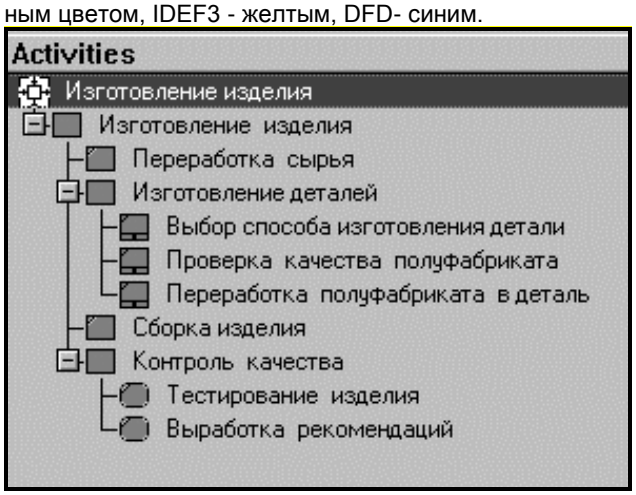


Рис.8. Смешанная модель в окне Model Explorer

Синтаксический анализ модели BPwin'a позволяет легко обнаружить "бесполезные" (не имеющие выхода), "неуправляемые" (не имеющие управления) и "простаивающие" работы. Более тонкий анализ позволяет выявить дублирующие, избыточные или неэффективные работы. Модель дает целостное представление о работе системы в целом и позволяет понять взаимосвязи всех составляющих системы. При этом часто выясняется, что обработка информации и использование ресурсов неэффективны, важная информация не доходит до соответствующего рабочего места и т.д. Признаком неэффективной организации работ является, например, отсутствие обратных связей по входу и управлению для многих, критически важных работ.

Невозможно построить эффективную ИС при неэффективной общей организации работы. Поэтому результатом анализа и критической оценки модели AS-IS должно быть перенаправление информационных потоков и усовершенствование бизнес- процессов в новой модели TO-BE, которая должна использоваться для реорганизации деятельности предприятия. Такой результат построения модели сам по себе самодостаточен, то есть если удается более рационально организовать бизнес-процессы на предприятии - это уже результат, оправдывающей капиталовложения. Однако при создании ИС модель процессов - это только первый шаг, за которым обычно следует построение модели данных.

Соответствие модели данных и модели процессов

Стрелки в модели процессов означают некоторую информацию, использующуюся в моделируемой системе. ERWin поддерживает два уровня представления модели данных – логический и физический. Логический уровень не зависит от конкретной реализации БД и позволяет наглядно представить данные для обсуждения с экспертами предметной области. Физический уровень является отображением системного каталога БД и зависит от конкретной реализации БД. На логическом уровне модели данных информация отображается в виде сущностей (соответствуют таблицам на физическом уровне), состоящих из атрибутов сущностей (соответствуют колонкам таблицы). Сущности состоят из совокупности отдельных записей - экземпляров сущностей (соответствуют записям в таблице). К модели данных предъявляются определенные требования (т.н. нормализация данных), которые призваны обеспечить компактность и непротиворечивость хранения данных. Основная идея нормализации данных – каждый факт должен хранится в одном месте. Это приводит к тому, что информация, которая моделируется в виде одной стрелки в модели процессов может содержаться в нескольких сущностях и атрибутах в модели данных. Кроме того, на диаграмме модели процессов могут присутствовать различные стрелки, изображающие одни и те же данные, но на разных этапах обработки (например, необработанные детали - обработанные детали - собранное изделие). Информация о таких стрелках находится в одних и тех же сущностях. Следовательно, одной и той же стрелке в модели процессов могут соответствовать несколько сущностей в модели данных и наоборот, одной сущности может соответствовать несколько стрелок.

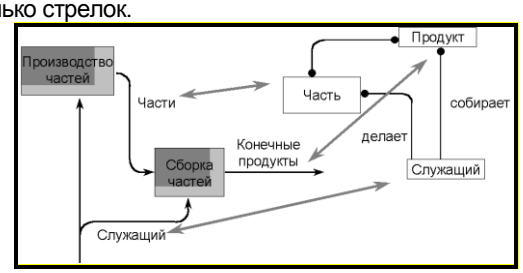


Рис.9. Преобразование стрелки в сущность

Стрелке в модели процессов может соответствовать отдельная сущность в модели данных. Так, стрелке "Части" на рис. 9 соответствует сущность "Часть", стрелке "Конечные продукты" – сущность "Продукт".

Информация о стрелке может содержаться только в нескольких атрибутах сущности. Разным атрибутам одной и той же сущности могут соответствовать разные стрелки. На рис. 10 стрелка "Новая часть" соответствует атрибутам "Номер части" и "Название части", стрелка "Наличное количество" – атрибутам "Количество".

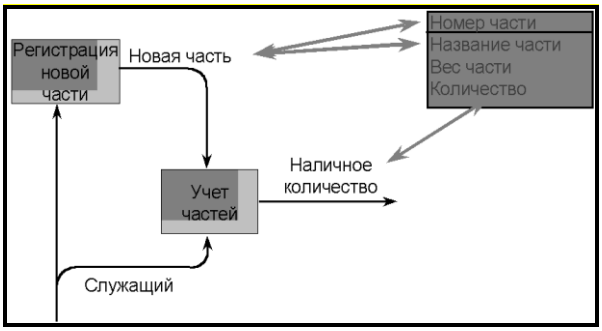


Рис.10. Преобразование стрелки в атрибут

Работы в модели процессов могут создавать или изменять данные, которые соответствуют входящим или выходящим стрелкам. Они могут воздействовать как целиком на сущности (создавая или модифицируя экземпляры сущности, рис. 11), так и на отдельные атрибуты сущности (рис. 12).

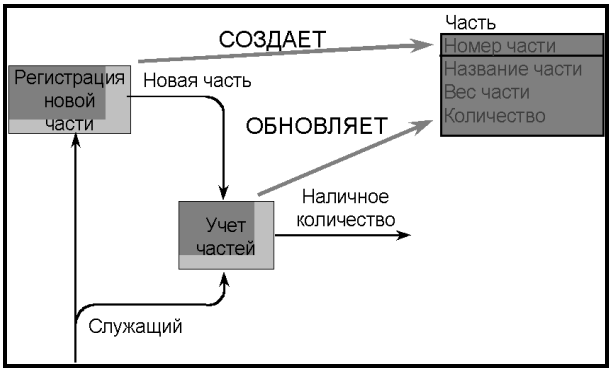


Рис.11. Воздействие работы на сущность

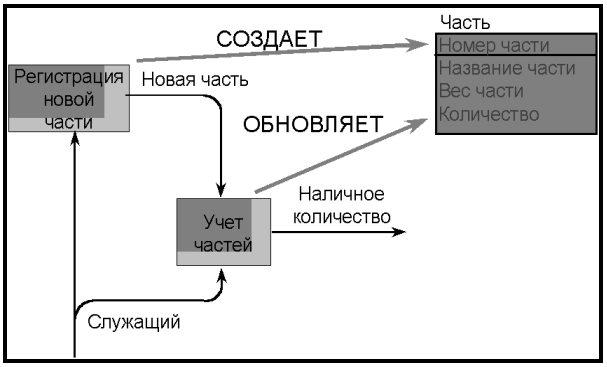


Рис.12. Воздействие работы на атрибут

BPWin позволяет связывать элементы модели данных, созданной с помощью ERWin, документировать влияние работ на данные и, тем самым, позволяет создать спецификации на права доступа к данным для каждого процесса (см. ниже).

2. МОДЕЛИ ДАННЫХ

Создание модели данных с помощью ERWin

Построение модели данных предполагает определение сущностей и атрибутов, то есть необходимо определить какая информация будет храниться в конкретной сущности или атрибуте. Сущность можно определить как объект, событие или концепцию, информация о которых должна сохраняться. Сущности должны иметь наименование с четким смысловым значением, именоваться существительным в единственном числе, не носить "технических" наименований и быть достаточно важными для того, чтобы их моделировать.

ERwin имеет развитый инструмент для облегчения проектирования модели данных. Интерфейс выполнен в стиле Windows-приложений, достаточно прост и интуитивно понятен. В дальнейшем будет описан интерфейс версии 3.5.

Кнопки панели инструментов описаны в табл. 2.

Таблица 2

Кнопки панели инструментов

	Создание, открытие, сохранение и печать модели.
	Вызов диалога Report Browser для генерации отчетов.
	Изменение уровня просмотра модели: уровень сущностей, уровень атрибутов и уровень определений.
	Изменение масштаба просмотра модели.
	Генерация схемы БД, выравнивание схемы с моделью и выбор сервера (доступны только на уровне физической модели)
	Вызов дополнительной панели инструментов для работы с репозиторием Model Mart. (Работа с Model Mart будет рассмотрена в следующем разделе).
	Переключение между областями модели – Subject Area.

Для создания моделей данных в ERwin можно использовать две нотации: IDEF1X и IE (Information Engineering). В примерах будет использоваться нотация IDEF1X.

Для внесения сущности в модель необходимо (убедившись предварительно, что Вы находитесь на уровне логической модели – переключателем между логической и физической моделью служит раскрывающийся список в правой части панели инструментов) кликнуть по кнопке сущности на панели инструментов (ERwin Toolbox)  , затем кликнуть по тому месту на диаграмме, где Вы хотите расположить новую сущность. Кликнув правой кнопкой мыши по сущности и выбрав из всплывающего меню пункт Entity Editor… можно вызвать диалог Entity Editor, в котором определяются имя, описание и комментарии сущности.

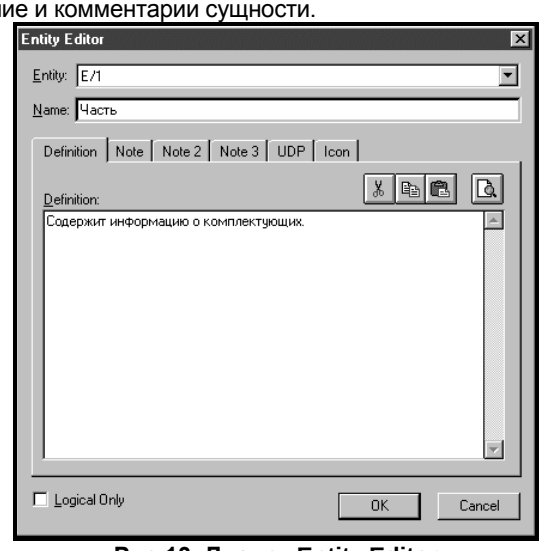


Рис.13. Диалог Entity Editor

Каждая сущность должна быть полностью определена с помощью текстового описания в закладке Definition. Закладки Note, Note2, Note3, UDP (User Defined Properties - Свойства, Определенные Пользователем) служат для

внесения дополнительных комментариев и определений сущности. В закладке Icon каждой сущности можно поставить в соответствие изображение (файл bmp), которое будет отображаться в режиме просмотра модели на уровне иконок (см. ниже).

Каждый атрибут хранит информацию об определенном свойстве сущности. Каждый экземпляр сущности должен быть уникальным. Атрибут или группа атрибутов, которые идентифицируют сущность, называется первичным ключом. Для описания атрибутов следует, кликнув правой кнопкой по сущности, выбрать в появившемся меню пункт Attribute Editor. Появляется диалог Attribute Editor.

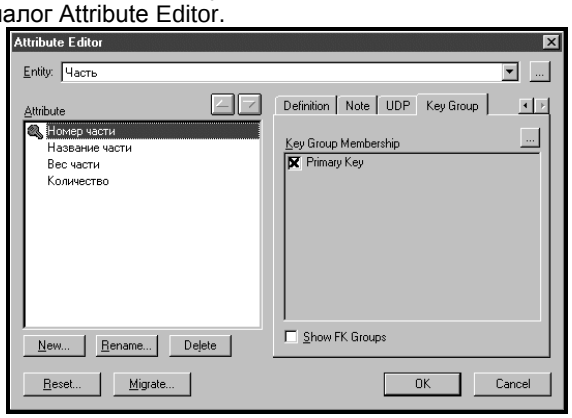


Рис.14. Диалог Attribute Editor

Кликнув по кнопке New…, в появившемся диалоге New Attribute следует указать имя атрибута, имя соответствующей ему колонки и домен. Домен атрибута будет использоваться при определении типа колонки на уровне физической модели [1]. Для атрибутов первичного ключа в закладке Key Group диалога Attribute Editor необходимо сделать пометку в окне выбора Primary Key. При определении первичного ключа может быть рассмотрено несколько наборов атрибутов. Такие наборы называются потенциальными ключами. Например, если рассматривается сущность “Сотрудник”, такими наборами могут быть:

- Имя, Фамилия, Отчество, Дата рождения;
- Номер паспорта;
- Табельный номер;
- Отдел.

К первичным ключам предъявляются определенные требования. Первичный ключ должен однозначно идентифицировать экземпляр сущности (этому требованию не удовлетворяет четвертый ключ, поскольку он может идентифицировать группу сотрудников, работающих в определенном отделе, но не каждого сотрудника). Первичный ключ должен быть компактен, то есть удаление любого атрибута из состава первичного ключа должно приводить к потере уникальности экземпляра сущности (если удалить Дату рождения из первого ключа, то невозможно будет идентифицировать полных тезок). Каждый атрибут из состава первичного ключа не должен принимать NULL – значений (например, если принять в качестве первичного ключа номер паспорта, необходимо быть уверенным, что все сотрудники имеют паспорта). Каждый атрибут первичного ключа не должен менять свое значение в течение всего времени существования экземпляра сущности (сотрудник может сменить фамилию и паспорт, поэтому первый и второй потенциальные ключи не могут стать первичными). Потенциальные ключи, не ставшие первичными, называются альтернативными. Атрибуты, или наборы атрибутов, использующиеся для доступа к группе экземпляров сущности, называются инверсионными ключами. Для описания альтернативных и инверсионных ключей необходимо кликнуть по кнопке … (диалог Attribute Editor, закладка Key Group) и в появившемся диалоге закладка Key Group Editor создать новую ключевую группу (либо инверсионную, либо альтернативную) и указать, какие атрибуты входят в ту или иную группу.

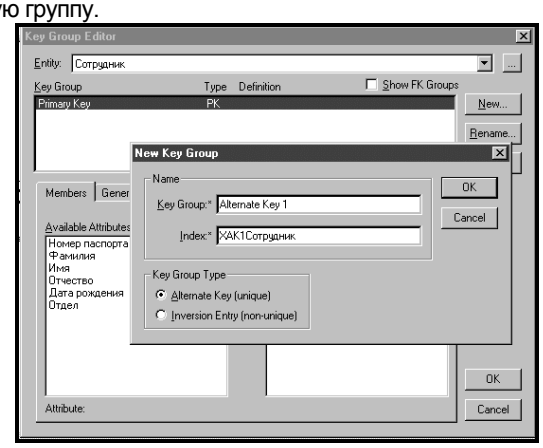


Рис. 15. Диалог Key Group Editor

ERwin имеет несколько уровней отображения диаграммы. Переключиться между ними можно кликнув по любому месту диаграммы, не занятому объектами модели и выбрав в появившемся меню пункт Display Level. В табл. 3 показаны уровни отображения модели.

На уровне атрибутов атрибуты альтернативного ключа помечаются номером (AKm.n), где m – номер ключа, n – номер атрибута в ключе. Инверсионные ключи помечаются номером (IEm.n). В дальнейшем при генерации БД на атрибутах альтернативных ключей могут быть сгенерированы уникальные индексы, на атрибутах инверсионного ключа – неуникальные. Имена индексов задаются в диалоге New Key Group (рис. 15). Атрибуты первичного ключа отображаются выше горизонтальной линии – прочие атрибуты – ниже.

К модели данных предъявляются определенные требования, называемые нормальными формами. Процесс приведения к нормальным формам называется нормализацией. Так, первая нормальная форма требует, чтобы все атрибуты были атомарными (не должно быть атрибута “Адрес” – должны быть атрибуты “Индекс”, “Страна”, “Область”, “Город”, “Улица”, “Дом”, “Квартира”). Вторая нормальная форма требует, чтобы каждый неключевой атрибут зависел от всего первичного ключа, не должно быть зависимости от части ключа. (здесь не ставится целью описание приведения к нормальным формам. Подробно вопросы нормализации освещены, например, в [2]). Для приведения ко второй нормальной форме необходимо создать новую сущность, перенести в нее атрибуты, зависящие от части ключа, сделать часть ключа первичным ключом новой сущности и установить идентифицирующую связь (см. ниже) от новой сущности к старой. Например, в сущности “Служащий” (слева на рис. 16) атрибут “Руководитель” отдела зависит от “Наименования отдела”. Справа изображены сущности, приведенные ко второй нормальной форме. Для установки связи между сущностями нужно воспользоваться кнопками

в палитре инструментов.

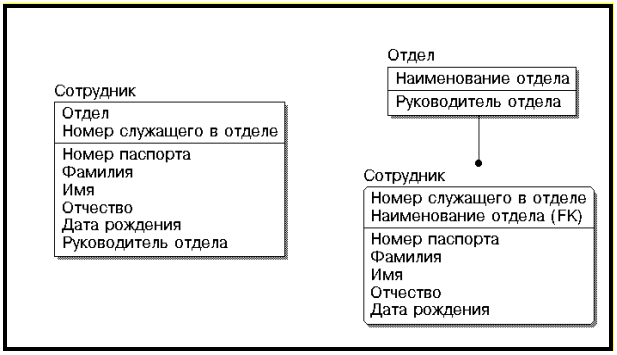
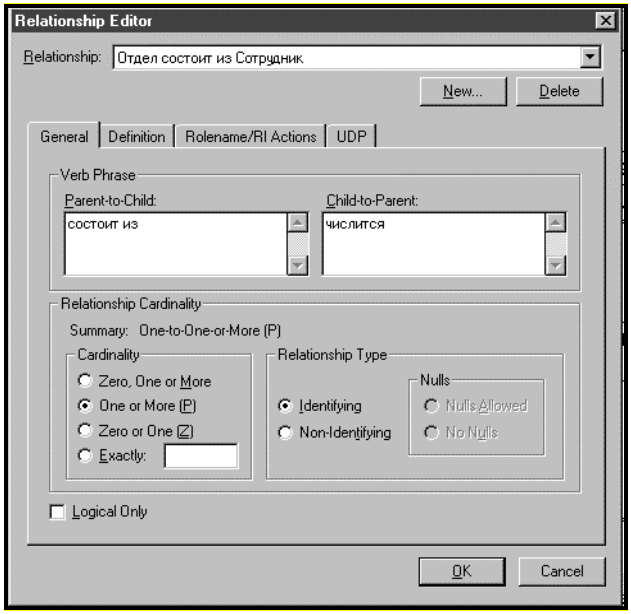


Рис. 16. Иллюстрация второй нормальной формы

На логическом уровне можно установить идентифицирующую связь один ко многим, связь многие ко многим и неидентифицирующую связь один ко многим (соответственно кнопки – слева направо в палитре инструментов).

Идентифицирующая связь устанавливается между независимой (родительский конец связи) и зависимой (дочерний конец связи) сущностями. Зависимая сущность изображается прямоугольником со скругленными углами (сущность "Служащий", справа на рис. 16). Экземпляр зависимой сущности определяется только через отношение к родительской сущности, то есть в структуре на рис.16 информация о служащем не может быть внесена и не имеет смысла без информации об отделе, в котором он работает. При установлении идентифицирующей связи атрибуты первичного ключа родительской сущности переносятся в состав первичного ключа дочерней сущности (миграция атрибутов). В дочерней сущности они помечаются как внешний ключ - (FK). При установлении неидентифицирующей связи дочерняя сущность остается независимой, а атрибуты первичного ключа родительской сущности мигрируют в состав неключевых компонентов родительской сущности.

Для редактирования свойств связи следует кликнуть правой кнопкой мыши по связи и выбрать на контекстном меню пункт Relationship Editor. В появившемся диалоге можно задать:



Мощность (cardinality) связи - служит для обозначения отношения числа экземпляров родительской сущности к числу экземпляров дочерней.

Verb Phrase – фраза, характеризующая отношение между родительской и дочерней сущностями.

Тип связи (идентифицирующая / не идентифицирующая).

Описание связи.

Правила ссылочной целостности (будут сгенерированы при генерации схемы БД).

Имя роли. Имя роли – это синоним атрибута внешнего ключа, которое необходимо, например, при циклической связи. В этом случае нельзя иметь два атрибута с одинаковым именем внутри одной сущности. При задании имени роли атрибут мигрирует в качестве внешнего ключа в состав неключевых атрибутов с именем роли.

Таблица 3

Уровни отображения моделей

Сущности Entity	Атрибуты Attribute	Первичный ключ Primary Key	Определение Definition	Иконки Icon
Часть	Сотрудник Табельный номер Номер паспорта (AK1.1) Фамилия (AK2.1) Имя (AK2.2) Отчество (AK2.3) Дата рождения (AK2.4) Отдел (IE1.1)	Часть Номер части	Часть Содержит информацию о комплектующих.	Часть

При переносе атрибутов внутри и между сущностями можно воспользоваться техникой "drag & drop", выбрав кнопку  в палитре инструментов.

Связь многие ко многим возможна только на уровне логической модели данных. При переходе к физическому уровню ERwin автоматически преобразует связь многие ко многим, добавляя новую, ассоциативную сущность и устанавливая две новые связи один ко многим от старых к новой сущности.

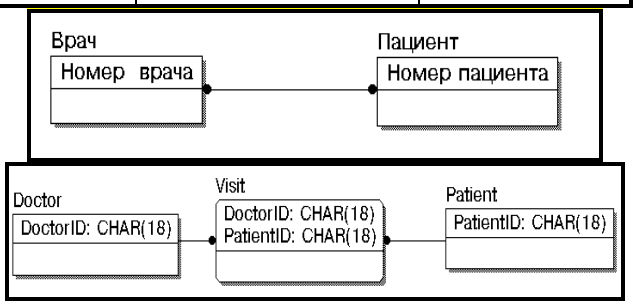


Рис. 17. Иллюстрация разрешения связи многие ко многим

Следует заметить, что автоматического решения проблемы связи многие ко многим не всегда оказывается достаточно. В данном примере один и тот же пациент может много раз посещать врача, поэтому для того, чтобы идентифицировать визит необходимо в состав первичного ключа таблицы “Visit” добавить, например, дату-время посещения.

Иерархия категорий представляет собой особый тип объединения сущностей, которые разделяют общие характеристики. Например, в организации работают служащие, занятые полный рабочий день, совместители и консультанты. Из их общих свойств можно сформировать обобщенную сущность (родовой предок), чтобы представить информацию общую для всех типов служащих. Специфическая для каждого типа информация может быть расположена в категориальных сущностях. Для моделирования категорий служит кнопка  в палитре инструментов. Для каждой категории можно указать дискриминатор – атрибут родового предка, который показывает как отличить одну категориальную сущность от другой (Атрибут “Тип” на рис. 18).

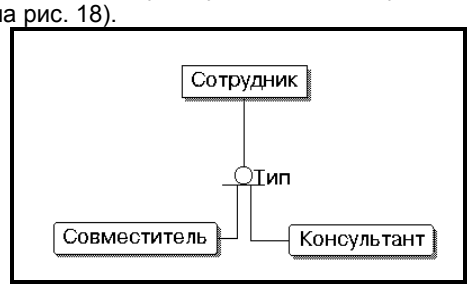


Рис. 18. Иерархия категорий

При создании реальных моделей данных количество сущностей и атрибутов может исчисляться сотнями. Для более удобной работы с большими моделями в ERwin’е предусмотрены предметные области (Subject Area), в которые можно включить тематически общие сущности. Для создания предметных областей нужно вызвать диалог Subject Area Editor (меню Edit/ Subject Area…), в котором указывается имя предметной области и входящие в нее сущности. Все изменения, сделанные в предметной области, автоматически отображаются на общей модели

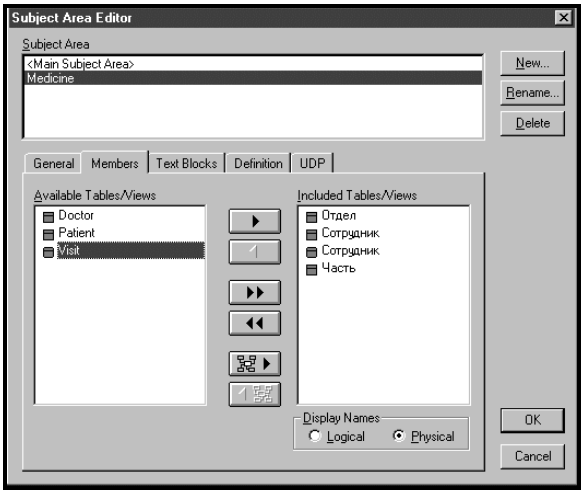


Рис. 19. Диалог Subject Area Editor.

Физический уровень представления модели зависит от выбранного сервера (меню Server/ Target Server..). На физическом уровне модель данных необходимо дополнить такой информацией как учет ограничений ссылочной целостности, хранимые процедуры, триггеры, индексы. Триггеры и хранимые процедуры представляют собой программный код и хранятся на сервере. ERwin обеспечивает мощный инструментарий для создания триггеров: шаблоны и библиотеки макросов. Макросы содержат наиболее часто используемые данные и конструкции. Для редактирования шаблонов триггеров используется редактор Trigger Template Editor (для его вызова следует кликнуть правой кнопкой по таблице и выбрать пункт Trigger в появившемся меню).

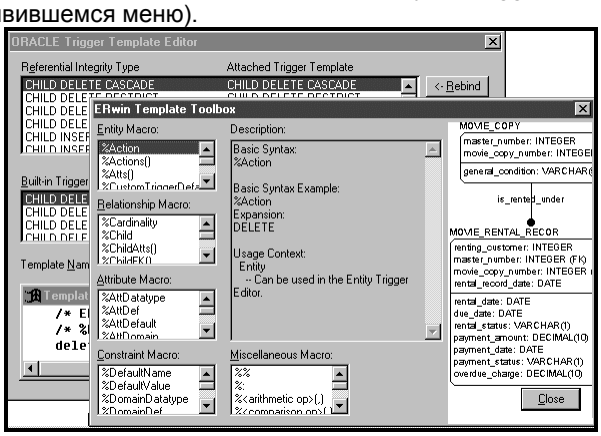


Рис. 20. Диалоги Trigger Template Editor и ERwin Template Toolbox

По умолчанию ERwin генерирует триггеры, дублирующие декларативную ссылочную целостность (опцию можно отменить).

После завершения проектирования модель может быть перенесена в среду целевой СУБД-сервера. Для этого нужно выбрать в главном меню Tasks / Forward Engineer. Можно либо сгенерировать схему БД, либо скрипт на диалекте SQL, соответствующем заранее выбранному серверу. Возможна обратная задача – по существующей схеме БД сгенерировать графическую модель данных. Возможно также выравнивание схемы БД с моделью данных. Для этого следует использовать соответствующую кнопку в панели инструментов (см. таблицу 1). В процессе выравнивания появляется диа-

лог, в котором предлагается указать объекты БД для переноса в графическую модель и объекты модели для переноса в схему БД.

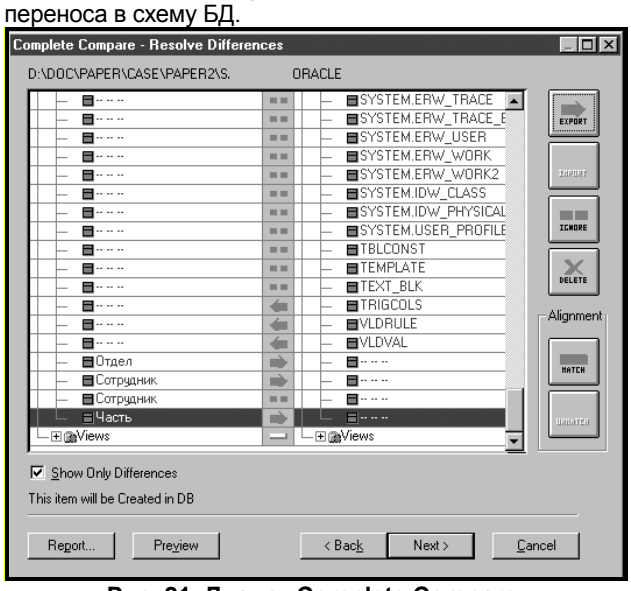
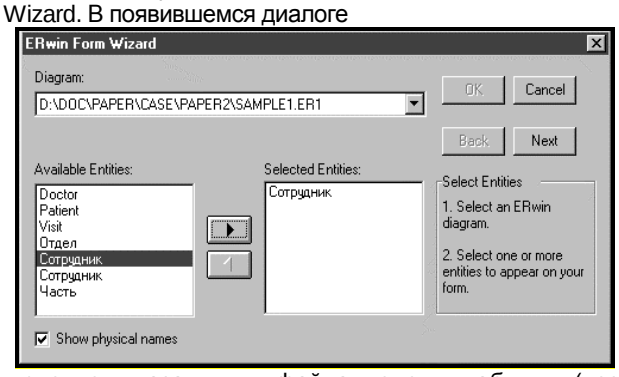


Рис. 21. Диалог Complete Compare-Resolve Differeces

Для генерации клиентской части во-первых, необходимо выбрать язык программирования (меню Client/Target Client…, можно выбрать Visual Basic либо Power Builder; далее в примере используется Visual Basic), затем на физическом уровне для каждой колонки необходимо указать свойства – тип визуального представления (style), код валидации и начальное значение (для задания свойств следует кликнуть по таблице правой кнопкой мыши, в контекстном меню выбрать VB Extended Att). Каждое из этих свойств можно предварительно описать в соответствующем редакторе (кнопка …).

После задания свойств для каждой колонки следует сохранить модель. Техника генерации приложения зависит от конкретной среды программирования. Рассмотрим в качестве примера MS Visual Basic 5.0. В среде Visual Basic 5.0 следует выбрать в меню Add-Ins / ERwin / Form Wizard. В появившемся диалоге



следует указать имя файла модели, таблицы (возможно построение формы по родительской и нескольким дочерним таблицам) и колонки, которые будут отображены в сгенерированной форме. В результате будет сгенерировано приложение, которое может быть откомпилировано и выполнено без дополнительного редактирования.

Связывание модели данных и модели процессов

После разработки модели данных ее следует связать с моделью процессов. Такая связь гарантирует завершенность анализа, гарантирует, что есть источник данных (Сущность) для всех потребностей данных (Работа) и позволяет делить данные между единицами и функциями бизнес-процессов.

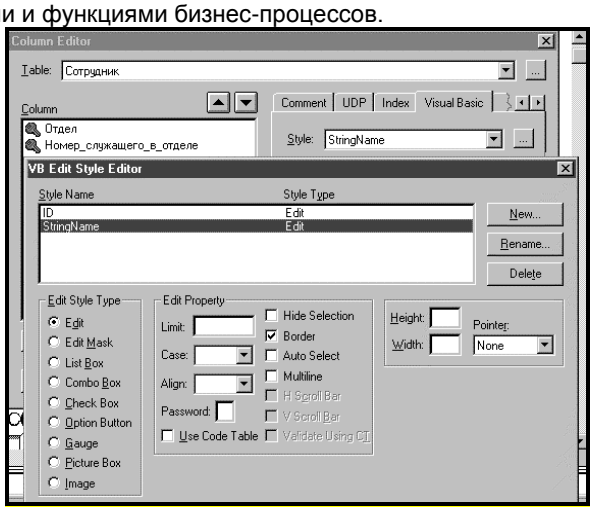
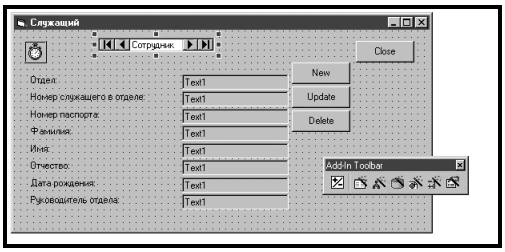
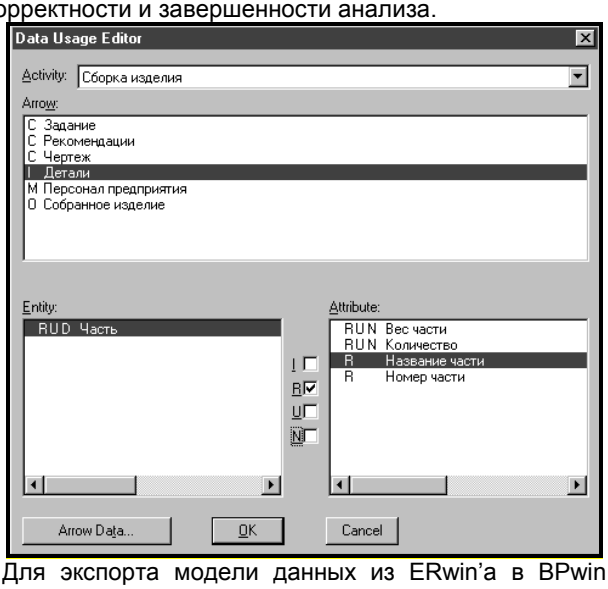


Рис. 22. Редактор стиля

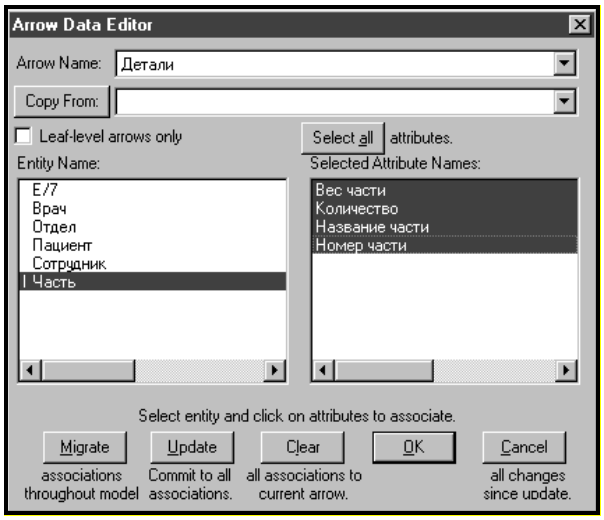


Каждая стрелка в модели процессов может быть связана с несколькими атрибутами различных сущностей. Связи объектов способствуют согласованности, корректности и завершенности анализа.



Для экспорта модели данных из ERwin'a в BPwin необходимо в ERwin'е открыть модель, войти в меню File, выбрать опцию Bpwin / Export, выбрать имя фай-

ла *.eax и нажать OK. Появится сообщение “Export Successful”.



Затем в BPwin’e нужно открыть желаемую модель процесса, выбрать из меню File / Import / Erwin (EAX)..., выбрать имя файла и нажать OK. Появится протокол импорта. Нужно закрыть диалог протокола и в следующем диалоге кликнуть по кнопке Accept Changes. Теперь можно связать сущности и атрибуты со стрелками. Правой кнопкой нужно кликнуть по стрелке и выбрать в контекстном меню Arrow Data. Появляется диалог Arrow Data Editor. В нем необходимо указать сущности и атрибут(ы), связанные со стрелкой и кликнуть по кнопке OK, чтобы сохранить изменения.

Если в процессе связывания стрелок с объектами модели данных окажется, что каких либо сущностей или атрибутов не хватает, их можно добавить (меню Edit / Entity/Attribute Dictionary), а затем экспортировать в ERwin (в BPwin’e меню File / Export / ERwin(BPX), в ERwin’e меню BPwin / Import) .

Как было указано выше, работы могут воздействовать на данные. Для документирования такого воздействия необходимо кликнуть правой кнопкой мыши по желаемой работе и выбрать Data Usage Editor.

В появившемся диалоге Data Usage Editor нужно в верхнем списке кликнуть по имени стрелки, с которой были связаны сущности и атрибуты. В нижнем левом окне появится список связанных сущностей. Если выбрать сущность, то, во-первых, в правом окне появится список соответствующих атрибутов, во- вторых, в центре открываются окна выбора CRUD (Create, Retrieve, Update, Delete). Если кликнуть по атрибуту, то значение окон выбора меняется на IRUN (Insert, Retrieve, Update, Nullify). Ассоциации CRUD и IRUN –это правила использования сущностей и атрибутов работами. Данные не могут использоваться работами произвольно. Стрелки входа представляют данные, которые работа преобразовывает в выход или потребляет. Такие данные могут быть восстановлены (Retrieve), обновлены (Update), удалены (Delete), но не могут быть созданы (Create). Стрелки контроля могут быть только восстановлены (Retrieve) и не могут быть изменены. Стрелки выхода могут быть обновлены (если им соответствуют данные стрелок входа) или созданы (Create).

Arrow Name	Entity Name	C_R_U_D	Attribute Name	I_R_U_N
Детали	Часть	RUD	Вес части	R U N
		RUD	Количество	R U N
		R U D	Название части	R U
		R U D	Номер части	R

Результат связывания объектов модели процессов можно отобразить в отчете Data Usage Report (меню Report / Data Usage Report).

Групповая разработка моделей данных и моделей процессов с помощью Logic Works Model Mart

ModelMart является системой групповой разработки крупных проектов, которая интегрирует инструментальные средства системных аналитиков и разработчиков БД. Одной из проблем, возникающей при многопользовательской работе с моделями, является разграничение прав доступа. Поскольку ModelMart является специализированным хранилищем моделей, помимо разграничения прав доступа на уровне модели возможно регулирование прав на уровне отдельных элементов модели. Для управления правами доступа в состав ModelMart включена утилита ModelMart Security Manager.

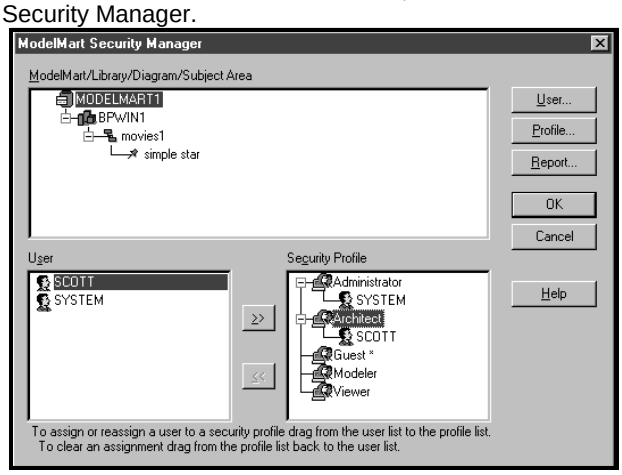
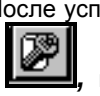


Рис.23 ModelMart Security Manager – диалог формирования групп пользователей

Для начала работы необходимо установить сеанс связи с ModelMart, нажав кнопку  (из дополнительной линейки инструментов ModelMart среды ERWin’a или BPWin’a) и набрать имя и пароль пользователя в появившемся диалоге. После успешного входа становится доступной кнопка  , которая вызывает диалог Security Manager.

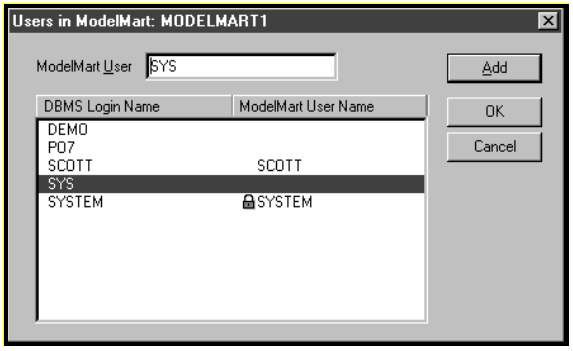


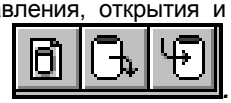
Рис.24. Users in ModelMart– диалог внесения новых пользователей

В окне диалога Security Manager можно задать группу пользователей и права каждой группы (кнопка Profile…) на создание, редактирование и удаление библиотек и моделей. В случае работы с моделями данных в ERWin'е можно также задать права на создание, редактирование и удаление для предметных областей (Subject Area) и сущностей. Нажав на кнопку Users…, можно вызвать диалог внесения новых пользователей ModelMart, которых необходимо предварительно завести как пользователей БД (в примере – Oracle).

Не все пользователи БД могут быть пользователями ModelMart, но все пользователи ModelMart должны быть пользователями БД.

Затем можно внести пользователей ModelMart в ту или иную группу пользователей ModelMart. На рис.1 показано, что пользователь SYSTEM внесен в группу Administrator, пользователь SCOTT- в группу Architect. Следовательно, если в течение жизненного цикла разработки проекта роль проектировщика меняется, администратор ModelMart может соответственно менять права доступа без изменения его прав как пользователя БД, что дает возможность гибкого управления проектами.

Рис.25. ModelMart Security Profile Manager – диалог задания прав группам пользователей

Если пользователь имеет соответствующие привилегии, он может создать библиотеку моделей ModelMart, нажав кнопку  . В состав библиотеки могут входить модели процессов BPWin'а или модели данных и отдельные предметные области моделей данных ERWin'а (диалог создания предметных областей вызывается кнопкой ) . Принцип работы с библиотеками моделей является весьма полезным при работе с большим количеством моделей, поскольку можно формировать библиотеки готовых решений (как для моделей процессов, так и для моделей данных) и добавлять в новый проект модели-блоки из заранее сформированных библиотек. Для создания новой модели в ModelMart , добавления, открытия и сохранения модели служат кнопки  . Создать или сохранить модель можно только в составе какой-либо библиотеки. При открытии модели возникает диалог Open ModelMart Diagramm, в котором можно указать опции блокировки модели.

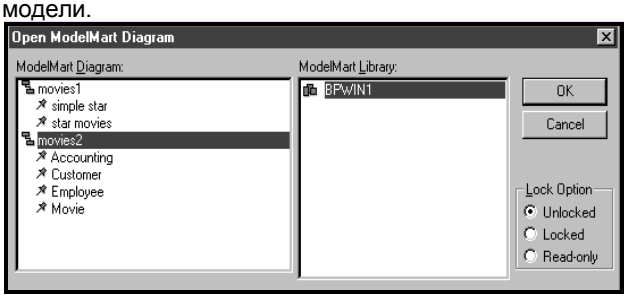


Рис.26. Диалог Open ModelMart Diagramm

Открытие модели в режиме Read Only означает, что измененную модель нельзя будет сохранить в репозитории. В режиме Locked модель блокируется и другие пользователи не смогут изменить модель. В режиме Unlocked все пользователи могут открыть и изменить модель, при попытке сохранить модель, измененную и сохраненную другим пользователем во время сеанса работы возникает диалог - Intelligent Conflict Resolution, показывающий различия текущей и имеющейся в репозитории моделей. Открытую модель можно перевести в режим Locked, нажав кнопку  .

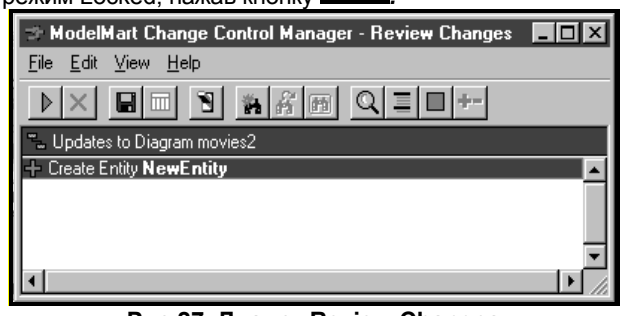


Рис.27. Диалог Review Changes

Кнопка  вызывает диалог ModelMart Merge Manager, который служит для слияния моделей. В зависимости от настройки, слияние может быть проведено в одну из существующих, либо во вновь создаваемую диа-

грамму. Обновление загруженной диаграммы можно осуществить кликнув по кнопке  .

Список изменений, сделанных в процессе работы с моделью, показывается в диалоге Review Changes (вызывается кнопкой ).

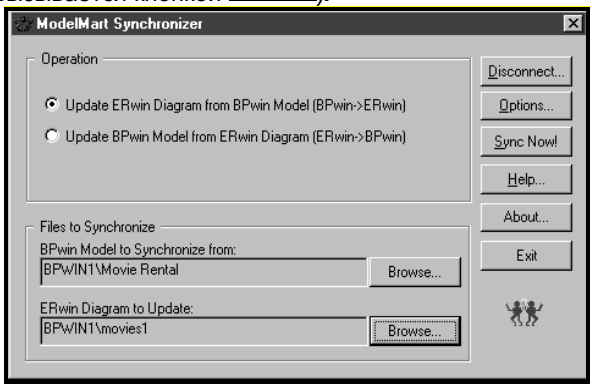


Рис.28. ModelMart Synchronizer.

Хранимые в репозитории модели можно сравнивать (кнопка ). В диалоге Version Manager следует выбрать сравниваемые версии и кликнуть по кнопке Diff. В появившемся диалоге Version Differences отображается список отличий версий.

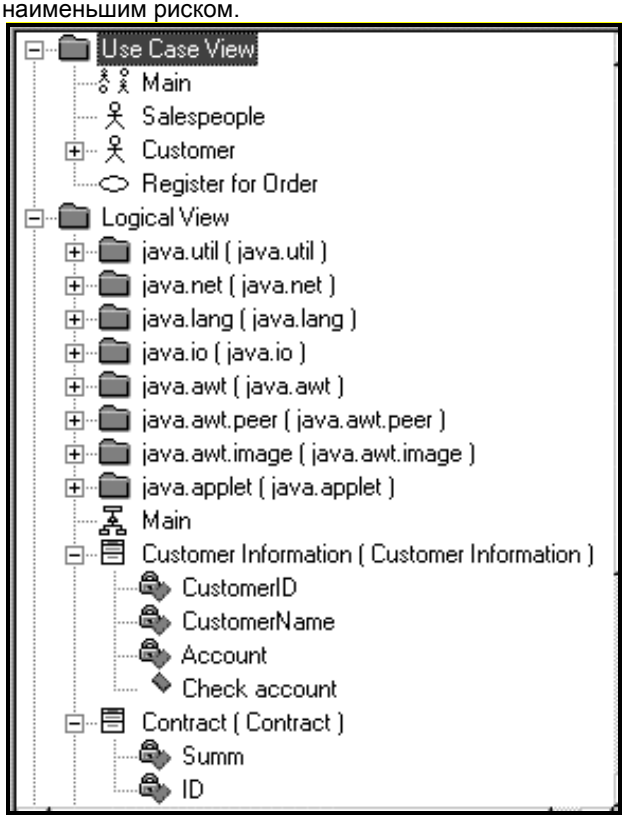
В репозитории ModelMart реализована функциональность синхронизации моделей процессов и моделей данных, которая ранее была реализована в ERWin'е и BPWin'е путем экспорта и импорта через файлы .bpx - .eax (см. предыдущий раздел). Для синхронизации моделей необходимо кликнуть по кнопке  .

В диалоге ModelMart Synchronizer следует указать хранящиеся в репозитории модели процессов и данных, указать направление синхронизации и запустить процесс синхронизации. Затем можно работать с синхронизированными моделями процессов и данных так же, как было выше.

Создание объектной модели с помощью Rational Rose

Классический структурный подход к созданию информационных систем предполагает последовательную реализацию этапов анализа, проектирования, создания модулей, объединения модулей в единую систему, тестирования и внедрения. Применение CASE-технологий и CASE – средств, подобных описанным выше ERWin"у и BPWin"у, позволяет в несколько раз сократить время разработки информационных систем и значительно снизить вероятность появления ошибок за счет автоматизации начальных этапов разработки (а, как следствие- более качественного планирования и проектирования) и автоматической генерации структуры сервера БД и кода клиентского приложения. Однако, эта технология не лишена недостатков. Код клиентского приложения генерируется на основе информации о структуре БД (см. предыдущий раздел). К структуре БД предъявляются определенные требования (нормализации), в результате чего данные хранятся в таблицах БД не всегда в той же форме, в которой они должны представляться на экранных формах. Другими словами, если код приложения генерируется не на основе описания предметной области, невозможно построить эффективное приложение со сложной бизнес-логикой. Кроме того, при структурном подходе к разработке информационных систем, риск остается большим на всех этапах создания системы вплоть до этапа тестирования, когда мы можем обнаружить допущенные ошибки и оценить состоятельность системы. В случае обнаружения ошибки, необходимо вернутся на тот этап разработки, на котором допущена ошибка, и заново пройти последующие этапы.

Альтернативой структурному подходу стали лишенные перечисленных недостатков объектно-ориентированные методы разработки информационных систем. В начале девяностых годов был предложен разработанный на основе наиболее популярных объектных методов - OMT (Rumbaudh), Booch и OOSE (Jacobsom) универсальный язык объектного проектирования – Unified Modeling Language, UML (The Unified Method, Draft Edition (0.8). Rational Software Corporation, October 1995). Одним из CASE-средств, поддерживающих язык UML, является выпущенный фирмой Rational Software программный пакет Rational Rose, который позволяет генерировать код приложения, в полной мере отвечающий бизнес-правилам и с наименьшим риском.



Снижение риска в объектной технологии достигается за счет реализации технологии итерационной разработки (так называемая спиральная модель жизненного цикла разработки). Разработка состоит из ряда итераций, которые в дальнейшем приводят к созданию информационной системы. Каждая итерация может приводить к созданию фрагмента или новой версии и включает этапы выработки требований, анализа, проектирования, реализа-

ции и тестирования. Поскольку тестирование проводится на каждой итерации, риск снижается уже на начальных этапах жизненного цикла разработки.

Модель представляет собой совокупность диаграмм, описывающих различные аспекты структуры и поведения информационной системы. Для просмотра модели в Rational Rose используется иерархический навигатор модели – Browser. В дальнейшем будет описан интерфейс версии Rational Rose for Java (version 4.0).

Рассмотрим в общих чертах некоторые диаграммы UML.

Диаграммы использования системы (Use Cases) показывают, какая функциональность должна быть реализована в системе, основные функции, которые должны быть включены в систему (use case), их окружение (actors) и взаимодействие функций с окружением. Воздействующие объекты (actors) не являются частью системы – это конечные пользователи или другие программы, взаимодействующие с проектируемой информационной системой. Функциональность (use case) - последовательность действий, выполняемых системой, которые приводят к определенным результатам, необходимым для конкретного воздействующего объекта.

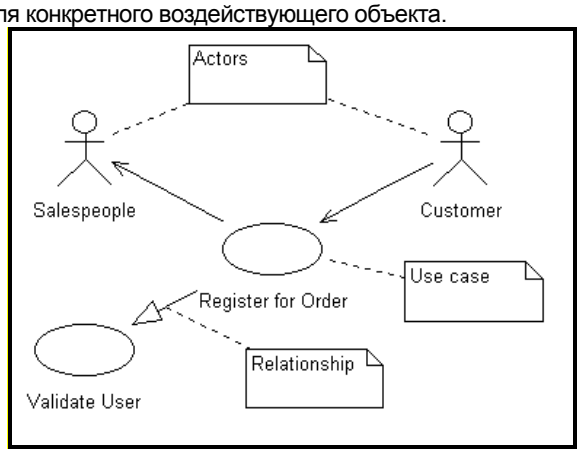


Рис.29. Диаграмма Use Cases. Здесь Customer, Salespeople – Actors; Register for Order, Validate User – Use case

Диаграммы Use Cases включают отношения и ассоциации, показывающие взаимодействие между воздействующими объектами и функциями (изображаются в виде стрелок) и примечания (note), которые могут быть привязаны к любому объекту диаграммы Use Cases. Для создания новой диаграммы Use Cases следует правой кнопкой мыши кликнуть в навигаторе модели по закладке Use Case View и выбрать во всплывающем меню пункты New / Use Case. Для внесения в диаграмму Use Case и установления связей между ними следует использовать кнопки палитры инструментов Rational Rose.

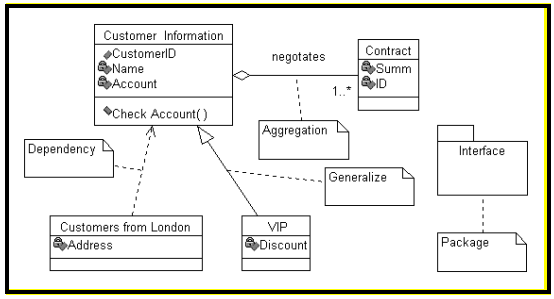


Рис.30. Диаграмма классов

Диаграммы классов. Под объектом в UML понимается некоторое абстрактное представление конкретного объекта предметной области. Каждый объект имеет состояние, поведение и индивидуальность. Например, объект "Проект" может иметь два состояния – "открыт" и "закрыт". Поведение объекта определяет, как объект взаимодействует с другими объектами. Индивидуальность означает, что каждый объект уникален и отличается от других объектов. Под классом понимается описание объектов, обладающих общими свойствами (атрибутами), поведением, общими взаимоотношениями с другими объектами и общей семантикой. Класс является шаблоном для создания новых объектов. Для внесения нового класса в диаграмму классов нужно использовать кнопку  в палитре инструментов.

Если система содержит большое количество классов, они могут быть объединены в пакеты (package).

Каждый класс может иметь атрибуты (свойства). Так, на рис. 30 класс Customer Information (информация о клиенте) имеет атрибуты CustomerID (идентификатор клиента), Name (имя) и Account (счет). Кроме того, каждый класс может методы (operations) – некоторые действия, которые описывают поведение объектов класса. На рис. 30 класс Customer Information имеет метод Check Account. Для внесения свойств класса следует правой кнопкой мыши кликнуть по классу и выбрать во всплывающем меню пункт Specification.

Классы могут иметь взаимосвязи (relationship), называемые отношениями. В нотации UML имеется несколько типов отношений. Отношение использования (associations, кнопка  палитры инструментов) показывает, что объект одного класса связан с одним или несколькими объектами другого класса. Отношение включения (aggregation, кнопка ) является частным случаем отношения использования. Оно показывает, что один объект является частью другого. При воздействии на один объект, связанный отношением включения, некоторые операции автоматически могут затронуть другой объект. Например, на рис. 30 класс Customer Information связан отношением включения с классом Contract. При удалении объекта класса Customer Information (информация о клиенте) должны удаляться все объекты класса Contract (относящиеся к данному клиенту контракты). Каждая связь может быть охарактеризована определенной фразой, называемой именем роли. Связь между классами Customer Information и Contract имеет имя negotiates. Каждая связь может иметь индикатор множественности, который показывает, сколько объектов одного

класса соответствует объекту другого класса. На рис. 30 связь negotiates имеет индикатор 1..* (один или много).

Наследование (inheritance) описывает взаимосвязь между классами, когда один класс (называется подклассом, subclass) наследует структуру и/или поведение одного или нескольких классов. На рис. 30 подкласс VIP наследует свойства и поведение класса Customer Information. Связь классов в иерархии наследования называется отношением наследования (generalization, кнопка ).

Временная диаграмма (Sequences) демонстрирует поведение объектов во времени. Она показывает объекты и последовательность сообщений, посылаемых объектами. Сообщения на диаграмме сценариев изображаются в виде стрелок.

Архитектура приложения описывается в диаграммах компонент (Component Diagram), которые описывают вхождение классов и объектов в программные компоненты системы (модули, библиотеки и т.д.) и диаграммы развертывания (Deployment Diagram), при помощи которых документируется размещение программных модулей на узлах (физических и логических устройствах) системы. Здесь эти диаграммы не рассматриваются.

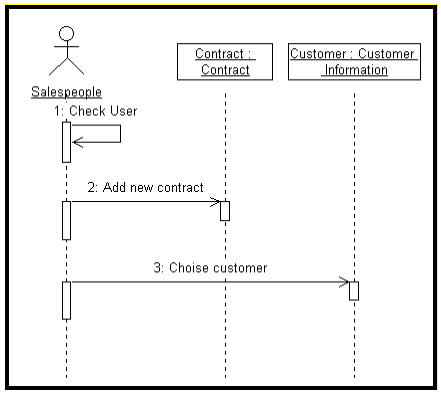


Рис.31. Временная (*Sequence*)диаграмма

Генерация кода осуществляется на основе диаграмм классов. Для генерации необходимо выбрать пункт меню Tools / Java / Generate Java.

Ниже приведен код на языке Java, соответствующий классу Customer Information:

```
  //## //## Source file: Customer__Information.java
  //## //## Subsystem: Component View
  //## //## Module: Customer Information
  //##begin module.cm preserve=no
  /* %X% %Q% %Z% %W% */
  //##end module.cm
  //##begin module.cp preserve=no
  //##end module.cp
  //##begin module.additionalImports preserve=no
  //##end module.additionalImports
  //##begin module.imports preserve=yes
  //##end module.imports
  //====================================
  //##begin module.declarations preserve=no
  //##end module.declarations
  //##begin module.additionalDeclarations preserve=yes
  //##end module.additionalDeclarations
  public class Customer__Information {
   //##begin Customer__Information.initialDeclarations preserve=yes
   //##end Customer__Information.initialDeclarations
   public int m_CustomerID;
   private int m_Name;
   private int m_Account;
   public Vector m_negotates = new Vector();
   public void Check_Account() {
   //##begin Customer__Information::Check Account%3561A0AF032A.body preserve=yes
   //##end Customer__Information::Check Account%3561A0AF032A.body
   }
   //##begin Customer__Information.additionalDeclarations preserve=yes
   //##end Customer__Information.additionalDeclarations
  }
```

При генерации кода Rational Rose включает строки комментария, начинающиеся последовательностью символов “//##” . Сгенерированный код (в отличие от кода, сгенерированного ERWin) не является готовым приложением. Здесь генерируются лишь заголовки методов (Check_Account), сами методы необходимо дописывать вручную.

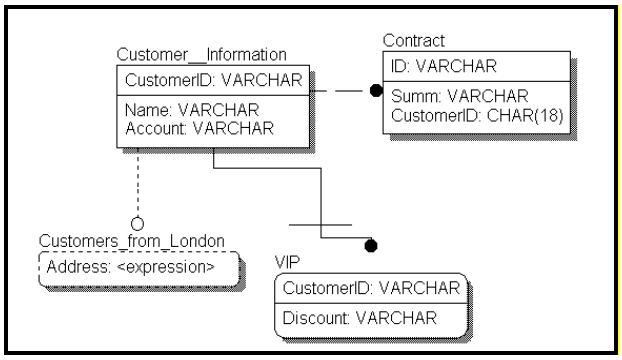


Рис. 32. Модель данных

Создание модели данных на основе объектной модели с помощью ERWin Translation Wizard

Rational Rose позволяет строить объектную модель, но не может построить модель данных или сгенерировать системный каталог сервера БД. Для решения этой задачи фирмой Logic Works выпущена утилита ERWin Translation Wizard, позволяющая перегрузить объектную модель в ERWin и автоматически получить на ее основе модель данных. После инсталляции ERWin Translation Wizard вызывается из среды Rational Rose. Для того, чтобы классы могли быть конвертированы в сущности модели данных, они должны быть определены как Persistent. Для этого необходимо правой кнопкой мыши кликнуть по классу, выбрать во всплывающем меню Specifications… / Detail / Persistence . ERWin Translation Wizard позволяет как сгенерировать диаграмму классов на основе модели данных, так и модель данных на основе диаграммы классов. На рис. 32 показана модель данных (физический взгляд), полученная на основе диаграммы классов, представленной на рис. 30. Модель данных может быть использована для генерирования системного каталога сервера БД (см. предыдущий раздел).

В таблице показано соответствие между объектами диаграммы классов и объектами модели данных при перегрузке моделей из Rational Rose в ERWin и обратно.

Объекты диаграммы классов	Объекты модели данных
Класс (Class)	Сущность, таблица (Entity, Table)
Атрибут класса (Attribute)	Атрибут сущности, колонка (Attribute, Column)
Отношение использования (association) Отношение включения (aggregation)	Неидентифицирующая связь (Non-identifying relationship)
Отношение наследования (generalization)	Иерархия подкатегорий, полная подкатегория (Complete sub-category)
Имя роли (Role name)	Наименование связи (Verb phrases)
Индикатор множественности (multiplicity indicators)	Мощность связи (Cardinality)
Класс – клиент в отношении зависимости (Dependency relationship - Client)	Временная таблица (View)
Отношение зависимости (Dependency)	View relationship

Литература

1. Маклаков С.В. Erwin расширяет свои возможности. - Компьютер пресс, 3, 1998
2. Дейта К. Дж. Введение в системы баз данных/ -Киев^ Диалектика, 1998

Маклаков Сергей