Областное государственное бюджетное
образовательное учреждение
среднего профессионального образования
«Иркутский авиационный техникум»

УТВЕРЖДАЮ
Директор ОГБОУ СПО «ИАТ»
_____ В.Г. Семенов

Комплект методический указаний по выполнению практических работ по междисциплинарному курсу МДК.03.02 Инструментальные средства разработки программного обеспечения

образовательной программы (ОП)
по специальности СПО

230115 Программирование в компьютерных системах

базовой подготовки

Иркутск 2013

Перечень практических (лабораторных) работ

№ работы	Название работы (в соответствии с рабочей программой)	Объём часов на выполнение работы	Страница
1.	BPwin. Инструментальная среда BPwin		
2.	BPwin. Изучение объектов диаграмм функциональной модели в стандарте IDEF0		
3.	BPwin. Постановка задачи. Создание контекстной диаграммы для данной задачи		
4.	BPwin. Создание диаграммы декомпозиции		
5.	BPwin. Создание диаграммы узлов. Создание диаграммы FEO. Каркас диаграммы		
6.	BPwin. Создание отчетов в пакете BPwin		
7.	BPwin. Расщепление и слияние моделей		
8.	BPwin. Создание диаграммы IDEF3		
9.	BPwin. Стоимостной анализ		
10.	Использование категорий UDP		
11.	BPwin. Расщепление модели. Слияние расщепленной модели с исходной. Копирование работ		
12.	BPwin. Создание модели TO-BE		
13.	BPwin. Создание диаграммы DFD		
14.	ERwin. Создание сущностей и атрибутов на диаграмме		
15.	ERwin. Создание связей между сущностями		
16.	ERwin. Индексирование		
17.	ERwin. Иерархия наследования		
18.	ERwin. Подмножества модели и хранимые отображения		
19.	ERwin. Установка цвета и шрифта, создание графических объектов на диаграмме		
20.	ERwin. Экспорт модели данных ERwin в модель процессов BPwin		
21.	ERwin. Выбор сервера		

22.	ERwin**.** Представления		
23.	ERwin**.** Правила валидации и значения по умолчанию		
24.	ERwin**.** Вычисление размера БД		
25.	ERwin. Прямое проектирование		
26.	UML. Методология объектно-ориентированного моделирования		
27.	UML. Методика построения диаграмм классов		
28.	UML. Генерация программного кода		
29.	UML. Отладка и тестирование разработанного программного средства		
30.	UML. Варианты заданий		

ПРАКТИЧЕСКАЯ РАБОТА № 1

Инструментальная среда Bpwin

Цель работы:

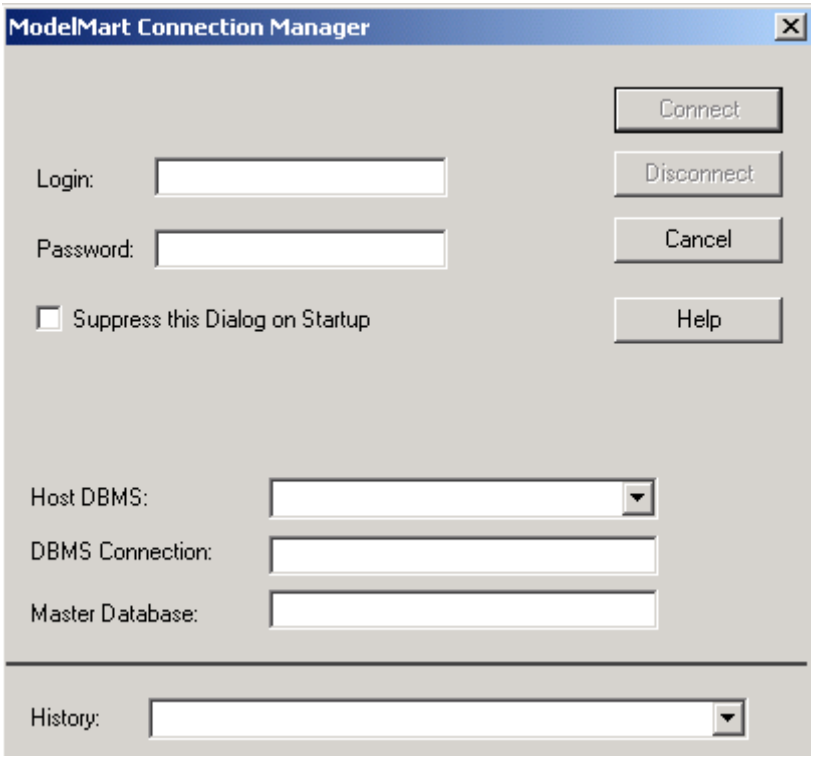
1. Овладение навыками работы в BPwin.
2. Освоение принципов построения основных элементов структурной диаграммы в методологии IDEF0

Исходные данные (задание):

Начало работы в BPwin

Стандарт IDEF0.

После запуска программы на экране появляется диалоговое окно, в котором предлагается продолжить дальнейшую работу без средства групповой разработки крупных проектов **ModelMart**:

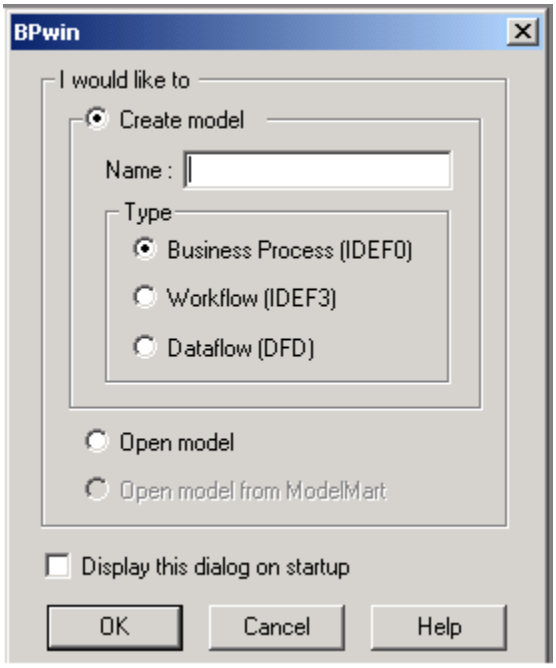


Для продолжения работы необходимо нажать клавишу **«Cancel»,** так как пакет **ModelMart** не установлен.

ModelMart –предназначен для коллективной разработки функциональных моделей.

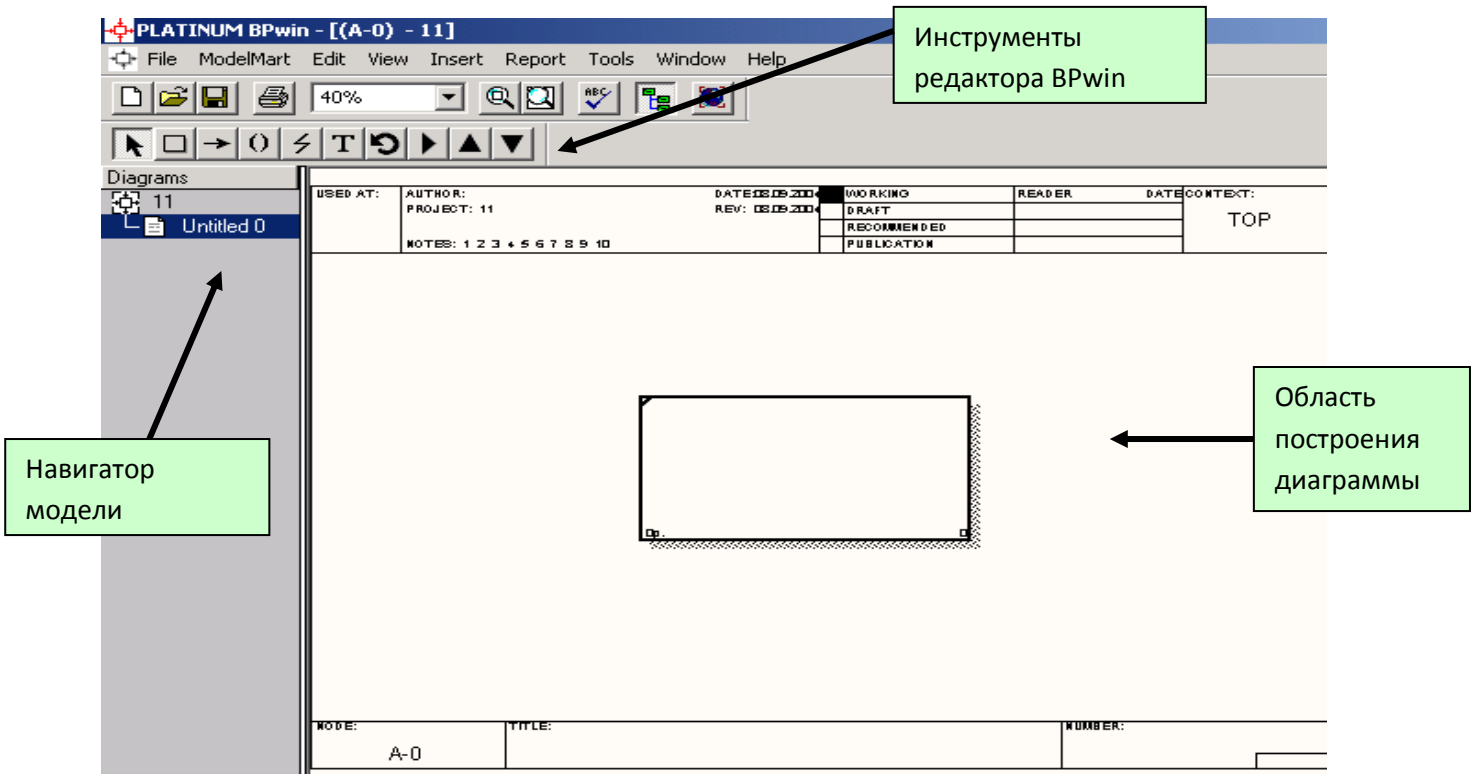
Дальше появится окно «I would like to». в котором в поле **Name** необходимо указать **Имя модели**, например, «Изготовление изделия»

Если необходимо создать новую модель, то нужно выполнить **File/New, если нужно** открыть существующую модель, то **File/Open**.



Задание №1. В поле **Name:** введите имя модели: **«Изготовление изделия».** Из группы **Type** выберите тип диаграммы **Business Process (IDEF0)** и нажмите ОК.

При первом открытии программы (при создании новой модели) область построения содержит контекстную диаграмму А-0.

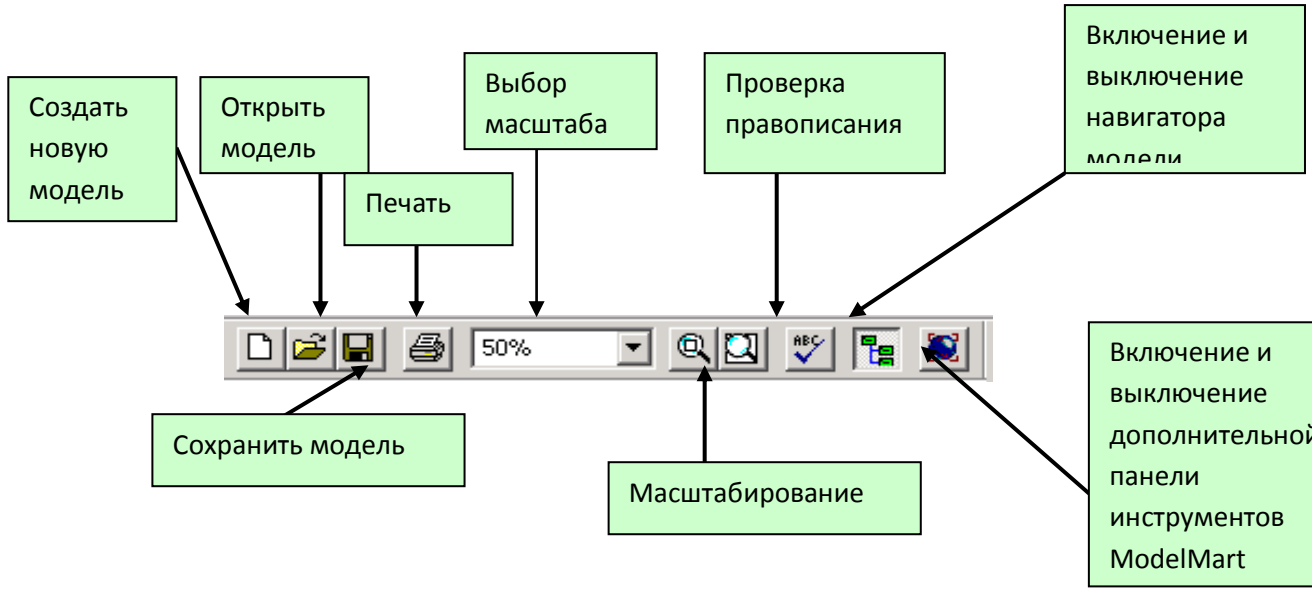


Основные инструменты

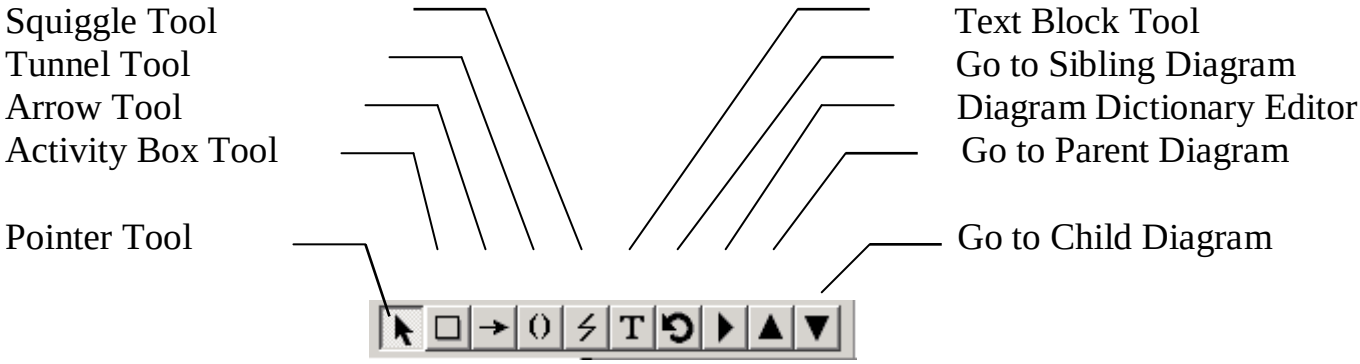
Все основные действия с диаграммами, такие как создание, редактирование и т.д, можно выполнить либо с помощью главного меню:

либо контекстно-зависимого меню (меню, появляющееся при нажатии правой кнопки мыши). Принципы работы с меню являются стандартными для среды Windows: объект сначала делается активным, затем над ним осуществляются необходимые действия.

На основной панели инструментов расположены элементы управления, в основном знакомые по другим Windows-интерфейсам



На основной панели инструментов (либо в любом желаемом месте экрана) расположены инструменты редактора BPwin для IDEF0 –диаграмм:



Режим редактирования (Pointer Tool) – используется для выбора и определения позиции объектов, добавленных в диаграмму.

Создание дуг (Arrow Tool) – используется для установки дуг.

Установка тоннеля для дуги (Tunnel Tool) – используется, чтобы ставить tunneled arrow (дуги со специальной нотацией, т.е. они не удовлетворяют обязательному условию, что дуги в родительских и диаграммах потомках соответствуют друг другу).

Создание метки дуги (Squiggle Tool) – используется для создания тильды  , которая соединяет дугу с её названием.

Создание текстового блока (Text Block Tool) – используется для создания текстовых блоков.

Переход на другую диаграмму того же уровня (Go to Sibling Diagram) – используется для отображения следующей диаграммы того же уровня.

Редактор словаря диаграмм (Diagram Dictionary Editor) – открывает диалоговое окно Diagram Dictionary Editor, где можно перейти на какую-либо диаграмму или создать новую диаграмму.

Переход на родительскую диаграмму (Go to Parent Diagram) - переход на родительскую диаграмму

Переход к диаграмме –потомку (Go to Child Diagram) – используется, чтобы отобразить диаграмму потомка или разложить выделенный блок на диаграмму потомка.

Установка цвета и шрифта объектов.

BPwin позволяет задавать параметры шрифтов, используемых для описания элементов моделей, а также их цвет. Для этого необходимо щелкнуть правой клавишей мыши по объекту и выбрать команду **Font Editor** для установки нужного шрифта или **Font Color** для установки нужного цвета объекта.

Model Eplorer – навигатор модели процессов

Model Eplorer - мощный инструмент, который используется для просмотра структуры модели и изменения любых объектов диаграмм в любой открытов модели BPwin.

Инструмент навигации **Model Eplorer** имеет две вкладки – **Activities**, **Diagrams, Objiect**.

Вкладка **Activities** показывает в виде раскрывающегося иерархического списка все работы модели. Щелчок по вкладке Activities переключает левое окно на диаграмму, на которой эта работа размещена. Для редактирования свойств работы следует щелкнуть по ней правой кнопкой мыши. Появится контекстное меню. В таблице 1 приведены значения пунктов меню.

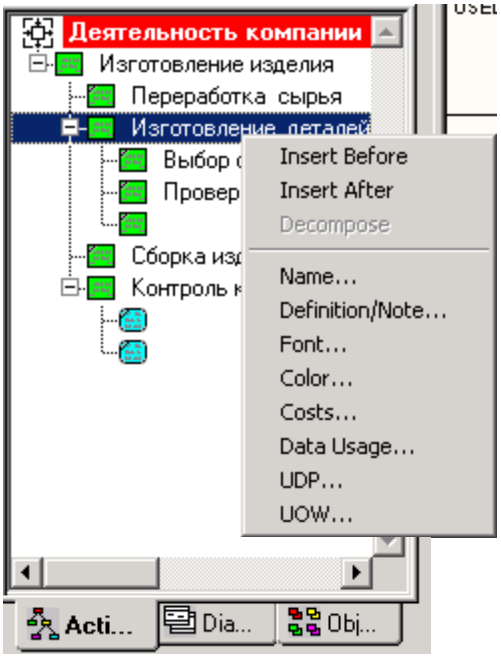


Таблица 1. Контекстное меню редактирования свойств работы

Пункт меню	Описание
Insert Before	Создать новую работу на той же самой диаграмме. Новая работа будет вставлена перед текущей.
Insert After	Создать новую работу на той же самой диаграмме. Новая работа

	будет вставлена после текущей.
Decompose	Декомпозировать работу. В результате будет создана новая диаграмма декомпозиции
Name	Вызов редактора имени работы
Definition/Note	Вызов редактора определения и примечания к работе
Font	Изменение шрифта работы
Color	Изменение цвета работы
Costs	Задание стоимости работы
Data Usage	Ассоциация работы с данными
UDP	Задание свойств, определяемых пользователем
UOW	Задание свойств для работ

Вкладка **Diagrams** служит для перехода на любую диаграмму модели.

Задание №2. Установите курсор на прямоугольник и введите текст (Изготовление изделия) в поле диаграммы, вызвав контекстное меню и выбрав команду Name.
Задание №3. Задайте параметры цветов и их шрифт.
Задание №4. Включите и выключите навигатор модели.

Контрольные вопросы:

1. Как начать работу в программе BPwin?.
2. Как создать новую модель?
3. Как открыть существующую модель?
4. Как указать Имя модели при создании новой модели?
5. Где находится область построения диаграммы?
6. Как включить или выключить навигатор модели?
7. Как сохранить модель?
8. Как установить шрифт объекта?
9. Как установить цвет объекта?
10. Для чего нужен навигатор модели?
11. Какие вкладки имеет навигатор модели?
12. Какую диаграмму содержит область построения диаграммы при создании новой модели?

Критерии оценки работы:
1. Полный ответ – 5 баллов.
2. Дополнительный уточняющий вопрос – 4 балла.
3. Краткий ответ – 3 балла.

ПРАКТИЧЕСКАЯ РАБОТА № 2

Изучение объектов диаграмм функциональной модели в стандарте IDEF0.

Цель работы:
Освоение принципов построения основных элементов структурной диаграммы в методологии IDEF0.
Задание №1. В поле **Name:** введите имя модели: **«Изготовление изделия».** Из группы **Type** выберите тип диаграммы **Business Process (IDEF0)** и нажмите ОК.
Задание №2. Прочитайте: Принципы построения модели IDEF0. Цель моделирования (Purpose). Точка зрения (Viewpoint). Модели AS-IS и TO-BE. Диаграммы IDEF0 Работа (Activity).
Задание №3. Построить Контекстную диаграмму «Производить изделия» по приведенному примеру.
Задание №4. По приведенному примеру построить диаграмму декомпозиции

Исходные данные (задание):

Принципы построения модели IDEF0.

На начальных этапах создания информационной системы необходимо понять, как работает организация, которую собираются автоматизировать. Для описания работы предприятия необходимо построить модель. Такая модель должна быть адекватна предметной области; то есть она должна содержать в себе знания всех участников бизнес-процессов организации.

Наиболее удобным языком моделирования бизнес-процессов является IDEF0, предложенный более 20 лет назад Дугласом Россом и называвшийся первоначально SADT(Система структурного анализа и проектирования).

В IDEF0 система представляется как совокупность взаимодействующих работ или функций.

Цель моделирования (Purpose)
Точка зрения (Viewpoint)

Построение модели системы должно начинаться с изучения всех документов, описывающих её функциональные возможности. Одним из таких документов является техническое задание, а именно разделы «Назначение разработки», «Цели и задачи системы» и «Функциональные характеристики системы». После изучения исходных документов и опроса заказчиков и пользователей системы необходимо сформулировать ***цель моделирования и определить точку зрения на модель.***

Модели AS-IS и TO-BE

Построение функциональной модели начинают с построения модели AS-IS (Как есть), то есть модели существующей организации работы. Модель AS-IS может строиться на основе изучения документации (должностных инструкций, положений о предприятии, приказов, отчетов и т.п.), анкетирования и опроса служащих предприятия и других источников. Полученная модель AS-IS служит для выявления неуправляемых работ, работ необеспеченных ресурсами, ненужных и неэффективных работ, дублирующихся работ и других недостатков в организации деятельности предприятия. Исправление недостатков, перенаправление информационных и материальных потоков приводит к созданию модели TO-BE(Как

будет)-модели идеальной организации бизнес-процессов. Как правило, строится несколько моделей TO-BE, среди которых определяют наилучший вариант.

Технология проектирования информационных систем подразумевает сначала создание модели AS-IS , её анализ и улучшение бизнес-процессов, то есть создание модели TO-BE, и только на основе модели TO-BE строится модель данных, прототип и затем окончательный вариант информационной системы. Построение системы на основе модели AS-IS автоматизирует несовершенные бизнес-процессы, а также дублирует, а не заменяет существующий документооборот

Диаграммы IDEF0

Модель в нотации IDEF0 представляет собой совокупность иерархически упорядоченных и взаимосвязанных диаграмм. Каждая диаграмма является единицей описания системы и располагается на отдельном листе.

Модель может содержать 4 типа диаграмм:

- Контекстную (главную) (в каждой модели может быть только одна контекстная диаграмма);
- Декомпозиции;
- Дерева узлов;
- Только для экспозиции (FEO)

Контекстная диаграмма является вершиной древовидной структуры диаграмм и представляет собой самое общее описание системы и её взаимодействия с внешней средой. После описания системы в целом производится разбиение её на крупные фрагменты. Этот процесс называется функциональной декомпозицией, а диаграммы называются диаграммами декомпозиции. После декомпозиции контекстной диаграммы проводится декомпозиция каждого большого фрагмента системы на более мелкие и т.д., до достижения нужного уровня подробности описания.

Диаграмма дерева узлов показывает иерархическую зависимость работ , но не взаимосвязи между работами.

Работа (Activity)

Работы обозначают поименованные процессы, функции или задачи, которые происходят в течение определенного времени и имеют распознаваемые результаты. Работы изображаются в виде прямоугольников (блоков). Все работы должны быть названы и определены. Имя работы должно быть глаголом (например, "Изготовить деталь", "Принять заказ" и т.д.). Работа «Изготовление детали» может иметь следующее определение : «Работа относится к полному циклу изготовления детали от контроля качества сырья до отгрузки готового упакованного изделия». При создании новой (меню File/New) автоматически создается **Контекстная** (Главная) диаграмма с единственной работой, изображающей систему в целом. Главная диаграмма имеет вид: (рис.1.)

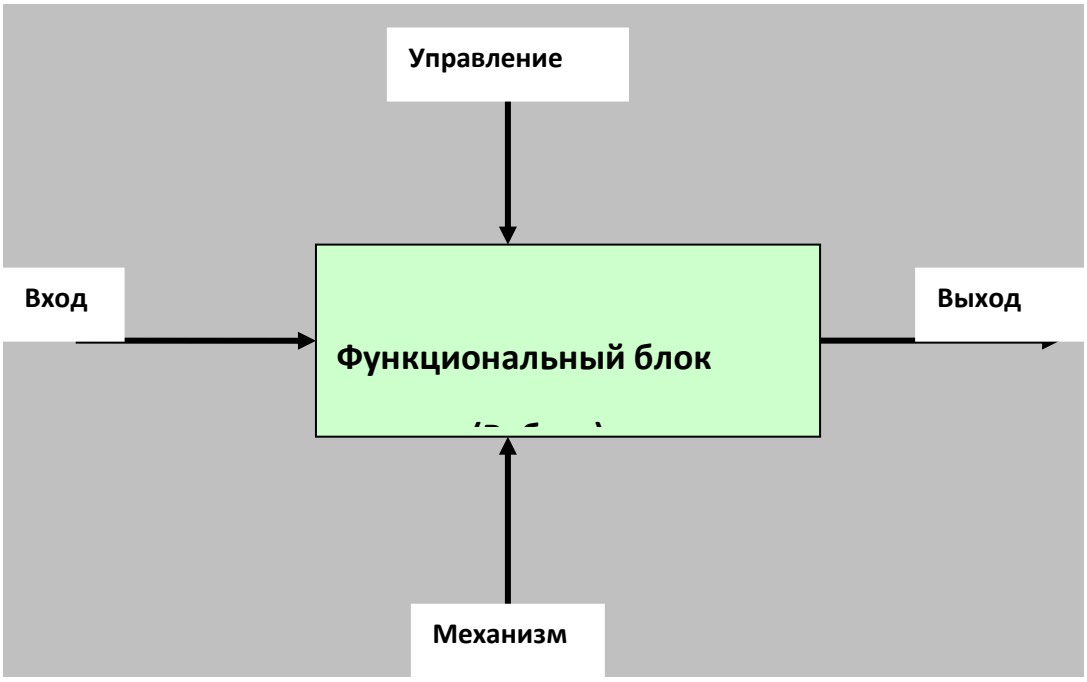


Рис.1.Главная диаграмма

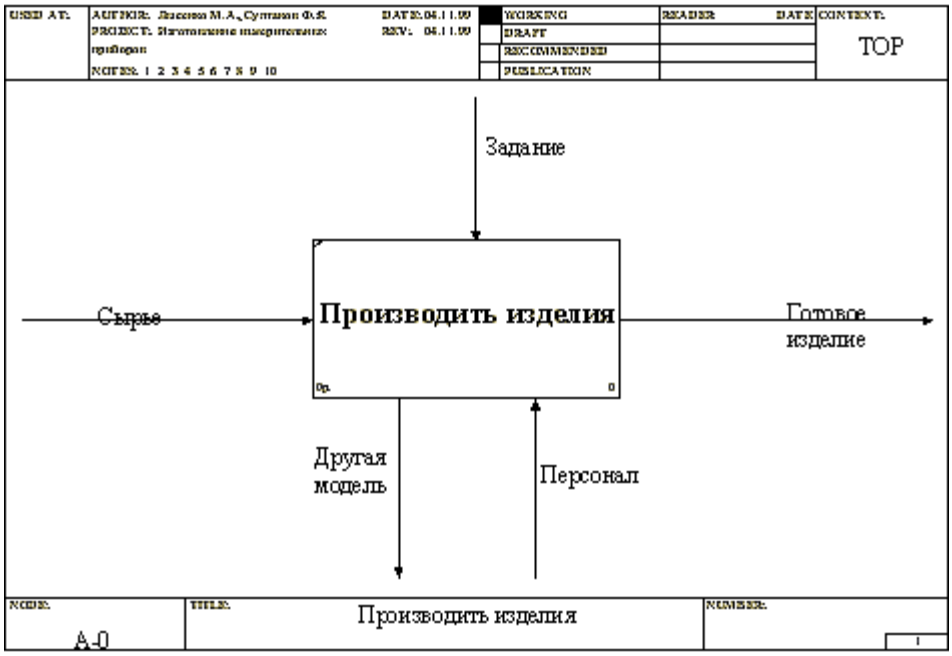


Рис.1.Контекстная диаграмма

Для внесения имени работы следует щелкнуть по работе, выбрать в меню Name и в появившемся диалоге внести имя работы, например «Изготовление изделия». Для описания других свойств работы служит диалог Activity Properties рис.2.

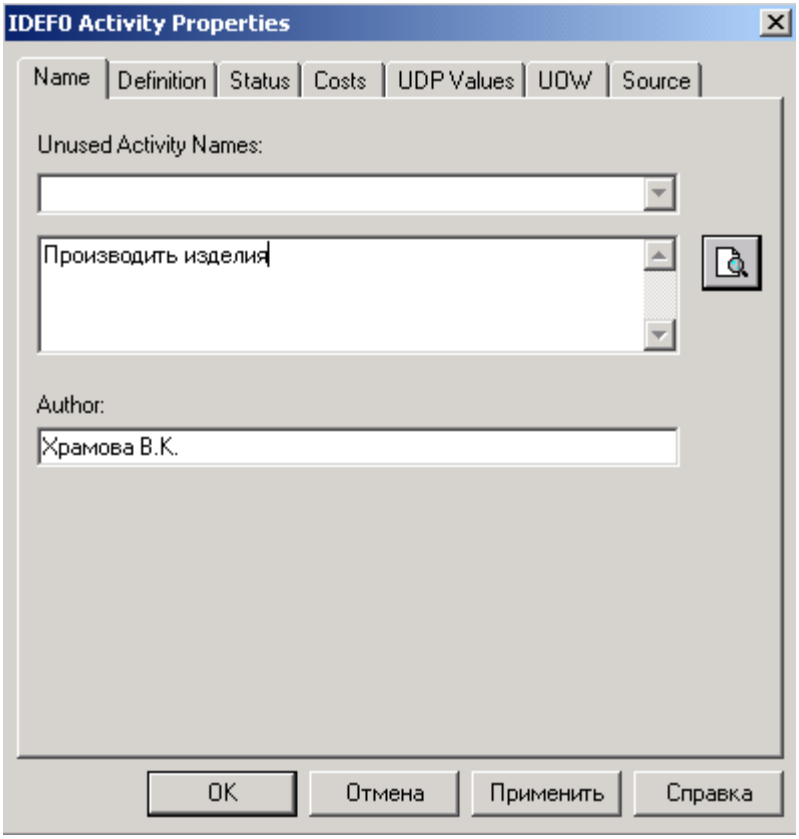


Рис.2. Редактор задания свойств работы.

После создания Контекстной диаграммы необходимо создать диаграммы Декомпозиции (Разбиение).

Диаграммы Декомпозиции содержат родственные работы, т.е. дочерние работы, имеющие родительскую работу.

Для создания диаграммы Декомпозиции следует щелкнуть по кнопке  . Возникает диалог Activity Box Count (рис.4), в котором следует указать нотацию новой диаграммы. Надо выбрать IDEF0 и нажать OK.

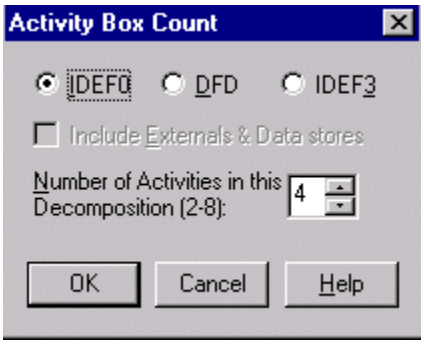


Рис.4.Выбор нотации диаграммы

На диаграмме декомпозиции работы нумеруются автоматически слева направо. Номер работы показывается в правом нижнем углу. В левом верхнем углу изображается небольшая диагональная черта, которая показывает, что данная работа не была декомпозирована.

Пример диаграммы декомпозиции

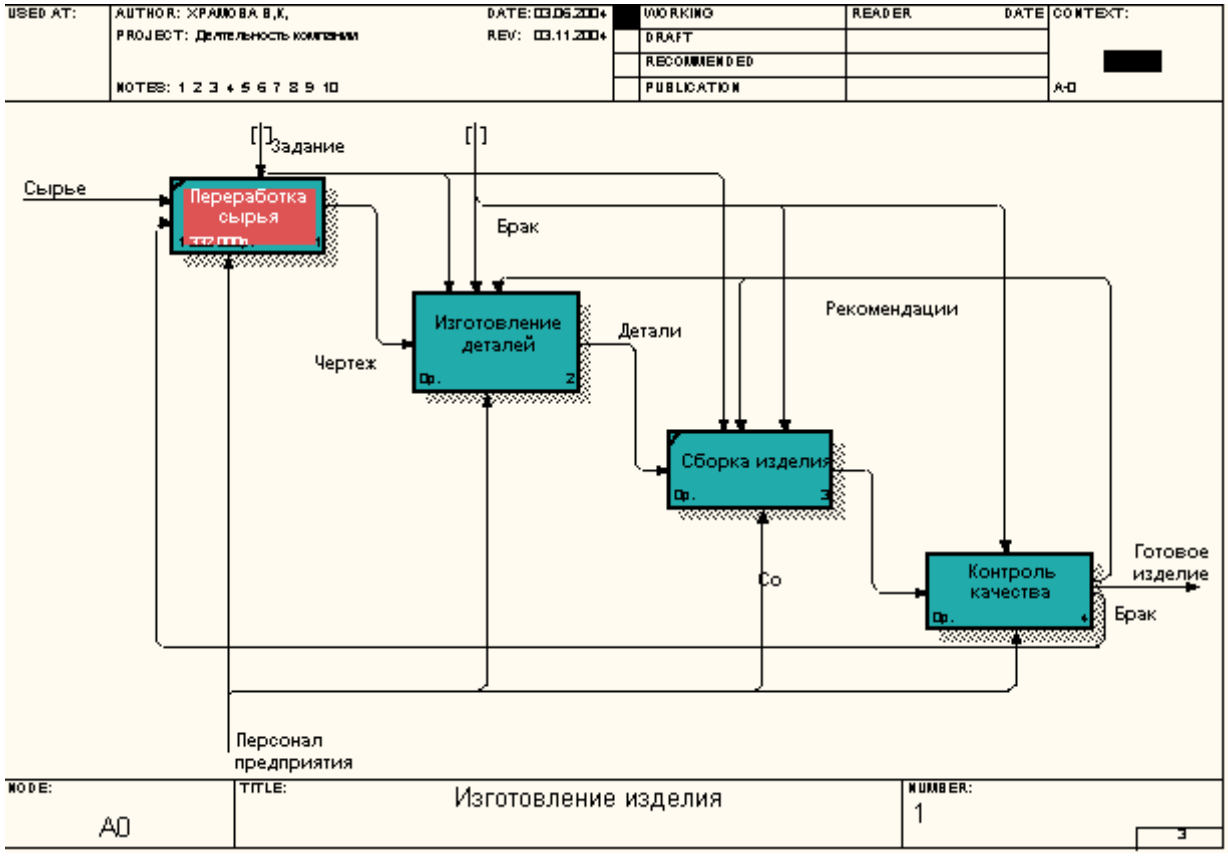


Рис.2.Диаграмма декомпозиции

На диаграмме декомпозиции работы нумеруются автоматически слева направо. Номер работы показывается в правом нижнем углу. В левом верхнем углу изображается небольшая диагональная черта, которая показывает, что данная работа не была декомпозирована.

Стрелки (Arrows).

Взаимодействие работ с внешним миром 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, которые используются или преобразуются работой для получения результата (выхода). Допускается, что работа может не иметь ни одной стрелки входа. Каждый тип стрелок подходит к определенной стороне прямоугольника, изображающего работу, или выходит из нее. Очень часто сложно определить, являются ли данные входом или управлением. В этом случае подсказкой может служить то, перерабатываются/изменяются ли данные в работе или нет. Если изменяются, то скорее всего это вход, если нет - управление.

Управление (Control) - правила, стратегии, процедуры или стандарты, которыми руководствуется работа. Каждая работа должна иметь хотя бы одну стрелку управления. Управление влияет на работу, но не преобразуется работой.

Выход (Output) - материал или информация, которые производятся работой. Каждая работа должна иметь хотя бы одну стрелку выхода. Работа без результата не имеет смысла.

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

Вызов (Call) - специальная стрелка, указывающая на другую модель работы. Рисуется как исходящая из нижней грани работы. Стрелка вызова используется для указания того, что некоторая работа выполняется за пределами моделируемой системы. Используются в механизме слияния и разделения моделей.

Каждый тип стрелок подходит к определенной стороне блока, или выходит из нее. Стрелка входа рисуется как входящая в левую грань работы. Стрелка управления рисуется как входящая в верхнюю грань. Выход рисуется как исходящая стрелка из правой грани. Механизм - входит в нижнюю.

Граничные стрелки.

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 надо:

- щелкнуть по кнопке с символом стрелки  в палитре инструментов. Дальше перенести курсор к левой стороне экрана, пока не появится начальная штриховая полоска;
- щелкнуть один раз по полоске (откуда выходит стрелка) и еще раз в левой части работы со стороны входа (где заканчивается стрелка);
- вернуться в палитру инструментов и выбрать опцию редактирования стрелки 
- щелкнуть правой кнопкой мыши на линии стрелки, во всплывающем меню выбрать пункт Name Editor и добавить имя стрелки в закладке Name диалога IDEF0 Arrow Properties.

Стрелки управления, входа, механизма и выхода изображаются аналогично. Для рисования стрелки выхода, например, следует щелкнуть по кнопке с символом стрелки в палитре инструментов, щелкнуть в правой части работы со стороны выхода (где начинается стрелка), перенести курсор к правой стороне экрана, пока не появится штриховая полоска, и щелкнуть один раз по ней. Имена вновь внесенных стрелок автоматически заносятся в словарь **(Arrow Dictionary).**

Словарь стрелок (Arrow Dictionary)

Словарь стрелок (Arrow Dictionary) редактируется при помощи специального редактора Arrow Dictionary Editor (рис.5), в котором определяется стрелка и вносится относящийся к ней комментарий. Словарь стрелок решает очень важную задачу. Диаграммы создаются аналитиком для того, чтобы провести сеанс экспертизы, т.е. обсудить диаграмму со специалистом предметной области. В любой предметной области формируется профессиональный жаргон, причем очень часто жаргонные выражения имеют нечеткий смысл и воспринимаются разными специалистами по-разному. В то же время аналитик - автор диаграмм вынужден употреблять те выражения, которые наиболее понятны экспертам. Поскольку формальные определения часто сложны для восприятия, аналитик вынужден употреблять профессиональный жаргон, а чтобы не возникало неоднозначных трактовок, в словаре стрелок каждому понятию можно дать расширенное и, если это необходимо, формальное определение.

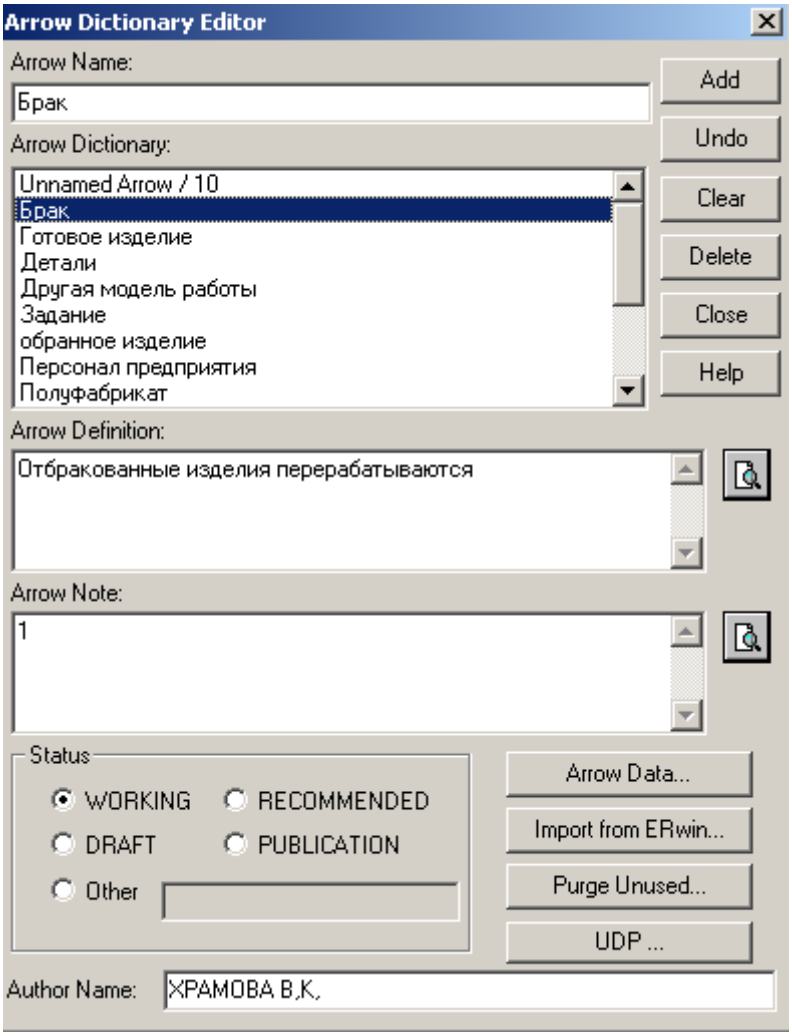


Рис.5.Редактор словаря стрелок

Внутренние стрелки.

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 щелкнуть по сегменту (например, выхода) одной работы и затем по сегменту (например, входа) другой.

В IDEF0 различают пять типов связей работ:

Связь по входу (output-input), когда стрелка выхода вышестоящей работы (далее - просто выход) направляется на вход нижестоящей;

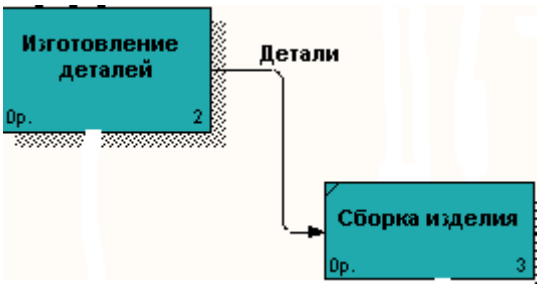


Рис.5.Связь по входу

Связь по управлению (output-control), когда выход вышестоящей работы направляется на управление нижестоящей. Связь по входу показывает доминирование вышестоящей работы. Данные или объекты выхода вышестоящей работы не меняются в нижестоящей;

Рис.5.Связь по управлению

Обратная связь по входу (output-input feedback), когда выход нижестоящей работы направляется на вход вышестоящей. Такая связь, как правило, используется для описания циклов;

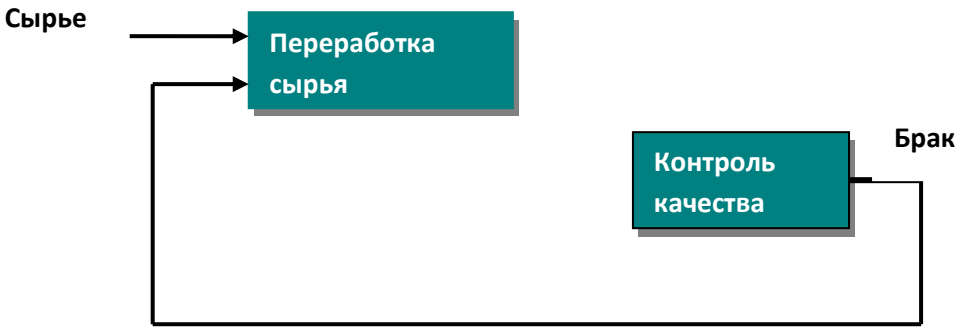


Рис.5.Обратная связь по входу

Обратная связь по управлению (output-control feedback), когда выход нижестоящей работы направляется на управление вышестоящей. Обратная связь по управлению часто свидетельствует об эффективности бизнес-процесса;

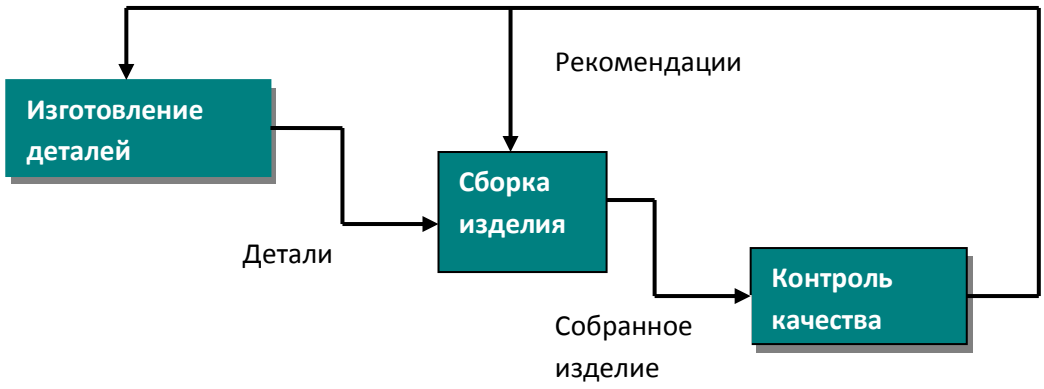


Рис.5.Обратная связь по управлению

Связь выход-механизм (output-mechanism), когда выход одной работы направляется на механизм другой. Эта взаимосвязь используется реже остальных и показывает, что одна работа подготавливает ресурсы, необходимые для проведения другой работы.

Явные стрелки.

Явная стрелка имеет источником одну-единственную работу и назначением тоже одну-единственную работу.

Разветвляющиеся и сливающиеся стрелки.

Одни и те же данные или объекты, порожденные одной работой, могут использоваться сразу в нескольких других работах. С другой стороны, стрелки, порожденные в разных работах, могут представлять собой одинаковые или однородные данные или объекты, которые в дальнейшем используются или перерабатываются в одном месте. Для моделирования таких ситуаций IDEF0 используются разветвляющиеся и сливающиеся стрелки. Для разветвления стрелки нужно в режиме редактирования стрелки щелкнуть по фрагменту стрелки и по соответствующему сегменту работы. Для слияния двух стрелок выхода нужно в режиме редактирования стрелки сначала щелкнуть по сегменту выхода работы, а затем по соответствующему фрагменту стрелки.

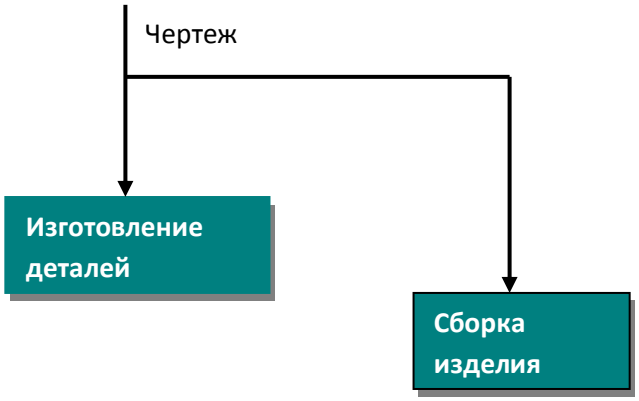


Рис.5.Пример именования разветвляющейся стрелки

Тоннелирование стрелок.

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 щелкнуть по квадратным скобкамграничной стрелки. Появится диалог Border Arrow Editor (рис.6).

Рис.6.Диалог для тоннелирования стрелок

Если щелкнуть по кнопке **Resolve Border Arrow**, стрелка мигрирует на диаграмму верхнего уровня, если по кнопке **Change To Tunnel** - стрелка будет затоннелирована и не попадет на другую диаграмму. Тоннельная стрелка изображается с круглыми скобками на конце. Тоннелирование может быть применено для изображения малозначимых стрелок. Если на какой-либо диаграмме

нижнего уровня необходимо изобразить малозначимые данные или объекты, которые не обрабатываются или не используются работами на текущем уровне, то их необходимо направить на вышестоящий уровень. Если эти данные не используются на родительской диаграмме, их нужно направить еще выше и т.д. В результате малозначимая стрелка будет изображена на всех уровнях и затруднит чтение всех диаграмм, на которых она присутствует. Выходом является тоннелирование стрелки на самом нижнем уровне. Такое тоннелирование называется "Не-в-родительской-диаграмме". Другим примером тоннелирования может быть ситуация, когда стрелка механизма мигрирует с верхнего уровня на нижний, причем на нижнем уровне этот механизм используется одинаково во всех работах без исключения. В этом случае стрелка механизма на нижнем уровне может быть удалена, после чего на родительской диаграмме она может быть затоннелирована ("Не-в-дочерней-работе").

Контрольные вопросы:

1. С какой модели начинается построение функциональной модели?.
2. Для чего нужно построение модели AS-IS ?
3. Для чего нужно построение модели TO-BE?
4. Какие типы диаграмм может содержать модель в стандарте IDEF0?
5. Как создать диаграммы декомпозиции?
6. Для чего нужны стрелки на диаграммах?
7. Как посмотреть словарь стрелок?

Критерии оценки работы:

1. Полный ответ – 5 баллов.
2. Дополнительный уточняющий вопрос – 4 балла.
3. Краткий ответ – 3 балла.

ПРАКТИЧЕСКАЯ РАБОТА № 3

Постановка задачи. Создание контекстной диаграммы для данной задачи.

Цель работы:

1. Знакомство с примером
2. Создание контекстной диаграммы для данного примера.

Исходные данные (задание):

Постановка задачи. Пример

В качестве примера рассмотрим деятельность вымышленной компании «Олди», которая существует 5 лет и занимается в основном **сборкой** и **продажей** настольных **компьютеров и ноутбуков.** Годовой оборот компании составляет примерно 20 млн. долларов. Компания закупает компоненты для компьютеров от трех независимых поставщиков, а не производит компоненты самостоятельно.. она только собирает и тестирует компьютеры. Компания реализует продукцию через магазины и специализируется на покупателях, для которых главный критерий при покупке – стоимость компьютера. Предполагаемый объем рынка для компании «Олди» в последующие 2 года – 50 млн.долл.

Несмотря на некоторое увеличение объема продаж, прибыли уменьшаются, растет конкуренция на рынке. Чтобы не потерять позиции компания решает проанализировать текущие бизнес-процессы и реорганизовать их с целью увеличения эффективности производства и продаж. Основные процедуры в компании таковы:

- Продавцы принимают заказы клиентов;
- Операторы группируют заказы по типам компьютеров;
- Операторы собирают и тестируют компьютеры;
- Операторы упаковывают компьютеры согласно заказам;
- Кладовщик отгружает клиентам заказы.

В настоящее время компания «Олди» использует купленную бухгалтерскую информационную систему, которая позволяет оформить заказ, счет и отследить платежи по счетам.

Улучшение деятельности компании должно касаться структур управления компанией, эффективности производства и внутреннего контроля. В результате реорганизация может потребовать внедрения новой корпоративной информационной системы (состоящей не только из одного бухгалтерского модуля).

Однако перед тем как пытаться производить какие-то улучшения необходимо разобраться в существующих бизнес-процессах.

Создание контекстной диаграммы.

Для составления контекстной диаграммы выполните следующие действия.

1. Запустите BPwin. Кнопка Пуск/Программы/
2. Появится диалоговое окно ModelMart Connection Manager. Нажмите кнопку Cancel.
3. Появится диалоговое окно Iwould like to. Впишите имя модели Деятельность компании «Олди» и выберите Type- IDEF0. Нажмите кнопку OK.

4. Появится окно Properties for New Models, в окно Autor впишите свою фамилию и нажмите кнопку OK.
5. Автоматически создается контекстная диаграмма.
6. Обратите внимание на кнопку  на панели инструментов. Эта кнопка включает и выключает инструмент просмотра и навигации - Model Explorer (появляется слева). Кнопка Activities/Diagram переключает режим Model Explorer. В режиме Activities щелчок правой кнопкой мыши по объекту в Model Explorer позволяет редактировать его свойства.
7. Перейдите в меню **Model/Model Properties**. В закладке General диалогового окна Model Properties введите имя модели {Деятельность компании «Олди»}, имя проекта (Project) {Модель деятельности «Олди»}, Имя автора свою фамилию и тип модели Time Frame{AS-IS}.
8. В закладке **Purpose** внесите Цель { Purpose: Моделировать текущие {AS-IS} бизнес-процессы компании «Олди» } и точку зрения {Viewpoint: Директор}.
9. В закладке **Definition** внесите определение {Это учебная модель, описывающая деятельность компании «Олди» и Scope(область действия) {Общее управление бизнесом компании: исследование рынка, закупка компьютеров, сборка, тестирование и продажа продуктов}.
10. В закладке **Sourse** (источник информации) введите {Материалы курса по BPwin}
11. В закладке **Status** установите WORKING и нажмите кнопку OK.
12. Перейдите на контекстную диаграмму и правой кнопкой мыши щелкните по работе. В контекстном меню выберите Name . В закладке Name внесите имя {Деятельность компании «Олди»}.
13. В закладке **Definition** внесите определение {Текущие бизнес-процессы компании «Олди»}.
14. В закладке **Status** установите WORKING.
15. В закладке **Source** внесите {Материалы курса по BPwin } и нажмите кнопку OK .

Создайте стрелки на контекстной диаграмме (табл.1.), для создания стрелок прочитайте «**Стрелки»** и **«Граничные стрелки».**

Таблица 1.

Наименование стрелки (Arrow Name)	**Описание (Arrow Definition)**	**Тип**
Бухгалтерская система	Оформление счетов, оплата счетов работа с заказами	Механизм
Звонки клиентов	Запросы информации, заказы, техническая поддержка и т.д.	Вход
Правила и процедуры	Правила продажи, инструкции по сборке, процедуры тестирования, критерии производительности и т.д.	Управление
Проданные продукты	Настольные и портативные компьютеры	Выход

16. Для создания **Arrow Name**, установите курсор на соответствующую стрелку, правой кнопкой мыши вызовите контекстное меню, выберите пункт Name и впишите название стрелки, например, Бухгалтерская система.

17. Перейдите на вкладку **Definition** и внесите описание для соответствующей стрелки, например, Оформление счетов, оплата счетов, работа с заказами.
18. Создайте отчет по модели, используя меню **Tools/Reports/Model Reports** (рис.1.)
19. Сохраните свою работу в своей папке.

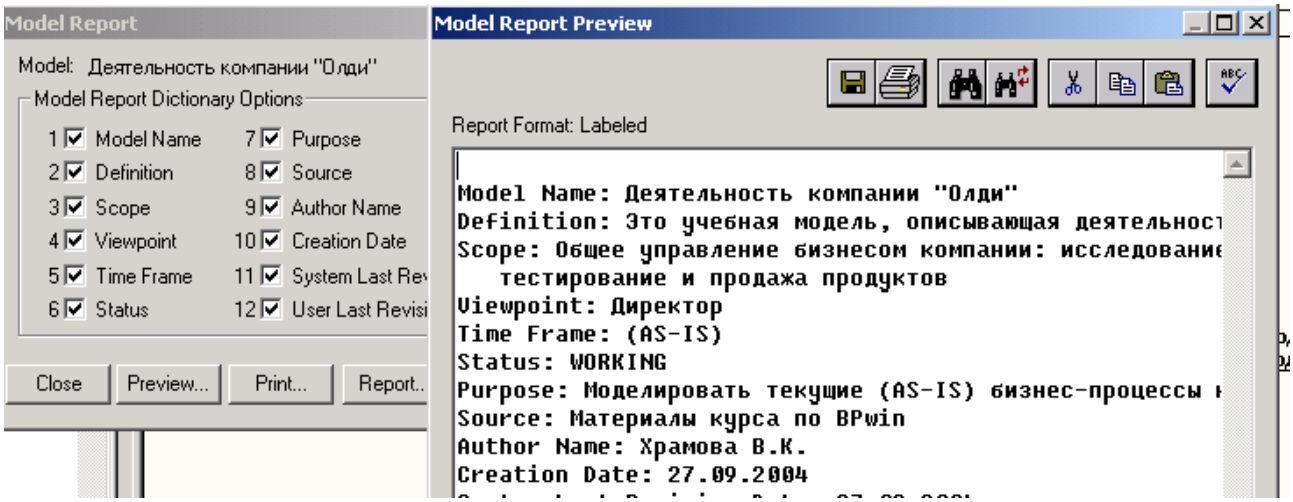


Рис.1.

Результат выполнения работы Контекстная диаграмма см. на рис.3.

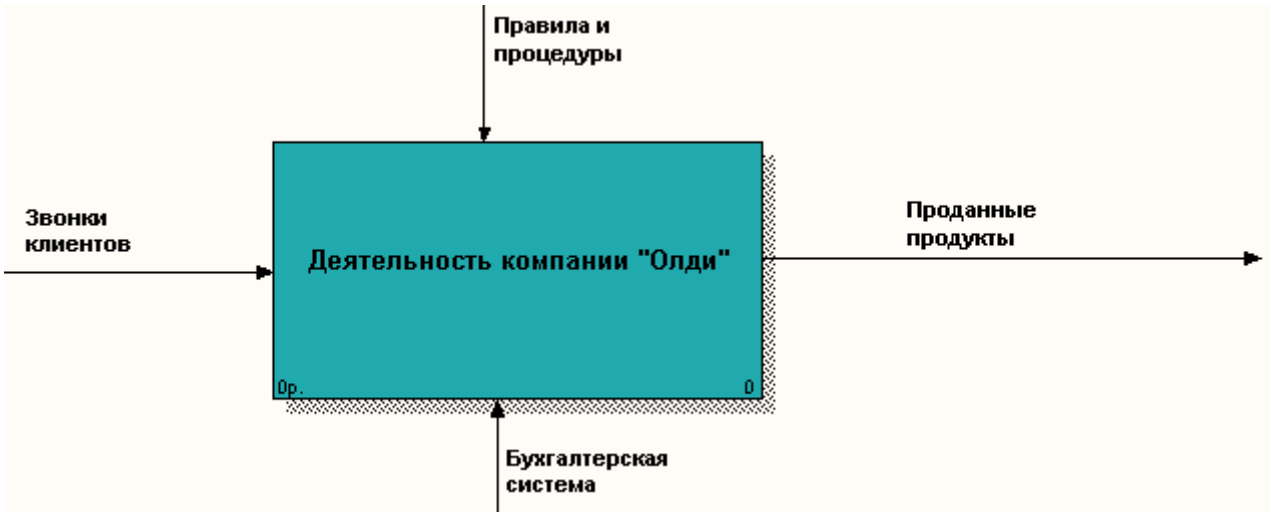


Рис.3. Контекстная диаграмма

Контрольные вопросы:

1. Как начать работу в программе BPwin?.
2. Как внести имя модели ?
3. Как внести имя проекта?
4. Как внести имя автора?
5. Как внести цель моделирования?
6. Как установить свойства диаграммы?
7. Как сохранить модель?
8. Как установить шрифт объекта?
9. Как установить цвет объекта?

10. Как создать отчет по модели?
11.Показать на диаграмме, где находится тип стрелки «Механизм»?
12.Показать на диаграмме, где находится тип стрелки «Вход»?
13.Показать на диаграмме, где находится тип стрелки «Управление»?
14.Показать на диаграмме, где находится тип стрелки «Выход»?

Критерии оценки работы:
1. Полный ответ – 5 баллов.
2. Дополнительный уточняющий вопрос – 4 балла.
3. Краткий ответ – 3 балла.

ПРАКТИЧЕСКАЯ РАБОТА № 4

Создание диаграммы декомпозиции.

Цель работы:

Создание диаграммы декомпозиции для данного примера.

Исходные данные (задание):

Создание диаграммы декомпозиции.

Перед выполнением этой работы внимательно прочитайте раздел «Работа» и раздел «Стрелки»

1. Выберите кнопку перехода ▼ на нижний уровень в палитре инструментов, в диалоговом окне Activity Box Count установите число работ 3 на диаграмме нижнего уровня нажмите кнопку OK (рис. 1.)

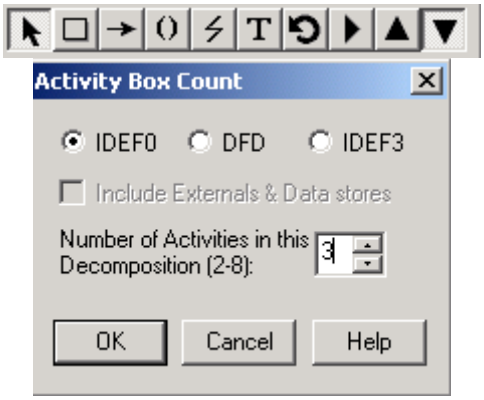


Рис.1.

Автоматически будет создана диаграмма декомпозиции. Правой кнопкой мыши щелкните по первой работе, выберите Name и внесите имя работы, например, «Продажи и маркетинг». Повторите операцию для всех трех работ. Затем внесите определение, статус и источник для каждой работы согласно табл. 2.

Таблица 2.

Описание работ для диаграммы декомпозиции

Функциональный блок Name)	Описание Definition)	Статус (Status)	Источник (Sourse)
Продажи, маркетинг	Телемаркетинг, презентации, выставки	WORKING	Материалы курса по BPwin
Сборка, тестирование компьютеров	Сборка и тестирование настольных и портативных компьютеров	WORKING	Материалы курса по BPwin
Отгрузка, получение	Отгрузка заказов клиентам и получение компонентов от	WORKING	Материалы курса по BPwin

	поставщиков		

2. Для изменения свойств работ после их внесения в диаграмму можно воспользоваться словарем объектов модели. Вызов словаря **Model/Diagram Object Editor** (рис. 2)

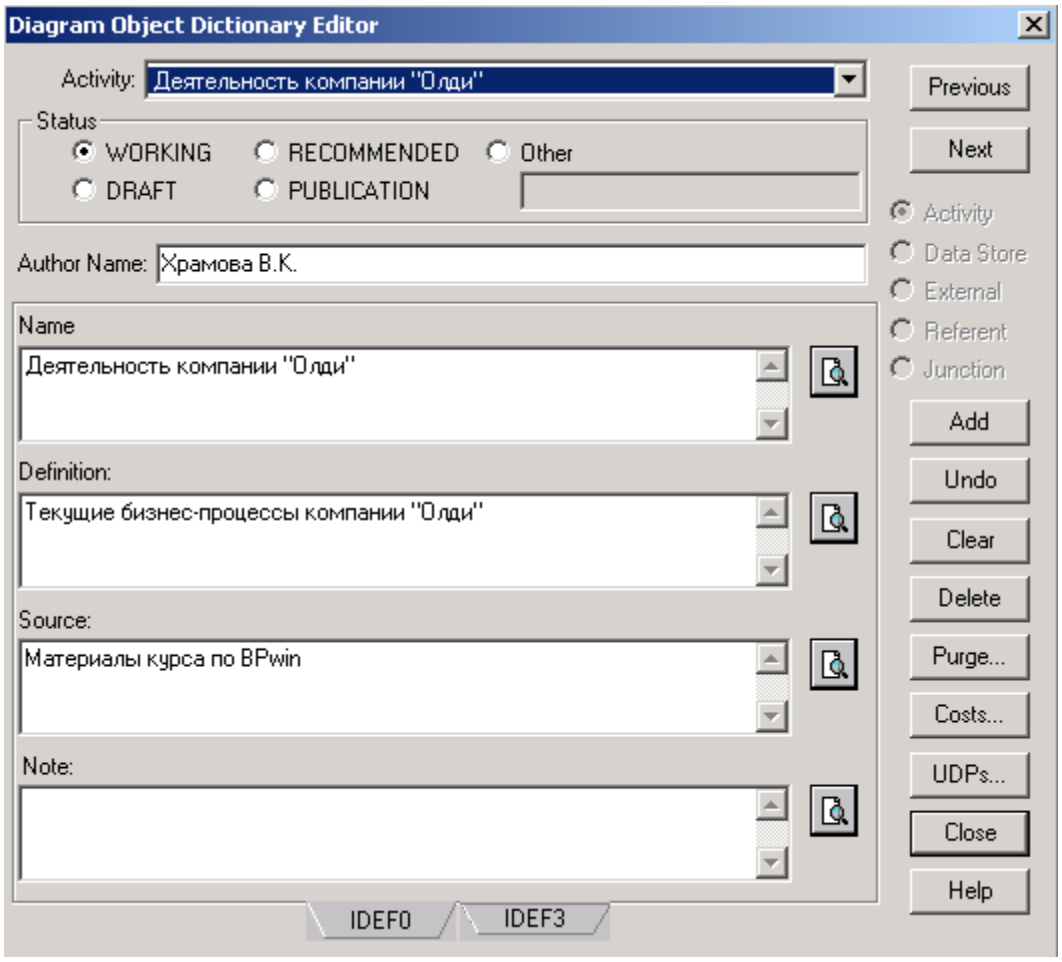


Рис.2.

3. Если Вы опишите имя и свойства работы в словаре, её можно будет внести в диаграмму позже с помощью кнопки  в палитре инструментов. Вы не можете удалить работу из словаря, если она используется на какой-либо диаграмме. Если вы удалите работу из диаграммы, из словаря она не удаляется. Имя и описание такой работы может быть использовано в дальнейшем. Для добавления работы в словарь щелкните по кнопкеClear, внесите имя и свойства работы, затем щелкните по Add. Для удаления всех работ, не использующихся в модели, щелкните по Purge

3. Перейдите в режим рисования стрелок. Свяжите граничные стрелки (кнопка  на палитре инструментов) с остальными так как показано на рис.2. Для этого прочитайте раздел «Разветвляющиеся и сливающиеся стрелки» в Лабораторной работе №2

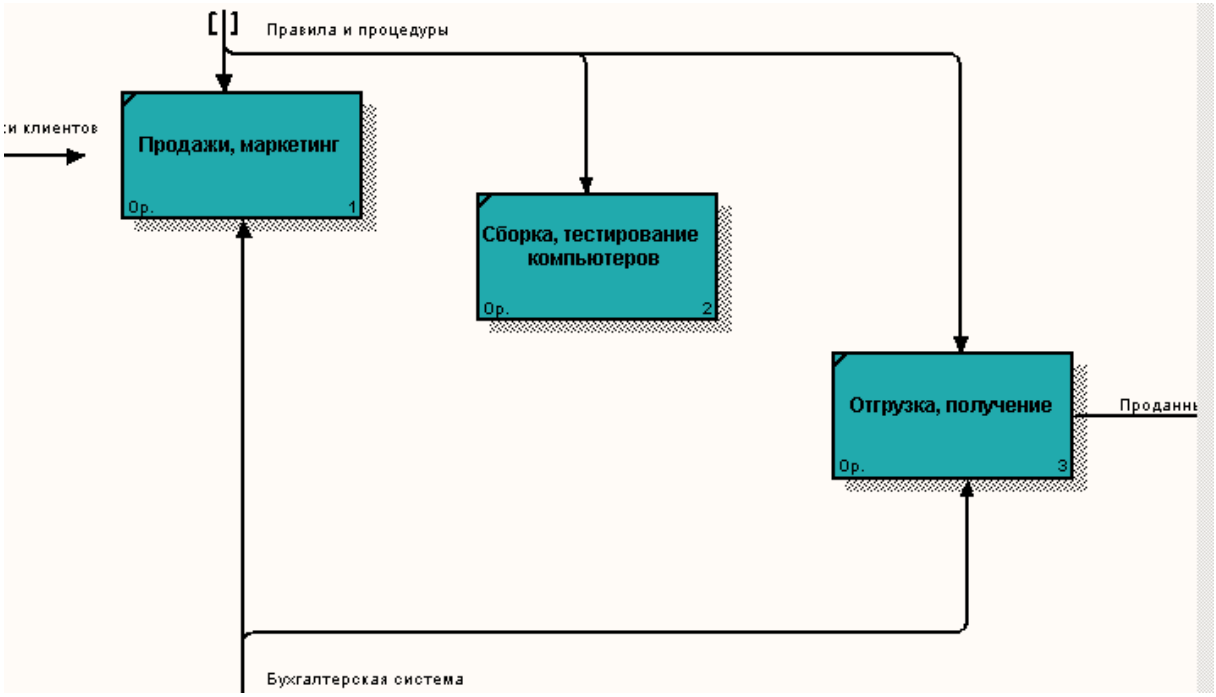


Рис. 3.

4. Правой кнопкой мыши щелкните по ветви стрелки управления работы «Сборка и тестирование компьютеров» и переименуйте её в «Правила сборки и тестирования» (рис.4.)

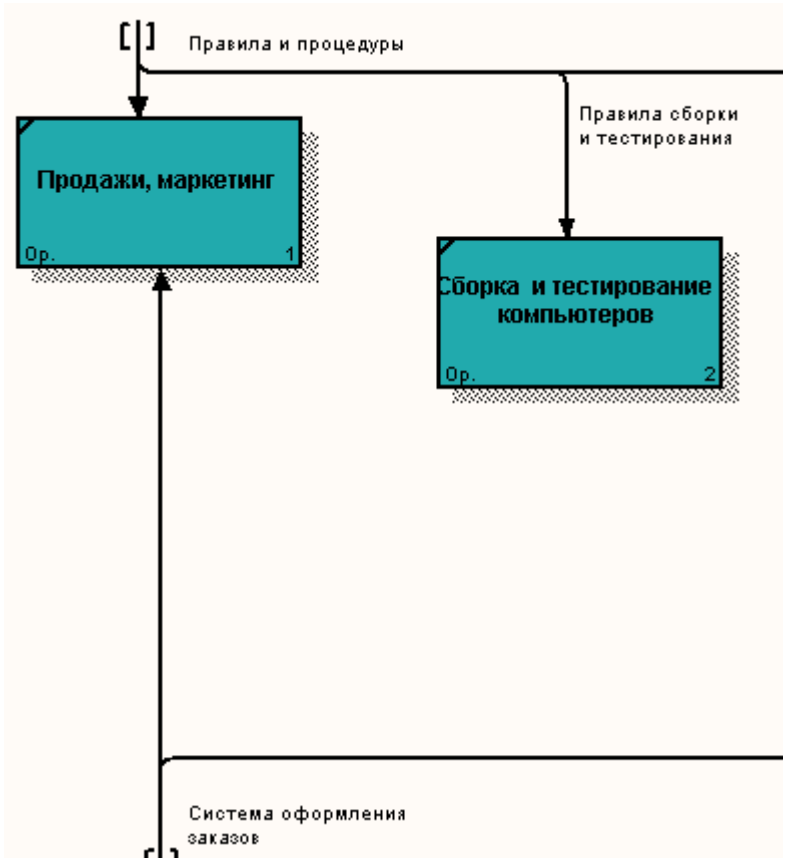


Рис. 4.

Внесите определение (Definition) для новой ветви: «Инструкция по сборке, процедуры тестирования, критерии производительности и т.д.»

Правой кнопкой мыши щелкните по ветви стрелки механизма работы «Продажи и маркетинг» и переименуйте её в «Систему оформления заказов»

5. Альтернативный метод внесения имен и свойств стрелок – использование словаря стрелок (вызов словаря – меню Model/Diagramm Objiect editor). Если вы опишите имя и свойство стрелки в словаре, её можно будет внести в диаграмму позже. Вы не можете удалить стрелку из словаря, если она используется на какой-либо диаграмме. Если Вы удалите стрелку из диаграммы, из словаря она не удаляется. Имя и описание такой стрелки может быть использовано в дальнейшем. Для добавления стрелки в словарь щелкните по кнопке Clear, внесите имя и свойства работы, затем щелкните по Add. Для удаления всех имен стрелок, не использующихся в модели, щелкните по Purge Unused.

6. Создайте новые внутренние стрелки так, как показано на Рис.5.

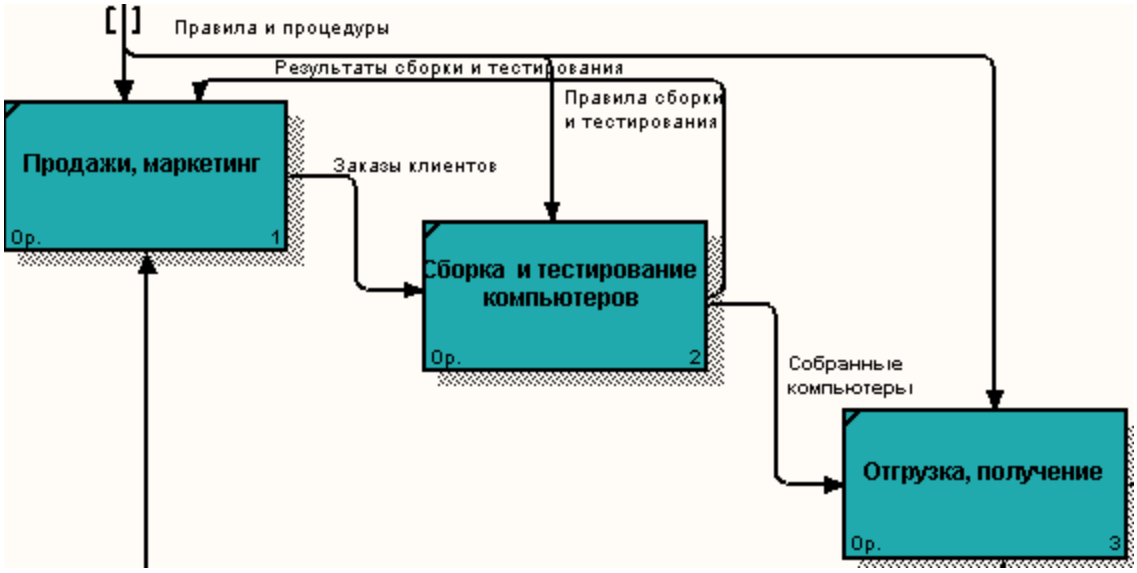
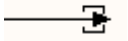


Рис.5.

7. Создайте стрелку обратной связи (по управлению) «Результаты сборки и тестирования» (рис.4.), идущую от работы «Сборка и тестирование компьютеров» к работе «Продажи и маркетинг». (В Л/Р №2 прочитайте материал: «Обратная связь по управлению»). Для большей наглядности измените стиль стрелки (толщина линий) и установите опцию Extra Arrowhead (из контекстного меню). Методом drag&drop перенесите имена стрелок так, чтобы их было удобнее читать. Если необходимо, установите Squiggle (из контекстного меню).

8. Создайте новую граничную стрелку выхода «Маркетинговые материалы» из работы «Продажи и маркетинг». Эта стрелка автоматически не попадает на диаграмму верхнего уровня и имеет квадратные скобки на наконечнике  . Из палитры выберите кнопку  , щелкните мышью по квадратным скобкам и в диалоговом окне Border Editor выберите Resolve Border Arrow. Для стрелки «Маркетинговые материалы» выберите опцию Trim из контекстного меню. Результат выполнения показан на рис.6. и в Приложении 1.

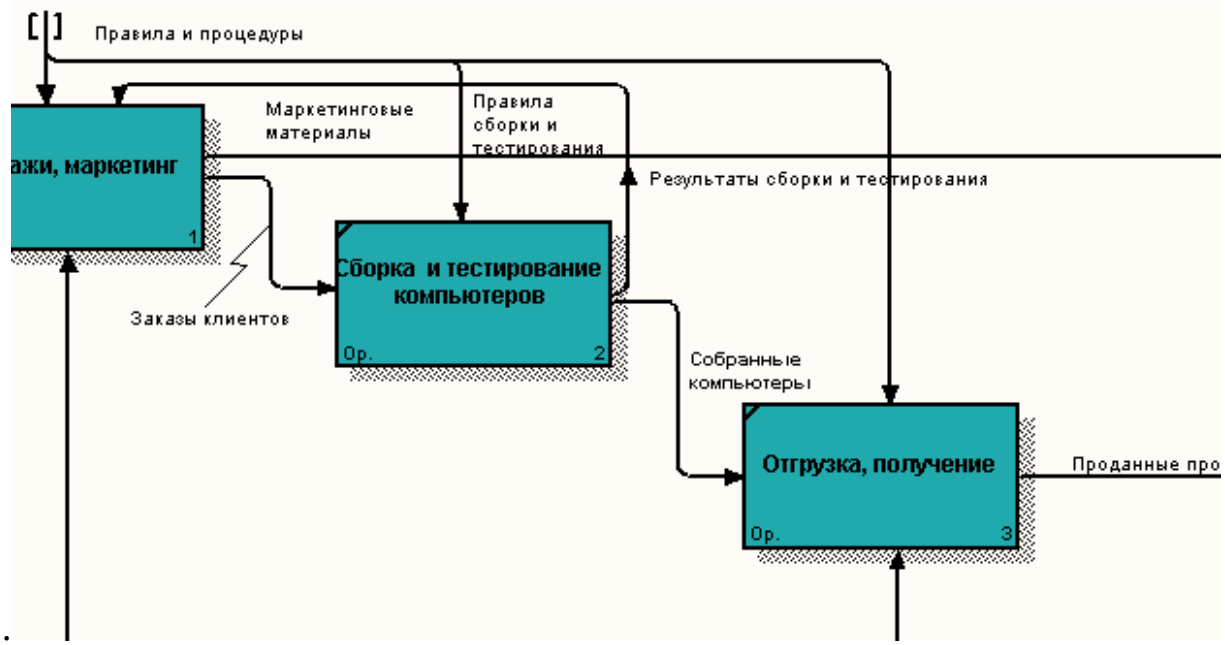


Рис.6.

Контрольные вопросы:

1. Как создать стрелку входа?
2. Как создать стрелку выхода ?
3. Как создать стрелку управления?
4. Как создать стрелку механизма?
5. Как посмотреть Словарь стрелок?
6. Как создать внутренние стрелки?
7. Как создать стрелки обратной связи?
8. Как создать разветвляющиеся и сливающиеся стрелки?
9. Как создать диаграмму декомпозиции?
10. Как создать отчет по модели?

Создание диаграммы декомпозиции А.2

Цель работы:

Создание диаграммы декомпозиции для данного примера.

Создание диаграммы декомпозиции.

Декомпозируем работу «**Сборка и тестирование компьютеров**».

- В результате проведения экспертизы получена следующая информация. Производственный отдел получает заказы клиентов от отдела продаж по мере их поступления.
- Диспетчер координирует работу сборщиков, сортирует заказы, группирует их и дает указание на отгрузку компьютеров, когда они готовы.
- Каждые 2 часа диспетчер группирует заказы – отдельно для настольных компьютеров и ноутбуков – и направляет на участок сборки.
- Сотрудники участка сборки собирают компьютеры согласно спецификациям заказа по сборке. Когда группа компьютеров, соответствующая группе заказов, собрана, она направляется на тестирование. Тестировщики тестируют каждый компьютер и в случае необходимости заменяют неисправные компоненты.

- Тестировщики направляют результаты тестирования диспетчеру, который на основании этой информации принимает решение о передаче компьютеров, соответствующих группе заказов, на отгрузку.

1. На основе этой информации внесите новые работы и стрелки (табл.1. и табл.2.).
2.

Таблица 1. Работы диаграммы декомпозиции A2

Имя работы (Activity Name)	Определение работы (Activity Definition)
Отслеживание расписания и управление сборкой и тестированием	Просмотр заказов, установка расписания и выполнения заказов, просмотр результатов тестирования, формирование групп заказов на сборку и отгрузку
Сборка настольных компьютеров	Сборка настольных компьютеров в соответствии с инструкциями и указаниями диспетчера
Сборка ноутбуков	Сборка ноутбуков в соответствии с инструкциями и указаниями диспетчера
Тестирование компьютеров	Тестирование компьютеров и компонентов. Замена неработающих компонентов.

Таблица 2. Стрелки диаграммы декомпозиции A2

Имя стрелки (Arrow Name)	Источник стрелки (Arrow Sourse)	Тип источника стрелки (Arrow Sourse Type)	Назначение стрелки (Arrow Dest.)	Тип назначения стрелки (Arrow Dest. Type)
Диспетчер	Персонал производственного отдела		Отслеживание расписания и управление сборкой и тестированием	Механизм
Заказы клиентов	Граница диаграммы	Управление	Отслеживание расписания и управление сборкой и тестированием	Управление
Заказы на настольные компьютеры	Отслеживание расписания и управление сборкой и	Выход	Сборка настольных компьютеров	Управление

	тестированием			
Заказы на ноутбуки	Отслеживание расписания и управление сборкой и тестированием	Выход	Сборка ноутбуков	Управление
Компоненты	«Tunnel»	Вход	Сборка настольных компьютеров	Вход
			Сборка ноутбуков	Вход
			Тестирование компьютеров	Вход
Настольные компьютеры	Сборка настольных компьютеров	Выход	Тестирование компьютеров	Вход
Ноутбуки	Сборка ноутбуков	Выход	Тестирование компьютеров	Вход
Персонал производственного отдела	«Tunnel»	Механизм	Сборка настольных компьютеров	Механизм
			Сборка ноутбуков	Механизм
Правила сборки и тестирования	Граница диаграммы		Сборка настольных компьютеров	Управление
			Сборка ноутбуков	Управление
			Тестирование компьютеров	Управление
Результаты сборки и тестирования	Сборка настольных компьютеров	Выход	Граница диаграммы	Выход
	Сборка ноутбуков	Выход		
	Тестирование компьютеров	Выход		
Результаты тестирования	Тестирование компьютеров	Выход	Отслеживание расписания и управление сборкой и тестированием	Вход
Собранные компьютеры	Тестирование компьютеров	Выход	Граница диаграммы	Выход
Тестировщик	Персонал производственного отдела		Тестирование компьютеров	Механизм

Указание передать компьютеры на отгрузку	Отслеживание расписания и управление сборкой и тестированием	Выход	Тестирование компьютеров	Управление

3. Тоннелируйте и свяжите на верхнем уровне граничные стрелки, если это необходимо. Результат выполнения работы показан в Приложении 1

Контрольные вопросы:

1. Как создать стрелку входа?
2. Как создать стрелку выхода ?
3. Как создать стрелку управления?
4. Как создать стрелку механизма?
5. Как посмотреть Словарь стрелок?
6. Как создать внутренние стрелки?
7. Как создать стрелки обратной связи?
8. Как создать разветвляющиеся и сливающиеся стрелки?
9. Как создать диаграмму декомпозиции?
10. Как создать отчет по модели?

Критерии оценки работы:

1. Полный ответ – 5 баллов.
2. Дополнительный уточняющий вопрос – 4 балла.
3. Краткий ответ – 3 балла.

ПРАКТИЧЕСКАЯ РАБОТА № 5

Создание диаграммы узлов. Создание диаграммы FEO. Каркас диаграммы

Цель работы:

1. Изучение и создание диаграммы дерева узлов.
2. Изучение и создание диаграммы FEO.
3. Изучение содержания каркаса диаграммы

Исходные данные (задание):

Задание №1. Внимательно прочитайте «Создание диаграмм дерева узлов»

Создание диаграммы дерева узлов.

Диаграмма дерева узлов показывает иерархию работ в модели и позволяет рассмотреть всю модель целиком, но не показывает взаимосвязи между работами(стрелки). Процесс создания модели работ является итерационным, следовательно, работы могут менять свое расположение в дереве узлов.

Для создания диаграммы дерева узлов следует выбрать в меню пункт **Diagram/Add Node Tree**. Возникает эксперт создания диаграммы дерева узлов **Node Tree Wizard**. В первом диалоге эксперта необходимо внести имя диаграммы дерева узлов, узел верхнего уровня и глубину дерева **Number of Levels (**по умолчанию 3). Дерево узлов не обязательно в качестве верхнего уровня должно иметь контекстную работу и произвольную глубину. В одной модели можно создавать множество диаграмм деревьев узлов. Имя дерева узлов по умолчанию совпадает с именем работы верхнего уровня

Второй диалог эксперта **Node Tree Wizard** (рис.2.) позволяет задать свойства диаграммы дерева узлов.

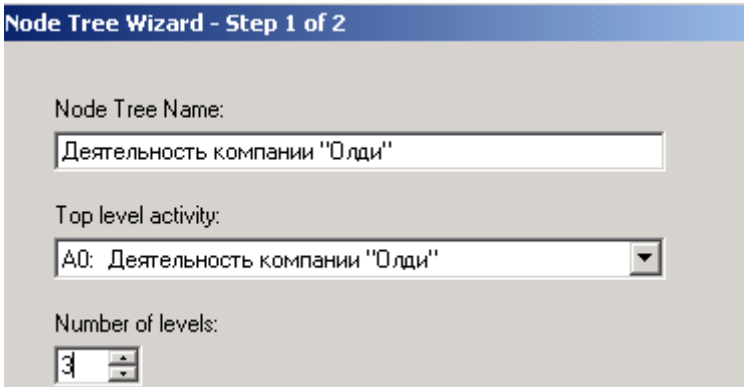


Рис.1. Первый диалог создания диаграммы дерева узлов

По умолчанию нижний уровень декомпозиции показывается в виде списка, а остальные работы в виде прямоугольников. Для отображения всего дерева в виде прямоугольников следует выбрать опцию **Bullet last level**. Группа **Connection Style** позволяет выбрать стиль соединительных линий.

Выберите меню **Diagram/Add Node Tree**. В первом диалоге вида **Node Tree Wizard** внесите имя диаграммы, укажите диаграмму корня дерева и количество уровней

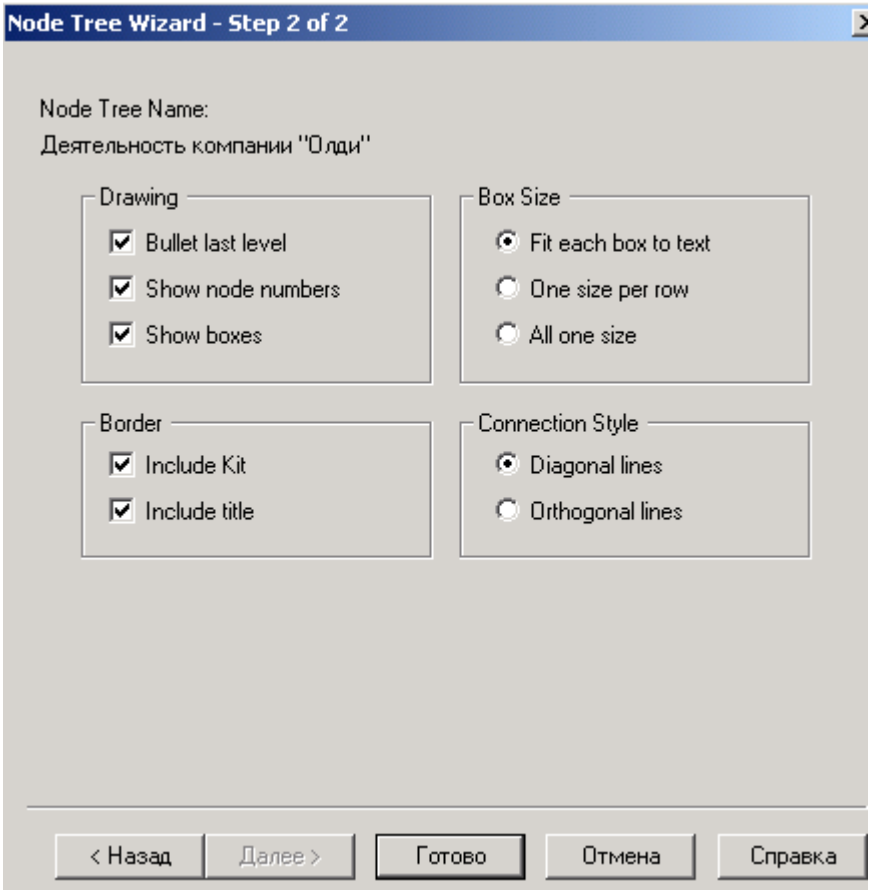


Рис.2. Диалог настройки диаграмм дерева узлов

Во втором диалоге установите опции, как на рис.2. Щелкните по кнопке «Готово». Создается диаграмма дерева узлов. Диаграмму дерева узлов можно модифицировать. Нижний уровень может быть отображен не в виде списка, а в виде прямоугольников, так же как и верхние уровни.

Для модификации диаграммы правой кнопкой мыши щелкните по свободному месту, не занятому объектами, выберите меню **Node Tree Diagram Properties** и во вкладке **Style** диалога **Node Tree Diagram Properties** отключите опцию **Bullet last level**. Щелкните по ОК.

1. Выберите пункт меню **Diagram/Add Node Tree**. В первом диалоге гида **Node Tree Wizard** внесите имя диаграммы, укажите диаграмму корня дерева и количество уровней.
2. Во втором диалоге установите опции, как показано на рис.2.
3. Щелкните по кнопке «Готово». Создается диаграмма дерева узлов. Диаграмму дерева узлов можно модифицировать. Нижний уровень может быть отображен не в виде списка, а в виде прямоугольников, так же как и верхние уровни.

Для модификации диаграммы правой кнопкой мыши щелкните по свободному месту, не занятому объектами, выберите меню **Node Tree Diagram Properties** и во вкладке Style диалога Node Tree Diagram отключите опцию **Bullet last level.** Щелкните по кнопке ОК.

Задание №2. Внимательно прочитайте «Создание FEO диаграммы»

Создание диаграммы FEO.

Диаграммы «только для экспозиции» (FEO) часто используются в модели для иллюстрации других точек зрения, для отображения отдельных деталей, которые не поддерживаются явно синтаксисом IDEF0. Диаграммы FEO позволяют нарушить любое синтаксическое правило, поскольку по сути являются просто картинками – копиями стандартных диаграмм и не включаются в анализ синтаксиса. Например, работа на диаграмме FEO может не иметь стрелок управления и выхода. С целью обсуждения определенных аспектов модели с экспертом предметной области может быть создана диаграмма только с одной работой и одной стрелкой, поскольку стандартная диаграмма декомпозиции содержит множество деталей, не относящихся к теме обсуждения и дезориентирующих эксперта. Но если FEO используется для иллюстрации альтернативных точек зрения (альтернативный контекст), рекомендуется все-таки придерживаться синтаксиса IDEF0. Для создания диаграммы FEO следует выбрать пункт меню **Diagram/Add FEO Diagram.**

Пример

Предположим, что при обсуждении бизнес-процессов возникла необходимость детально рассмотреть взаимодействие работы **«Сборка и тестирование компьютеров»** с другими работами. Чтобы не портить диаграмму декомпозиции, создайте FEO- диаграмму, на которой будут только стрелки работы **«Сборка и тестирование компьютеров».**

1. Выберите пункт меню **Diagram/Add FEO Diagram**.
2. В диалоге **Add New FEO Diagram** выберите тип и внесите имя диаграммы FEO.
3. Для определения диаграммы перейдите в **Diagram/ Diagram Properties** и во вкладку **Diagram Text** внесите определение.
4. Удалите лишние стрелки на диаграмме FEO. Результат показан на рис.3.

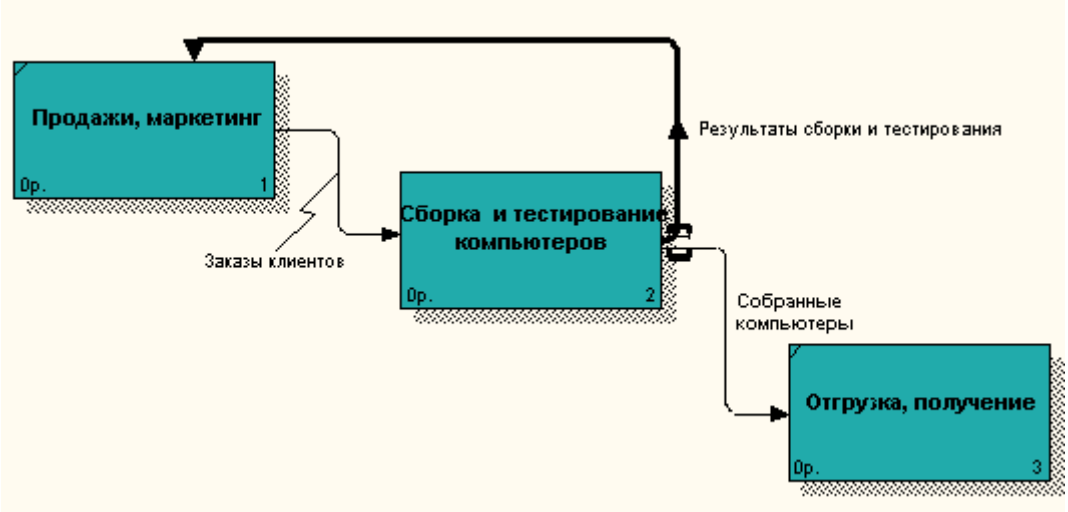


Рис.3. Диаграмма FEO

Для перехода между стандартной диаграммой, деревом узлов и FEO используйте кнопку  на палитре инструментов.

Каркас диаграммы.

На рис.4 показан типичный пример контекстной диаграммы с граничными рамками, которые называются каркасом диаграммы. Каркас содержит заголовок (верхняя часть рамки, табл.1) и подвал (нижняя часть, табл.2). Заголовок каркаса используется для отслеживания диаграммы в процессе моделирования. Нижняя

часть используется для идентификации и позиционирования в иерархии диаграмм. Значения полей каркаса задаются в диалоге Diagram Properties (в меню Edit/Diagram Properties).

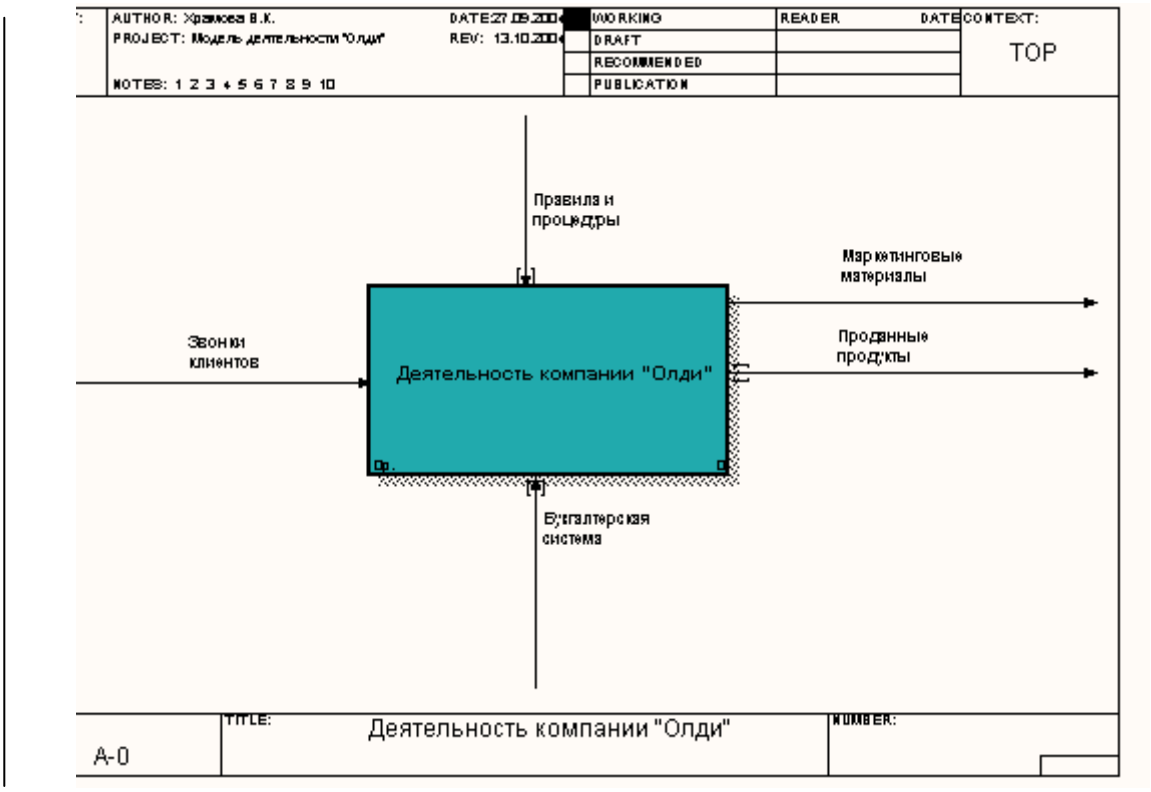


Рис.4. Контекстная диаграмма

Поля заголовка каркаса (слева направо)

Таблица 1.

Поле	Смысл
Used At	Используется для указания на родительскую работу в случае, если на текущую диаграмму ссылались посредством стрелки вызова.
Author, Date, Rev, Project	Имя создателя диаграммы, дата создания и имя проекта, в рамках которого была создана диаграмма. REV - дата последнего редактирования диаграммы.
Notes 1 2 3 4 5 6 7 8 9 10	Используется при проведении сеанса экспертизы. Эксперт должен (на бумажной копии диаграммы) указать число замечаний, вычеркивая цифру из списка каждый раз при внесении нового замечания.
Status	Статус отображает стадию создания диаграммы, отображая все этапы публикации.
Working	Новая диаграмма, кардинально обновленная диаграмма или новый автор диаграммы.
Draft	Диаграмма прошла первичную экспертизу и готова к дальнейшему обсуждению.
Recommended	Диаграмма и все ее сопровождающие документы прошли экспертизу. Новых изменений не ожидается.

Publication	Диаграмма готова к окончательной печати и публикации.
Reader	Имя читателя (эксперта).
Date	Дата прочтения (экспертизы).
Context	Схема расположения работ в диаграмме верхнего уровня. Работа, являющаяся родительской, показана темным прямоугольником, остальные - светлым. На контекстной диаграмме (A-0) показывается надпись TOP. В левом нижнем углу показывается номер по узлу родительской диаграммы.

Поля подвала каркаса (слева направо)

Таблица 2.

Поле	Смысл
Node	Номер узла диаграммы (номер родительской работы)
Title	Имя диаграммы. По умолчанию - имя родительской работы
Number	C-Number, уникальный номер версии диаграммы
Page	Номер страницы, может использоваться как номер страницы при формировании папки

Контрольные вопросы:

1. Как создать диаграмму дерева узлов?
2. Как создать FEO диаграмму?
3. Какую информацию содержит каркас диаграммы?
4. Какую информацию содержат поля подвала каркаса диаграммы?
5. В каком поле находится номер узла диаграммы?
6. В каком поле находится номер версии диаграммы?
7. В каком поле находится имя диаграммы?

Критерии оценки работы:

1. Полный ответ – 5 баллов.
2. Дополнительный уточняющий вопрос – 4 балла.
3. Краткий ответ – 3 балла.

ПРАКТИЧЕСКАЯ РАБОТА № 6

Создание отчетов в пакете BPwin

Цель работы:

Изучение способов создания отчетов в BPwin.

Исходные данные (задание):

BPwin имеет мощный инструмент генерации отчетов. Отчеты по модели вызываются из пункта меню **Report**. Всего имеется семь типов отчетов:

1. **Model Report**. Этот отчет включает информацию о контексте модели - имя модели, точку зрения, область, цель, имя автора, дату создания и др.
2. **Diagram Report**. Отчет по конкретной диаграмме. Включает список объектов (работ, стрелок, хранилищ данных, внешних ссылок и т.д.).
3. **Diagram Object Report**. Наиболее полный отчет по модели. Может включать полный список объектов модели (работ, стрелок с указанием их типа и др.) и свойства, определяемые пользователем.
4. **Activity Cost Report**. Отчет о результатах стоимостного анализа.
5. **Arrow Report**. Отчет по стрелкам. Может содержать информацию из словаря стрелок, информацию о работе-источнике, работе-назначении стрелки и информацию о разветвлении и слиянии стрелок.
6. **Data Usage Report**. Отчет о результатах связывания модели процессов и модели данных.
7. **Model Consistency Report**. Отчет, содержащий список синтаксических ошибок модели.

Синтаксические ошибки IDEF0 с точки зрения BPwin разделяются на три типа:

- во-первых, это ошибки, которые BPwin выявить не в состоянии. BPwin не позволяет анализировать синтаксис естественного языка (английского и русского) и смысл имен объектов и поэтому игнорирует ошибки этого типа. Выявление таких ошибок - ручная работа.
- ошибки второго типа BPwin просто не допускает. Например, каждая грань работы предназначена для определенного типа стрелок. BPwin просто не позволит создать на диаграмме IDEF0 внутреннюю стрелку, выходящую из левой грани работы и входящую в правую грань.
- третий тип ошибок BPwin позволяет допустить, но отмечает их. Полный их список можно получить в отчете Model Consistency Report. Это единственный неопциональный отчет в BPwin. Список ошибок может содержать, например, неименованные работы и стрелки (unnamed arrow, unnamed activity), несвязанные стрелки (unconnected border arrow), неразрешенные стрелки (unresolved (square tunneled) arrow connections), работы, не имеющие по крайней мере одной стрелки выхода и одной стрелки управления, и т.д.

При выборе пункта меню, который соответствует какому-либо отчету, появляется диалог настройки отчета. Для каждого из семи типов отчетов он выглядит по-

своему. Рассмотрим типичный диалог Arrow Report (рис.7). Раскрывающийся список Standard Reports позволяет выбрать один из стандартных отчетов. Стандартный отчет - это запоминаемая комбинация переключателей, флажков и других элементов управления диалога. Для создания собственного стандартного отчета необходимо задать опции отчета, ввести имя отчета в поле списка выбора и щелкнуть по кнопке New. BPwin сохраняет информацию о стандартном отчете в файле BPWINRPT.INI. Все определения этого файла доступны из любой модели. Единственное ограничение - свойства, определяемые пользователем (User Defined Properties). Они сохраняются в виде указателя и поэтому доступны только из родной модели. Стандартный отчет можно изменить или удалить.

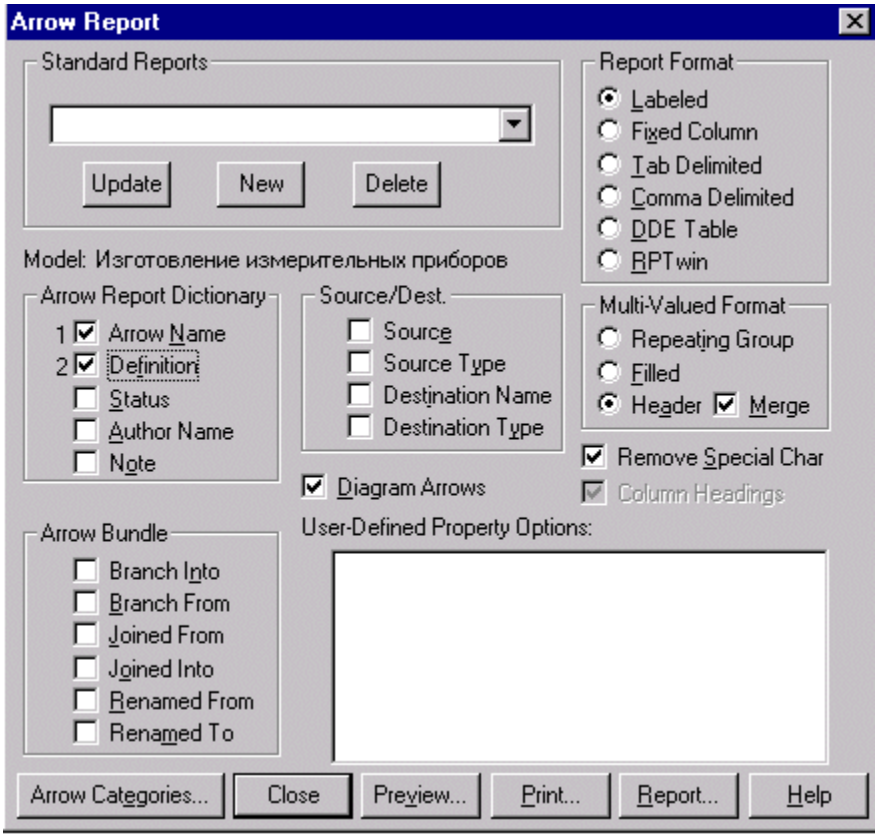


Рис.7.Диалог настройки отчета

В правом верхнем углу диалога находится группа управляющих элементов для выбора формата отчета. Доступны следующие форматы:

- **Labeled** - отчеты включают метку поля, затем, в следующей строке, печатается содержимое поля;
- **Fixed Column** - каждое поле печатается в собственной колонке;
- **Tab-Comma Delimited** - каждое поле печатается в собственной колонке. Колонки разделяются знаком табуляции или запятыми;
- **DDE Table** - данные передаются по DDE приложению, например MS Word или Excel;
- **RPTwin** - отчет создается в формате Platinum RPTwin - специализированного генератора отчетов, который входит в поставку BPwin.

Опция **Ordering** (на отчете по стрелкам отсутствует) сортирует данные по какому-либо значению.Опция Multi-Valued Format регулирует вывод полей в отчете при группировке данных:

- **Repeating Group** - детальные данные объединяются в одно поле, между значениями вставляется +.
- **Filled** - дублирование данных для каждого заголовка группы;
- **Header** (опция по умолчанию) - печатается заголовок группы, затем - детальная информация.

Задание №1**.** Создать отчет о контексте модели, используя различные опции.

Задание №2**.** Создать отчеты по конкретным диаграммам.

Задание №3**.** Создать полный отчет по модели.

Задание №4**.** Создать отчет о результатах стоимостного анализа.

Задание №5**.** Создать отчет по стрелкам.

Задание №6**.** Создать отчет, содержащий список синтаксических ошибок модели.

Контрольные вопросы:

1. Как создать отчет о контексте модели?
2. Как создать отчеты по конкретным диаграммам?
3. Как создать полный отчет по модели?
4. Как создать отчет о результатах стоимостного анализа?
5. Как создать отчет по стрелкам?
6. Как создать отчет, содержащий список синтаксических ошибок модели?

Критерии оценки работы:
1. Полный ответ – 5 баллов.
2. Дополнительный уточняющий вопрос – 4 балла.
3. Краткий ответ – 3 балла.

ПРАКТИЧЕСКАЯ РАБОТА № 7

Расщепление и слияние моделей.

Цель работы:

1. Научиться выполнять «Расщепление модели»
2. Научиться выполнять «Слияние модели».

Исходные данные (задание):

Возможность слияния и расщепления моделей обеспечивает коллективную работу над проектом. Так, руководитель проекта может создать декомпозицию верхнего уровня и дать задание аналитикам продолжить декомпозицию каждой ветви дерева в виде отдельных моделей. После окончания работы над отдельными ветвями все подмодели могут быть слиты в единую модель. С другой стороны, отдельная ветвь модели может быть отщеплена для использования в качестве независимой модели, для доработки или архивирования.

BPwin применяет для слияния и разветвления моделей стрелки вызова.

Для слияния необходимо выполнить следующие условия:

- Обе сливаемые модели должны быть открыты в BPwin;
- Имя-модели-источника, которое присоединяют к модели-цели, должно совпадать с именем стрелки вызова работы в модели-цели;
- Стрелка вызова должна исходить из недекомпозируемой работы (работа должна иметь диогональную черту в левом верхнем углу);
- Имена контекстной работы подсоединяемой модели-источника и работы на модели-цели, к которой мы подсоединяем модель источник, должны совпадать;
- Модель-источник должна иметь по крайней мере одну диаграмму декомпозиции.

Для слияния моделей нужно щелкнуть правой кнопкой мыши по работе со стрелкой вызова в модели-цели и во всплывающем меню выбрать пункт Merge Model.

Появляется диалог в котором следует указать опции слияния модели Рис. 2.

После подтверждения слияния (кнопка ОК) модель-источник подсоединяется к модели цели, стрелка вызова исчезает, а работа от которой отходила стрелка вызова становится декомпозируемой – к ней подсоединяется диаграмма декомпозиции первого уровня модели-источника. Стрелки, касающиеся работы на диаграмме модели цели, автоматически не мигрируют в декомпозицию, а отображаются как неразрешенные. Их следует тоннелировать вручную.

В процессе слияния модель-источник остается неизменной и к модели цели подключается фактически её копия.

Разделение моделей производится аналогично. Для отщепления ветви от модели следует щелкнуть правой кнопкой мыши по декомпозированной работе (работа не должна иметь диагональной черты в левом верхнем углу) и выбрать пункт Split Model. В появившемся диалоге Split Option следует указать имя создаваемой модели. После подтверждения расщепления в старой модели работа

станет недекомпозированной (признак-диагональная черта в левом верхнем углу), будет создана стрелка вызова, причем её имя будет совпадать с именем новой модели и будет создана новая модель, причем имя контекстной работы будет совпадать с именем работы, от которой была оторвана декомпозиция.

Расщепление модели

1. Посмотрите на Model Explorer и запомните содержимое дерева.
2. Перейдите на диаграмму A0. Правой кнопкой мыши щелкните по работе «Сборка и тестирование компьютеров» и выберите Split Model.
3. В диалог Split Option внесите имя новой модели «Сборка и тестирование компьютеров», установите опции как на рисунке и щелкните по ОК Рис.1.

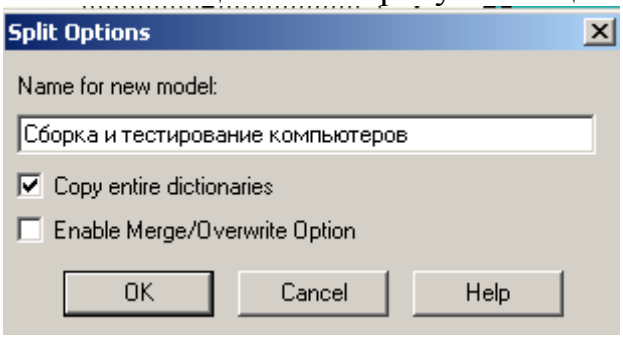


Рис.1. Диалог Split Option

4. Посмотрите на результат: в Model Explorer появилась новая модель, а на диаграмме A0 модели «Деятельность компании» появилась стрелка вызова «Сборка и тестирование компьютеров».
5. Создайте в модели «Сборка и тестирование компьютеров» новую стрелку, «Неисправные компоненты». На диаграмме A0 это будет граничная стрелка выхода, на диаграмме A0 - граничная стрелка выхода от работ «Сборка настольных компьютеров», «Тестирование компьютеров» и «Сборка ноутбуков».

Слияние модели

1. Посмотрите на Model Explorer и запомните содержимое дерева.
2. Перейдите на диаграмму A0 модели «Деятельность компании».
3. Правой кнопкой мыши щелкните по работе «Сборка и тестирование компьютеров» и выберите Merge Model.
4. В диалоге Merge Model включите опцию Cut/Paste entire dictionaries щелкните по ОК.

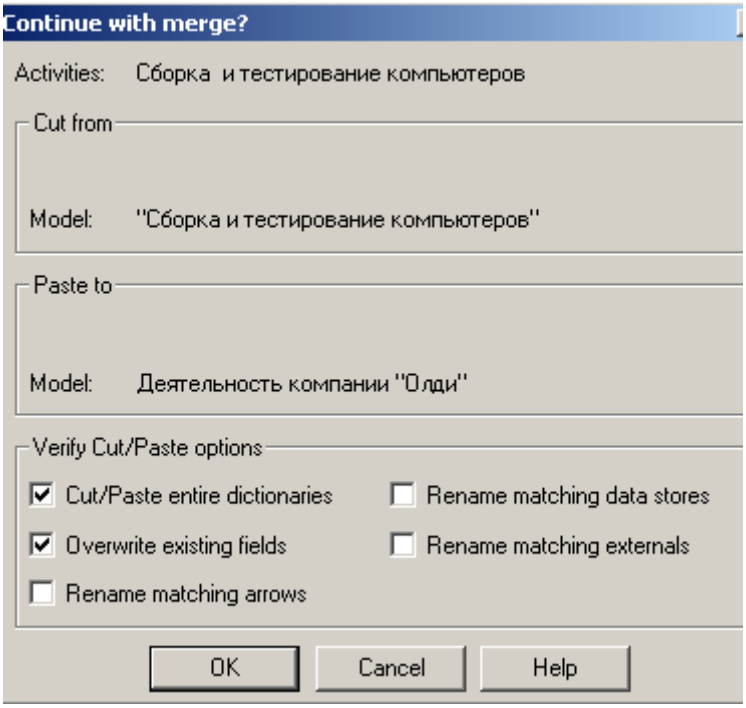


Рис.2. Диалог Continue with merge?

Посмотрите на результат. В Model Explorer видно, что две модели слились. Модель «Сборка и тестирование компьютеров» осталась и может быть сохранена в отдельном файле. На диаграмме A0 модели «Деятельность компании» исчезла стрелка вызова «Сборка и тестирование компьютеров». Появилась неразрешенная граничная стрелка «Неисправные компоненты». Направьте эту стрелку ко входу работы «Отгрузка и получение».

Контрольные вопросы:

1. Как выполнить расщепление модели?
2. Как выполнить слияние модели?

Критерии оценки работы:

1. Полный ответ – 5 баллов.
2. Дополнительный уточняющий вопрос – 4 балла.
3. Краткий ответ – 3 балла.

ПРАКТИЧЕСКАЯ РАБОТА № 8

Создание диаграммы IDEF3.

Цель работы:

1. Научиться выполнять «Создание диаграмм в стандарте IDEF3»
2. Научиться создавать перекрестки.
3. Научиться создавать сценарий работы.

Исходные данные (задание):

Перед выполнением лабораторной работы внимательно прочитайте ***«Методологию описания бизнес-процессов IDEF3»***

1. Перейдите на диаграмму A2 и декомпозируйте работу ***«Сборка настольных компьютеров»***. В диалоге ***Activity Box Count*** (рис. 1.) установите число работ 4 и нотацию IDEF3.

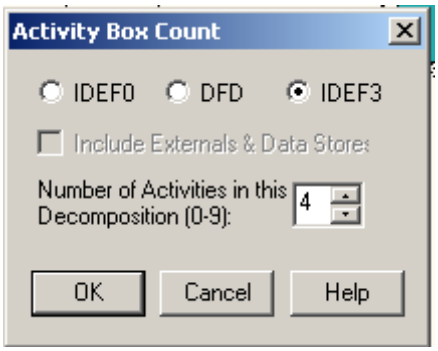


Рис.1. Выбор нотации IDEF3 в диалоге Activity Box Count

Возникает диаграмма IDEF3, содержащая работы (UOW). Правой кнопкой мыши щелкните по работе, выберите в контекстном меню Name и внесите имя работы «***Подготовка компонентов***». Затем во вкладке **Difinition** внесите определение «Подготавливаются все компоненты компьютера согласно спецификации заказа»

2. Во вкладку **UOW** внесите свойства работы (Табл.1.)

Таблица 1. Свойства UOW

Objiects	Компоненты: винчестеры, корпуса, материнские платы, видеокарты, звуковые карты, дисководы CD-ROM и флоппи, модемы, программное обеспечение
Facts	Доступные операционные системы: Windows 98, Windows NT, Windows 2000
Constrains	Установка модема требует установки дополнительного программного обеспечения

3. Внесите в диаграмму еще три работы (кнопка  .
Внесите имена следующих работ:
   - Установка материнской платы и винчестера
   - Установка модема
   - Установка дисковода CD-ROM

- Установка флоппи-дисковода
- Инсталляция операционной системы
- Инсталляция дополнительного программного обеспечения

4. С помощью кнопки  палитры инструментов создайте объект ссылки. Внесите имя объекта внешней ссылки «***Компоненты***».

Свяжите стрелкой объект ссылки и работу «***Подготовка компонентов».***

5. Свяжите стрелкой работы «***Подготовка компонентов***» (выход) и «***Установка материнской платы и винчестера***». Измените стиль стрелки на Object Flow.
   В IDEF3 имя стрелки может отсутствовать, хотя BPwin показывает отсутствие имени, как ошибку. Результат выполнения показан на рис.2.

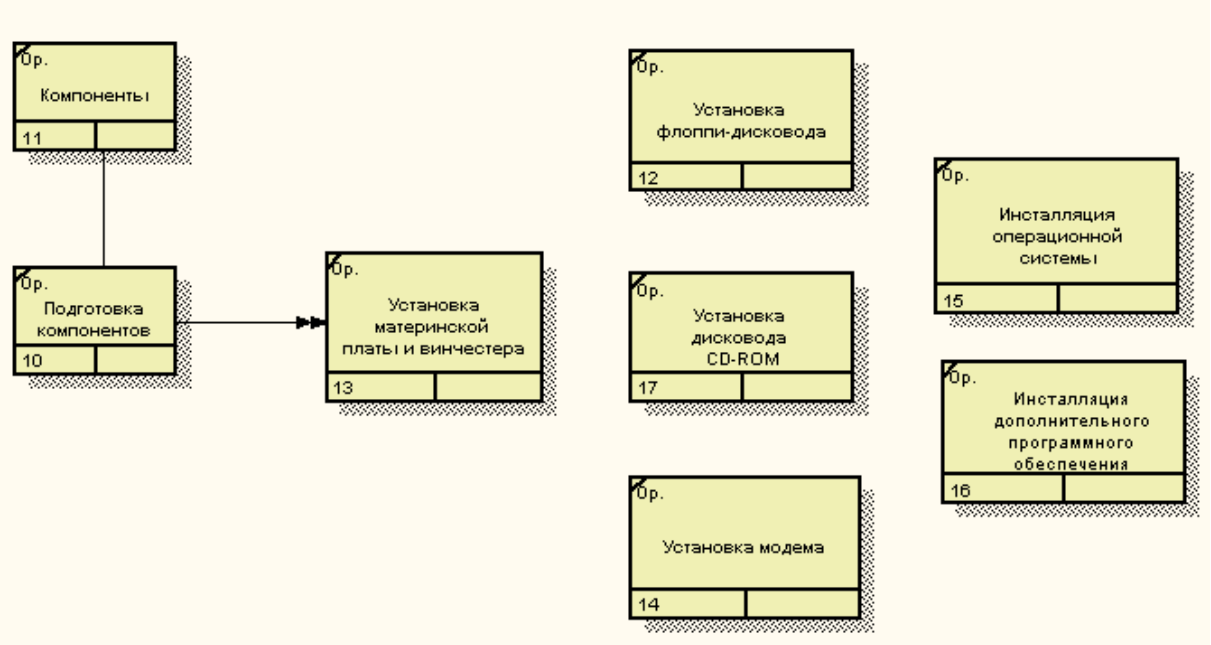


Рис.2. Результат создания UOW и объекта ссылки

6. С помощью кнопки  на палитре инструментов внесите два перекрестка типа асинхронного «или» и свяжите работы с перекрестками, как показано на рис.3.
7. Правой кнопкой щелкните по перекрестку для разветвления (fan-out), выберите Name и внесите имя «Компоненты, требуемые в спецификации заказа».
8. Создайте два перекрестка типа исключающего «ИЛИ» и свяжите работы, как показано на рис.3.

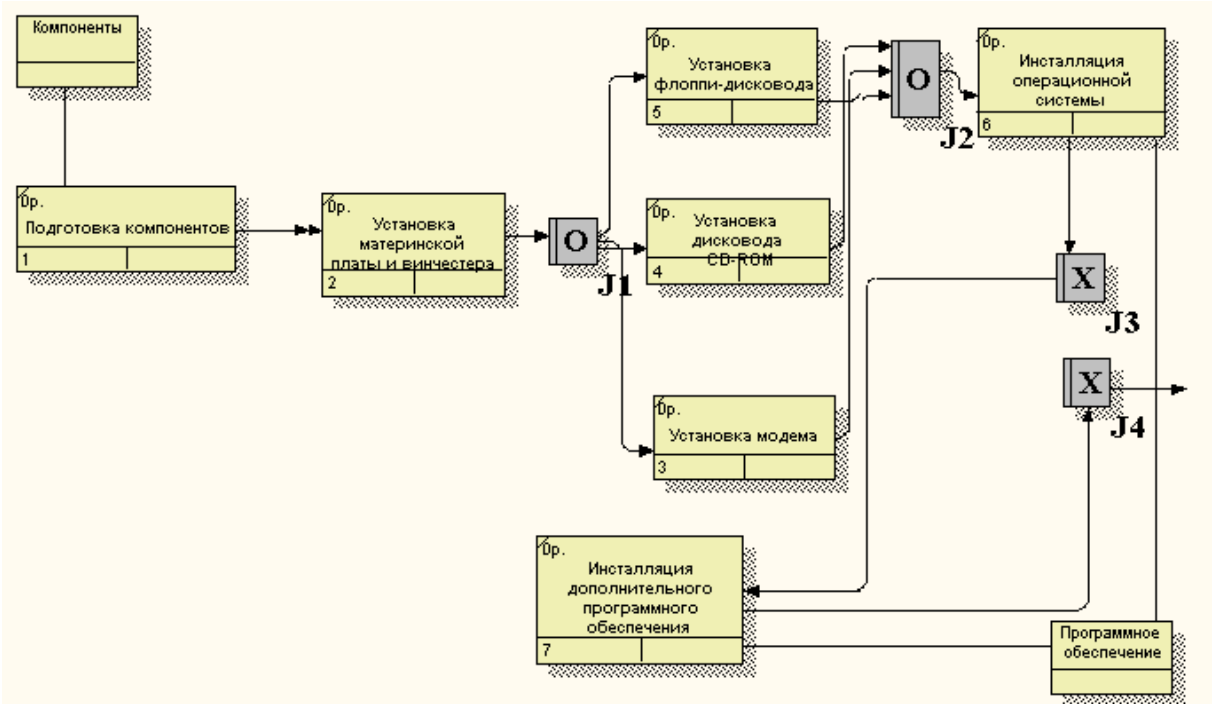


Рис.3. Результат выполнения работы

Создание сценария

1. Создайте диаграмму сценария на основе диаграммы IDEF3 «Сборка настольных компьютеров» (A22.1). Выберите пункт меню **Diagram/Add IDEF3 Scenario**. Высветится окно Add New IDEF3 Scenario Diagram, в окно Name of New Diagram введите имя сценария «Сценарий для сборки настольных компьютеров». Установите «галочку» в окно Copy contents jf sourse diagram. Нажмите кнопку ОК.
2. Удалите элементы, не входящие в сценарий. Результат выполнения на (рис.4)
3. Посмотреть созданные сценарии можно выполнив меню Diagram Manager, установив «точку» в IDEF3 scenario, выбрав нужный сценарий.

Контрольные вопросы:

1. Как создать диаграмму декомпозиции в стандарте IDEF3?
2. Как задать свойства для работы?
3. Как внести в диаграмму новую работу?
4. Как создать объект ссылки?
5. Как создать перекрестки на диаграмме?
6. Как создать сценарий диаграммы?
7. Как посмотреть созданные сценарии?

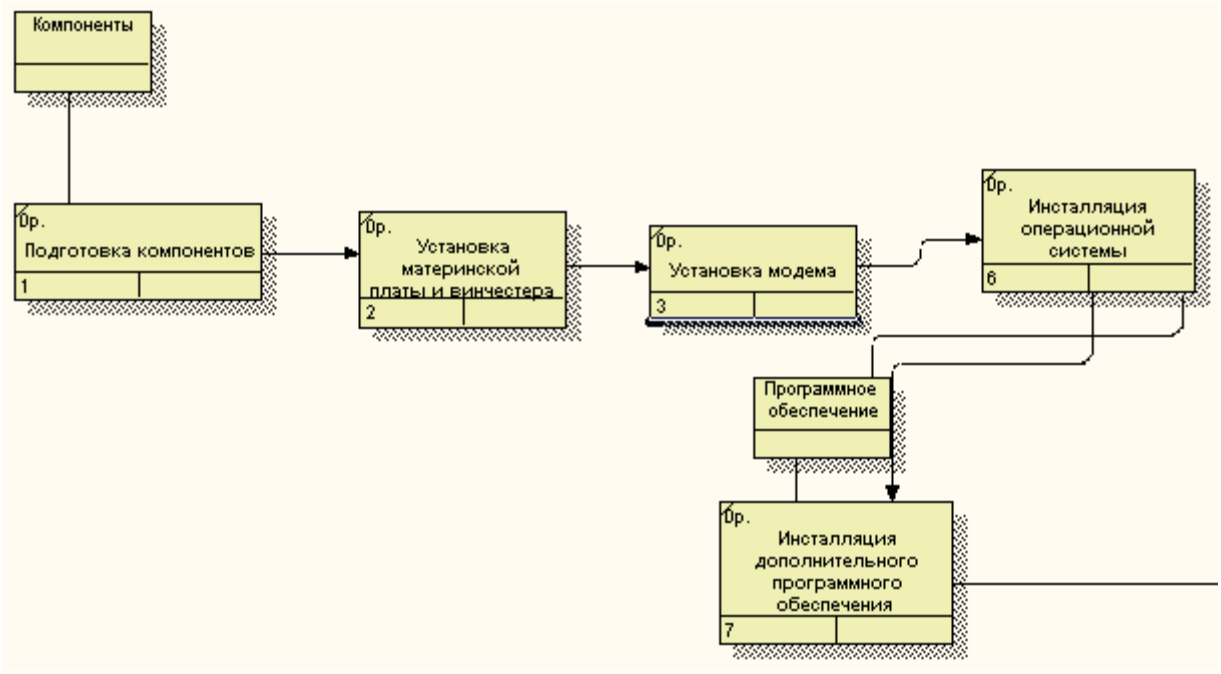


Рис.4. Сценарий

. Создание диаграммы IDEF3 (декомпозиция). Создание сценария.

Цель работы:

3. Научиться выполнять «Создание диаграмм в стандарте IDEF3»
4. Научиться создавать перекрестки

В результате проведения экспертизы с тестировщиками выявлена следующая информация:

a. Каждый тестировщик имеет собственную периферию (монитор, клавиатуру, мышь) для проверки компьютера;

b. Каждый тестировщик подсоединяет кабель питания и периферию для настольного компьютера и кабель питания для ноутбука;

c. Каждый тестировщик запускает с дискеты программу диагностики, которая тестирует компоненты компьютера;

d. Если программа диагностики определяет неработающий компонент, то тестировщик заменяет его исправным. Тестирование и замена компонентов проводится до тех пор, пока все компоненты компьютера не будут исправлены;

e. Каждый проверенный компьютер хранится до тех пор, пока диспетчер не даст распоряжение об отгрузке партии;

f. Неисправные компоненты направляются на отгрузку для возврата поставщикам.

Задание №1. На основании этой информации необходимо декомпозировать (в нотации IDEF3) работу «Тестирование компьютеров» диаграммы A2.

Создайте UOW:

- Подключение периферии;
- Запуск программы диагностики;
- Формирование партии;

- Замена неисправных компонентов;

Создайте 4 объекта ссылок:

- Периферия;
- Компьютер;
- Заказы;
- Компоненты

Соедините работы и объекты ссылок стрелками. Создайте перекрестки и задайте имена стрелок: «Исправные компьютеры», «Неисправные компьютеры»

Задание №2. Создать сценарий на основе диаграммы IDEF3 «Тестирование компьютеров» (A24.1), описывающий путь неисправных компонентов..

Создание сценария

4. На основе диаграммы IDEF3 «Тестирование компьютеров» (A24.1). создайте сценарий, описывающий путь неисправных компонентов. В сценарий должны входить только объекты, содержащие неисправные компоненты. Выберите пункт меню **Diagram/Add IDEF3 Scenario**. Высветится окно **Add New IDEF3 Scenario Diagram**, в окно **Name of New Diagram** введите имя сценария ***«Сценарий, описывающий путь неисправных компьютеров».*** Установите «**галочку**» в окно **Copy contents jf sourse diagram**. Нажмите кнопку ОК.
5. Удалите элементы, не входящие в сценарий. Результат выполнения на (рис.4)
6. Посмотреть созданные сценарии можно выполнив меню ***Diagram Manager***, установив «**точку**» в **IDEF3 scenario**, выбрав нужный сценарий.

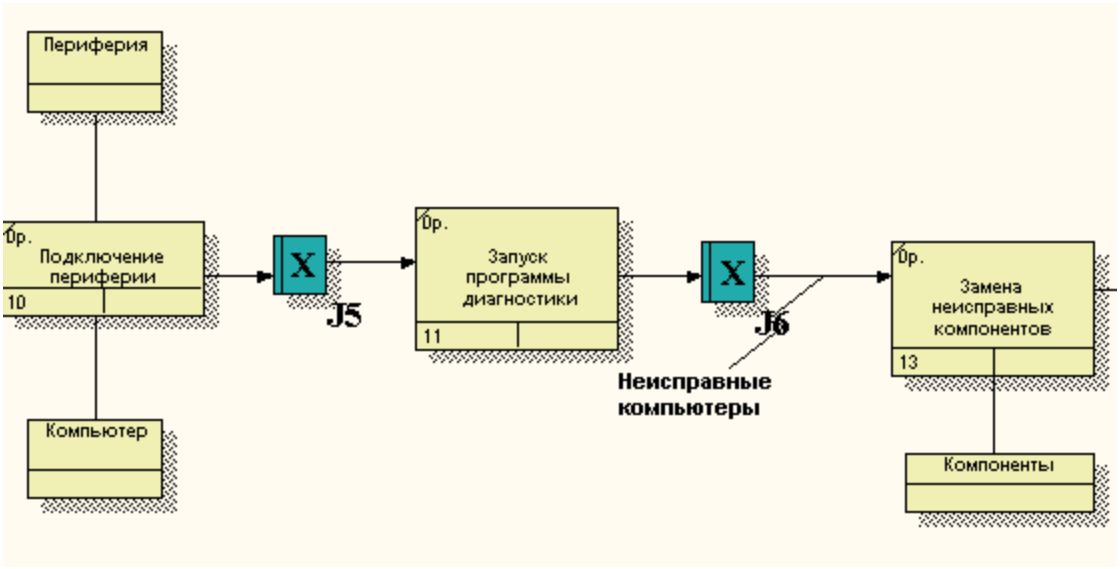


Рис.4. Сценарий, описывающий путь неисправных компьютеров

Контрольные вопросы:

1. Как создать диаграмму декомпозиции в стандарте IDEF3?
2. Как создать сценарий для работы?
3. Как посмотреть созданные сценарии?

Критерии оценки работы:
1. Полный ответ – 5 баллов.
2. Дополнительный уточняющий вопрос – 4 балла.
3. Краткий ответ – 3 балла.

ПРАКТИЧЕСКАЯ РАБОТА № 9

Стоимостной анализ (Activity Based Costing)

Цель работы:

Научиться выполнять расчеты общей стоимости работ.

Исходные данные (задание):

BPwin представляет аналитику два инструмента для оценки модели – стоимостный анализ, основанный на работах ABC и свойства определяемые пользователем UDP. ABC является широко распространенной методикой, используемой международными корпорациями и государственными организациями для идентификации истинных затрат в организации.

Стоимостной анализ представляет собой соглашение об учете, используемое для сбора затрат, связанных с работами, с целью определить общую стоимость процесса. С помощью стоимостного анализа можно решить такие задачи, как определение действительной стоимости производства продукта, идентификация работ, которые стоят больше всего (те, которые должны быть улучшены в первую очередь) и др.

При проведении стоимостного анализа сначала задаются единицы измерения времени и денег.

1. Выполните меню Model/Model Proporties выберите вкладку ABC Units (рис.1.) установите единицы измерения денег и времени –рубли и часы
2.

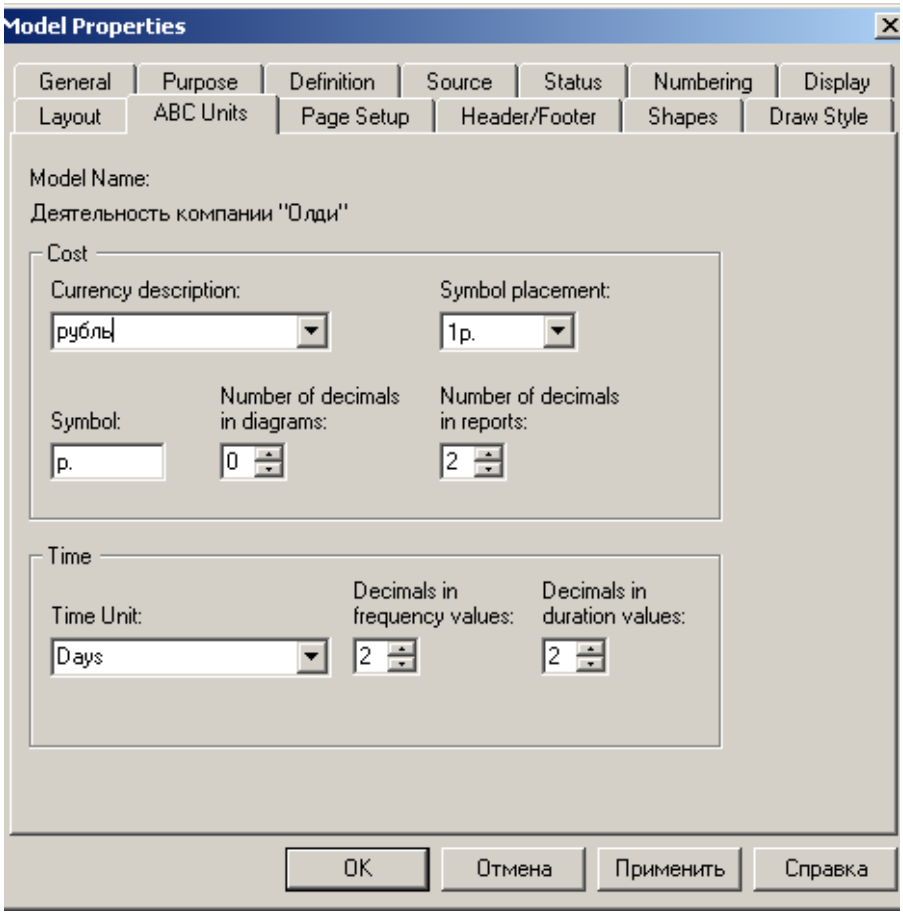


Рис.1. Вкладка ABC Units диалога Model Properties

3. Выполните Dictionary/Cost Center и в диалоге Cost Center Dictionary внесите название и определение центров затрат (Табл.1)

Таблица 1. Центры затрат ABC.

Центр затрат	***Определение***
Управление	Затраты на управление, связанные с составлением графика работ, формированием партий компьютеров, контролем над сборкой и тестированием
Рабочая сила	Затраты на оплату рабочих, занятых сборкой и тестированием компьютеров
Компоненты	Затраты на покупку компонентов

Для отображения стоимости каждой работы выполните Model/Model Properties и во вкладку Display диалога Model Properties включите опцию ABC Data (рис.2.)

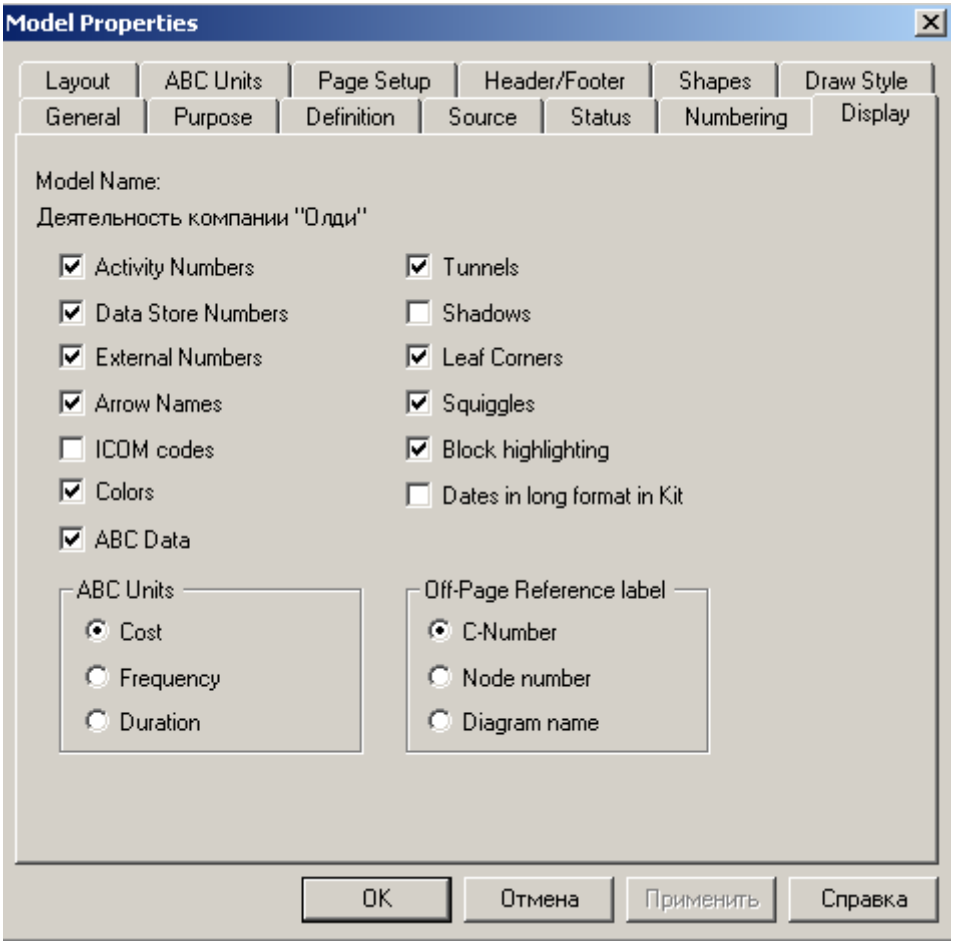


Рис.2. Вкладка Display диалога Model Properties

Для назначения стоимости работе следует установить курсор на нужную работу, щелкнуть правой кнопкой мыши и выбрать в контекстном меню Cost (рис.3.)

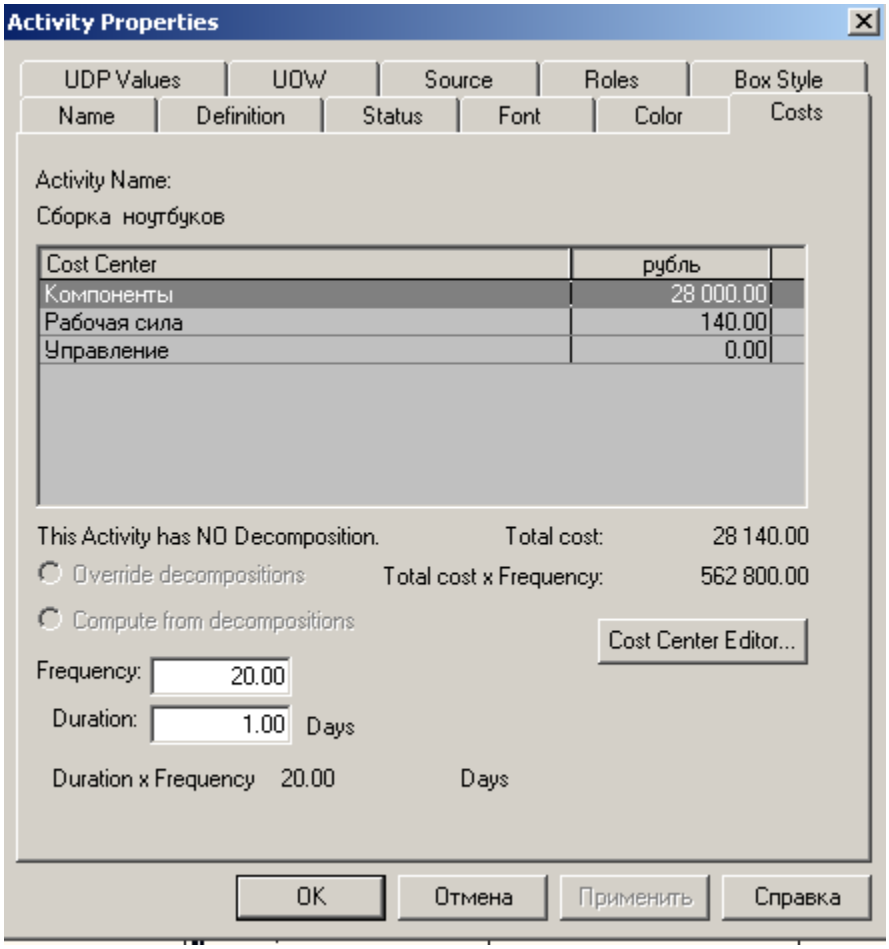


Рис.3 Вкладка Cost диалога Activity Properties

4. Для работ на диаграмме A2 внесите параметры ABC (Табл.2.)

Таблица 2. Стоимости работ на диаграмме A2.

Имя работы (Activity Name)	Центр затрат (Cost Center)	Сумма центра затрат (Cost Center)	Продолжительность (Duration), день	Частота (Frequency)
Отслеживание расписания и управление сборкой и тестированием	Управление	500,00	1,00	1,00
Сборка настольных компьютеров	Рабочая сила	100,00	1,00	12,00
	Компоненты	16000,00		
Сборка ноутбуков	Рабочая сила	140,00	1,00	20,00
	Компоненты	28000,00		
Тестирование компьютеров	Рабочая сила	60,00	1,00	32,00

Посмотрите результат – стоимость работы верхнего уровня. (Отображение стоимости в нижнем левом углу прямоугольника работы «Сборка и тестирование компьютеров»)
4. Сгенерируйте отчет Activity Cost Report

Контрольные вопросы:

1. Как установить для расчета стоимости единицы измерения денег и времени – рубли и часы
2. Как задать центр затрат и определение затрат?
3. Как посмотреть стоимость каждой работы?

Критерии оценки работы:

1. Полный ответ – 5 баллов.
2. Дополнительный уточняющий вопрос – 4 балла.
3. Краткий ответ – 3 балла.

ПРАКТИЧЕСКАЯ РАБОТА № 10

Использование категорий UDP.

Цель работы:

1. Научиться использовать категории UDP.
2. Научиться создавать формулы в отчете RPTwin.

Исходные данные (задание):

ABC позволяет оценить стоимостные и временные характеристики системы. Если стоимостных показателей недостаточно, то имеется возможность внесения собственных метрик - свойств, определенных пользователем UDP. UDP – это свойства, определяемые пользователем. UDP можно поставить в соответствие одно или несколько ключевых слов. Ключевые слова могут быть использованы для отбора UDP при печати отчетов или при присвоении свойств работам и стрелкам. Ключевые слова должны быть описаны в словаре UDP Keyword List (рис.1.).Для внесения нового ключевого слова следует щелкнуть по кнопке  и в таблице диалога UDP Keyword List задать значение ключевого слова.

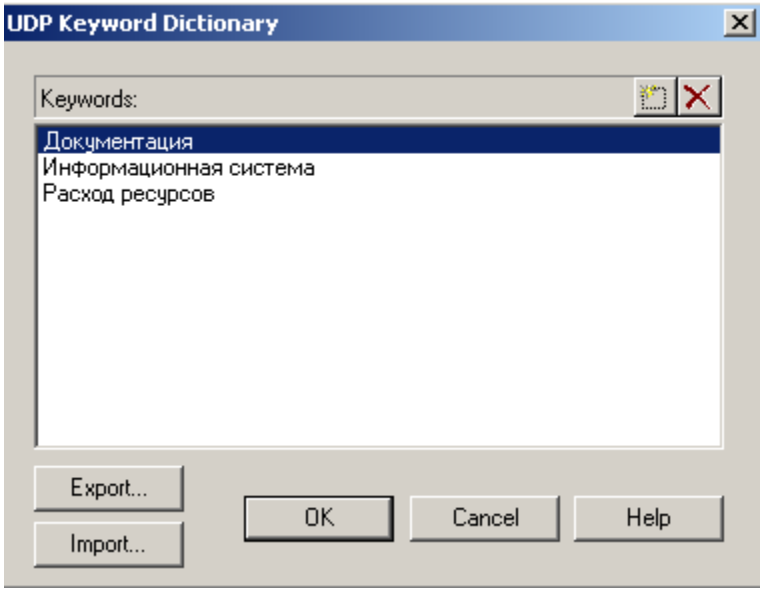


Рис.1. Диалог описания ключевых слов UDP.

Для создания нового свойства UDP следует в словаре ***UDP Dictionary*** перейти к нижней строке списка и дважды щелкнуть по полю Name. В поле UDP Type описывается тип свойства. Имеется возможность задания 18 различных типов UDP.

Для присвоения свойству ключевого слова следует перейти к полю **Keyword** и выбрать из списка необходимые ключевые слова. Одному свойству может соответствовать несколько разных ключевых слов, одно ключевое слово может соответствовать разным свойствам.

Каждой работе можно поставить в соответствие набор **UDP**. Для этого следует щелкнуть правой кнопкой мыши по работе и выбрать пункт меню **UDP**. Во вкладке **UDP Values** диалога ***Activity Properties*** можно задать значения UDP. Свойства типа List отображаются списком выбора, который заполнен предварительно

определенными значениями. Кнопка **Filter** служит для задания фильтра по ключевым словам.

Кнопка **Dictionary** вызывает диалог, который позволяет создавать и редактировать как UDP, так и ключевые слова UDP

1. Выполните меню ***Dictionary/UDP Keywords***, появится диалоговое окно ***UDP Keywords Dictionary***, в поле **Keywords** внесите следующие ключевые слова
   - Расход ресурсов
   - Документация
   - Информационная система
2. Создайте UDP. Для этого перейдите в ***Dictionary/UDP*** и в словарь внесите имя UDP, например «Приложение».
3. Для UDP типа **List** необходимо в поле **Value** задать список значений. Для UDP – «Приложение». Внесите значение «Модуль оформления заказов». Затем внесите другие значения в соответствии с табл.1. Для подключения к UDP ключевого слова перейдите к полю Keywords и щелкните по полю выбора.

Таблица 1. Наименование и свойства UDP

Наименование UDP	Тип UDP Datatype	Значение Value	Ключевое слово Keywords
Приложения	Text List (Multiple Selection)	Модуль оформления заказов. Модуль создания и контроля расписания выполнения работ. Модуль учета комплектующих и оборудования. Модуль процедур сборки и поиска неисправностей.	Информационная система
Дополнительная документация	Command List	Winword.EXE sample1.doc Winword.EXE sample2.doc POWERPNT.EXE sample3.ppt	Документация
История изменения	Paragraph Text		Документация
Загрязнение окружающей среды	Text List (Single Selection)	Очень высокое Высокое Среднее Низкое	
Расход электроэнергии	Real Number		Расход ресурсов

4. Для назначения UDP работе следует щелкнуть по ней правой кнопкой мыши и выбрать в контекстном меню UDP. Появится вкладка **UDP Values** диалога **Activity Properties** рис.2.
   Внесите значения UDP для работ (табл.2)

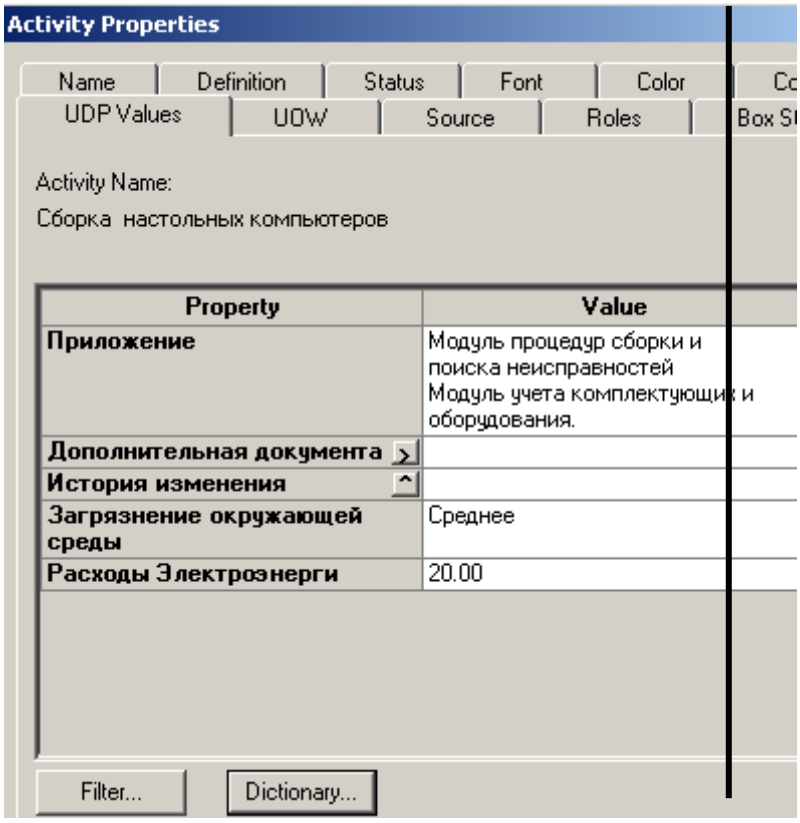


Рис.2. Вкладка UDP диалога Values Activity Properties

Таблица 2. Значения UDP

Имя работы (Activity Name)	Дополнительная документация	Приложения	История изменения	Расход электроэнергии	Загрязнение окружающей среды
Сборка настольных компьютеров		Модуль учета комплектующих и оборудования. Модуль процедур сборки и поиска неисправностей		20,00	Среднее
Сборка ноутбуков		Модуль учета комплектующих и оборудования. Модуль процедур сборки и поиска неисправностей		25,00	Среднее
Тестирование компьютеров		Модуль учета комплектующих и оборудования. Модуль процедур сборки и поиска		40,00	Среднее

		неисправностей			
Отслеживания расписания и управление сборкой и тестированием	Winword.EXE sample2.doc	Модуль создания и контроля расписания выполнения работ	История изменения спецификаций	10,00	Низкое

5. УБРАТЬ СТРОКУ После внесения UDP типа Command или Command List щелчок по кнопке >> приведет к запуску приложения.
6. В диалоге **Activity Properties** щелкните по кнопке **Filter**. В появившемся диалоге **Diagram object UDP Filter** Рис.3. отключите ключевые слова «***Информационная система***». Щелкните по ОК. В результате в диалоге **Activity Properties** не будут отображаться UDP с ключевыми словами «Информационная система». Свойства UDP можно присвоить не только работам, но и стрелкам.

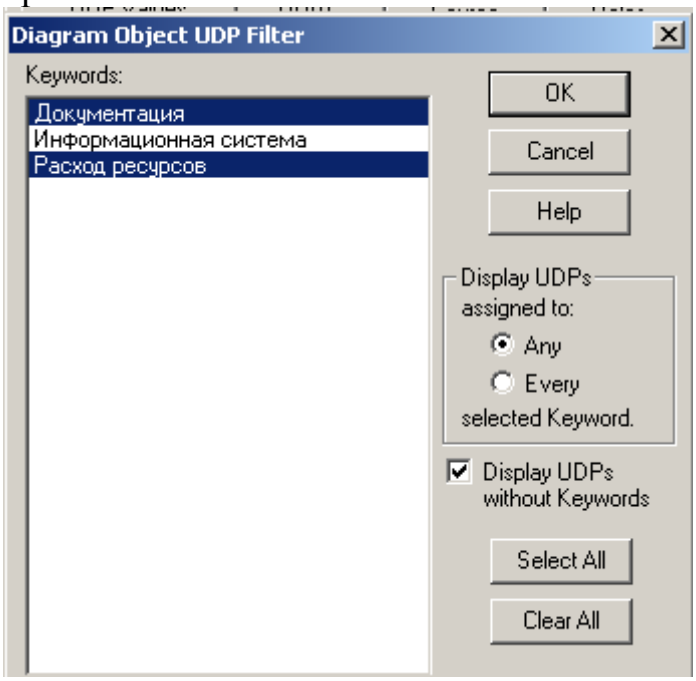


Рис.3. Диалог Diagram object UDP Filter

7. Посмотрите отчет по UDP,выполнив **Tools/Report/Diagram Objiect Report**. Выберите опции отчета:
   Start from Activity: A2. Сборка и тестирование компьютеров
   Number of Levels: 2
   User Defined Properties: Расход электроэнергии
   Report Format: RPTwin
8. Щелкните по кнопке **Report** . В появившемся диалоге «Сохранение файла» щелкните по кнопке «**Сохранить**». Запускается генератор отчетов RPTwin и появляется диалог New Report. Выберите тип отчета **Columnar**. Нажатие на кнопку  позволяет просмотреть отчет. Отразим в отчете суммарный расход электроэнергии

9. Выберите в меню ***Insert/Formula Field***, затем переместите маркер в секцию отчета **Page Footer**, затем щелкните один раз. Появляется диалог ***Formula Editor*** рис.4

.

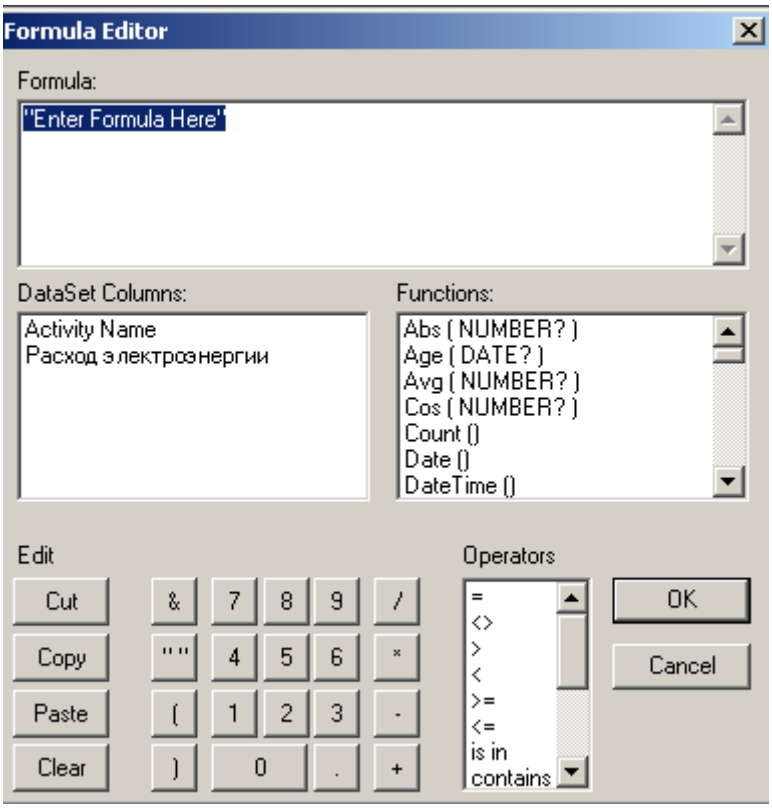


Рис.4. Диалог Formula Editor

10. В поле **Formula** формула внесите текст формулы:

Sum ({Расход электроэнергии»})

11. Щелкните по кнопке ОК. Отчет показывается в окне просмотра. В нижней части страницы расположено суммирующее поле – результат вычисления формулы.

Контрольные вопросы:

1. Как внести ключевые слова UDP?
2. Как создать UDP?
3. Как назначить UDP работе?
4. Как создать отчет в RPTwin?
5. Как создать формулу в отчете?

Критерии оценки работы:

1. Полный ответ – 5 баллов.
2. Дополнительный уточняющий вопрос – 4 балла.
3. Краткий ответ – 3 балла.

ПРАКТИЧЕСКАЯ РАБОТА № 11

Расщепление модели. Слияние расщепленной модели с исходной. Копирование работ

Цель работы:

1. Научиться выполнять «Расщепление модели»
2. Научиться выполнять слияние расщепленной модели с исходной.
3. Научиться выполнять копирование работ в другую модель и перемещать работы в той же самой модели.

Исходные данные (задание):

Перед выполнением лабораторной работы внимательно прочитайте разделы «Расщепление модели», «Слияние модели», «Тоннелирование стрелок»

1. Перейдите на диаграмму A0 и щелкните правой кнопкой мыши по работе ***«Огрузка, получение»***. В контекстном меню выберите команду **Split Model**. В появившемся диалоге **Split Option** установите опцию **Enable Merge/Overwrite Option**, внесите имя новой модели –***«Отгрузка и получение»*** и щелкните по ОК. Обратите внимание, что у работы ***«Отгрузка, получение»*** появилась стрелка вызова. **BPwin** создал также новую модель ***«Отгрузка и получение»***.
2. Внесите следующие свойства новой модели:

- Time Frame: AS-IS
- Purpose: Документировать работу «Отгрузка и получение»
- Viewpoint: Начальник отдела
- Definition: Модель создается для иллюстрации возможностей BPwin по расщеплению и слиянию моделей
- Scope: Работы по получению комплектующих и отправке готовой продукции

3. Декомпозируйте контекстную работу на три работы (Табл.1.)

Табл.1 Декомпозиция работы «Отгрузка и получение»

Имя работы (Activity Name)	Определение работы (Activity Definition)
Получить комплектующие	Физически получить комплектующие и сделать соответствующие записи в информационной системе
Доставить комплектующие	Доставить комплектующие сборщикам и тестировщикам
Отгрузить товар и возврат	Отгрузить товар клиентам и неисправные компоненты (возврат) поставщикам

4. Свяжите граничные стрелки как показано на Рис.1.

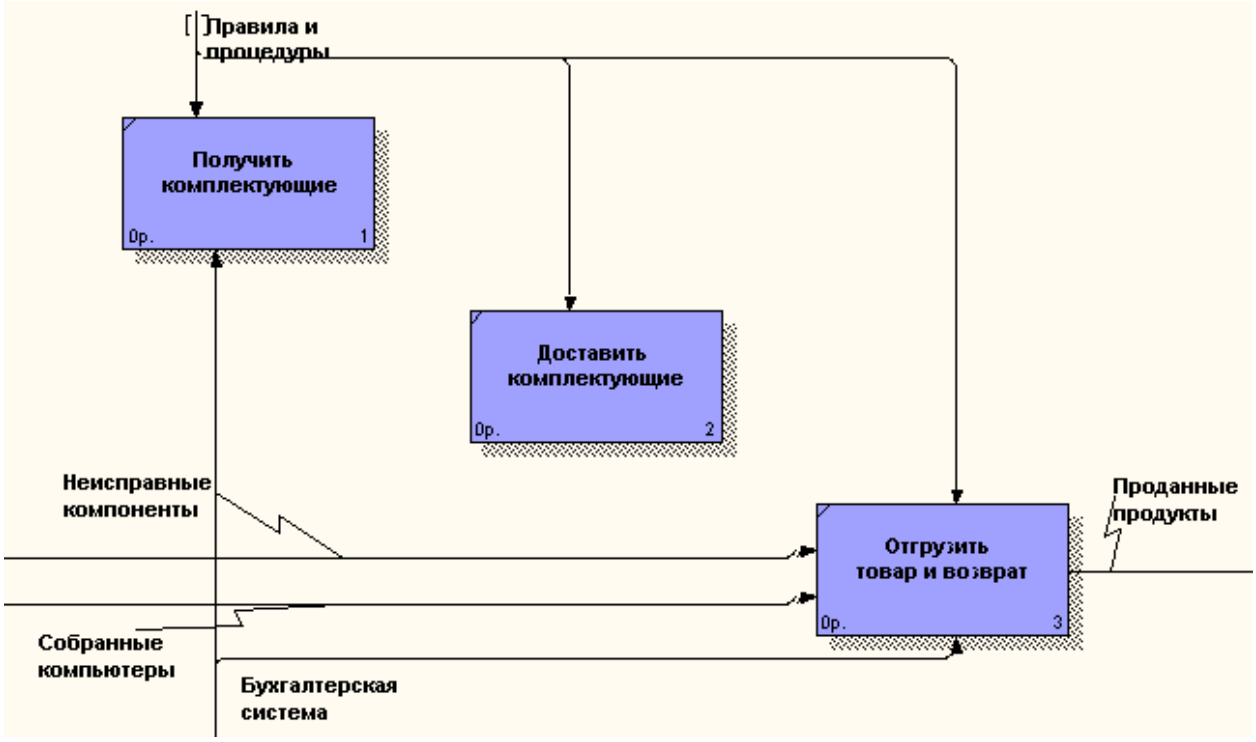


Рис.1. Внутренние стрелки на декомпозиции работы «Отгрузка и получение»

5. Внесите следующие внутренние и граничные стрелки (Табл.2)

Табл.2. Внутренние и граничные стрелки на декомпозиции работы «Отгрузка и получение»

Имя стрелки (Arrow Name)	Определение работы (Arrow Definition)
Возврат поставщику	Неисправные компоненты
Компоненты	Выберите название из списка (словаря)
Компоненты от поставщика	
Проверенные компоненты	Проверенные и подготовленные для передачи сборщикам и тестировщикам компоненты

6. Тоннелируйте граничные стрелки (Resolve Border Arrow). Результат выполнения показан на Рис. 2.

Слияние расщепленной модели с исходной

1. Перейдите в модель ***«Деятельность компании»***. На диаграмме А0 щелкните правой кнопкой мыши по работе ***«Отгрузка, получение».*** В контекстном меню **Merge Model**. В появившемся диалоге **Merge Model** установите опцию **Cut/Paste entire dictionaries** и щелкните по ОК. Обратите внимание, что у работы ***«Отгрузка и получение***»исчезла стрелка вызова и появилась новая декомпозиция. Появились новые стрелки с квадратными скобками. Тоннелируйте эти стрелки **(Resolve Border Arrow)**.
2. На диаграмме А0 тоннелируйте и свяжите стрелки согласно рисунка в Приложении 1.

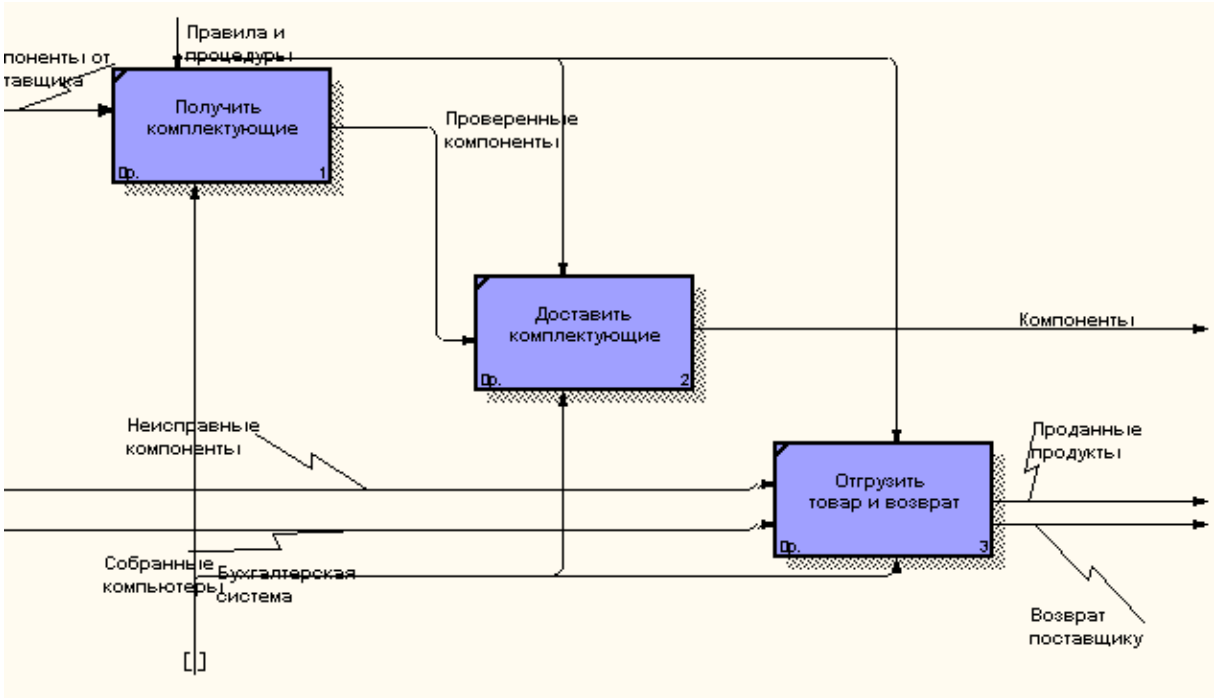


Рис.2. Результат выполнения упражнения

Копирование работ

Копирование работ в другую модель

1. Создайте новую модель «ТЕСТ». Декомпозируйте контекстную работу в новой модели, но не вносите имена работ.
2. Переключите **Model Explorer** во вкладку **Activity** в технике drap&drop перенесите какую-нибудь работу из модели «Деятельность компании» на диаграмму декомпозиции модели «ТЕСТ». В появившемся диалоге **Continue with Merge?** Установите опцию **Paste/ Merge entire dictionaries** и щелкните по ОК. В результате работа из модели ***«Деятельность компании»*** копируется на новую диаграмму модели «ТЕСТ».

Перемещение работ в той же самой модели

1. Щелкните по работе в модели «ТЕСТ» и переместите работу на место не названной работы на другой диаграмме. В появившемся диалоге **Continue**

with Merge? щелкните по ОК. в результате работа переносится из одной диаграммы на другую.

Контрольные вопросы:

1. Как выполнить расщепление модели?
2. Как выполнить слияние расщепленной модели с исходной?
3. Как выполнить копирование работ в другую модель?

Критерии оценки работы:
1. Полный ответ – 5 баллов.
2. Дополнительный уточняющий вопрос – 4 балла.
3. Краткий ответ – 3 балла.

ПРАКТИЧЕСКАЯ РАБОТА № 12

. Создание модели TO-BE (реинжиниринг бизнес-процессов)

Цель работы:

Научиться создавать модели TO-BE

Исходные данные (задание):

Модель **TO-BE** создается на основе анализа модели **AS-IS**. Анализ может проводиться как по формальным признакам (отсутствие выходов или управлений у работ, отсутствие обратных связей и т.д.), так и по неформальным – на основе знаний предметной области.

Допустим, в результате анализа принимается решение реорганизовать функции производства и тестирования компьютеров и оставить функциональности ***«Продажи маркетинг»*** и ***«Отгрузка и получение»*** пока без изменений.

Принято решение сформировать отдел дизайна, который должен формировать конфигурацию компьютеров, разрабатывать корпоративные стандарты, подбирать приемлемых поставщиков, разрабатывать инструкции по сборке, процедуры тестирования и устранения неполадок для всего производственного отдела.

Работа ***«Сборка и тестирование компьютеров»*** должна быть реорганизована и названа ***«Производство продукта»***. Будут созданы работы ***«Разработать конфигурацию», «Планировать производство»*** и ***«Собрать продукт».***

Рассмотрим новые роли персонала:

- Дизайнер должен разрабатывать систему;
- Дизайнер должен разрабатывать стандарты на продукцию, документировать и передавать спецификации в отдел маркетинга и продаж;
- Дизайнер должен определять, какие компоненты (аппаратные и программные) должны закупаться для сборки компьютеров,;
- Дизайнер должен обеспечивать документацией и управлять процедурами сборки, тестирования и устранения неполадок.

Функции диспетчера в работе ***«Сборка и тестирование компьютеров»*** должны быть заменены на функции планировщика.

- Планировщик должен обрабатывать заказы клиентов и генерировать заказы на сборку;
- Планировщик должен получит коммерческий прогноз из отдела маркетинга и формировать требования на закупку компонент;
- Планировщик должен собирать информацию от поставщиков и должен быть ответственен за оформление заказов на поставку;
- Планировщик должен составлять расписания производства на основании заказов на сборку, полученных в результате работы ***«Планировать производство»***;
- Планировщик должен получать копии заказов клиентов и отвечать за упаковку и комплектацию заказанных компьютеров, передаваемых в работу ***«Отгрузка и получение»*** .

Задание состоит из пяти этапов. Выполняя каждый этап, Вы должны использовать приобретенные навыки.

1. Расщепление и модификация модели.
2. Слияние расщепленной модели с исходной моделью.
3. Использование **Model Explorer** для реорганизации дерева декомпозиции.
4. Модификация диаграммы IDEF3 ***«Собрать продукт»*** с целью отображения новой информации.
5. Добавление декомпозиции ***«Продажи и маркетинг»***

!!! Перед выполнением данного задания необходимо создать копию выполненной работы «Деятельность компании «Олди» »

Этап 1. Расщепление и модификация модели.

1. Измените свойства модели ***«Деятельность компании***, выполнив Model/Model Properties

- Model Name: Предлагаемая модель компании.
- Time Frame TO-BE.
- Purpose: Документировать предлагаемые изменения бизнес-процессов компании.

2. Переименуйте работу ***«Сборка и тестирование компьютеров»*** в ***«Производство продукта».*** Расщепите эту работу в модель с тем же названием.
3. Модифицируйте отщепленную модель. Переместите работу ***«Тестирование компьютеров»*** с диаграммы A0 ***«Производство продукта»*** на диаграмму A2.1 ***«Сборка настольных компьютеров»***
4. Переименуйте работу ***«Сборка настольных компьютеров»*** на диаграмме A0 в ***«Сборка продукта»***
5. Удалите работу ***«Сборка ноутбуков»***
6. Переименуйте стрелку ***«Заказы на настольные компьютеры»*** в ***«Заказы на изготовление»***
7. Переименуйте работу ***«Отслеживание расписания и управление сборкой и тестированием»*** в ***«Планирование производства»***
8. Создайте работу ***«Разработать конфигурацию»***
9. Создайте ветвь стрелки ***«Персонал производственного отдела»***, назовите её «***Дизайнер***» и направьте как механизм к работе ***«Разработать конфигурацию»***
10. Создайте стрелку ***«Стандарты на продукцию»*** и направьте её от выхода «Разработать конфигурацию» к границе диаграммы. Тоннелируйте эту стрелку **(Resolve Border Arrow**). Создайте ветвь этой стрелки, идущую к управлению работы ***«Планирование производства»***, и назовите её «Списком необходимых компонентов»
11. Удалите стрелку ***«Правила сборки и тестирования»***. Создайте ветвь стрелки ***«Стандарты на продукцию»***, идущую к управлению работы ***«Сборка продукта»***, и назовите её ***«Правилами сборки и тестирования»***.
12. Переименуйте стрелку «***Диспетчер***», в «***Планировщика производства»***

13. Добавьте стрелку ***«Прогноз продаж»*** как граничную управляющую к работе ***«Планирование производства»***
14. Добавьте стрелку ***«Информация от поставщика»*** как граничную управляющую к работе ***«Планирование производства»***
15. Добавьте стрелку ***«Заказ поставщику»*** как граничную стрелку выхода от работы ***«Планирование производства»***.
16. Тоннелируйте эти стрелки (Resolve Border Arrow).
17. На диаграмме A0 тоннелируйте стрелку (Resolve Border Arrow) ***«Собранные компьютеры»*** и свяжите её на диаграмме A0 с выходом работы «Сборка продукта».

Этап 2. Слияние расщепленной модели с исходной моделью.

1. Перейдите к работе ***«Производство продукта»*** в модели ***«Деятельность компании «ОЛДИ»»*** . Щелкните правой кнопкой мыши по работе. В контекстном меню выберите Merge Model. В появившемся диалоге Merge Model установите опцию Cut/Paste entire dictionaries, опцию Overwrite existing fields и щелкните по ОК.
   Модели должны слиться
2. На диаграмме A0 тоннелируйте стрелки (Resolve Border Arrow) ***«Информация от поставщика»*** и ***«Заказ поставщику»***.
3. Направьте стрелку ***«Прогноз продаж»*** с выхода ***«Продажи и маркетинг»*** на управление ***«Производство продукта»***.
4. Направьте стрелку ***«Стандарты на продукцию»*** с выхода ***«Производство продукта»*** на управление ***«Продажи и маркетинг»***.
5. Удалите ветвь стрелки управления ***«Правила и процедуры»*** работы ***«Производство продукта».***
6. Закройте модель ***«Производство продукта»***.

Этап 3. Использование Model Explorer для реорганизации дерева декомпозиции.

Существуют причины, по которым работа «***Разработать конфигурацию»*** должна быть на верхнем уровне, на диаграмме **A0**. действительно, дизайнер разрабатывает стандарты на продукцию, включая правила сборки и тестирования, и список необходимых для закупки компонентов. Тем самым дизайнер управляет производством продукта в целом, кроме того, управляет работой **«Продажи и маркетинг».**

Было бы логично перенести эту работу на уровень выше

Используя возможности Model Explorer , перенесите работу ***«Разработать конфигурацию»*** с диаграммы A2 ***«Производство продукта»*** на диаграмму A0

Разрешите и перенаправьте стрелки согласно рисункам, приведенным в Приложении 2.

Этап 4. Модификация диаграммы IDEF3 «Собрать продукт» с целью отображения новой информации.

Так же как в модели **AS-IS** , сборка продукта состоит из сборки компонентов и установки программного обеспечения. Однако теперь в работу ***«Сборка продукта»*** включена работа ***«Тестирование компьютеров»***.

Тестирование начинается после окончания процесса сборки компьютера и окончания процесса установки программного обеспечения. Если компьютер неисправен, в процессе тестирования у него заменяют компоненты, информация о неисправных компонентах может быть направлена на работу ***«Подготовка компонентов»***. Такая информация может помочь более тщательно подготавливать компоненты к сборке. Результатом процесса тестирования являются заказанные компьютеры и неисправные компоненты.

Модифицируйте диаграмму **IDEF3** ***«Сборка продукта»*** в соответствии с приведенной информацией. Результат представлен на рисунке, приведенном в Приложении 2.

Этап 5. Декомпозиция работы «Продажи и маркетинг»

Работа по продажам и маркетингу заключается в ответах на телефонные звонки клиентов, предоставлении клиентам информации о ценах, оформлении заказов, внесении заказов в ИС и исследовании рынка.

На основе этой информации декомпозируйте работу ***«Продажи и маркетинг» (IDEF0)***.

Создайте следующие работы:

- Предоставление информации о ценах;
- Оформлении заказов;
- Исследование рынка

Контрольные вопросы:

1. Как выполнить расщепление модели?
2. Как выполнить слияние расщепленной модели с исходной?
3. Как посмотреть созданные сценарии?
4. Как выполнить модификацию модели?
5. Как выполнить декомпозицию работы?

Критерии оценки работы:

1. Полный ответ – 5 баллов.
2. Дополнительный уточняющий вопрос – 4 балла.
3. Краткий ответ – 3 балла.

ПРАКТИЧЕСКАЯ РАБОТА № 13

Создание диаграммы DFD. Использование Off-Page Reference на диаграмме DFD

Цель работы:

1. Научиться создавать диаграммы DFD
2. Научиться использовать межстраничные ссылки Off Page Reference на диаграмме DFD.

Исходные данные (задание):

Создание диаграммы DFD

При оформлении заказа важно проверить, существует ли такой клиент в базе данных и, если не существует, внести его в базу данных и затем оформить заказ. Оформление заказа начинается со звонка клиента. В процессе оформления заказа база данных клиентов может просматриваться и редактироваться. Заказ должен включать как информацию о клиенте, так и информацию о заказанных продуктах. Оформление заказа подразумевает чтение и запись информации о прочих заказах.

В процессе декомпозиции согласно правилам **DFD** необходимо преобразовать граничные стрелки во внутренние, начинающиеся и заканчивающиеся на внешних ссылках

1. Декомпозируйте работу ***«Оформление заказов»*** на диаграмме A2.
2. В диалоге Activity Box Count выберите количество работ 2 и нотацию DFD рис.1.

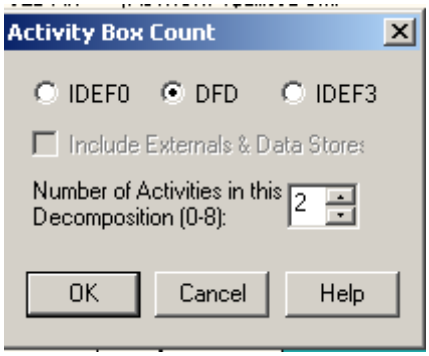


Рис.1. Выбор нотации DFD в диалоге Activity Box Count

3. Щелкните по ОК и внесите новую диаграмму, DFD A22, имена работ
- Проверка и внесение клиента;
- Внесение заказа

4. Используя кнопку  на палитре инструментов, внесите хранилища данных:
- Список клиентов;
- Список продуктов;
- Список заказов

5. Удалите граничные стрелки с диаграммы DFD A22
6. Используя кнопку  на палитре инструментов, внесите внешнюю ссылку:
- Звонки клиентов

7. Создайте внутренние ссылки согласно Рис.2. При именовании стрелок используйте словарь.

8.

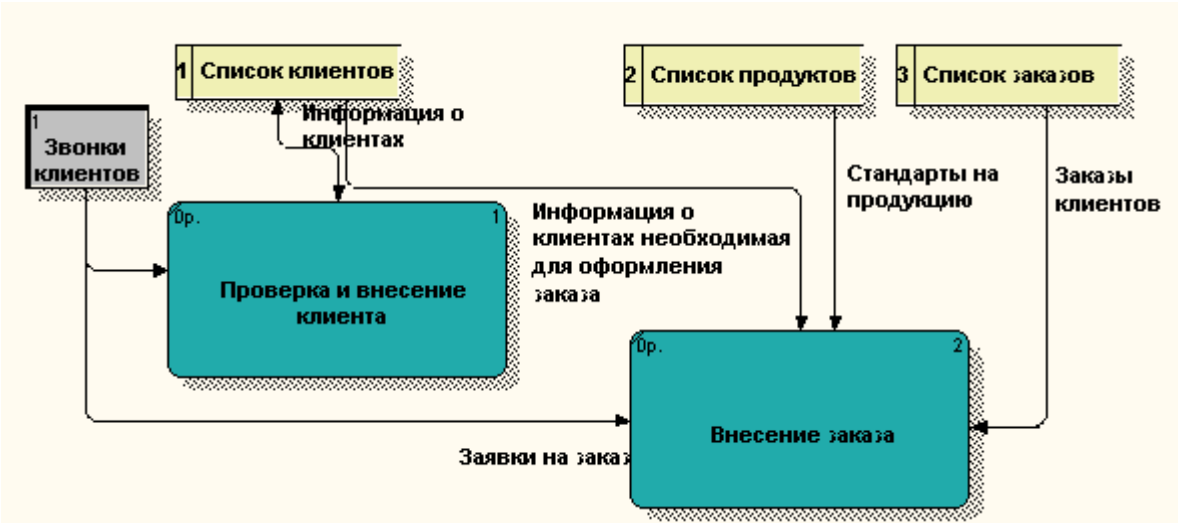


Рис.2. Диаграмма A22

9. Обратите внимание, что стрелки «***Информация о клиентах***» и ***«Заказы клиентов»*** двунаправленные. Для того чтобы сделать стрелку двунаправленной, щелкните правой кнопкой мыши по стрелке, выберите в контекстном меню пункт **Style** и во вкладке **Style** выберите опцию **Bidirectional**
10. На родительской диаграмме A2 тоннелируйте (Change to Tunnel) стрелки, подходящие и исходящие из работы «Оформление заказов» Рис.3.

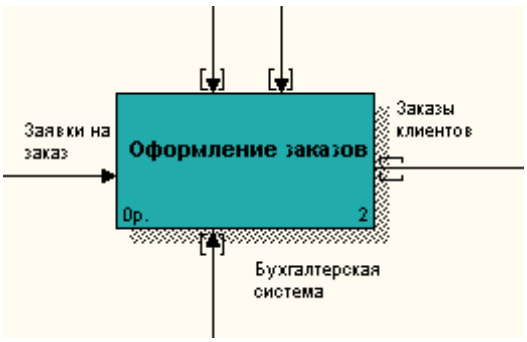


Рис.3. Диаграмма A2

Использование Off-Page Reference на диаграмме DFD

Межстраничные ссылки (Off Page Reference).

Нотация DFD включает межстраничные ссылки - инструмент, позволяющий описать переход стрелки (т.е. передачу данных или объектов) с одной диаграммы на другую. Для создания межстраничной ссылки на диаграмме DFD следует создать новую граничную стрелку. У границы диаграммы эта стрелка будет помечена квадратными скобками, так же как неразрешенная стрелка на диаграмме IDEF0. Затем следует щелкнуть правой кнопкой мыши по квадратным скобкам и выбрать в контекстном меню пункт Off Page Reference.

Появляется диалог Off Page Arrow Reference. В нем необходимо указать диаграмму, на которую будет направлена стрелка, и, если это программа в нотации IDEF0, границу, от которой будет исходить стрелка (Destination border).

В результате будет создана межстраничная ссылка как на диаграмме-источнике, так и на диаграмме назначении. Межстраничная ссылка может быть

помечена как C-number диаграммы, как номер диаграммы по узлу (как на рис.5.) или как имя диаграммы. Для изменения метки следует перейти в меню Model/Model Properties и во вкладке Display диалога Model Properties и в группе Off Page Reference label выбрать нужную опцию.

Некоторые стрелки с диаграмм IDEF0 и DFD (не только родительских) могут показываться на диаграмме DFD. Для отображения таких стрелок используется инструмент Off-Page Reference.

1. Декомпозируйте работу ***«Исследование рынка»*** на диаграмме A2 на диаграмму DFD. Удалите граничные стрелки. Создайте следующие работы.:
   - Разработка прогнозов продаж
   - Разработка маркетинговых материалов
   - Привлечение новых клиентов
2. Используя кнопку  на палитре инструментов, внесите хранилища данных:
   - Список клиентов
   - Список продуктов;
   - Список заказов
3. Добавьте две внешние ссылки:
   - Маркетинговые материалы
   - Прогноз продаж
4. Свяжите объекты диаграммы **DFD** стрелками, как показано на Рис.4.

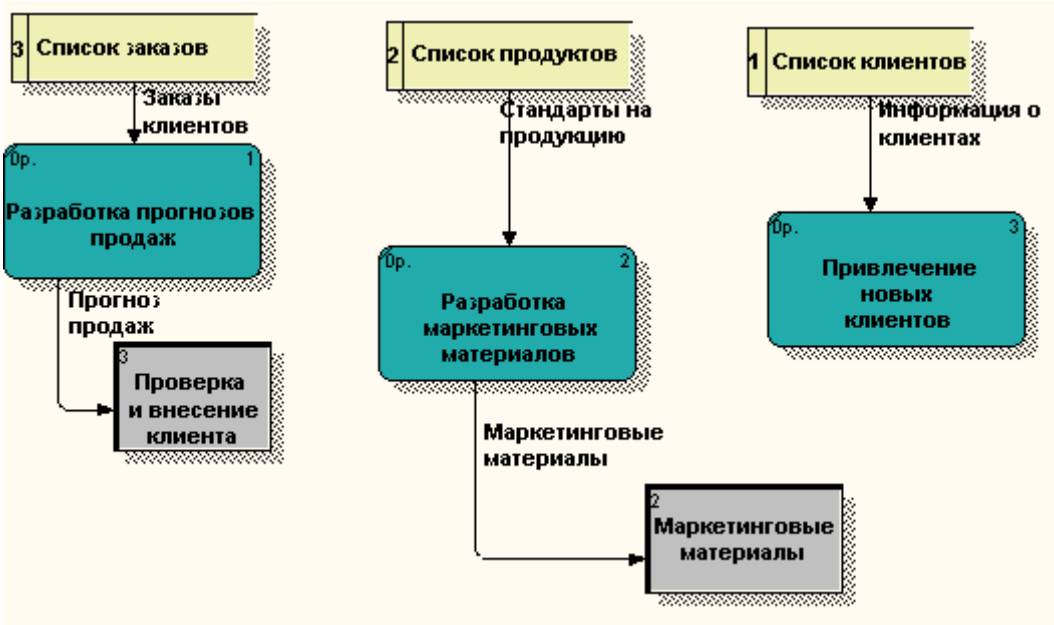


Рис.4. Диаграмма A22

5. На родительской диаграмме A2 тоннелируйте (Правой кнопкой мыши по тоннели, выбрать Arrow Tonnel, затем Change …to Tunnel) стрелки, подходящие и исходящие из работы «Исследование рынка»
6. В случае внесения новых клиентов в работу «Проверка и внесение клиента» на диаграмме A22 ***«Оформление заказов»*** информация должна направляться к работе ***«Привлечение новых клиентов»*** диаграммы ***A23 «Исследование рынка»***. Для этого необходимо использовать инструмент Off-Page Reference. На диаграмме A22

«Оформление заказов» создайте новую граничную стрелку, исходящую из работы ***«Проверка и внесение клиента»***, назовите ее ***«Информацией о новом клиенте»***.

7. Правой кнопкой мыши щелкните по наконечнику стрелки и выберите в меню Off-Page Reference. В появившемся диалоге Off-Page Arrow Reference Рис.5. выберите в качестве диаграммы A23D ***«Исследование рынка»***. Нажмите на кнопку OK and Go To Diagram.

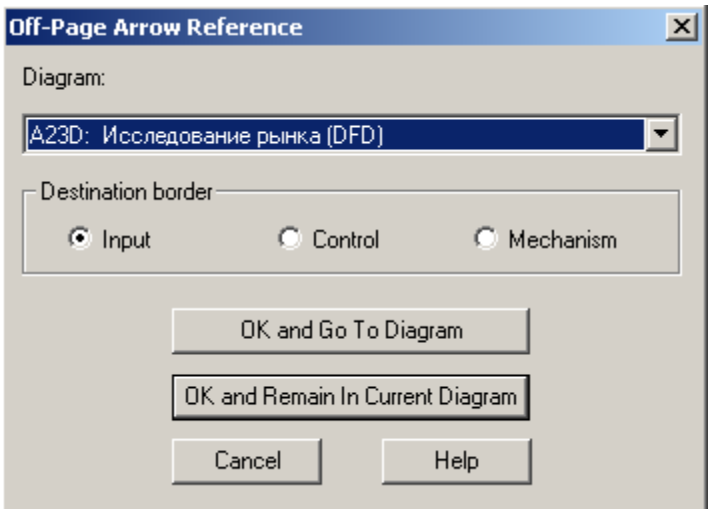


Рис.5 Создание межстраничной ссылки на диаграмме DFD

8. Перейдите в меню Model/ Model Proporties во вкладку Display. Установите опцию Off-Page Reference Label- Node number.

9. Перейдите на диаграмму A23D ***«Исследование рынка»*** и направьте стрелку ***«Информацией о новом клиенте»*** на вход работы ***«Привлечение новых клиентов»***. Результат представлен на Рис.6.

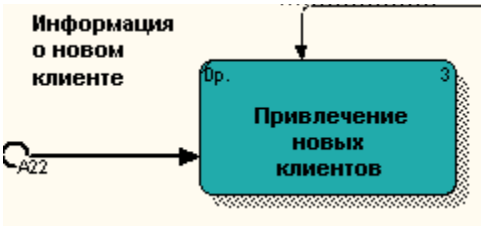


Рис.6. Межстраничная ссылка на диаграмме A23

Контрольные вопросы:

1. Как создать диаграмму DFD?
2. С помощью какой кнопки на палитре инструментов можно создать хранилище данных?
3. Как создать межстраничные ссылки ?

Критерии оценки работы:

1. Полный ответ – 5 баллов.
2. Дополнительный уточняющий вопрос – 4 балла.
3. Краткий ответ – 3 балла.

ПРАКТИЧЕСКАЯ РАБОТА № 14

Создание логического уровня модели данных
Создание сущностей и атрибутов на диаграмме

Цель работы: Изучение процесса создания объектов модели данных сущностей и атрибутов, используя панель инструментов; также процесса создания новых, переименования, удаления атрибутов в сущности; освоения навыков задания свойств сущностям и атрибутам.

.

Исходные данные (задание):

Основные понятия работы - сущность и атрибут

Создание модели данных в ERwin, как правило, начинается с создания логической модели данных.

В ERwin сначала создается сущность, и ей задается имя, потом в ней создаются атрибуты, и одному или нескольким из них при необходимости задается параметр Primary Key (первичный ключ).

Задания:

1. Создать структуру модели данных, состоящую из следующих сущностей: Заказ, Заказчик, Дизайн и Реализация.

2. Для каждой сущности соответственно создать следующие атрибуты:

Номер заказа. Дата заказа, Наименование продукции, Количество для сущности Заказ.

Номер заказчика. Фамилия, Имя, Отчество, Адрес, Телефон, Статус заказчика для сущности Заказчик.

Номер чертежа. Размер, Цвет, Стиль для сущности Дизайн.

Номер реализации. Дата реализации, Наименование продукции. Количество, Стоимость для сущности Реализация.

Задать свойство Primary Key всем первым атрибутам в списках выше.

1. Для каждой сущности задать свойство Definition по своему усмотрению.

2. Для атрибутов задать свойства Definition и Domain по своему усмотрению.

Технология работы

1. **Создание сущности**. Для того, чтобы начать создание сущности необходимо открыть или создать файл проекта ERwin в меню File, стандартным методом. Затем, нажав на кнопку сущности на панели инструментов щелкнуть в любом месте диаграммы, затем задать имя сущности.

2. **Создание атрибутов**. После создания сущности, нужно создать в ней атрибут, для этого необходимо правой кнопкой мыши щелкнуть на сущности и в контекстном меню выбрать Attributes... После этого, в правой части появившегося диалога, снизу, нажать на кнопку New; и задать имя нужного атрибута. Для редактирования имени существует кнопка Rename, для удаления - Delete. Для задания уже созданному атрибуту свойства Primary Key. Для отображения первичного ключа на диаграмме, нужно в не занятом диаграммами месте щелкнуть

правой кнопкой мыши и в entity display выделить Primary Key Designator.

3. **Задание свойств сущностям и атрибутам**. Для задания свойства Definition сущности нужно щелкнув по сущности правой кнопкой и в контекстном меню выбрать пункт Entity Properties, в появившемся диалоге выбрать закладку Definition и ввести нужный текст. Для задания аналогичного свойства атрибуту необходимо также щелкнуть по сущности выбрав пункт Attributes, и в появившемся диалоге выбрать закладку Definition. Для задания типа данных атрибуту нужно выбрать в том же диалоговом окне вкладку General и выбрать нужный домен для атрибута. Для смены стандартной иконки для сущности, во вкладке Icon нужно щелкнуть по раскрывающемуся списку и выбрать иконку, если список пуст, то, используя кнопку Import можно импортировать любой тип иконки. Для отображения иконки сущности на диаграмме, нужно в не занятом диаграммами месте щелкнуть правой кнопкой мыши и в Entity Display выделить Entity Icon.

Контрольные вопросы

1. В чём назначение стандарта IDEF1?
2. Основные понятия нотации IDEF1.
3. Какие уровни представления и уровни отображения имеет ERwin? Как происходит переключение между уровнями?
4. Каким образом осуществляется взаимосвязь между двумя отдельными сущностями?
5. Что называется моделью данных и концептуальной схемой?
6. Что описывает сущность в IDEF1X и в чём её отличие от сущности в 1DEF1?
7. Назначение атрибутов сущностей в IDEF1X.
8. Как графически описывается сущность в диаграмме IDEF1X?
9. Как идентифицировать уникальным образом запись сущности?
10. Какие правила существуют для выбора первичного ключа?
11. Как классифицируются сущности в IDEF1X?

Критерии оценки работы:

1. Полный ответ – 5 баллов.
2. Дополнительный уточняющий вопрос – 4 балла.
3. Краткий ответ – 3 балла.

ПРАКТИЧЕСКАЯ РАБОТА № 15

Создание логического уровня модели данных
Создание связей между сущностями

Цель работы: .Изучение процесса создания объектов модели данных - связей, используя панель инструментов, также освоения навыков задания свойств связям.

Исходные данные (задание):

Основное понятие работы - связь

Связь является логическим соотношением между сущностями. Каждая связь должна именоваться глаголом или глагольной фразой (Relationship Verb Phrases). Имя связи выражает некоторое ограничение или бизнес-правило и облегчает чтение диаграммы. ERwin использует два типа связей идентифицирующая и неидентифицирующая. Идентифицирующая связь устанавливается между независимой (родительский конец связи) и зависимой (дочерний конец связи) сущностями. Когда рисуется идентифицирующая связь, ERwin автоматически преобразует дочернюю сущность в зависимую. Зависимая сущность изображается прямоугольником со скругленными углами. При установлении идентифицирующей связи атрибуты первичного ключа родительской сущности автоматически переносятся в состав первичного ключа дочерней сущности.

При установлении неидентифицирующей связи дочерняя сущность остается независимой, а атрибуты первичного ключа родительской сущности мигрируют в состав неключевых компонентов родительской сущности. Неидентифицирующая связь служит для связывания независимых сущностей.

Задания:

1. Создать идентифицирующую связь между сущностями Заказчик и Заказ, Заказ и Дизайн. Чтобы сущность Заказ была дочерней для сущности Заказчик, а для сущности Дизайн - родительской.

2. Создать неидентифицирующую связь между сущностями Заказ и Реализация. Чтобы сущность Заказ была родительской для сущности Реализация.

3. Задать свойство Verb Phrase для всех связей по принципу Заказчик размещает Заказ, Дизайн описывает Заказ, Заказ обеспечивает Реализацию.

Технология работы

1. Создание связи. Для создания связи между сущностями нужно нажать соответствующую кнопку на панели инструментов (в зависимости от типа нужной связи), щелкнуть на родительской сущности, затем на дочерней.

2. Задание свойства Verb Phrase. Для задания свойства Verb Phrase нужно вызвать диалог Relationship, щелкнув по связи правой кнопкой и выбрав пункт Relationship Properties..Затем выбрать вкладку General и вписать в поле Parent-to-Chaild необходимый глагол или глагольную фразу. Для отображения свойства Verb Phrase на диаграмме, нужно в не занятом диаграммами месте щелкнуть правой кнопкой мыши и в Relationship Display выделить Verb Phrase.

Контрольные вопросы

1. Для чего устанавливается связь между сущностями в IDEF1X?
2. Какие типы связей существуют между сущностями?
3. Как графически отличить зависимую и независимую сущности, идентифицирующие и неидентифицирующие связи?
4. Для чего и как задаются свойства сущностям и атрибутам?
5. Для чего предназначен диалог Relationship?

Критерии оценки работы:

1. Полный ответ – 5 баллов.
2. Дополнительный уточняющий вопрос – 4 балла.
3. Краткий ответ – 3 балла.

ПРАКТИЧЕСКАЯ РАБОТА № 16

Создание логического уровня модели данных
Индексирование

Цель работы: Изучение процесса задания свойств объектам модели данных - атрибутам; получение навыков использования индексов в ERwin.

Исходные данные (задание):

Основное понятие работы - индекс

В таблице БД данные обычнбо хранятся в том же порядке, в котором их ввели в таблицу. Многие реляционные СУБД имеют страничную организацию, при которой физически таблица может храниться фрагментарно в разных областях диска, причем строки таблицы располагаются на страницах неупорядоченно. Хотя такой способ хранения и позволяет быстро вводить новые данные, но для того чтобы, найти нужную строку, придется просмотреть всю таблицу. В промышленных системах каждая может содержать миллионы строк, поэтому простой перебор ведет к катастрофическому падению производительности ИС.

Каждая сущность должна иметь по крайней мере один потенциальный ключ. Многие сущности имеют только один потенциальный ключ. Такой ключ становится первичным. Некоторые сущности могут иметь более одного возможного ключа. Тогда один из них становится первичным, а остальные -альтернативными ключами. Альтернативный ключ (**Alternate Key**) - это потенциальный ключ, не ставший первичным. ERwin позволяет выделить атрибуты альтернативных ключей, и по умолчанию в дальнейшем, при генерации схемы БД, по этим атрибутам будет генерироваться уникальный индекс.

При работе ИС часто бывает необходимо обеспечить доступ к нескольким экземплярам сущности, объединенным каким-либо одним признаком. Для повышения производительности в этом случае используются неуникальные индексы. ERwin позволяет на уровне логической модели назначить атрибуты, которые будут участвовать в неуникальных индексах. Атрибуты, участвующие в неуникальных индексах называются **Inversion Entries** (инверсионные входы). Inversion Entries - это атрибут или группа атрибутов, которые не определяют экземпляр сущности уникальным образом, но часто используются для обращения к экземплярам сущности. ERwin генерирует неуникальный индекс для каждого Inversion Entry.

Задание:

Задать индексы для атрибутов сущностей Заказ, Заказчик и Дизайн. Для сущности Заказчик объединить в группу альтернативного ключа (Alternate Key) сущности Имя и Фамилия, и в инверсионный вход (Inversion Entry) - Адрес и Телефон. Для сущности Заказ: в инверсионный вход - Дата заказа и Количество. Для сущности Дизайн объединить в группу альтернативного ключа (Alternate Key) сущности Цвет и Стиль, и в инверсионный вход (Inversion Entry) - Размер.

Технология работы

Для создания индексов (на логическом уровне) нужно выбрать в меню

Model\Key Groups, далее в списке Entity выбрать нужную сущность, в списке Key Group отображается ключ (например. Primary Key). Ниже в закладке Members списка Available Attributes отображаются неиндексированные атрибуты, а в списке Key Group Members - индексированные атрибуты. При нажатии кнопки New выводится диалог New Key Groups, где вы задаете имя индекса и тип индекса (Alternate Key или Inversion Entry). В списке Key Group появляется заданный ключ, которому нужно присвоить значение атрибута из списка Available Attributes, выбрав атрибут нужно щелкнуть на кнопку перехода, после чего атрибут появится в правой части окна списка Key Group Members, атрибуту будет присвоено значение созданного типа ключа. С помощью кнопок New, Delete, Rename можно соответственно создавать, переименовывать и удалять индексы. (Индексы можно задавать и на физическом уровне воспользовавшись меню Model\ Indexes).

Контрольные вопросы

1. Какие существуют виды ключей и как они устанавливаются для каждой сущности?
2. Что называется нормализацией данных? Как связана нормализация данных с FA?
3. Характеристика и назначение домена. Как вызывается диалог «Доменный словарь»?
4. Как создать домен в логической модели и новый домен? Как переопределить свойства домена?
5. Дать понятие индекса и его назначения для СУБД.
6. Что называется альтернативным ключом, как его создать и включить атрибут в качестве ключа?
7. Как создать отдельные индексы на основе ключей?

Критерии оценки работы:

1. Полный ответ – 5 баллов.
2. Дополнительный уточняющий вопрос – 4 балла.
3. Краткий ответ – 3 балла.

ПРАКТИЧЕСКАЯ РАБОТА № 17

Создание логического уровня модели данных
Иерархия наследования

Цель работы: Изучение процесса создания иерархии наследования, и освоение навыков в создании и редактировании иерархии наследования

Исходные данные (задание):

.

Основное понятие работы - иерархия наследования

Иерархия наследования (или иерархия категорий) представляет собой особый тип объединения сущностей, которые разделяют общие характеристики. Например, заказчики, могут иметь разный юридический статус. Из их общих свойств можно сформировать обобщенную сущность (родовой предок) Заказчик, чтобы представить информацию, общую для всех типов. Специфическая для каждого типа информация может быть расположена в категориальных сущностях (потомках) Статус заказчика и Статус заказчика 1.

Обычно иерархию наследования создают, когда несколько сущностей имеют общие по смыслу атрибуты, либо когда сущности имеют общие по смыслу связи, либо когда это диктуется бизнес-правилами. Иерархии категорий делятся на два типа - полные и неполные. В полной категории одному экземпляру родового предка обязательно соответствует экземпляр в каком-либо потомке. Если категория еще не выстроена полностью и в родовом предке могут существовать экземпляры, которые не имеют соответствующих экземпляров в потомках, то такая категория будет неполной. Иерархии категорий также могут представлять собой комбинацию из полных и неполных типов.

Задание:

Создать иерархию наследования полную категорию для сущности Заказчик с именами таблиц Статус заказчика и Статус заказчика 1.

Технология работы

Сначала создаем сущности Статус заказчика и Статус заказчика1, затем создаем атрибуты: Организация, Должность, Лицевой счет для сущности Статус заказчика; и Данные паспорта для сущности Статус заказчика1.

Затем щелкаем по значку категории на панели инструментов левой кнопкой мыши, затем по сущности Статус заказчика. Для установления второй связи щелкаем по значку категории на панели инструментов, затем по символу категории установленной первой связи, затем по сущности Статус заказчика 1.

Контрольные вопросы

1. Что называется иерархией наследования, для чего и как её создают?
2. Какие типы иерархии категорий существуют, как они различаются на диаграмме? Как редактировать категории?
3. Что относится к основным компонентам диаграммы ERwin? Назначение

каждого компонента и взаимосвязь с иерархией наследования

4. Какие существуют типы зависимых сущностей и какие из них используются в иерархии наследования?

5. Что называется дискриминатором и для чего он указывается?

Критерии оценки работы:

1. Полный ответ – 5 баллов.
2. Дополнительный уточняющий вопрос – 4 балла.
3. Краткий ответ – 3 балла.

ПРАКТИЧЕСКАЯ РАБОТА № 18

Создание логического уровня модели данных
Подмножества модели и хранимые отображения

Цель работы: Изучение процесса создания подмножества модели и сохраняемых отображений, освоение навыков в создании подмножеств модели и сохраняемых отображений.

Исходные данные (задание):

Основное понятие работы - подмножества модели и хранимое отображение

При создании реальных моделей данных количество сущностей и атрибутов может исчисляться сотнями. Для более удобной работы с большими моделями в ERwin предусмотрены **nolмножества модели (Subject Areas)**, в которые можно включить тематически общие сущности. В подмножество модели может входить произвольный набор сущностей, связей и текстовых комментариев. Все изменения, сделанные в любой Subject Area, автоматически отображаются на общей модели. Одна и та же сущность может входить в несколько Subject Area. ERwin позволяет разбить модель на несколько Subject Area, каждая из которых может соответствовать определенной задаче, например финансовой, производственной, маркетинговой и т. д. Subject Area можно создавать как в логической, так и в физической модели данных.

Хранимое отображение (Stored Display) - представление подмножества модели, отображающее специфический аспект структуры данных. Одна Subject Area может включать в себя несколько хранимых отображений. В хранимое отображение входят те же самые сущности и связи, что и в Subject Area, но они могут по-разному располагаться на экране, иметь разные уровни, различный масштаб и цвет объектов или фона. При создании Subject Area в нее могут не входить либо родительская, либо дочерняя сущность. По умолчанию связи с сущностями, которые не вошли в Subject Area ("висящие связи"), не показываются. Хранимое отображение позволяет отобразить линии связей не только ортогональными, но и диагональными.

Задание:

1. Создать подмножества модели с названием «Прием заказов» и «Исполнение заказов». В подмножество модели «Прием заказов» будут входить сущности Заказчик и Заказ и Дизайн, а в подмножество модели «Исполнение заказов» сущности Заказ и Реализация.

2. Создать три хранимых отображения: уровень сущностей, уровень атрибутов и уровень определений.

Технология работы

Для создания, удаления или редактирования подмножеств модели нужно

вызвать диалог Subject Areas (меню Model/Subject Areas...), в котором указывается имя подмножества и входящие в нее сущности. Щелкая по кнопке New задаем имя подмножества, в закладке Members в левой части находится список Available Objects, в котором отображаются все сущности. Выделяя сущность с помощью кнопок перехода переносим сущность в правую часть Included Objects, которая будет входить в подмножество модели.

Для создания хранимого отображения служит диалог Stored Displays (меню Format/Stored Display Settings...). Щелкая на кнопке New, диалогового окна Stored Displays задаем имя хранимого отображения (например, уровень сущностей), в закладке Logical в списке Display Level ставим переключатель Entity, для отображения имени связи в списке Relationship Option ставим флажок Verb Phrase.

Контрольные вопросы

1. Каково назначение подмножеств модели Subject Areas?
2. Как включать и перемещать сущности в Subject Area?
3. Для чего предназначено хранимое отображение Stored Display?
4. Как хранимое отображение позволяет представить связи диагональными линиями?
5. Какими способами можно переключаться между хранимыми отображениями?

Критерии оценки работы:

1. Полный ответ – 5 баллов.
2. Дополнительный уточняющий вопрос – 4 балла.
3. Краткий ответ – 3 балла.

ПРАКТИЧЕСКАЯ РАБОТА № 19

Создание логического уровня модели данных
Установка цвета и шрифта, создание графических объектов на диаграмме

Цель работы: Изучение процесса установки цвета и шрифта, также создания графических объектов, освоение навыков работы с диалогом Default Fonts & Colors.

ERwin позволяет устанавливать цвет, шрифт для сущностей атрибутов и связей, а также создавать графические объекты на диаграмме.

Исходные данные (задание):

Задание:

Сущности Заказ и Заказчик сделать желтым цветом, а сущности Дизайн и Реализация - зеленым. Связи сделать красным цветом и все сущности поместить в закрашенный серый прямоугольник со скругленными краями, сверху сделать надпись «Дизайнерский участок» черного цвета.

Технология работы

1. Для того, чтобы изменить цвет и шрифт сущности нужно выделить сущность, затем щелкнуть правой кнопкой и из контекстного меню выбрать пункт Object Font & Color..., появляется диалог Table Font & Color, в котором можно менять цвет и шрифт.
2. Для того, чтобы изменить цвет связи на диаграмме, нужно выделить связь, затем щелкнуть правой кнопкой и из контекстного меню выбрать пункт Object Font & Color..., появляется диалог Relationship Font & Color для изменения цвета связи.
3. Для создания графических объектов на диаграмме сначала нужно установить соответствующую панель инструментов, меню View\Toolbars\Drawing. На панели инструментов щелкнуть по нужному объекту, затем на диаграмме. Размеры объекта можно изменять с помощью мыши.
4. Имеется возможность изменить шрифт и цвет для всех объектов модели или для какой-либо отдельной категории объектов. Для этого служит диалог Default Fonts & Colors (пункт меню Format/Default Fonts & Colors...). Каждая закладка на диалоге позволяет редактировать шрифт и цвет для определенной категории объектов.

Контрольные вопросы

1. Какие способы существуют для установки шрифта и цвета объектов в Erwin?
2. Как редактировать шрифт и цвет конкретного объекта?
3. Как объединить графические элементы в графический объект?
4. Для чего предназначены панели Drawing и Alignment?

Критерии оценки работы:
1. Полный ответ – 5 баллов.
2. Дополнительный уточняющий вопрос – 4 балла.
3. Краткий ответ – 3 балла.

ПРАКТИЧЕСКАЯ РАБОТА № 20

Создание логического уровня модели данных
Экспорт модели данных ERwin в модель процессов BPwin

Цель работы: Изучение процесса экспорта модели данных в модель процессов.

Исходные данные (задание):

При создании КИС модель данных может содержать сотни таблиц, в таком случае возникает необходимость сравнить модель данных с моделью процессов, для гарантии того, что она не конфликтует с существующими моделями объектов.

ERwin позволяет производить экспорт только на уровне логической модели. Для успешного связывания моделей необходимо, чтобы версии ERwin и BPwin соответствовали друг другу.

Задание:

Экспортировать модель данных в модель процессов, связать сущность и атрибуты с работами и стрелками в модели процессов.

Технология работы

Для экспорта модели данных из ERwin в BPwin необходимо в ERwin открыть модель(проверить, чтобы никакие объекты не были выделены) и выбрать пункт меню File/Export/BPwin.

В появившемся диалоге Select BPwin Export File необходимо выбрать каталог, указать имя создаваемого файла экспорта *.eax и нажать Сохранить.

Затем в BPwin нужно открыть модель процессов, выбрать в меню пункт File/Import/ERwin (EAX), в диалоге Open выбрать имя файла (*.eax) и нажать ОК. Появится диалог Import Differences Preview, в котором показывается протокол импорта. Для внесения данных в модель процессов следует щелкнуть по кнопке Accert. Кнопка Cancel отменяет импорт.

После внесения данных в модель процессов можно связать сущности и атрибуты со стрелками. Правой кнопкой мыши нужно щелкнуть по стрелке и выбрать в контекстном меню Arrow Data. Появляется вкладка Arrow Data диалога Arrow Properties. Для связывания атрибута со стрелкой достаточно щелкнуть по иконке выбора в иерархическом списке атрибутов. При этом сущность в модели процессов может быть связана с несколькими атрибутами различных сущностей.

В свою очередь работы тоже могут воздействовать на данные. Для документирования такого воздействия необходимо щелкнуть правой кнопкой мыши по работе и выбрать пункт меню Data Usage Editor.

В появившемся диалоговом окне Data Usage Editor в виде иерархического списка показывается все работы модели, стрелки, которыые касаются работ, сущности и атрибуты. Для задания ассоциации достаточно щелкнуть по окну в иерархическом списке.

Для сущностей задается ассоциация CRUD (Create, Read, Update, Delete), для атрибутов - IRUN (Insert, Read, Update, Nullify). Ассоциация CRUD и IRUN - это

правила использования сущностей и атрибутов работами, т.е. то, что могут делать работы с входящими и исходящими данными. Данные не могут использоваться работами произвольно.

Для проверки соответствия связи атрибутов и сущностей с работами и стрелками, открываем модель данных и модель процессов, щелкаем по работе в модели процессов появляется диалог Data Usage Editor, в модели данных щелкаем правой кнопкой мыши по соответствующей сущности и из контекстного меню выбираем пункт Attributes, и в диалоговом окне Attributes появляется список атрибутов, которые соответствуют стрелкам в модели процессов.

Контрольные вопросы

1. Провести сравнение созданных моделей ERwin и BPwin на уровне характеристики объектов.
2. Как происходит связывание объектов модели данных со стеклами и работами?
3. Каким образом графически происходит преобразование стрелки в сущность (информация о стрелке в нескольких сущностях) ?
4. Рассказать алгоритм экспорта модели данных из ERwin в BPwin.
5. Для чего предназначен диалог Arrow Properties - вкладка Arrow Data?
6. Для чего предназначено диалоговое окно Data Usage Editor?
7. Как применяются правила использования сущностей и атрибутов работами(ассоциации CRUN и IRUN)?
8. Провести графическое сравнение сущностей и атрибутов со стрелками и работами.

Критерии оценки работы:

1. Полный ответ – 5 баллов.
2. Дополнительный уточняющий вопрос – 4 балла.
3. Краткий ответ – 3 балла.

ПРАКТИЧЕСКАЯ РАБОТА № 21

Создание физического уровня модели данных

Выбор сервера

Цель работы: Изучение процесса работы с типами данных конкретной СУБД, получение навыков работая с физической моделью в ERwin.

Исходные данные (задание):

Физический уровень представления модели зависит от выбранною сервера. В физической модели содержится информация обо всех объектах БД. Поскольку стандартов на объекты БД не существует (например, нет стандарта на типы данных), физическая модель зависит от конкретной реализации СУБД. Следовательно, одной и той же логической модели могут соответствовать несколько разных физических моделей. Если в логической модели не имеет значения какой конкретно тип данных имеет атрибут, то в физической модели важно описать всю информацию о конкретных физических объектах -таблицах, колонках, индексах, процедурах и т.д.

Задание:

1. На основе полученной логической модели создать два файла модели данных ERwin назвать их: «laccess.erl» и «lsql.erl», где физический уровень первого файла будет соответствовать СУБД Access 2000, второй СУБД SQL Server 2000.

2. Задать типы данных всем атрибутам сущностей в соответствии с выбранной СУБД в обоих файлах.

Технология работы

Для переключения между логической и физической моделью данных служит список выбора в левой части панели инструментов ERwin.

Физический уровень представления модели зависит от выбранного сервера. Для выбора СУБД служит редактор Target Server (меню Datebase/Choose Datebase... доступно только на физическом уровне). Выбираем СУБД, версию СУБД и в группе переключателей Default Non - Key Null Option по умолчанию ставим NOT NULL. Группа кнопок Default Non-Key Null Option позволяет разрешить или запретить значения NULL для неключевых колонок. •

Чтобы задать тип данных атрибуту нужно щелкнуть правой кнопкой мыши по сущности и из контекстного меню выбрать пункт Columns, где в левой части списка Column представлены все атрибуты данной сущности, справа находится набор вкладок. General для всех СУБД, остальные вкладки соответствуют выбранному серверу. General позволяет присвоить колонку определенному домену, создать колонку только на физическом уровне и включить ее в состав первичного ключа

При выборе СУБД Access 2000, в диалоговом окне Columns справа находятся вкладки соответствующие данной СУБД. Закладка Access позволяет каждому атрибуту присвоить тип данных. Тип данных присваивается автоматически на основе выбранного домена. Если домен соответствует типу < Default > т.е. по

умолчанию, в этом случае атрибуту будет присвоен тип данных выбранный по умолчанию в редакторе Target Server.

Тип данных для каждого атрибута лучше всего определять самостоятельно, во избежания осложнений при генерации модели.

Например, для сущности Заказчик, атрибуту Номер заказчика присвоем значение LONG INTEGER - т.е. длинное целое. Атрибутам Фамилия, Имя, Отчество присвоем значение TEXT (40), количество символов в скобках задается самостоятельно. Атрибутам Адрес и Телефон присвоем значение MEMO - т.е. текстовые и числовые значения. Атрибуту Статус заказчика присвоем значение TEXT (20) - т.е. текстовые символы.

При выборе СУБД SQL Server 2000, в диалоговом окне Columns, закладка SQL Server позволяет каждому атрибуту присвоить тип данных.

Например, для сущности Реализация, атрибуту Номер реализации присвоем значение Int - т.е. длинное целое. Атрибуту Дата реализации присвоем значение Datetime - т.е. значение дата\время. Атрибуту Наименование продукции присвоем значение Char (50) - т.е. символьные значения. Атрибуту Количество присвоем значение Char (20) - т.е. символьные значения. Атрибуту Стоимость присвоем значение Money - т.е. денежные значения.

При генерации модели, все заданные типы данных будут соответствовать типам данных выбранной СУБД.

Контрольные вопросы

1. Как создается физическая модель данных на основе сущностей и атрибутов?
2. Для чего предназначен редактор Target Server? В чём отличия баз данных SQL DBMS от Desktop DBMS?
3. Как задается тип данных атрибуту сущности БД?
4. Для чего предназначена группа кнопок Default Non-Key Null Option?
5. Как изменяется тип данных каждой колонки при изменении СУБД':'

Критерии оценки работы:

1. Полный ответ – 5 баллов.
2. Дополнительный уточняющий вопрос – 4 балла.
3. Краткий ответ – 3 балла.

ПРАКТИЧЕСКАЯ РАБОТА № 22

Создание физического уровня модели данных
Представления

Цель работы: Изучение процесса создания представлений, освоения навыков работы с представлениями.

Исходные данные (задание):

Основное понятие работы - представление

Представления (view), или, как их иногда называют, временные или производные таблицы, представляют собой объекты БД, данные в которых не хранятся постоянно, как в таблице, а формируются динамически при обращении к представлению. Представление не может существовать само по себе, а определяется только в терминах одной или нескольких таблиц. Применение представлений позволяет разработчику БД обеспечить каждому пользователю или группе пользователей свой взгляд на данные, что решает проблемы простоты использования и безопасности данных.

Задание:

Создать три представления:

1. Название «О Заказчике», атрибуты входящие в представление: Номер заказчика. Фамилия, Имя, Отчество, Дата заказа, Наименование продукции, Количество.

2. Название «О _Дизайне». атрибуты входящие в представление: Наименование продукции, Размер, Цвет, Стиль.

3. Название «О Заказе», атрибуты входящие в представление: Номер заказа. Наименование продукции, Дата заказа, Дата Реализации, Количество, Стоимость.

Технология работы

Для внесения представления нужно щелкнуть по кнопке представления на палитре инструментов, затем по свободному месту диаграммы. По умолчанию представление получает номер V_n, где n - уникальный порядковый номер представления. Для установления связи нужно щелкнуть по кнопке связи, затем по родительской таблице и, наконец, по представлению. Связи с представлениями и прямоугольники представлений показываются на диаграмме пунктирными линиями.

Для редактирования представления служит диалог Views. Для его вызова следует щелкнуть правой кнопкой мыши по представлению и выбрать в меню пункт Database View Columns...

Контрольные вопросы

1. Характеристика и назначение объектов БД - view.
2. В чем графические отличия сущностей от представлений?
3. Для чего предназначен редактор View Columns?
4. Какую функцию выполняют временные таблицы в модели данных?

Критерии оценки работы:
1. Полный ответ – 5 баллов.
2. Дополнительный уточняющий вопрос – 4 балла.
3. Краткий ответ – 3 балла.

ПРАКТИЧЕСКАЯ РАБОТА № 23

Создание физического уровня модели данных

Правила валидации и значения по умолчанию

Цель работы: Изучение процесса создания правил валидации и значений по умолчанию, освоение навыков работы с диалогом Validation Rules.

Исходные данные (задание):

Основное понятие работы - правила валидации и значения по умолчанию

ERwin поддерживает правила валидации для колонок, а также значение, присваиваемое колонкам по умолчанию. Правило валидации задает список допустимых значений для конкретной колонки и/или правила проверки допустимых значений. Значение но умолчанию - значение, которое нужно ввести в колонку, если никакое другое значение не задано явным образом во время ввода данных. С каждой колонкой или доменом можно связать значение по умолчанию(если выбранная СУБД поддерживает домен.)

Правила валидации и значения по умолчанию можно задать в диалоге Validation Rules меню Model. В нем можно задать максимальное и минимальное значение.

Задание:

1. Для сущности заказ, атрибута Количество создать правило валидации с именем "Ограничение заказов", которое содержит выражение. Количество BETWEEN 1 AND. Использование этого правила валидации гарантирует, что диапазон вводимых значений будет от 1 до 10.

2. Для сущности заказчик, атрибута Адрес, создать список допустимых значений с именем "Проверка адреса" "Местный", "Иногородний", "Иностранный"

Технология работы

Правой кнопкой щелкнуть по сущности и из контекстного меню выбрать Columns, затем выделить нужный атрибут и в строке Alid щелкнуть по раскрывающемуся списку, для вызова диалога Validation Rules.

В верхней части редактора Validation Rules содержиться список всех существующих правил валидации. Для создания нового правила валидации следует щелкнуть по кнопке New, ввести имя правила в поле Name диалога New Validation Rules и щелкнуть по кнопке ОК. После этого можно ввести выражение для правила валидации. Поля Min и Max служат для задания нижней и верхней границ диапазона значения. СУБД выдаст сообщения об ошибки, если вводимое количество находится вне границ заданного диапазона.

Для того, чтобы создать список всех допустимых значений, которые можно хранить в колонке, и связать его с правилом валидации. Для этого нужно переключить радиокнопку Valid Values List. В списке Valid Value задаем значение - Местный, в списке Definition - Проживает в нашем городе. При выходе из редактора Valid Values (кнопка ОК) ERwin автоматически изменяет правило валидации,

используя введенные допустимые значения.

Контрольные вопросы

1. Для чего предназначены правила валидации?
2. Какие вкладки содержит диалог Validation Rules?
3. Как ввести новое значения в список допустимых значений?
4. Как создается список всех допустимых значений?

Критерии оценки работы:

1. Полный ответ – 5 баллов.
2. Дополнительный уточняющий вопрос – 4 балла.
3. Краткий ответ – 3 балла.

ПРАКТИЧЕСКАЯ РАБОТА № 24

Создание физического уровня модели данных

Вычисление размера БД

Цель работы: Изучение процесса вычисления размера БД с помощью диалога Volumetrics, освоение навыков работы с диалогом Volumetrics.

Исходные данные (задание):

ERwin позволяем рассчитать приблизительный размер БД в целом, а также таблиц, индексов, и других объектов через определенный период времени, после начала эксплуатации ИС.

Задание:

Вычислить приблизительный размер БД через 24 месяца, если каждый месяц будет выполнятся 50 заказов (начальное значение 0), таблица заказчиков будет пополнятся на 25 заказчиков ежемесячно (начальное значение 10), таблица реализации будет пополнятся на 50 заказов (начальное значение 0), и таблица дизайна будет увеличиваться каждый месяц на 30 (начальное значение 0).

Технология работы

Для расчета размеров физических объектов служит диалог Volumetrics, который вызывается из меню Tools/ Volumetrics...

Редактор Volumetrics имеет три закладки - Settings, Report и Parameters.

Settings. Служит для задания основных параметров, на основе которых вычисляется размер БД:

В группе **Table Row Counts** для выделенной в левом списке Table таблицы задается начальное количество строк (Inital), максимальное количество строк (Max) и прирост количества строк в месяц (Grow Bu). Если параметры Max и Grow Bu используются одновременно, рост размера таблицы прекращается по достижении максимального размера. Те же самые параметры можно задать для каждой таблицы в закладке Volumetrics редактора Tables. Сразу после задания параметров Inital, Max и Grow Bu в группе Sizing Estimates, расположенной в левом нижнем углу диалога, показывается средний размер строки, начальный размер таблицы и индексов.

Таблица **Column Properties** позволяет задать свойства колонок таблицы. Имена колонок, их тип и размер (allocated) не редактируются. Можно изменять ширину поля Avg Width (для тех типов данных, для которых это допускается) и параметр Pet NULL (средний ожидаемый процент строк, в которых текущее поле принимает значение NULL). ERwin автоматически определяет в зависимости от выбранной СУБД, какие поля таблицы Column Properties могут изменяться.

Группа **Include Indexes** позволяет учесть или игнорировать индексы, создаваемые на внешних (FK, Foreign Key), первичных (PK, Primary Key), альтернативных (AK, Alternate Key) ключах или инверсионных входах (IE, Inverse Entry) при расчете размера БД.

Группа **Storage** позволяет задать объект физической памяти, в котором будет храниться выбранная таблица. Если объект физической памяти не описан, его можно определить в соответствующем редакторе.

Report. В ней отображаются результаты расчета размера БД (рис. 1.14). Группа Options позволяет выбрать тип объектов, по которым проводится расчет, Time - временной диапазон (начальное состояние или определенное время после начала эксплуатации).

Parameters. Служит для задания дополнительных параметров, используемых для расчета размера БД:

TableFactor. Этот фактор показывает накладные расходы на хранение таблицы в БД. Например, значение TableFactor = 2 увеличит размер таблиц вдвое.

IndexFactor показывает накладные расходы на хранение индекса в БД. Например, значение IndexFactor = I увеличит размер индекса с 1 М до 1,5 М .

RowOverhead используется для дополнительного пересчета количества байт каждой строки. Например, если значение RowOverhead т 10, размер каждой строки таблицы будет увеличен на 10 байт.

BlobFactor и **BlobBlockSize** используется для пересчета Blob-колонок, хранящихся физически вне таблицы.

BytesPerChar используется для задания количества байт, необходимых для хранения одного символа строкового типа. Для ASCII - это I, для других кодировок значение может быть больше 1, например для UNICODE - 2.

LogPercent используется для вычисления размеров log-файлов БД. LogPercent = 100 увеличивает БД вдвое.

Контрольные вопросы

1. Как размер БД связан с эксплуатацией ИС?
2. Как задаются основные параметры, на основе которых вычисляется размер БД?
3. Как рассчитать размер БД через определенное время после начала эксплуатации?
4. Как влияют временные таблицы на размер БД?

Критерии оценки работы:

1. Полный ответ – 5 баллов.
2. Дополнительный уточняющий вопрос – 4 балла.
3. Краткий ответ – 3 балла.

ПРАКТИЧЕСКАЯ РАБОТА № 25

Создание физического уровня модели данных

Прямое проектирование

Цель работы: Изучение процесса прямого проектирования БД с помощью диалога Forward Engineer; освоение навыков, работая с диалогом Forward Engineer.

Исходные данные (задание):

Основное понятие работы - прямое проектирование

Процесс генерации физической схемы БД из логической модели данных называется прямым проектированием (Forward Engineering). При генерации физической схемы ERwin включает триггеры ссылочной целостности, хранимые процедуры, индексы, ограничения и другие возможности, доступные при определении таблиц в выбранной СУБД. Процесс генерации логической модели из физической БД называется обратным проектированием (Reverse Engineering). ERwin позволяет создать модель данных путем обратного проектирования имеющейся БД. После того как модель создана, можно переключиться па другой сервер (модель будет конвертирована) : и произвести прямое проектирование структуры БД для другой СУБД. Кроме режима прямого обратного проектирования ERwin поддерживает синхронизацию между логической моделью и системным каталогом СУБД на протяжении всего жизненного цикла создания ИС.

Задание:

1. Создать один пустой файл в СУБД MS Access 2000 и пустую папку в списке Datebase MS SQL Server 2000 под названием «Rabola» сохранить и закрыть.

2. Открыть файл «1 access.erl» путем прямого проектирования сгенерировать код для создания системного каталога БД в файл «Rabota.mdb».

3. Открыть файл «lsql.erl» путем прямого проектирования сгенерирован, код для создания системного каталога БД в список Datebase в папку под названием Rabota.

Технология работы

Для генерации системного каталога БД следует выбрать пункт меню Tools/Forward Engineer/Schema Generation или нажать кнопку на панели инструментов. Появляется диалог Schema Generation.

Диалог Schema Generation имеет три закладки:

Options. Служит для задания опций генерации объектов БД - триггеров, таблиц, представлений, колонок, индексов и т. д. Для задания опций генерации какого-либо объекта следует выбрать объект в левом списке закладки, после чего включить соответствующую опцию в правом списке.

В закладке **Summary** отображаются все опции, заданные в закладке Options. Список опций в Summary можно редактировать так же, как и в Options.

Comment. Позволяет внести комментарий для каждого набора опций.

Каждый набор опций может быть именован (окно Option Set, кнопки New,

Rename и Delete) и использован многократно.

Кнопка **Preview** вызывает диалог Schema Generation Preview, в котором отображается SQL-скрипт, создаваемый ERwin для генерации системного каталога СУБД. Нажатие на кнопку Generate приведет к запуску процесса генерации схемы.

Кнопка **Print** предназначена для вывода на печать создаваемого ERwin SQL-скрипта.

Кнопка Report сохраняет тот же скрипт в ERS или SQL текстовом файле. Эти команды можно в дальнейшем редактировать любым текстовым редактором и выполнять при помощи соответствующей утилиты сервера.

Кнопка Generate запускает процесс генерации схемы. Возникает диалог связи с БД, устанавливается сеанс связи с сервером и начинает выполняться SQL-скрипт. При этом возникает диалог Generate Database Schema.

В процессе установки сеанса связи с Access:

Поле User Name - заполняем Admin (сокращение от Administrator);

Поле Datebase - прописываем путь к базе данных Access, заранее созданной (Rabota.mdb).

Поле System Datebase - прописывем путь к файлу system.mdw (который может находиться в папке C\Program Files\Microsoft Office 2000\Office).

Остальные поля не заполняем и нажимаем кнопку Connect.

В процессе установки сеанса связи SQL Server:

В диалоговом окне SQL Server Connection ставим переключатель в строке Use Windows authentication.

В поле Datebase - заполняем именем заранее созданной БД (Rabota);

Поле Server Name - заполняем именем сервера.

Нажимаем кнопку Connect.

По умолчанию в диалоге Generate Database Schema ключена опция Stop If Failure. Это означает, что при первой же ошибке выполнение скрипта прекращается. Щелкнув по кнопке Contunue, можно продолжить выполнение. Кнопка Abort прерывает выполнение. При выключенной опции Stop If Failure скрипт будет выполняться, несмотря на встречающиеся ошибки.

Дополнительная информация

Можно создать один файл модели данных, физический уровень которого соответствует MS Access 2000, код которого путем прямого проектирования сгенерировать в БД, которую затем заполнить данными.

После этого физический уровень модели конвертировать, т.е. сменить СУБД на MS SQL Server 2000. Имена таблиц и связи модели данных менять нельзя. При смене СУБД ERwin предлагает автоматически преобразовать тип данных, связанный с каждым атрибутом, на ближайший, доступный для новой СУБД. Для автоматического преобразования следует в ответ на вопрос нажать Yes, после этого проверить типы данных для всех атрибутов. Затем путем прямого проектирования сгенерировать код в БД SQL Server:

После этого открываем MS Access 2000, в диалоге Microsoft Access выбираем переключатель Мастера, страницы и проекты баз данных нажимаем Ok, затем в закладке Общие выбираем Про ект (существующая база данных) нажимаем Ок. Задаем имя БД (*.adp) нажимаем Ок. После этого появляется диалог Data Link Properties, в строке Select or Enter a server name - задаем имя сервера, после этого в списке Enter Information to Log on the server - ставим переключатель в строке Use

Windows NT Integrated security. В раскрывающемся списке Select the database on the server выбираем созданную ранее БД, в которую был сгенерирован код модели, нажимаем Ок. БД связывается с сервером, в результате структура переносится в БД *.adp.

Для того чтобы импортировать данные из заполненной БД *.mdb, нужно открыть базу данных *.adp, из меню Файл выбрать пункт Внешние данные Импорт, затем выбрать БД *.mdb, из которой будем импортировать данные, нажать Ок. На экране появится диалоговое окно Импорт объектов, задаем все объекты БД (таблицы, формы и т.д.) выделяя их в списке, после этого нажимаем Ok, запуская процесс импорта объектов. После окончания процесса импорта все одноименные импортированные таблицы в имени будут содержать либо 1, либо знак подчеркивания. Следующим шагом следует удалить пустые таблицы, а импортированные переименовать (убрать 1 или знак подчеркивания), чтобы их имена совпадали с именами таблиц хранящихся на сервере. После этого открываем структуру БД в SQL Server, где в таблицах должны отражаться данные из *.adp.

Контрольные вопросы

1. Дать определение-триггера, хранимой процедуры.
2. Что называется триггером ссылочной целостности?
3. Для чего предназначен индекс на физическом уровне?
4. Что называется прямым и обратным проектированием?
5. Как выполнить генерацию системного каталога?
6. Что является результатом прямого проектирования для СУБД (MS Access 2000 и MS SQL Server 2000)?
7. В чем эффективность прямого проектирования?

Критерии оценки работы:

1. Полный ответ – 5 баллов.
2. Дополнительный уточняющий вопрос – 4 балла.
3. Краткий ответ – 3 балла.

ПРАКТИЧЕСКАЯ РАБОТА № 26

Работа с инструментальными средствами, поддерживающими методологию объектно-ориентированного моделирования

Цель работы: Ознакомление с основными элементами определения, представления, проектирования и моделирования программных систем с помощью языка UML.

Исходные данные (задание):

1. Теоретическая часть

Существует множество технологий и инструментальных средств, с помощью которых можно реализовать оптимальный проект ИС, начиная с этапа анализа и заканчивая созданием программного кода системы. В большинстве случаев эти технологии предъявляют весьма жесткие требования к процессу разработки и используемым ресурсам, а попытки трансформировать их под конкретные проекты оказываются безуспешными. Эти технологии представлены CASE-средствами верхнего уровня или CASE-средствами полного жизненного цикла (upper CASE tools или fulllife-cycle CASE tools). Они не позволяют оптимизировать деятельность на уровне отдельных элементов проекта, и, как следствие, многие разработчики перешли на CASE-средства нижнего уровня (lower CASE tools). Однако они столкнулись с новой проблемой — проблемой организации взаимодействия между различными командами, реализующими проект.

Унифицированный язык объектно-ориентированного моделирования UnifiedModelingLanguage (UML) явился средством достижения компромисса между этими подходами. Существует достаточное количество инструментальных средств, поддерживающих с помощью *UML* жизненный цикл информационных систем, и, одновременно, *UML* является достаточно гибким для настройки и поддержки специфики деятельности различных команд разработчиков.

Создание UML началось в октябре 1994 г., когда Джим Рамбо и ГрадиБуч из RationalSoftwareCorporation стали работать над объединением своих методов OMT и Booch. В настоящее время консорциум пользователей UML Partners включает в себя представителей таких грандов информационных технологий, как RationalSoftware, Microsoft, IBM, HewlettPackard, Oracle, DEC, Unisys, IntelliCorp, PlatinumTechnology.

UML представляет собой *объектно-ориентированный* язык моделирования, обладающий следующими основными характеристиками:

- является языком визуального моделирования, который обеспечивает разработку репрезентативных моделей для организации взаимодействия заказчика и разработчика ИС, различных групп разработчиков ИС;
- содержит механизмы расширения и специализации базовых концепций языка.

UML — это стандартная нотация визуального моделирования программных систем, принятая консорциумом ObjectManagingGroup (OMG) осенью 1997 г., и на сегодняшний день она поддерживается многими объектно-ориентированными

CASE-продуктами.

UML включает внутренний набор средств моделирования, которые сейчас приняты во многих методах и средствах моделирования. Эти концепции необходимы в большинстве прикладных задач, хотя не каждая концепция необходима в каждой части каждого приложения. Пользователям языка предоставлены возможности:

- строить модели на основе средств ядра, без использования механизмов расширения для большинства типовых приложений;
- добавлять при необходимости новые элементы и условные обозначения, если они не входят в ядро, или специализировать компоненты, систему условных обозначений (нотацию) и ограничения для конкретных предметных областей.

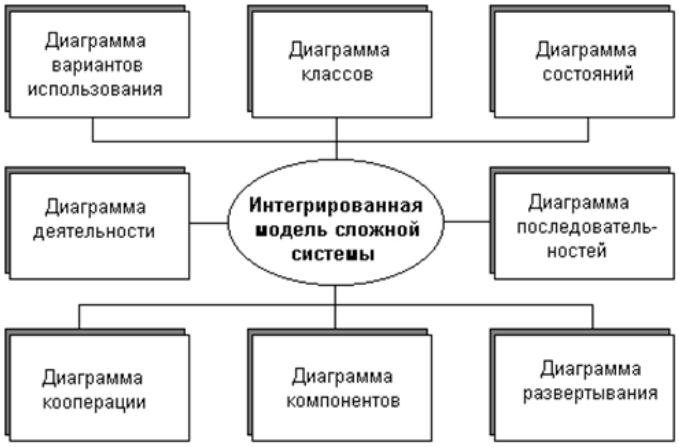


Рисунок 1.1 - Интегрированная модель сложной системы в нотации языка UML

Стандарт UML предлагает следующий набор диаграмм для моделирования:

✓ диаграммы вариантов использования (usecasediagrams) – для моделирования бизнес процессов организации и требований к создаваемой системе);

✓ диаграммы классов (classdiagrams) – для моделирования статической структуры классов системы и связей между ними;

✓ диаграммы поведения системы (behaviordiagrams):

✓ диаграммы взаимодействия (interaction diagrams):

✓ диаграммы последовательности (sequence diagrams) и

✓ кооперативные диаграммы (collaborationdiagrams) – для моделирования процесса обмена сообщениями между объектами;

✓ диаграммы состояний (statechartdiagrams) – для моделирования поведения объектов системы при переходе из одного состояния в другое;

✓ диаграммы деятельностей (activitydiagrams) – для моделирования поведения системы в рамках различных вариантов использования, или моделирования деятельностей;

✓ диаграммы реализации (implementationdiagrams):

✓ диаграммы компонентов (componentdiagrams) – для моделирования иерархии компонентов (подсистем) системы;

✓ диаграммы развертывания (deploymentdiagrams) – для моделирования физической архитектуры системы.

Диаграммы вариантов использования

Понятие варианта использования (use case) впервые ввел Ивар Якобсон и придал ему такую значимость, что в настоящее время вариант использования превратился в основной элемент разработки и планирования проекта.

Вариант использования представляет собой последовательность действий (транзакций), выполняемых системой в ответ на событие, инициируемое некоторым внешним объектом (действующим лицом). Вариант использования описывает типичное взаимодействие между пользователем и системой. В простейшем случае вариант использования определяется в процессе обсуждения с пользователем тех функций, которые он хотел бы реализовать. На языке UML вариант использования изображают следующим образом:

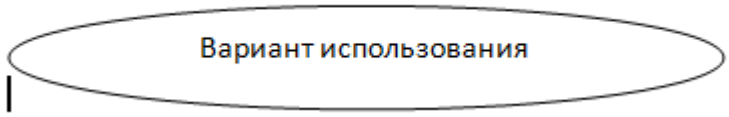


Рисунок 1.2 - Вариант использования

Действующее лицо (actor) – это роль, которую пользователь играет по отношению к системе. Действующие лица представляют собой роли, а не конкретных людей или наименования работ. Несмотря на то, что на диаграммах вариантов использования они изображаются в виде стилизованных человеческих фигурок, действующее лицо может также быть внешней системой, которой необходима некоторая информация от данной системы. Показывать на диаграмме действующих лиц следует только в том случае, когда им действительно необходимы некоторые варианты использования. На языке UML действующие лица представляют в виде фигур:

Рисунок 1.3 - Действующее лицо (актер)

Действующие лица делятся на три основных типа:

- пользователи;
- системы;
- другие системы, взаимодействующие с данной;
- время.

Время становится действующим лицом, если от него зависит запуск каких-либо событий в системе.

Связи между вариантами использования и действующими лицами

В языке UML на диаграммах вариантов использования поддерживается несколько типов связей между элементами диаграммы. Это связи коммуникации (communication), включения (include), расширения (extend) и обобщения (generalization).

Связь коммуникации – это связь между вариантом использования и действующим лицом. На языке UML связи коммуникации показывают с помощью однонаправленной ассоциации (сплошной линии).

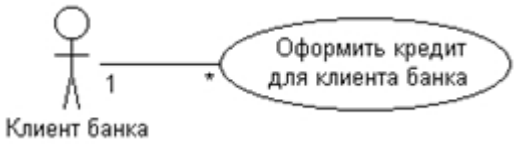


Рисунок 1.4 - Пример связи коммуникации

Связь включения применяется в тех ситуациях, когда имеется какой-либо фрагмент поведения системы, который повторяется более чем в одном варианте использования. С помощью таких связей обычно моделируют многократно используемую функциональность.

Связь расширения применяется при описании изменений в нормальном поведении системы. Она позволяет варианту использования только при необходимости использовать функциональные возможности другого.

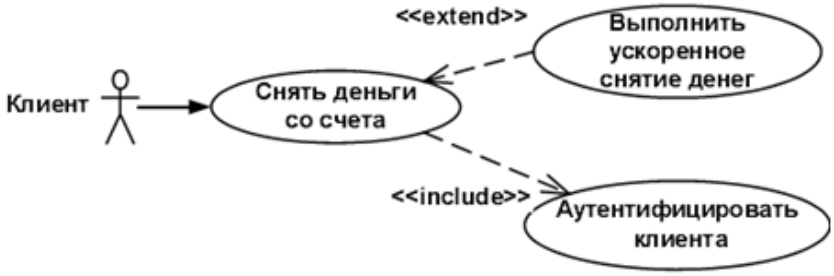


Рисунок 1.5 - Пример связи включения и расширения

С помощью связи обобщения показывают, что у нескольких действующих лиц имеются общие черты.

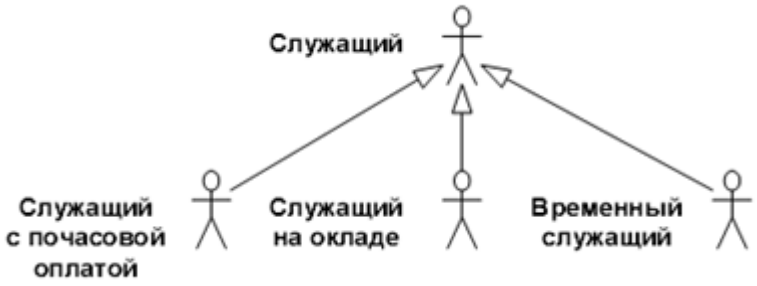


Рисунок 1.6 - Пример связи обобщения

<u>Диаграммы взаимодействия</u> (interactiondiagrams)

Диаграммы взаимодействия (interactiondiagrams) описывают поведение взаимодействующих групп объектов. Как правило, диаграмма взаимодействия охватывает поведение объектов в рамках только одного варианта использования. На такой диаграмме отображается ряд объектов и те сообщения, которыми они обмениваются между собой.

Сообщение (message) – это средство, с помощью которого объект-отправитель запрашивает у объекта получателя выполнение одной из его операций.

Информационное (informative) сообщение – это сообщение, снабжающее объект-получатель некоторой информацией для обновления его состояния.

Сообщение-запрос (interrogative) – это сообщение, запрашивающее выдачу

некоторой информации об объекте-получателе.

Императивное (imperative) сообщение – это сообщение, запрашивающее у объекта-получателя выполнение некоторых действий.

Существует два вида диаграмм взаимодействия: диаграммы последовательности (sequencediagrams) и кооперативные диаграммы (collaborationdiagrams).

Диаграмма последовательности (sequencediagrams)

Диаграмма последовательности отражает поток событий, происходящих в рамках варианта использования.

Все действующие лица показаны в верхней части диаграммы. Стрелки соответствуют сообщениям, передаваемым между действующим лицом и объектом или между объектами для выполнения требуемых функций.

На диаграмме последовательности объект изображается в виде прямоугольника, от которого вниз проведена пунктирная вертикальная линия. Эта линия называется линией жизни (lifeline) объекта. Она представляет собой фрагмент жизненного цикла объекта в процессе взаимодействия.

Каждое сообщение представляется в виде стрелки между линиями жизни двух объектов. Сообщения появляются в том порядке, как они показаны на странице сверху вниз. Каждое сообщение помечается как минимум именем сообщения. При желании можно добавить также аргументы и некоторую управляющую информацию. Можно показать само делегирование (selfdelegation) – сообщение, которое объект посылает самому себе, при этом стрелка сообщения указывает на ту же самую линию жизни.

Диаграмма кооперации (collaborationdiagram)

Диаграммы кооперации отображают поток событий через конкретный сценарий варианта использования, упорядочены по времени, а кооперативные диаграммы больше внимания заостряют на связях между объектами.

На диаграмме кооперации представлена вся та информация, которая есть и на диаграмме последовательности, но кооперативная диаграмма по-другому описывает поток событий. Из нее легче понять связи между объектами, однако, труднее уяснить последовательность событий.

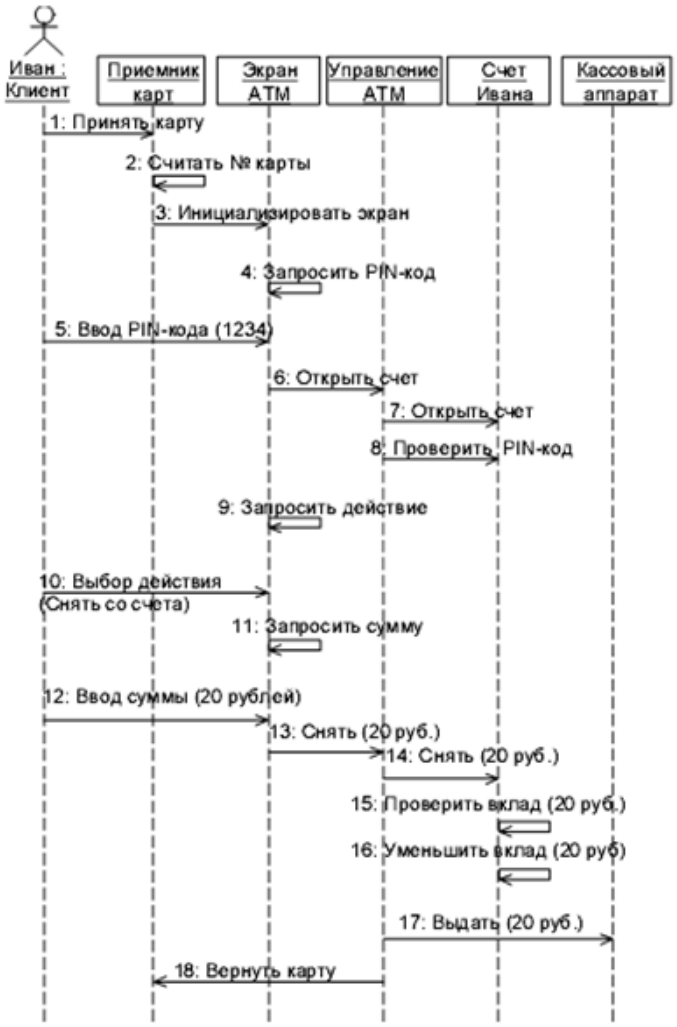


Рисунок 1.7 - Пример диаграммы последовательности

На кооперативной диаграмме так же, как и на диаграмме последовательности, стрелки обозначают сообщения, обмен которыми осуществляется в рамках данного варианта использования. Их временная последовательность указывается путем нумерации сообщений.

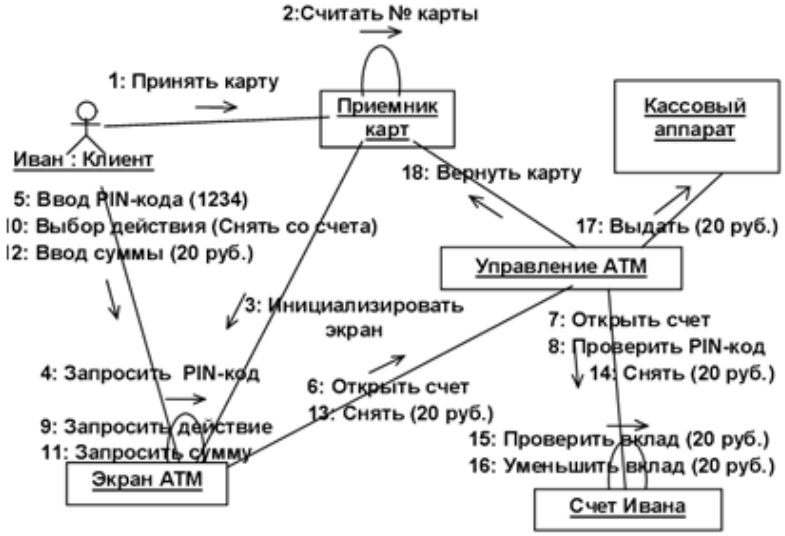


Рисунок 1.8 - Пример диаграммы кооперации

Диаграммы классов

Диаграмма классов определяет типы классов системы и различного рода статические связи, которые существуют между ними. На диаграммах классов изображаются также атрибуты классов, операции классов и ограничения, которые накладываются на связи между классами.

Диаграмма классов UML - это граф, узлами которого являются элементы статической структуры проекта (классы, интерфейсы), а дугами - отношения между узлами (ассоциации, наследование, зависимости).

На диаграмме классов изображаются следующие элементы:

- Пакет (package) - набор элементов модели, логически связанных между собой;
- Класс (class) - описание общих свойств группы сходных объектов;
- Интерфейс (interface) - абстрактный класс, задающий набор операций, которые объект произвольного класса, связанного с данным интерфейсом, предоставляет другим объектам.

Класс - это группа сущностей (объектов), обладающих сходными свойствами, а именно, данными и поведением. Отдельный представитель некоторого класса называется объектом класса или просто объектом.

Под поведением объекта в UML понимаются любые правила взаимодействия объекта с внешним миром и с данными самого объекта.

На диаграммах класс изображается в виде прямоугольника со сплошной границей, разделенного горизонтальными линиями на три секции:

- Верхняя секция (секция имени) содержит имя класса и другие общие свойства (в частности, стереотип).
- В средней секции содержится список атрибутов
- В нижней - список операций класса, отражающих его поведение (действия, выполняемые классом).

Любая из секций атрибутов и операций может не изображаться (а также обе сразу). Для отсутствующей секции не нужно рисовать разделительную линию и как-либо указывать на наличие или отсутствие элементов в ней.

На усмотрение конкретной реализации могут быть введены дополнительные секции, например, исключения (Exceptions).

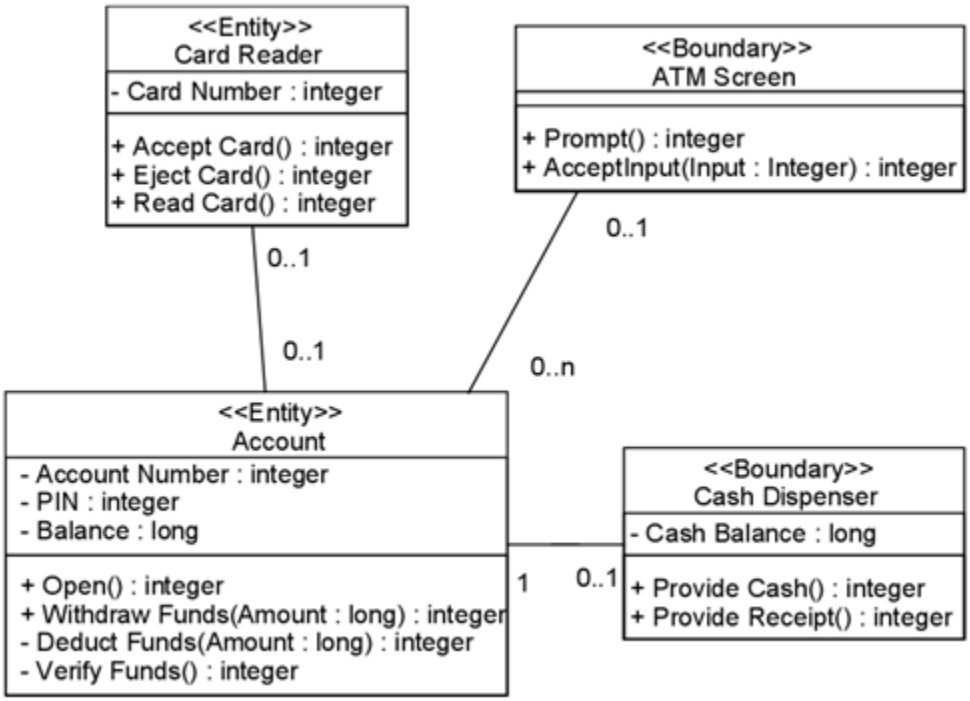


Рисунок 1.9 -. Пример диаграммы классов

Стереотипы классов – это механизм, позволяющий разделять классы на категории.

В языке UML определены три основных стереотипа классов:

- Boundary (граница);
- Entity (сущность);
- Control (управление).

Граничными классами (boundaryclasses) называются такие классы, которые расположены на границе системы и всей окружающей среды. Это экранные формы, отчеты, интерфейсы с аппаратурой (такой как принтеры или сканеры) и интерфейсы с другими системами.

Чтобы найти граничные классы, надо исследовать диаграммы вариантов использования. Каждому взаимодействию между действующим лицом и вариантом использования должен соответствовать, по крайней мере, один граничный класс. Именно такой класс позволяет действующему лицу взаимодействовать с системой.

Классы-сущности (entityclasses) содержат хранимую информацию. Они имеют наибольшее значение для пользователя, и потому в их названиях часто используют термины из предметной области. Обычно для каждого класса-сущности создают таблицу в базе данных.

Управляющие классы (controlclasses) отвечают за координацию действий других классов. Обычно у каждого варианта использования имеется один управляющий класс, контролирующий последовательность событий этого варианта использования. Управляющий класс отвечает за координацию, но сам не несет в себе никакой функциональности, так как остальные классы не посылают ему большого количества сообщений. Вместо этого он сам посылает множество сообщений. Управляющий класс просто делегирует ответственность другим классам, по этой причине его часто называют классом-менеджером.

В системе могут быть и другие управляющие классы, общие для нескольких

вариантов использования. Например, может быть класс SecurityManager (менеджер безопасности), отвечающий за контроль событий, связанных с безопасностью. Класс TransactionManager (менеджер транзакций) занимается координацией сообщений, относящихся к транзакциям с базой данных. Могут быть и другие менеджеры для работы с другими элементами функционирования системы, такими как разделение ресурсов, распределенная обработка данных или обработка ошибок.

Помимо упомянутых выше стереотипов можно создавать и свои собственные.

Атрибуты – это элемент информации, связанный с классом. Атрибуты хранят инкапсулированные данные класса.

Так как атрибуты содержатся внутри класса, они скрыты от других классов. В связи с этим может понадобиться указать, какие классы имеют право читать и изменять атрибуты. Это свойство называется видимостью атрибута (attributevisibility).

У атрибута можно определить четыре возможных значения этого параметра:

- Public (общий, открытый). Это значение видимости предполагает, что атрибут будет виден всеми остальными классами. Любой класс может просмотреть или изменить значение атрибута. В соответствии с нотацией UML общему атрибуту предшествует знак « + ».
- Private (закрытый, секретный). Соответствующий атрибут не виден никаким другим классом. Закрытый атрибут обозначается знаком « – » в соответствии с нотацией UML.
- Protected (защищенный). Такой атрибут доступен только самому классу и его потомкам. Нотация UML для защищенного атрибута – это знак « # ».
- PackageorImplementation (пакетный). Предполагает, что данный атрибут является общим, но только в пределах его пакета. Этот тип видимости не обозначается никаким специальным значком.

В общем случае, атрибуты рекомендуется делать закрытыми или защищенными. Это позволяет лучше контролировать сам атрибут и код.

С помощью закрытости или защищенности удается избежать ситуации, когда значение атрибута изменяется всеми классами системы. Вместо этого логика изменения атрибута будет заключена в том же классе, что и сам этот атрибут. Задаваемые параметры видимости повлияют на генерируемый код.

Операции реализуют связанное с классом поведение. Операция включает три части – имя, параметры и тип возвращаемого значения.

Параметры – это аргументы, получаемые операцией «на входе». Тип возвращаемого значения относится к результату действия операции.

На диаграмме классов можно показывать как имена операций, так и имена операций вместе с их параметрами и типом возвращаемого значения. Чтобы уменьшить загруженность диаграммы, полезно бывает на некоторых из них показывать только имена операций, а на других их полную сигнатуру.

В языке UML операции имеют следующую нотацию:

Имя Операции (аргумент: тип данных аргумента,
аргумент2:тип данных аргумента2,...): тип возвращаемого значения

Следует рассмотреть четыре различных типа операций:

- Операции реализации;
- Операции управления;
- Операции доступа;
- Вспомогательные операции.

Операции реализации (implementoroperations) реализуют некоторые бизнес-функции. Такие операции можно найти, исследуя диаграммы взаимодействия. Диаграммы этого типа фокусируются на бизнес-функциях, и каждое сообщение диаграммы, скорее всего, можно соотнести с операцией реализации.

Каждая операция реализации должна быть легко прослеживаема до соответствующего требования. Это достигается на различных этапах моделирования. Операция выводится из сообщения на диаграмме взаимодействия, сообщения исходят из подробного описания потока событий, который создается на основе варианта использования, а последний – на основе требований. Возможность проследить всю эту цепочку позволяет гарантировать, что каждое требование будет реализовано в коде, а каждый фрагмент кода реализует какое-то требование.

Операции управления (manageroperations) управляют созданием и уничтожением объектов. В эту категорию попадают конструкторы и деструкторы классов.

Операции доступа. Атрибуты обычно бывают закрытыми или защищенными. Тем не менее, другие классы иногда должны просматривать или изменять их значения. Для этого существуют операции доступа (accessoperations). Такой подход дает возможность безопасно инкапсулировать атрибуты внутри класса, защитив их от других классов, но все же позволяет осуществить к ним контролируемый доступ. Создание операций Get и Set (получения и изменения значения) для каждого атрибута класса является стандартом.

Вспомогательные операции (helperoperations) называются такие операции класса, которые необходимы ему для выполнения его ответственностей, но о которых другие классы не должны ничего знать. Это закрытые и защищенные операции класса.

Чтобы идентифицировать операции, выполните следующие действия:

1. Изучите диаграммы последовательности и кооперативные диаграммы. Большая часть сообщений на этих диаграммах является операциями реализации. Рефлексивные сообщения будут вспомогательными операциями.
2. Рассмотрите управляющие операции. Может потребоваться добавить конструкторы и деструкторы.
3. Рассмотрите операции доступа. Для каждого атрибута класса, с которым должны будут работать другие классы, надо создать операции Get и Set.

Связь представляет собой семантическую взаимосвязь между классами. Она дает классу возможность узнавать об атрибутах, операциях и связях другого класса. Иными словами, чтобы один класс мог послать сообщение другому на диаграмме последовательности или кооперативной диаграмме, между ними должна существовать связь.

Существуют четыре типа связей, которые могут быть установлены между классами: ассоциации, зависимости, агрегации и обобщения.

Ассоциации (association) – это семантическая связь между классами. Их рисуют на диаграмме классов в виде обыкновенной линии.

Рисунок 1.10 - Связь ассоциация

Ассоциации могут быть двунаправленными, как в примере, или однонаправленными. На языке UML двунаправленные ассоциации рисуют в виде простой линии без стрелок или со стрелками с обеих ее сторон. На однонаправленной ассоциации изображают только одну стрелку, показывающую ее направление.

Направление ассоциации можно определить, изучая диаграммы последовательности и кооперативные диаграммы. Если все сообщения на них отправляются только одним классом и принимаются только другим классом, но не наоборот, между этими классами имеет место однонаправленная связь. Если хотя бы одно сообщение отправляется в обратную сторону, ассоциация должна быть двунаправленной.

Ассоциации могут быть рефлексивными. Рефлексивная ассоциация предполагает, что один экземпляр класса взаимодействует с другими экземплярами этого же класса.

Зависимости. Связи зависимости (dependency) также отражают связь между классами, но они всегда однонаправлены и показывают, что один класс зависит от определений, сделанных в другом. Например, класс А использует методы класса В. Тогда при изменении класса В необходимо произвести соответствующие изменения в классе А.

Зависимость изображается пунктирной линией, проведенной между двумя элементами диаграммы, и считается, что элемент, привязанный к концу стрелки, зависит от элемента, привязанного к началу этой стрелки.

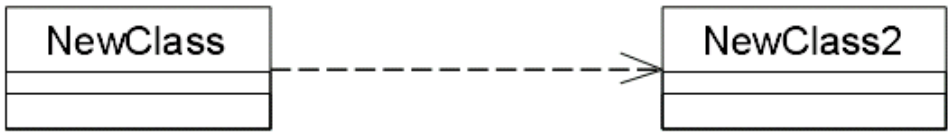


Рисунок 1.11 - Связь зависимость

При генерации кода для этих классов к ним не будут добавляться новые атрибуты. Однако, будут созданы специфические для языка операторы, необходимые для поддержки связи.

Агрегации (aggregations) представляют собой более тесную форму ассоциации. Агрегация – это связь между целым и его частью. Например, у вас может быть класс Автомобиль, а также классы Двигатель, Покрышки и классы для других частей автомобиля. В результате объект класса Автомобиль будет состоять из объекта класса Двигатель, четырех объектов Покрышек и т. д. Агрегации визуализируют в виде линии с ромбиком у класса, являющегося целым:

Рисунок 1.11 - Связь агрегация

В дополнение к простой агрегации UML вводит более сильную разновидность агрегации, называемую композицией. Согласно композиции, объект-часть может принадлежать только единственному целому, и, кроме того, как правило, жизненный цикл частей совпадает с циклом целого: они живут и умирают вместе с ним. Любое удаление целого распространяется на его части.

Такое каскадное удаление нередко рассматривается как часть определения агрегации, однако оно всегда подразумевается в том случае, когда множественность роли составляет 1..1; например, если необходимо удалить Клиента, то это удаление должно распространиться и на Заказы (и, в свою очередь, на Строки заказа).

Обобщения (Наследование) - это отношение типа общее-частное между элементами модели. С помощью обобщений (generalization) показывают связи наследования между двумя классами. Большинство объектно-ориентированных языков непосредственно поддерживают концепцию наследования. Она позволяет одному классу наследовать все атрибуты, операции и связи другого. Наследование пакетов означает, что в пакете-наследнике все сущности пакета-предка будут видны под своими собственными именами (т.е. пространства имен объединяются). Наследование показывается сплошной линией, идущей от класса-потомка к классу-предку (в терминологии ООП - от потомка к предку, от сына к отцу, или от подкласса к суперклассу). Со стороны более общего элемента рисуется большой полый треугольник.

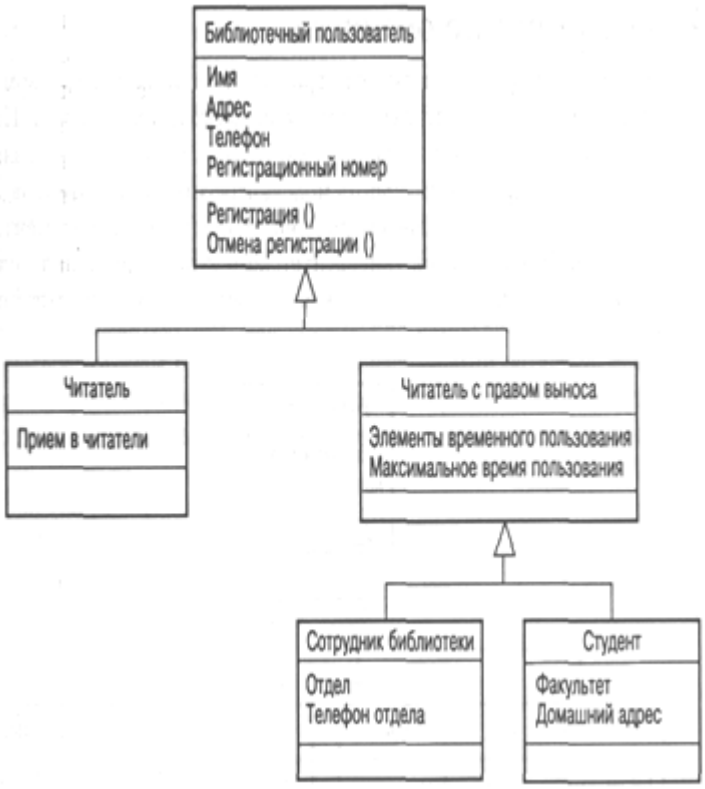


Рисунок 1.12 - Пример связи наследование

Помимо наследуемых, каждый подкласс имеет свои собственные уникальные атрибуты, операции и связи.

Множественность (multiplicity) показывает, сколько экземпляров одного класса взаимодействуют с помощью этой связи с одним экземпляром другого класса в данный момент времени.

Например, при разработке системы регистрации курсов в университете можно определить классы Course (курс) и Student (студент). Между ними установлена связь: у курсов могут быть студенты, а у студентов – курсы. Вопросы, на который должен ответить параметр множественности: «Сколько курсов студент может посещать в данный момент? Сколько студентов может за раз посещать один курс?»

Так как множественность дает ответ на оба эти вопроса, её индикаторы устанавливаются на обоих концах линии связи. В примере регистрации курсов мы решили, что один студент может посещать от нуля до четырех курсов, а один курс могут слушать от 0 до 20 студентов.

В языке UML приняты определенные нотации для обозначения множественности.

Таблица 1.1 - Обозначения множественности связей в UML

Множественность	Значение
0..*	Ноль или больше
1..*	Один или больше
0..1	Ноль или один
1..1 (сокращенная запись: 1)	Ровно один

Имена связей. Связи можно уточнить с помощью имен связей или ролевых имен. Имя связи – это обычно глагол или глагольная фраза, описывающая, зачем она нужна. Например, между классом Person (человек) и классом Company (компания) может существовать ассоциация. Можно задать в связи с этим вопрос, является ли объект класса Person клиентом компании, её сотрудником или владельцем? Чтобы определить это, ассоциацию можно назвать «employs» (нанимает):

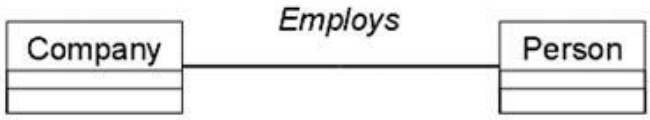


Рисунок 1.13 - Пример имен связей

Роли. Ролевые имена применяют в связях ассоциации или агрегации вместо имен для описания того, зачем эти связи нужны. Возвращаясь к примеру с классами Person и Company, можно сказать, что класс Person играет роль сотрудника класса Company. Ролевые имена – это обычно имена существительные или основанные на них фразы, их показывают на диаграмме рядом с классом, играющим соответствующую роль. Как правило, пользуются или ролевым именем, или именем связи, но не обоими сразу. Как и имена связей, ролевые имена не обязательны, их

дают, только если цель связи не очевидна. Пример ролей приводится ниже:

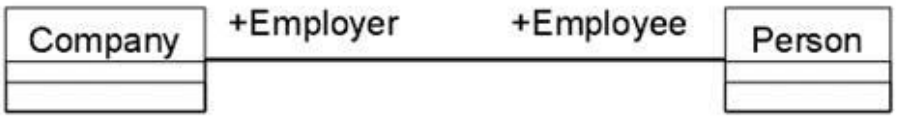


Рисунок 1.14 - Пример ролей связей

Пакет. Механизм пакетов. В контексте диаграмм классов, пакет - это вместилище для некоторого набора классов и других пакетов. Пакет является самостоятельным пространством имен.

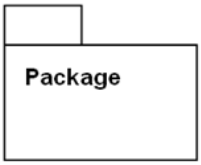


Рисунок 1.15 - Обозначение пакета в UML

В UML нет каких-либо ограничений на правила, по которым разработчики могут или должны группировать классы в пакеты. Но есть некоторые стандартные случаи, когда такая группировка уместна, например, тесно взаимодействующие классы, или более общий случай - разбиение системы на подсистемы.

Пакет физически содержит сущности, определенные в нем (говорят, что "сущности принадлежат пакету"). Это означает, что если будет уничтожен пакет, то будут уничтожено и все его содержимое.

Существует несколько наиболее распространенных подходов к группировке.

Во-первых, можно группировать их по стереотипу. В таком случае получается один пакет с классами-сущностями, один с граничными классами, один с управляющими классами и т.д. Этот подход может быть полезен с точки зрения размещения готовой системы, поскольку все находящиеся на клиентских машинах пограничные классы уже оказываются в одном пакете.

Другой подход заключается в объединении классов по их функциональности. Например, в пакете Security (безопасность) содержатся все классы, отвечающие за безопасность приложения. В таком случае другие пакеты могут называться EmployeeMaintenance (Работа с сотрудниками), Reporting (Подготовка отчетов) и ErrorHandling (Обработка ошибок). Преимущество этого подхода заключается в возможности повторного использования.

Механизм пакетов применим к любым элементам модели, а не только к классам. Если для группировки классов не использовать некоторые эвристики, то она становится произвольной. Одна из них, которая в основном используется в UML, – это зависимость. Зависимость между двумя пакетами существует в том случае, если между любыми двумя классами в пакетах существует любая зависимость.

Таким образом, диаграмма пакетов представляет собой диаграмму, содержащую пакеты классов и зависимости между ними. Строго говоря, пакеты и зависимости являются элементами диаграммы классов, то есть диаграмма пакетов – это форма диаграммы классов.

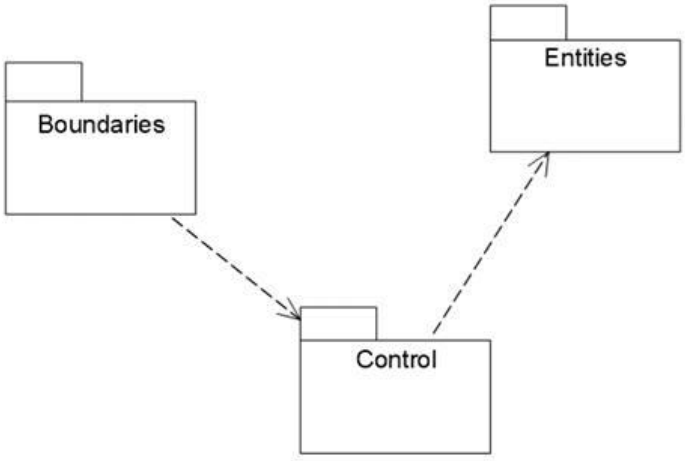


Рисунок 1.16 - Пример диаграммы пакетов

Зависимость между двумя элементами имеет место в том случае, если изменения в определении одного элемента могут повлечь за собой изменения в другом. Что касается классов, то причины для зависимостей могут быть самыми разными:

- один класс посылает сообщение другому;
- один класс включает часть данных другого класса; один класс использует другой в качестве параметра операции.

Если класс меняет свой интерфейс, то любое сообщение, которое он посылает, может утратить свою силу.

Пакеты не дают ответа на вопрос, каким образом можно уменьшить количество зависимостей в вашей системе, однако они помогают выделить эти зависимости, а после того, как они все окажутся на виду, остается только поработать над снижением их количества. Диаграммы пакетов можно считать основным средством управления общей структурой системы.

Пакеты являются жизненно необходимым средством для больших проектов. Их следует использовать в тех случаях, когда диаграмма классов, охватывающая всю систему в целом и размещенная на единственном листе бумаги формата А4, становится нечитаемой.

Диаграммы состояний

Диаграммы состояний определяют все возможные состояния, в которых может находиться конкретный объект, а также процесс смены состояний объекта в результате наступления некоторых событий.

Существует много форм диаграмм состояний, незначительно отличающихся друг от друга семантикой.

На диаграмме имеются два специальных состояния – начальное (start) и конечное (stop). Начальное состояние выделено черной точкой, оно соответствует состоянию объекта, когда он только что был создан. Конечное состояние обозначается черной точкой в белом кружке, оно соответствует состоянию объекта непосредственно перед его уничтожением. На диаграмме состояний может быть одно и только одно начальное состояние. В то же время, может быт столько конечных состояний, сколько вам нужно, или их может не быть вообще. Когда

объект находится в каком-то конкретном состоянии, могут выполняться различные процессы. Процессы, происходящие, когда объект находится в определенном состоянии, называются действиями (actions).

С состоянием можно связывать данные пяти типов: деятельность, входное действие, выходное действие, событие и история состояния.

Деятельностью (activity) называется поведение, реализуемое объектом, пока он находится в данном состоянии. Деятельность – это прерываемое поведение. Оно может выполняться до своего завершения, пока объект находится в данном состоянии, или может быть прервано переходом объекта в другое состояние. Деятельность изображают внутри самого состояния, ей должно предшествовать слово do (делать) и двоеточие.

Входным действием (entryaction) называется поведение, которое выполняется, когда объект переходит в данное состояние. Данное действие осуществляется не после того, как объект перешел в это состояние, а, скорее, как часть этого перехода. В отличие от деятельности, входное действие рассматривается как непрерываемое. Входное действие также показывают внутри состояния, ему предшествует слово entry (вход) и двоеточие.

Выходное действие (exitaction) подобно входному. Однако, оно осуществляется как составная часть процесса выхода из данного состояния. Оно является частью процесса такого перехода. Как и входное, выходное действие является непрерываемым.

Выходное действие изображают внутри состояния, ему предшествует слово exit (выход) и двоеточие.

Поведение объекта во время деятельности, при входных и выходных действиях может включать отправку события другому объекту. В этом случае описанию деятельности, входного действия или выходного действия предшествует знак « ^ ».

Соответствующая строка на диаграмме выглядит как

Do: Цель.Событие (Аргументы)

где: *Цель* – это объект, получающий событие;

Событие – это посылаемое сообщение;

Аргументы - являются параметрами посылаемого сообщения.

Деятельность может также выполняться в результате получения объектом некоторого события. При получении некоторого события выполняется определенная деятельность.

Переходом (Transition) называется перемещение из одного состояния в другое. Совокупность переходов диаграммы показывает, как объект может перемещаться между своими состояниями. На диаграмме все переходы изображают в виде стрелки, начинающейся на первоначальном состоянии и заканчивающейся последующим.

Переходы могут быть рефлексивными. Объект может перейти в то же состояние, в котором он в настоящий момент находится. Рефлексивные переходы изображают в виде стрелки, начинающейся и завершающейся на одном и том же состоянии.

У перехода существует несколько спецификаций. Они включают события, аргументы, ограждающие условия, действия и посылаемые события.

Событие (event) – это то, что вызывает переход из одного состояния в другое. Событие размещают на диаграмме вдоль линии перехода.

На диаграмме для отображения события можно использовать как имя операции, так и обычную фразу.

Большинство переходов должны иметь события, так как именно они, прежде всего, заставляют переход осуществиться. Тем не менее, бывают и автоматические переходы, не имеющие событий. При этом объект сам перемещается из одного состояния в другое со скоростью, позволяющей осуществиться входным действиям, деятельности и выходным действиям.

Ограждающие условия (guardconditions) определяют, когда переход может, а когда не может осуществиться. В противном случае переход не осуществится.

Ограждающие условия изображают на диаграмме вдоль линии перехода после имени события, заключая их в квадратные скобки.

Ограждающие условия задавать необязательно. Однако если существует несколько автоматических переходов из состояния, необходимо определить для них взаимно исключающие ограждающие условия. Это поможет читателю диаграммы понять, какой путь перехода будет автоматически выбран.

Действием (action), как уже говорилось, является непрерываемое поведение, осуществляющееся как часть перехода. Входные и выходные действия показывают внутри состояний, поскольку они определяют, что происходит, когда объект входит или выходит из него. Большую часть действий, однако, изображают вдоль линии перехода, так как они не должны осуществляться при входе или выходе из состояния.

Действие рисуют вдоль линии перехода после имени события, ему предшествует косая черта.

Событие или действие могут быть поведением внутри объекта, а могут представлять собой сообщение, посылаемое другому объекту. Если событие или действие посылается другому объекту, перед ним на диаграмме помещают знак « ^ ».

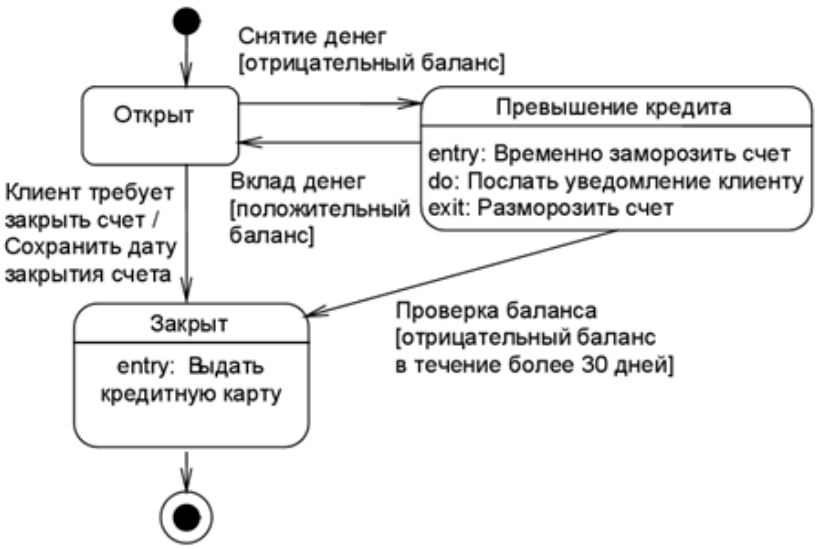


Рисунок 1.17 - Пример диаграммы состояний

Диаграммы состояний не надо создавать для каждого класса, они применяются только в сложных случаях. Если объект класса может существовать в нескольких

состояниях и в каждом из них ведет себя по-разному, для него может потребоваться такая диаграмма.

Диаграммы размещения

Диаграмма размещения (deploymentdiagram) отражает физические взаимосвязи между программными и аппаратными компонентами системы. Она является хорошим средством для того, чтобы показать маршруты перемещения объектов и компонентов в распределенной системе.

Каждый узел на диаграмме размещения представляет собой некоторый тип вычислительного устройства – в большинстве случаев, часть аппаратуры. Эта аппаратура может быть простым устройством или датчиком, а может быть и мэйнфреймом.

Диаграмма размещения показывает физическое расположение сети и местонахождение в ней различных компонентов.

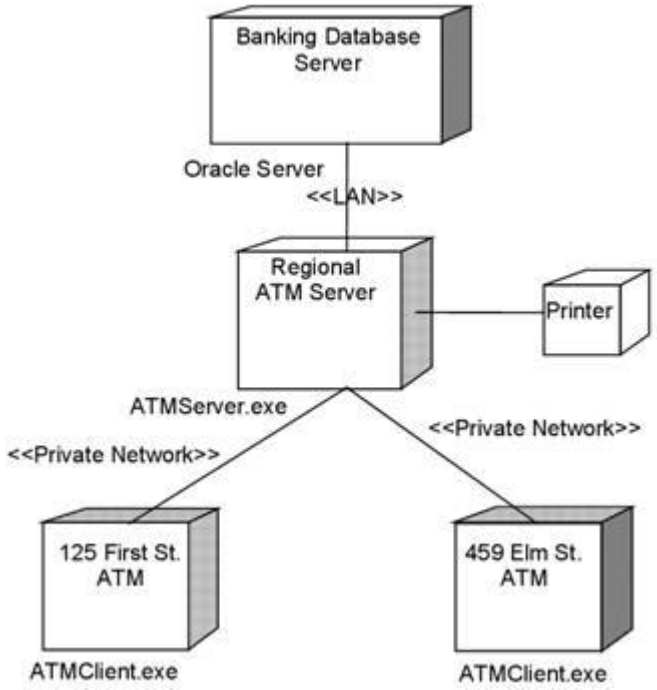


Рисунок 1.18 - Пример диаграммы размещения

Диаграмма размещения используется менеджером проекта, пользователями, архитектором системы и эксплуатационным персоналом, чтобы понять физическое размещение системы и расположение её отдельных подсистем.

Диаграммы компонентов

Диаграммы компонентов показывают, как выглядит модель на физическом уровне. На них изображены компоненты программного обеспечения и связи между ними. При этом на такой диаграмме выделяют два типа компонентов: исполняемые компоненты и библиотеки кода.

Каждый класс модели (или подсистема) преобразуется в компонент исходного кода. После создания они сразу добавляются к диаграмме компонентов. Между отдельными компонентами изображают зависимости, соответствующие зависимостям на этапе компиляции или выполнения программы.

Диаграммы компонентов применяются теми участниками проекта, кто отвечает за компиляцию системы. Из нее видно, в каком порядке надо компилировать компоненты, а также какие исполняемые компоненты будут созданы системой. На такой диаграмме показано соответствие классов реализованным компонентам. Она нужна там, где начинается генерация кода.

Объединение диаграмм компонентов и развертывания

В некоторых случаях допускается размещать диаграмму компонентов на диаграмме развертывания. Это позволяет показать какие компоненты выполняются и на каких узлах.

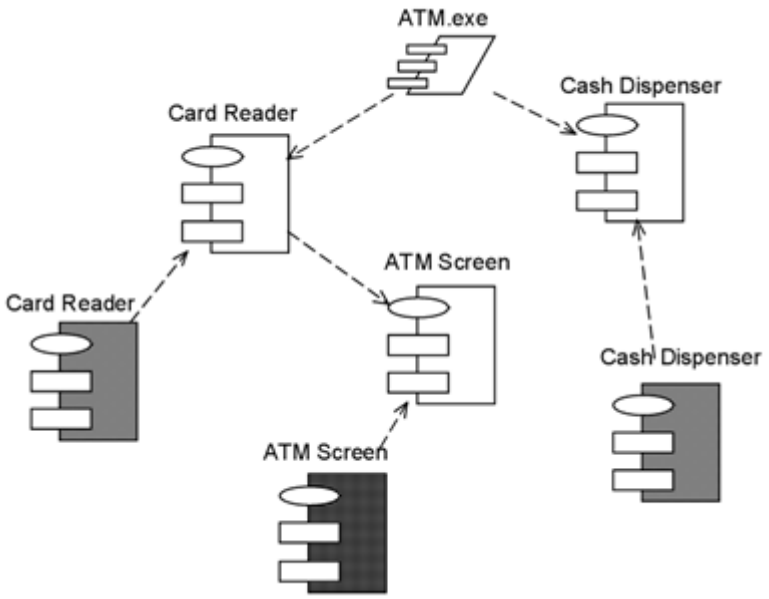


Рисунок 1.19 - Пример диаграммы компонентов

2. Практическая часть

1. Изучить предлагаемый теоретический материал.
2. Постройте диаграмму вариантов использования для выбранной информационной системы.
3. Выполните реализацию вариантов использования в терминах взаимодействующих объектов и представляющую собой набор диаграмм:
   - диаграмм классов, реализующих вариант использования;
   - диаграмм взаимодействия (диаграмм последовательности и кооперативных диаграмм), отражающих взаимодействие объектов в процессе реализации варианта использования.
4. Разделить классы по пакетам, используя один из механизмов разбиения.
5. Постройте диаграмму состояний для конкретных объектов информационной системы.
6. Построить отчёт, включающий все полученные уровни модели, описание функциональных блоков, потоков данных, хранилищ и внешних объектов.

В отчете следует указать:

1. Цель работы
2. Введение
3. Программно-аппаратные средства, используемые при выполнении работы.
4. Основную часть (описание самой работы), выполненную согласно

требованиям к результатам выполнения.

5. Заключение (выводы)

Контрольные вопросы

1. Дайте определение понятию «вариант использования».
2. Какие типы связи могут присутствовать на диаграмме вариантов использования?
3. Дайте определение понятию «действующее лицо».
4. Какие типы сообщений могут присутствовать на диаграммах взаимодействия?
5. Дайте определение понятию класс, объект класса.
6. Кем и для чего может быть использована диаграмма размещения?

Критерии оценки работы:

1. Полный ответ – 5 баллов.
2. Дополнительный уточняющий вопрос – 4 балла.
3. Краткий ответ – 3 балла.

ПРАКТИЧЕСКАЯ РАБОТА № 27

Работа с CASE – средствами проектирования программного обеспечения

Цель работы: Освоить методику построения диаграмм классов.

Исходные данные (задание):

Теоретическая часть

Class diagram (диаграмма классов) — основная диаграмма для создания кода приложения. При помощи диаграммы классов создается внутренняя структура системы, описывается наследование и взаимное положение классов друг относительно друга. Здесь описывается логическое представление системы. Именно логическое, так как классы — это лишь заготовки, на основе которых затем будут определены физические объекты.

Таким образом, диаграмма классов описывает общее представление системы и является противоположной Collaboration diagram, в которой представлены объекты системы. Однако такое разделение не является строгим правилом, и возможно смешанное представление классов и объектов.

2.1 Особенности разработки диаграмм классов в среде IBM Rational Rose 2003

Диаграмма *классов* является основным логическим представлением модели и содержит детальную информацию о внутреннем устройстве объектно-ориентированной программной системы или, используя современную терминологию, об архитектуре программной системы. Активизировать рабочее окно диаграммы *классов* можно несколькими способами:

- окно диаграммы *классов* появляется по умолчанию в рабочем окне диаграммы после создания нового проекта;
- щелкнуть на кнопке с изображением диаграммы классов на стандартной панели инструментов;
- раскрыть логическое представление (Logical View) в браузере проекта и дважды щелкнуть на пиктограмме Main (Главная);
- выполнить операцию главного меню: Browse→Class Diagram (Обзор→ Диаграмма *классов*).

При этом появляется новое окно с чистым рабочим листом диаграммы *классов* и специальная панель инструментов, содержащая кнопки с изображением графических примитивов, необходимых для разработки диаграммы *классов* (таблица 2.1). Назначение отдельных кнопок панели можно узнать также из всплывающих подсказок.

2.2 Добавление класса на диаграмму классов и редактирование его свойств

Для добавления класса на диаграмму классов нужно с помощью левой кнопки мыши нажать кнопку с изображением пиктограммы класса на специальной панели

инструментов, отпустить левую кнопку мыши и щелкнуть левой кнопкой мыши на свободном месте рабочего листа диаграммы. На диаграмме появится изображение класса с маркерами изменения его геометрических размеров и предложенным средой именем по умолчанию NewClass.

Продолжая разработку модели приложения для АИС своего варианта, например, «Трудоустройство», в качестве сквозного примера проекта, построим для этой модели следующую диаграмму классов. С этой целью следует изменить предложенное по умолчанию имя диаграммы Main на - Диаграмма классов «Трудоустройство», а имя добавленного на диаграмму класса - на «Matrix» (рисунок 2.1).

Для класса «Matrix» можно уточнить его назначение в модели с помощью указания стереотипа и пояснительного текста в форме документации. С этой целью двойным щелчком левой кнопкой мыши на изображении этого класса на диаграмме или в браузере проекта следует открыть диалоговое окно спецификации свойств этого класса (рисунок 2.2) и на вкладке General (Общие) выбрать из вложенного списка Stereotype стереотип.

Таблица 2.1 - Назначение кнопок специальной панели инструментов для диаграммы классов

Графическое изображение	**Всплывающая подсказка**	**Назначение кнопки**
	Selection Tool	Превращает изображение курсора в форму стрелки для последующего выделения элементов на диаграмме
	Text Box	Добавляет на диаграмму текстовую область
	Note	Добавляет на диаграмму примечание
	Anchor Note to Item	Добавляет на диаграмму связь примечания с соответствующим графическим элементом диаграммы
	Class	Добавляет на диаграмму *класс*
	Interfase	Добавляет на диаграмму *интерфейс*
	Unidirectional Association	Добавляет на диаграмму направленную *ассоциацию*
	Association Class	Добавляет на диаграмму *ассоциацию класс*
	Package	Добавляет на диаграмму пакет
	Dependency or Instantiates	Добавляет на диаграмму отношение зависимости
	Generalization	Добавляет на диаграмму отношение обобщения
	Realize	Добавляет на диаграмму отношение реализации

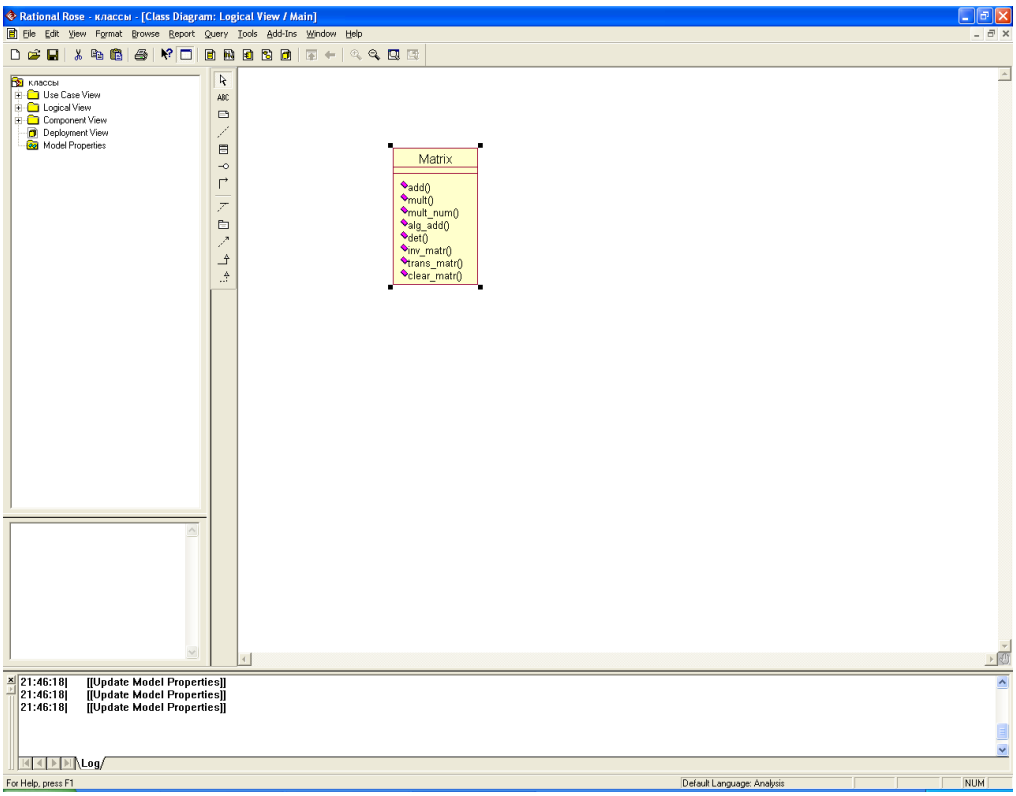


Рисунок 2.1 - Диаграмма классов после добавления на нее класса «Matrix»

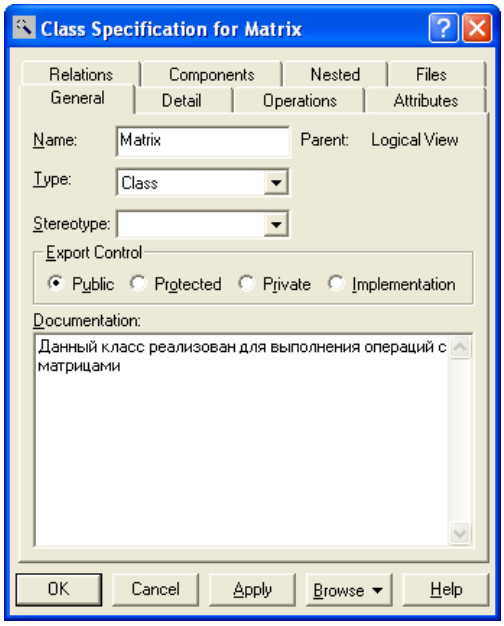


Рисунок 2.2 - Диалоговое окно спецификации свойств класса

Для отдельного класса можно уточнить также и другие его свойства, доступные для редактирования на вкладке Detail (Подробно) окна спецификации свойств этого класса. Например, на этой вкладке с помощью вложенного списка Multiplicity (Кратность) можно задать количество объектов или экземпляров данного класса, для чего следует выбрать строку с буквой n. Данное значение означает, что у класса может быть любое конечное число экземпляров (рисунок 2.3). Поле ввода с именем Space (Пространство) служит для указания объема абсолютной или относительной памяти, которая требуется, по оценке разработчика, для реализации каждого объекта данного класса. Применительно к рассматриваемой модели это поле можно оставить

пустым.

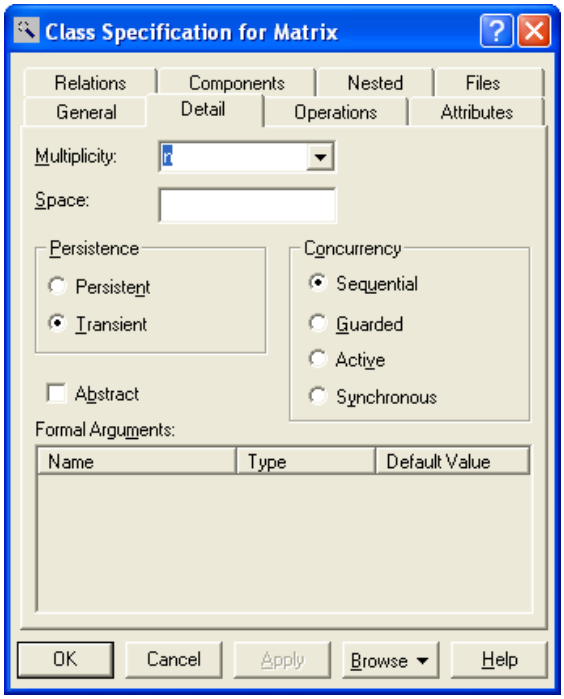


Рисунок 2.3 - Диалоговое окно спецификации свойств класса

Далее можно задать устойчивость классов в группе выбора Persistence. При этом выбор свойства Persistent (Устойчивый) означает, что информация об объектах данного класса должна быть сохранена в системе. Выбор свойства Transient (Временный) означает, что нет необходимости сохранять информацию об объектах данного класса в системе после завершения работы программного приложения. Применительно к рассматриваемой модели следует выбрать свойство Persistent.

В группе выбора Concurrency (Параллельность) можно специфицировать условия на возможность реализации объектов данного класса в параллельных потоках управления. Для выбора могут быть использованы следующие свойства:

Sequential (Последовательный) - свойство по умолчанию, которое означает, что объекты класса будут вести себя нормально только при наличии одного потока управления, т. е. соответствующие операции объектов должны выполняться последовательно. В то же время при наличии нескольких потоков управления стабильное поведение объектов класса не гарантируется.

Guarded (Безопасный) - означает, что при наличии нескольких потоков управления объекты класса будут вести себя ожидаемым от них образом. Для этого объекты в различных потоках должны взаимодействовать друг с другом для того, чтобы гарантировать отсутствие конфликта между ними.

Active (Активный) - означает, что класс должен иметь свой собственный поток управления.

Synchronous (Синхронный) - означает, что объекты класса будут вести себя ожидаемым от них образом при наличии нескольких потоков управления. При этом нет необходимости во взаимодействии объектов в различных потоках управления, поскольку объекты данного класса могут самостоятельно разрешать возможные конфликты.

Для того, чтобы специфицировать класс как абстрактный, т.е. не имеющий

экземпляров, следует на этой же вкладке выставить отметку в свойстве Abstract (Абстрактный).

2.3 Добавление и редактирование атрибутов классов

Из всех графических элементов среды IBM Rational Rose 2003 класс обладает максимальным набором свойств, главными из которых являются его *атрибуты* и *операции*.

Добавить *атрибут* к созданному ранее классу можно одним из следующих способов:

- с помощью *операции* контекстного меню New Attribute (Новый *атрибут*) для класса, выделенного на диаграмме классов. В этом случае активизируется курсор ввода текста в области графического изображения класса на диаграмме.

- с помощью *операции* контекстного меню: New→Attribute (Новый→*Атрибут*) для класса, выделенного в браузере проекта. В этом случае активизируется курсор ввода текста в области иерархического представления класса в браузере проекта под именем соответствующего класса.

- с помощью *операции* контекстного меню Insert (Вставить), вызванного при позиционировании курсора в области открытой вкладки *атрибутов* в диалоговом окне свойств Class Specification соответствующего класса.

После добавления *атрибута* к классу по умолчанию ему присваивается имя name и некоторый *квантор видимости* (рисунок 2.4).

Напомним, что имена *атрибутов* и *операций* классов должны начинаться со строчной буквы. Видимость *атрибутов* на диаграмме классов изображается в форме специальных пиктограмм. Используемые пиктограммы видимости изображаются перед именем соответствующего *атрибута* и имеют следующий смысл (таблица 2.2).

Для редактирования свойств *атрибутов* предназначено специальное диалоговое окно спецификации *атрибута* Class Attribute Specification, которое открывается двойным щелчком мыши на строке выбранного *атрибута* в окне спецификации свойств класса. В окне свойств отдельного *атрибута* класса можно задать *тип* данных *атрибута* и его начальное *значение*, а также назначить *атрибуту* стереотип из раскрывающегося списка или изменить его *квантор видимости*.

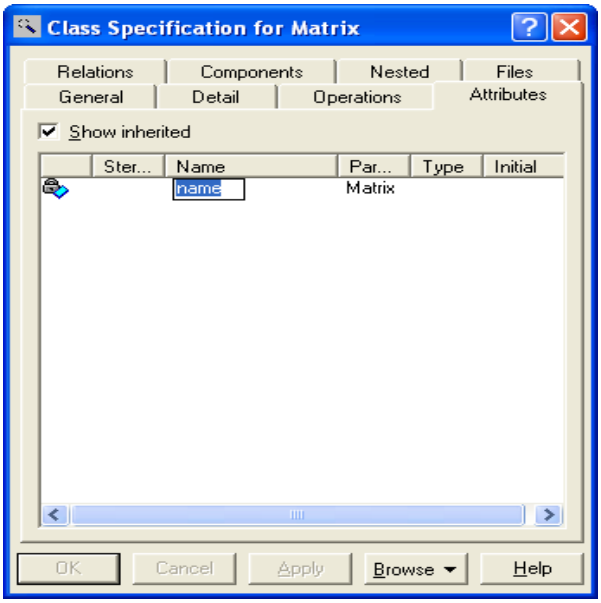


Рисунок 2.4 - Диалоговое окно спецификации свойств класса после добавления нового атрибута

Таблица 2.2 - Пиктограммы видимости атрибутов классов

Графическое изображение	Текстовый аналог	Назначение пиктограммы
	Public	Общедоступный или открытый. В нотации языка UML такому *атрибуту* соответствует знак «+»
	Protected	Защищенный. В нотации языка UML такому *атрибуту* соответствует знак «#»
	Private	Закрытый. В нотации языка UML такому *атрибуту* соответствует знак «-»
	Implementation	Реализация. В нотации языка UML такому *атрибуту* соответствует знак «~»

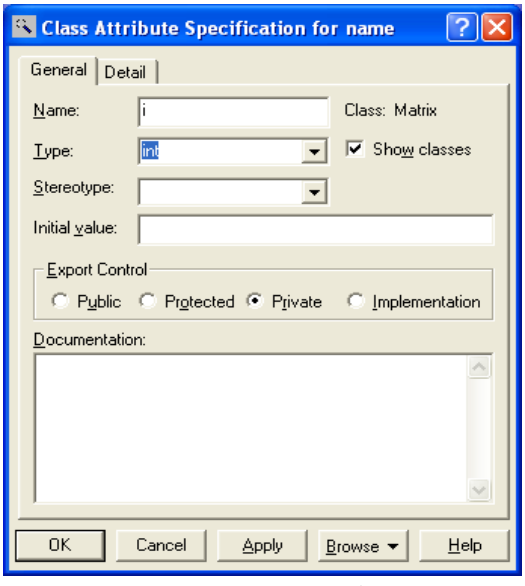


Рисунок 2.5 - Диалоговое окно спецификации свойств атрибута

2.4 Добавление и редактирование операций классов

Функционирование любой системы основано на выполнении отдельными его элементами тех или иных действий. В нашей модели все действия представляются с помощью *операций* классов. Таким образом, следующий этап разработки диаграммы классов связан со спецификацией *операций* классов.

Добавить *операцию* к созданному ранее классу можно одним из следующих способов:

- с помощью *операции* контекстного меню New Operation (Новая *операция*) для класса, выделенного на диаграмме классов. В этом случае активизируется курсор ввода в области графического изображения класса на диаграмме.
- с помощью *операции* контекстного меню: New→Operation (Новая→*Операция*) для класса, выделенного в браузере проекта. В этом случае активизируется курсор ввода в области иерархического представления класса в браузере под именем соответствующего класса.
- с помощью *операции* контекстного меню Insert (Вставить), вызванного при позиционировании курсора в области открытой вкладки *операций* в диалоговом окне свойств Class Specification соответствующего класса.

После добавления *операции* к классу по умолчанию ей присваивается имя opname и некоторый *квантор видимости*. Видимость *операций* на диаграмме классов также изображается в форме специальных пиктограмм. Используемые пиктограммы видимости изображаются перед именем соответствующей *операции* и имеют следующий смысл (таблица 2.3).

Таблица 2.3 - Пиктограммы видимости операций классов

Графическое изображение	Текстовый аналог	Назначение пиктограммы
	Public	Общедоступный или открытый. В нотации языка UML такому *атрибуту* соответствует знак «+»
	Protected	Защищенный. В нотации языка UML такому *атрибуту* соответствует знак «#»
	Private	Закрытый. В нотации языка UML такому *атрибуту* соответствует знак «-»
	Implementation	Реализация. В нотации языка UML такому *атрибуту* соответствует знак «~»

Каждая из *операций* классов имеет собственное диалоговое окно спецификации свойств Operation Specification, которое может быть открыто по двойному щелчку на имени *операции* на соответствующей вкладке спецификации класса или на имени этой *операции* в браузере проекта (рисунок 2.6).

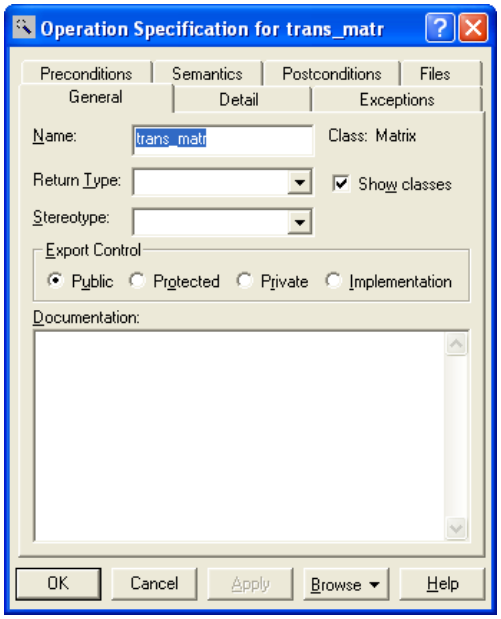


Рисунок 2.6 - Диалоговое окно спецификации свойств операции

Для *операций* классов кроме *квантора видимости* можно также задать: *аргументы* и их тип, *тип возвращаемого результата*, стереотип *операции*, а также определить протокол и размер, задать исключительные ситуации, специфицировать предусловия и постусловия и целый ряд других свойств. Для отдельной *операции* эти дополнительные свойства доступны для редактирования на вкладке Detail (Подробно) диалогового окна спецификации свойств выбранной *операции* (рисунок 2.7).

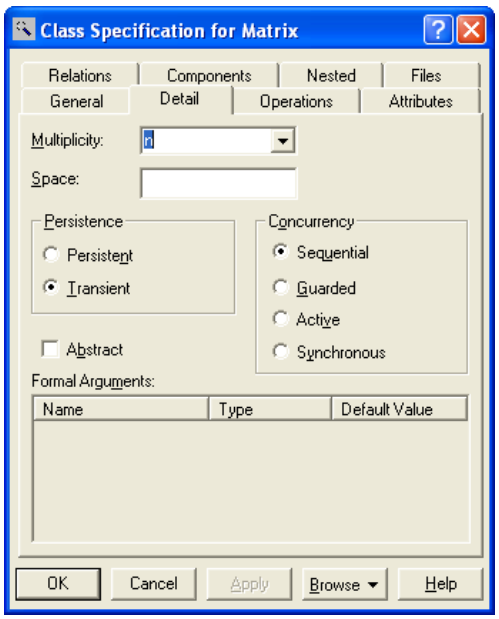


Рисунок 2.7 - Диалоговое окно спецификации свойств операции, открытое на вкладке Detail (Подробно)

На вкладке Detail в многостраничном поле Arguments (Аргументы) можно определить *аргументы редактируемой операции*. Для этого следует выполнить *операцию* контекстного меню Insert (Вставить). После этого в этом поле появится аргумент данной *операции* с именем по умолчанию argname. Для редактирования свойств аргумента предназначено специальное окно свойств аргумента.

На вкладке Detail в поле Protocol (Протокол) можно специфицировать порядок выполнения *операций* класса, например, указать, что одна *операция* не может быть вызвана раньше другой. Соответствующий текст в данное поле вводится с клавиатуры и попадает в генерируемый код в форме комментария. В поле Qualification (Квалификация) можно уточнить детали реализации *операции*, связанные с конкретным языком программирования. Соответствующий текст также вводится в данное поле с клавиатуры и попадает в генерируемый код в форме комментария.

Далее на этой же вкладке в полях Size (Размер) и Time (Время) можно специфицировать предполагаемый объем памяти и время, необходимое для выполнения *операции*. Соответствующая информация попадает в генерируемый код в форме комментария.

В группе выбора Concurrency (Параллельность) можно специфицировать условия на возможность параллельного выполнения данной *операции*. Для выбора могут быть использованы следующие свойства:

Sequential (Последовательная) - свойство по умолчанию, которое означает, что данная *операция* класса может быть выполнена только при наличии одного потока управления, т. е. соответствующая *операция* класса должна выполняться последовательно. При наличии нескольких потоков управления выполнение данной *операции* класса не гарантируется.

Guarded (Безопасная) - означает, что при наличии нескольких потоков управления выполнение данной *операции* класса гарантируется только в том случае, когда обеспечено взаимодействие объектов друг с другом в различных потоках.

Synchronous (Синхронная) - означает, что выполнение данной *операции* класса гарантируется при наличии нескольких потоков управления. При этом нет необходимости во взаимодействии объектов в различных потоках управления, поскольку данная *операция* класса будет выполняться в отдельном потоке управления вплоть до своего завершения.

2.5 Добавление ассоциации на диаграмму классов и редактирование ее свойств

Добавление на диаграмму *ассоциации* между двумя классами выполняется следующим образом. На специальной панели инструментов необходимо нажать кнопку с изображением пиктограммы направленной *ассоциации* и отпустить левую кнопку мыши. Если *ассоциация* - направленная, то на диаграмме классов надо выделить первый элемент *ассоциации* или источник, от которого исходит стрелка, и, не отпуская нажатую левую кнопку мыши, переместить ее указатель ко второму элементу отношения или приемнику, к которому направлена стрелка. После перемещения ко второму элементу кнопку мыши следует отпустить, в результате чего на диаграмму классов будет добавлена направленная *ассоциация* с именем Untitled между двумя выбранными классами (рисунок 2.8).

Рисунок 2.8 - Фрагмент диаграммы классов после добавления на неё

направленной ассоциации

Изменим имя для данной *ассоциации*, предложенное средой по умолчанию. Это можно выполнить с помощью окна спецификации свойств *ассоциации*. Доступ к диалоговому окну спецификации свойств *ассоциации* Association Specification можно получить после выделения линии *ассоциации* на диаграмме классов или в браузере проекта и двойного щелчка на ней левой кнопки мыши (рисунок 2.9).

Для задания имени *ассоциации* следует на вкладке General (Общие) в поле ввода Name (Имя) ввести текст ее имени: Соответствует и нажать кнопку Apply или OK, чтобы сохранить результаты редактирования имени *ассоциации*. Для *ассоциации* можно задать также *кратность* каждого из концов *ассоциации*, стереотип, использовать ограничения и роли, а также некоторые другие свойства.

Если *ассоциация* является ненаправленной, то порядок выбора классов может быть произвольный, а после добавления *ассоциации* на диаграмму классов следует изменить значение соответствующего свойства данной *ассоциации*. С этой целью необходимо перейти на вкладку Role A Detail в окне спецификации свойств *ассоциации* и убрать отметку у свойства Navigable (Навигация).

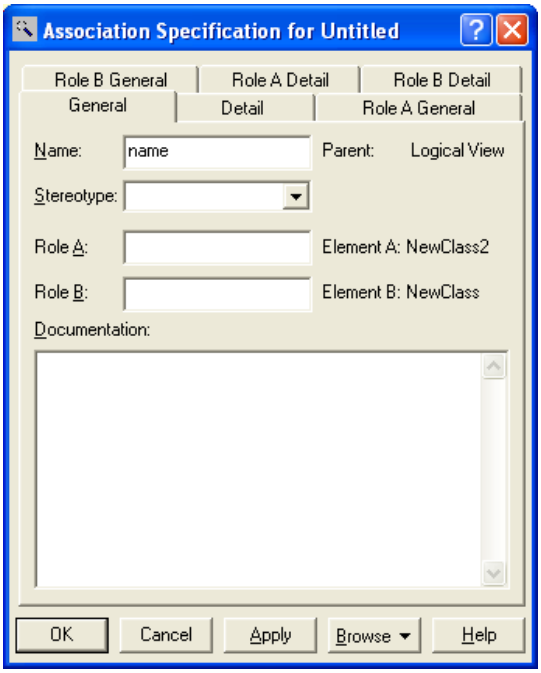


Рисунок 2.9 - Диалоговое окно спецификации свойств ассоциации

2.6 Добавление отношений агрегации и композиции на диаграмму классов и редактирование их свойств

Агрегация – это частный случай ассоциации, описывающий объекты, состоящие из частей. Например, газонокосилка состоит из ножа, двигателя, нескольких колес и корпуса. Здесь газонокосилка является агрегатом (незакрашенный ромбик), а остальные детали – составляющими частями (стрелка).

Композиция – это частный случай агрегации. Композиция подразумевает, что части принадлежат целому. Например, компания, состоящая из отделений, которые в свою очередь, состоят из отделов.

- добавить на диаграмму отношение *агрегации* между двумя классами можно следующими способами:

- щелкнуть на кнопке с изображением отношение *агрегации* на специальной панели инструментов и провести линию *агрегации* от одного класса к другому.

Провести линию *ассоциации* между выбранными классами и изменить ее свойства таким образом, чтобы превратить данную *ассоциацию* в *агрегацию*.

В первом случае может оказаться, что по умолчанию на специальной панели инструментов диаграммы классов отсутствует кнопка с пиктограммой *агрегации*. В этом случае необходимо предварительно добавить ее на панель инструментов одним из описанных ранее способов. Во втором случае следует открыть окно спецификации свойств *ассоциации* Association Specification и на вкладке деталей соответствующего конца *ассоциации* выставить отметку в строке выбора Aggregate (*Агрегация*).

В качестве примера изменим тип созданной ранее *ассоциации* и сделаем ее агрегацией. С этой целью на вкладке Role B Detail деталей конца *ассоциации* одного класса следует выставить отметку в строке выбора Aggregate(рисунок 2.10).

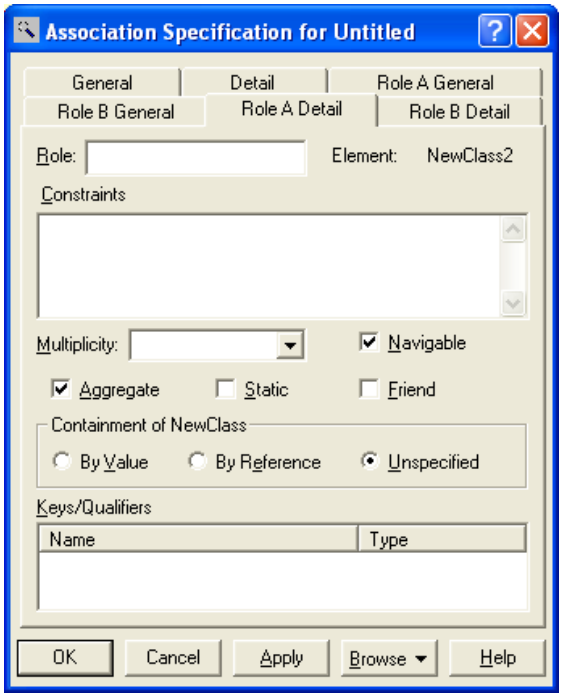


Рисунок 2.10 -Диалоговое окно спецификации свойств ассоциации

Соответствующий фрагмент диаграммы классов после изменения *ассоциации* между классами на отношение *агрегации* будет иметь следующий вид (рисунок 2.11).

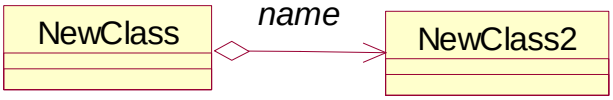


Рисунок 2.11 - Фрагмент диаграммы классов модели после добавления на нее отношения агрегации

Для изображения отношения *композиции* можно также вначале изобразить обычную *ассоциацию*, после чего, открыв окно ее свойств на вкладке деталей

соответствующего конца *ассоциации*, выставить отметку в строке выбора Aggregate (*Агрегация*) и в секции Containment (Локализация) выбрать опцию By Value (По значению). По умолчанию эта опция не специфицирована, т.е. выставлена отметка опции Unspecified (рисунок 2.12).

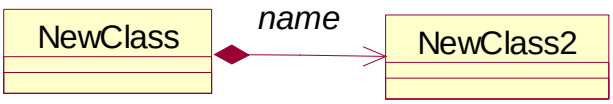


Рисунок 2.12 - Фрагмент диаграммы классов модели после добавления на нее отношения композиции

2.7 Добавление отношения обобщения на диаграмму классов и редактирование ее свойств

Добавление на диаграмму отношения *обобщения* между двумя классами выполняется следующим образом. На специальной панели инструментов необходимо нажать кнопку с изображением пиктограммы *обобщения* и отпустить левую кнопку мыши. Далее на диаграмме классов надо выделить первый элемент *обобщения* или потомок, от которого исходит стрелка, и, не отпуская нажатую левую кнопку мыши, переместить ее указатель ко второму элементу отношения или предок, к которому направлена стрелка. После перемещения ко второму элементу кнопку мыши следует отпустить, в результате чего на диаграмму классов будет добавлена линия *обобщения* с именем Untitled между двумя выбранными классами. Пример приведен на рисунке 2.13.

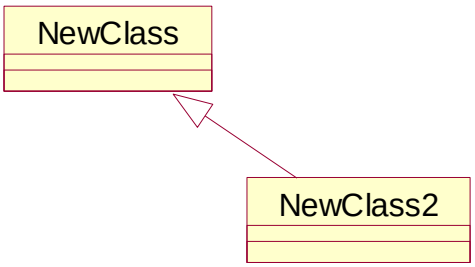


Рисунок 2.13 – Пример диаграммы классов после добавления на неё отношения обобщения

Изменим имя отношения *обобщения*, предложенное средой по умолчанию. Это можно выполнить с помощью окна спецификации свойств *обобщения*. Доступ к диалоговому окну спецификации свойств отношения *обобщения* Generalize Specification можно получить после выделения линии *обобщения* на диаграмме классов или в браузере проекта и двойного щелчка на ней левой кнопки мыши (рисунок 2.14).

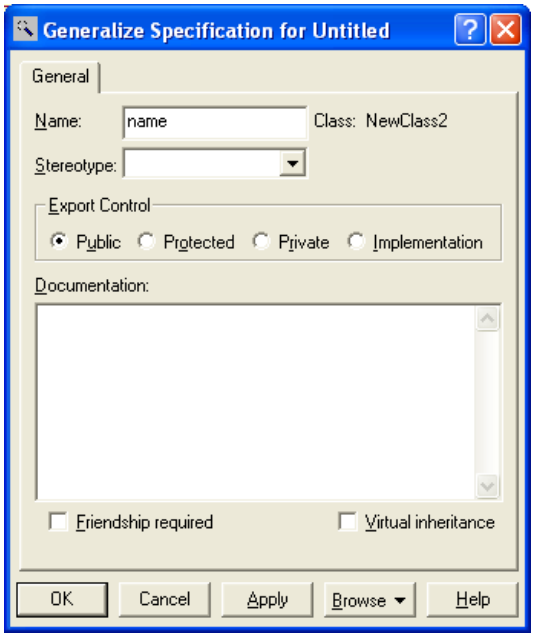


Рисунок 2.14 - Диалоговое окно спецификации свойств отношения обобщения

Для задания имени *обобщения* следует на единственной вкладке General (Общие) в поле ввода Name (Имя) ввести текст ее имени: следует и нажать кнопку Apply или OK, чтобы сохранить результаты редактирования имени *ассоциации*.

Пример построения диаграммы классов

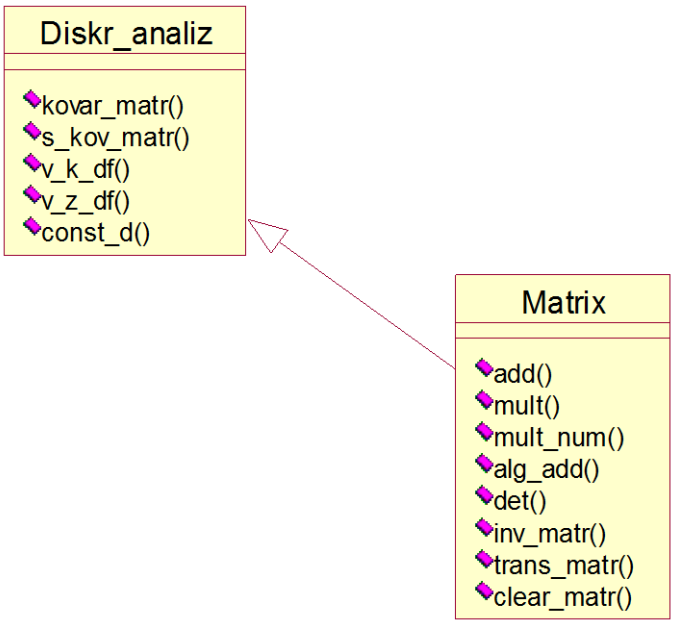


Рисунок 2.15 – Диаграмма классов для алгоритма дискриминантного анализа

Задание на лабораторное занятие

1. Изучить теоретический материал
2. Согласно заданному варианту, разработайте диаграмму классов для реализации метода многомерного статистического анализа.

Контрольные вопросы

1. Каково назначение диаграммы классов?
2. Какими способами можно создать диаграмму?
3. Какие инструменты доступны для диаграммы?
4. Какие команды предоставляет контекстное меню класса?
5. Как настроить свойства атрибутов класса?
6. Как настроить свойства методов класса?
7. Какие типы отношений классов вы знаете?

Работа с CASE – средствами кодирования программного обеспечения

Цель работы: Подготовка модели к генерации программного кода; генерация программного кода

Теоретическая часть

Одно из самых мощных свойств Rational Rose – возможность генерации программного кода, представляющего модель. Варианты генерации программ меняются в зависимости от установленной версии Rose. В настоящее время применяются следующие три различные версии:

- Rose Modeler – позволяет создавать модель системы, но не поддерживает генерацию программного кода и обратное проектирование.
- Rose Professional – позволяет генерировать программный код на одном языке.
- Rose Enterprise – позволяет генерировать программный код на Ada 83, Ada 95, ANSI C++, CORBA, Java, COM, Visual Basic, Visual C++, C++ и XML. Кроме того, поддерживается генерация кода и обратное проектирование баз данных.

Лабораторный практикум построен на изучении Case-средства Rational Rose версии - Enterprise Edition.

Подготовка к генерации программного кода

Процесс генерации программного кода состоит из пяти основных этапов:

1. Проверка модели.
2. Создание компонентов.
3. Отображение классов на компоненты.
4. Установка свойств генерации программного кода.
5. Генерация программного кода.

В разных языках не все этапы обязательны. Так, программы, разработанные на языке C++, генерируются и без предварительного создания компонентов. Генерировать код программ на любом языке можно, не выполняя проверки модели, хотя во время генерации это порой приводит к различным ошибкам.

Хотя не все этапы обязательны, первые пять из них рекомендуется выполнять до начала процесса генерации. Проверка модели поможет выявить ее неточности и недостатки, нежелательные с точки зрения последующей генерации программ. Этапы, на которых рассматриваются компоненты, - это способ отобразить логическую схему системы на ее физическую реализацию, причем на этих этапах собирается большой объем полезной информации. Если пропустить эти этапы, то для создания компонентов Rose воспользуется пакетной структурой логического представления.

2.1.1 Проверка модели

В Rose существует не зависящее от языка средство проверки моделей, применяемое для обеспечения корректности модели перед генерацией программного кода. Такая проверка помогает выявить в модели неточности и ошибки, не позволяющие генерировать программы надлежащим образом.

В общем случае проверка модели может выполняться на любом этапе работы над проектом. Однако после завершения разработки графических диаграмм она является обязательной, поскольку позволяет выявить целый ряд ошибок

разработчика. К числу таких ошибок и предупреждений относятся, например, не используемые ассоциации и классы, оставшиеся после удаления отдельных графических элементов с диаграмм, а также операции, не являющиеся именами сообщений на диаграммах взаимодействия.

Для проверки модели следует выполнить операцию главного меню: Tools→ Check Model (Инструменты→Проверить модель). Результаты проверки разработанной модели на наличие ошибок отображаются в окне журнала. Прежде чем приступить к генерации текста программного кода разработчику следует добиться устранения всех ошибок и предупреждений, о чем должно свидетельствовать чистое окно журнала (рис. 5.1).

Рисунок 5.1- Вид журнала при отсутствии ошибок по результатам проверки модели

К наиболее распространенным ошибкам относятся, например, сообщения на диаграмме Последовательности или Кооперативной диаграмме, не отображенные на операцию, либо объекты этих диаграмм, не отображенные на класс.

2.1.2 Создание компонентов

Второй этап процесса генерации программного кода - создание компонентов для классов. Существуют компоненты самых разных типов: файлы исходного программного кода, исполняемые файлы, библиотеки исполнения в реальном времени, компоненты ActiveX, апплеты и т.д. Перед генерацией программ можно отобразить каждый из классов на соответствующий компонент исходного программного кода.

После создания компонентов можно добавить зависимости между ними на диаграмме Компонентов. Зависимости между компонентами - это зависимости во время компиляции системы.

При генерации программ на C++, Java или Visual Basic выполнять подобный шаг не обязательно. В Java и Visual Basic Rose создаст автоматически соответствующий компонент для каждого из классов.

Для создания компонента:

1. Откройте диаграмму Компонентов (Component).

2. С помощью значка Component панели инструментов Diagram введите новый компонент в диаграмму.

2.1.3 Отображение классов на компоненты

Каждый компонент исходного кода - это файл с исходным программным кодом для одного или нескольких классов. В C++ каждый класс отображается на два компонента с исходным кодом: файл заголовка и основной файл (тело). В PowerBuilder на один компонент отображается несколько классов. Компонентом с исходным программным кодом в PowerBuilder является файл библиотеки PowerBuilder (.pbl). В Java каждый компонент - это один файл .java. Компоненты

также создаются для элементов управления ActiveX, апплетов, файлов DDL, исполняемых файлов, а также других исходных и скомпилированных файлов.

Третий этап процесса генерации программного кода - отображение каждого из классов на соответствующие компоненты. В PowerBuilder необходимо отобразить каждый класс на компонент перед генерацией программы, в то время как в C++, Java и Visual Basic этот шаг не является обязательным. Rose может генерировать программный код самостоятельно. При генерации в Rose программ Java и Visual Basic производится еще и генерация нужных компонентов и отображение классов. Однако для C++ компоненты не создаются автоматически, а кроме того, ни для одного из языков не генерируются зависимости. Поэтому рекомендуется выполнять этот шаг независимо от применяемого языка программирования. Для отображения класса на компонент:

1. Щелкните правой кнопкой мыши на компоненте, на диаграмме Компонентов или в браузере.
2. Выберите Open Specification в контекстном меню.
3. Выберите вкладку Realizes (Реализует).
4. Во вкладке Realizes щелкните правой кнопкой мыши на нужном классе (классах) и выберите Assign (Присвоить) в контекстном меню.

В браузере имя компонента будет показано в круглых скобках вслед за именем класса в Логическом представлении (Logical view).

ИЛИ

1. Найдите класс в окне Logical view браузера.
2. Перетащите класс на нужный компонент в окне Component view.
3. В окне Logical view имя компонента будет показано в круглых скобках за именем класса.

2.1.4 Установка свойств генерации программного кода

Для каждого языка в Rose предусмотрен ряд определенных свойств генерации программного кода. Можно установить несколько параметров генерации программного кода для классов, атрибутов, компонентов и других элементов модели. Этими свойствами определяется способ генерации программ. В Rose предлагаются общепринятые параметры по умолчанию.

2.1.4.1 Настройка свойств C++

Для того чтобы пользоваться возможностями C++, необходимо описать назначение этих свойств, список которых доступен во вкладке C++ спецификаций класса. (см. рис. 5.2)

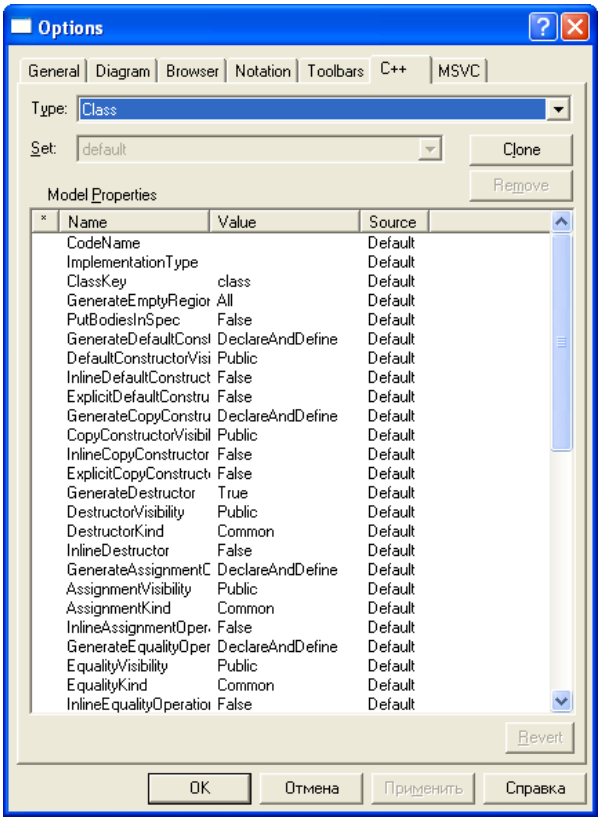


Рисунок 5.2 – Окно свойств генерации кода на C++

Перед генерацией программного кода рекомендуется анализировать эти свойства и вносить необходимые изменения.

Назначение свойств:

1) **CodeName** - устанавливает имя класса в создаваемом коде. Данное свойство необходимо устанавливать только в том случае, если имя класса должно быть отлично от имени заданного в модели Rational Rose. Данное свойство необходимо использовать для создания работоспособного кода C++, если для классов в модели используются русские имена.

2) **ImplementationType** - позволяет использовать простые типы вместо определения класса, устанавливаемого Rational Rose по умолчанию. При задании этого параметра создается директива typedef.

3) **ClassKey** - используется для задания типа класса, такого как class, struct, или union. Если тип не указан, то создается класс.

4) **GenerateEmptyRegion** - свойство указывает, как будет создаваться пустой раздел protected: None - пустой раздел не будут создан; Preserved - пустой раздел будет создан, если будет установлено свойство «preserve=yes»; Unpreserved — пустой раздел будет создан, если будет установлено свойство «preserve=no»; All — всегда будет создаваться.

5) **PutBodiesInSpec** - если установлено как true, то в заголовочный файл попадет и описание тела класса. Используется для компиляторов, которым необходимо определение шаблона класса в каждом компилируемом файле.

6) **GenerateDefaultConstructor** - позволяет установить, необходимо ли создавать конструктор для класса по умолчанию. Может принимать следующие значения: DeclareAndDefine - создается определение для конструктора и скелет конструктора в теле класса; Declare Only - создается только определение;

DoNotDeclare - не создается ни определения, ни скелета конструктора.

7) **DefaultConstructorVisibility** - устанавливает раздел, в котором будет определен конструктор по умолчанию: public, protected, private, implementation.

8) **InlineDefaultConstructor** - устанавливает, будет ли конструктор по умолчанию создаваться как inline подстановка. Если конструктора по умолчанию нет, то данное свойство не оказывает на код никакого эффекта.

9) **ExplicitDefaultConstructor** - устанавливает конструктор по умолчанию как explicit (явно заданный).

10) **InlineRelationalOperations** - определяет, будут ли функции операторов сравнения создаваться как inline подстановка.

11) **GenerateStorageMgmtOperations** - определяет, будут ли переопределяться операторы new и delete в классе.

12) **StorageMgmtVisibility** - определяет раздел, в который будут помещены операторы new и delete.

13) **InlineStorageMgmtOperations** - определяет, будут ли операторы new и delete определены как inline подстановка.

14) **GenerateSubscriptOperation** - определяет, будет ли переопределен оператор [].

15) **Subscript Visibility** определяет - раздел, в который будет помещен оператор [].

16) **SubscriptKind** - определяет вид функций оператора []: Common - обычная, Virtual - виртуальная, Abstract - абстрактная.

17) **SubscriptResultType** - определяет тип возвращаемого выражения для оператора [].

18) **InlineSubscriptOperation** - определяет, будет ли оператор [] определен как inline подстановка.

19) **GenerateDereferenceOperation** - определяет, будет ли переопределен оператор *.

20) **Dereference Visibility** - определяет раздел, в который будет помещен оператор *.

21) **DereferenceKind** - определяет вид функций оператора *: Common - обычная, Virtual - виртуальная, Abstract - абстрактная.

22) **DereferenceResultType** - определяет тип возвращаемого выражения для оператора *.

23) **InlineDereferenceOperation** - определяет, будет ли оператор * определен, как inline подстановка.

24) **GenerateIndirectionOperation** - определяет, будет ли переопределен оператор ->.

25) **IndirectionVisibility** - определяет раздел, в который будет помещен оператор ->.

26) **IndirectionKind** - определяет вид функций оператора ->: Common - обычная, Virtual - виртуальная, Abstract - абстрактная.

27) **IndirectionResultType** - определяет тип возвращаемого выражения для оператора ->.

28) **InlineIndirectionOperation** - определяет, будет ли оператор -> определен как inline подстановка.

29) **GenerateStreamOperations** - определяет, будут ли переопределены

операторы потоков (« и »).

2.1.5 Выбор класса, компонента или пакета

При генерации программного кода за один раз можно создать класс, компонент или целый пакет. Программный код генерируется с помощью диаграммы или браузера. При генерации кода из пакета можно выбрать либо пакет Логического представления на диаграмме Классов, либо пакет представления Компонентов на диаграмме Компонентов. При выборе пакета Логического представления генерируются все его классы. При выборе пакета представления Компонентов генерируются все его компоненты.

Программный код можно генерировать одновременно для нескольких классов, компонентов или пакетов. На диаграмме с помощью клавиши Ctrl выберите классы, компоненты или пакеты, для которых нужно сгенерировать программный код, а затем - соответствующую команду генерации в меню.

2.1.6 Генерация программного кода

Если у вас установлены Rose Professional или Rose Enterprise, то в меню Tools предлагается несколько вариантов, специфичных для конкретного языка программирования (см. рис. 5.3).

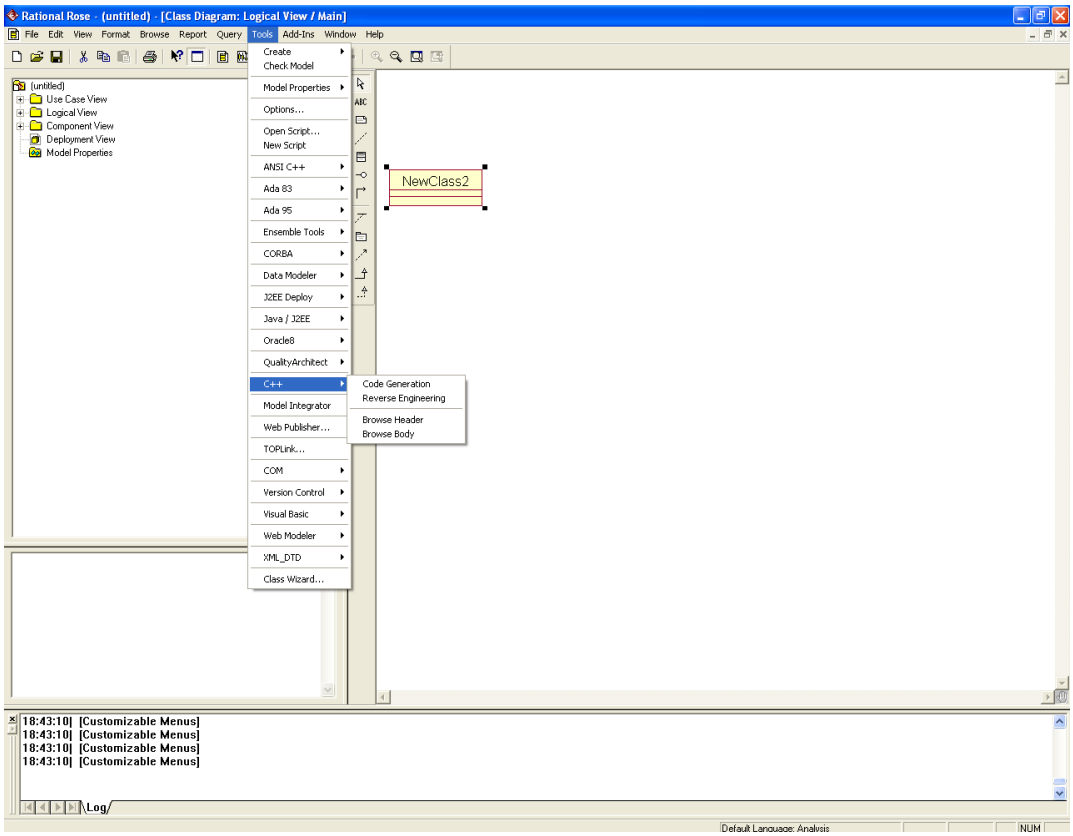


Рисунок 5.3 - Пункты меню генерации кода

Чтобы показать или скрыть эти пункты меню, выберите пункт Add-Ins → Add-Ins (Надстройки → Менеджер надстроек). В диалоговом окне Add-In Manager (см. рис. 5.4) с помощью флажков покажите или скройте нужные варианты для различных языков.

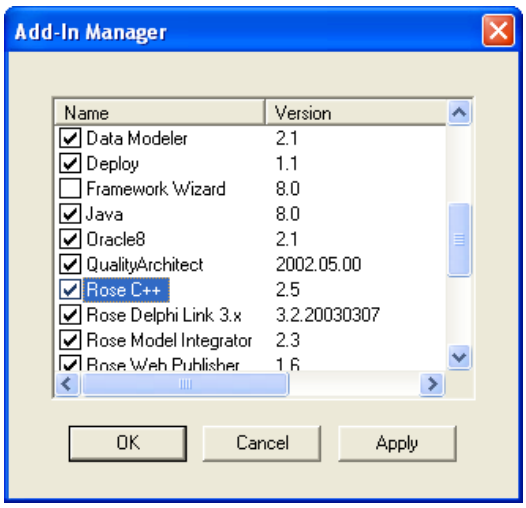


Рисунок 5.4 - Менеджер надстроек Add-Ins

Генерация программного кода в среде IBM Rational Rose 2003 возможна для отдельного класса или компонента. Для этого нужный элемент модели предварительно следует выделить в браузере проекта и выполнить операцию контекстного меню: Tools→C++→Code Generation - (Язык C++→Генерировать код). В результате этого будет открыто диалоговое окно с предложением выбора классов для генерации программного кода на выбранном языке программирования (рис. 5.5). После выбора соответствующих классов и нажатия кнопки OK программа IBM Rational Rose 2003 выполняет кодогенерацию.

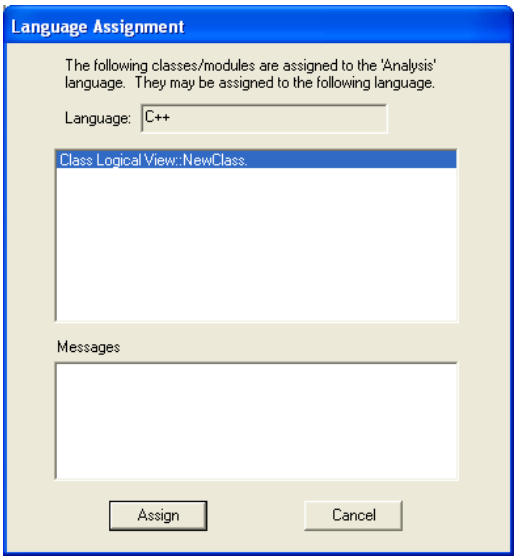


Рисунок 5.5 - Окно выбора классов для генерации программного кода

Затем происходит компиляция и выдается окно статуса (Code Generation Status). Здесь можно увидеть информацию о том, какой класс был закодирован и количество ошибок и предупреждений (рис. 5.6). Если у вас произошла, какая-либо ошибка или же предупреждение, то их можно увидеть на рабочем поле в Rational Rose, для этого и существует самое нижнее окно, в нем передаются все ваши действия и ошибки, произошедшие в ходе кодогенерации.

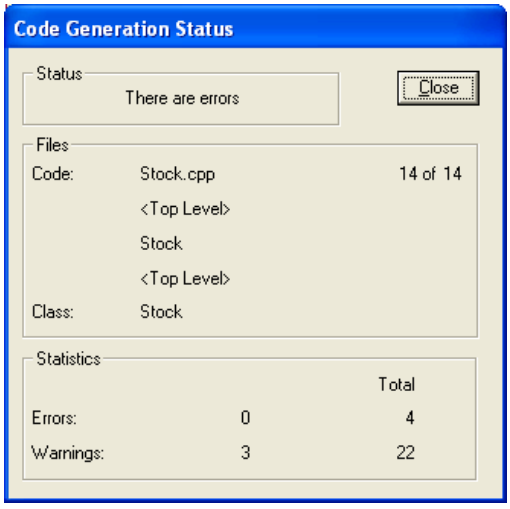


Рисунок 5.6 – Окно статуса компиляции

2.1.7 Результаты генерации

В результате кодогенерации Rational Rose создает два файла с расширением ".h" и ".cpp", названия у них те же, что и название класса. Итак, выполнив эти действия, нажимаем правой клавишей на класс, появляется окошко, в нем ищем "C++", и видим два пункта Browse Header и Browse Body, и в зависимости от того какой из файлов нам нужен ".h" (заголовочный) или ".cpp" (непосредственно реализация), выбираем их. Эти файлы открываются с помощью блокнота и теперь легко можно увидеть скелет класса, с различными комментариями, которые писали вы на диаграммах, и комментарии которые вставляет сама Rose. Теперь можно открыть один из файлов в C++ и доработать класс, описать работу функций, добавить различные нововведения.

Следует заметить, что при установленной на компьютер разработчика интегрированной среды сгенерированные файлы с текстом программного кода автоматически открываются в этой среде после двойного щелчка на пиктограмме этих файлов. Тем не менее, лучше копировать содержимое этих файлов в предварительно созданные программные проекты для полного контроля в этих средах процесса программирования и отладки приложений.

Сгенерированные программой IBM Rational Rose 2003 файлы с текстом программного кода содержат минимум информации. Для включения дополнительных элементов в программный код следует изменить свойства генерации программного кода, установленные по умолчанию.

В заключение следует отметить, что эффект от использования средства IBM Rational Rose 2003 проявляется при разработке *масштабных проектов* в составе команды или проектной группы. Однако ситуация покажется не столь тривиальной, когда станет необходимо выполнить проект с несколькими десятками вариантов использования и сотней *классов*. Именно для подобных проектов явно выявляется преимущество использования средства IBM Rational Rose 2003 и нотации языка UML для документирования и реализации соответствующих моделей.

Задание на лабораторное занятие

1. Изучить теоретический материал
2. Сгенерировать программный код на C++ для диаграммы классов, разработанной вами в предыдущей лабораторной работе.

Содержание отчета

- титульный лист;
- постановка задачи;
- листинг сгенерированного кода;
- вывод.

Контрольные вопросы

1. Какие диаграммы необходимо предварительно разработать, чтобы выполнить кодогенерацию?
2. Как посмотреть исходный код?
3. Какие установки свойств доступны на вкладке C++?
4. Какова структура создаваемого кода?
5. Что необходимо добавить в шаблоны классов для получения работоспособного приложения?
6. Какие шаги нужно предпринять для обновления модели по исходному коду?
7. Какие основные этапы кодогенерации вы знаете? Расскажите кратко о каждом из них?

Работа с CASE – средствами тестирования программного обеспечения

Цель работы: отладка разработанного программного средства; тестирование разработанного программного средства.

Теоретическая часть

Отладка ПС – это деятельность, направленная на обнаружение и исправление ошибок в ПС с использованием процессов выполнения его программ. *Тестирование* ПС – это процесс выполнения его программ на некотором наборе данных, для которого заранее известен результат применения или известны правила поведения этих программ. Указанный набор данных называется тестовым или просто тестом. Таким образом, отладку можно представить в виде многократного повторения трех процессов: тестирования, в результате которого может быть констатировано наличие в ПС ошибки, поиска места ошибки в программах и документации ПС и редактирования программ и документации с целью устранения обнаруженной ошибки.

Отладка программного средства невозможна без представления физической структуры. На этом этапе проектирования необходимо разработать *диаграмму компонентов*.

Диаграммой компонентов (Component diagram) называется диаграмма UML, на которой показаны компоненты системы и зависимости между ними.

Компонентом называется физический модуль кода. Компонентами бывают как библиотеки исходного кода, так и исполняемые файлы. Например, .h и .cpp и .exe - будут отдельными компонентами.

Особенности разработки диаграмм компонентов в среде IBM Rational Rose 2003

Диаграмма *компонентов* служит частью физического представления модели, играет важную роль в процессе ООАП и является необходимой для генерации программного кода. Для разработки диаграмм *компонентов* в браузере проекта предназначено отдельное представление *компонентов* (Component View), в котором уже содержится диаграмма *компонентов* с пустым содержанием и именем по умолчанию Main (Главная).

Активизация диаграммы *компонентов* может быть выполнена одним из следующих способов:

Щелкнуть на кнопке с изображением диаграммы *компонентов* на стандартной панели инструментов.

Раскрыть представление *компонентов* в браузере (Component View) и дважды щелкнуть на пиктограмме Main (Главная).

Через пункт меню Browse→Component Diagram (Браузер→Диаграмма *компонентов*).

В результате выполнения этих действий появляется новое окно с чистым рабочим листом диаграммы *компонентов* и специальная панель инструментов, содержащая кнопки с изображением графических примитивов, необходимых для разработки диаграммы *компонентов* (табл. 6.1).

Таблица 6.1 - Назначение кнопок специальной панели инструментов диаграммы компонентов

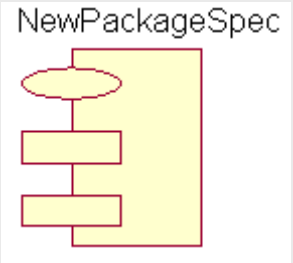
Графическое изображение	Всплывающая подсказка	Назначение кнопки
	Selection Tool	Превращает изображение курсора в форму стрелки для последующего выделения элементов на диаграмме
	Text Box	Добавляет на диаграмму текстовую область
	Note	Добавляет на диаграмму примечание
	Anchor Note to Item	Добавляет на диаграмму связь примечания с соответствующим графическим элементом диаграммы
	Component	Добавляет на диаграмму *компонент*
	Package	Добавляет на диаграмму пакет
	Dependency	Добавляет на диаграмму отношение зависимости
	Subprogram Specification	Добавляет на диаграмму спецификацию подпрограммы
	Subprogram Body	Добавляет на диаграмму тело подпрограммы
	Main Program	Добавляет на диаграмму главную программу
	Package Specification	Добавляет на диаграмму спецификацию пакета
	Package Body	Добавляет на диаграмму тело пакета
	Task Specification	Добавляет на диаграмму спецификацию задачи
	Task Body	Добавляет на диаграмму тело задачи
	Generic Subprogram	Добавляет на диаграмму типовую подпрограммы(по умолчанию отсутствует)
	Generic Package	Добавляет на диаграмму типовой пакет (по умолчанию отсутствует)
	Database	Добавляет на диаграмму базу данных (по умолчанию отсутствует)

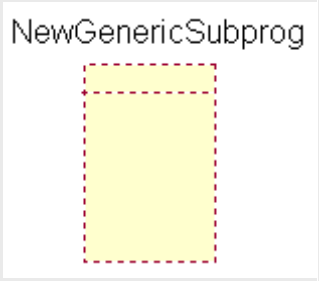
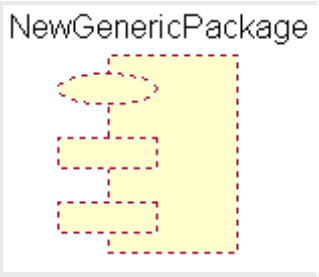
Программа IBM Rational Rose 2003 не поддерживает следующие графические *стереотипы*. Графическое изображение этих *стереотипов* и их краткая характеристика приводятся в следующей таблице (табл. 6.2). При этом каждому из

компонентов, как правило, соответствует отдельный файл исходной сборки программного приложения.

Использование *стереотипов* существенно увеличивают наглядность графического представления диаграммы *компонентов* и позволяют архитектору уточнить характер реализации модели программистом на выбранном языке программирования.

Таблица 6.2. Графическое изображение стереотипов компонентов и их характеристика

Графическое изображение и имя по умолчанию	**Название стереотипа**	**Характеристика стереотипа *компонента***
	Subprogram Specification	Спецификация подпрограммы. Содержит описание переменных, процедур и функций и не содержит определений классов
	Subprogram Body	Тело подпрограммы. Содержит реализацию процедур и функций, не относящихся к каким-то классам, при этом не содержит определений классов или реализаций операций других классов
	Main Program	Главная программа. Реализует базовую логику работы программного приложения и содержит ссылки на другие *компоненты* модели
	Package Specification	Спецификация пакета. Содержит определение класса, его атрибутов и операций. В языке программирования C++ спецификации пакета соответствует отдельный файл с расширением «h»
	Package Body	Тело пакета. Содержит код реализации операций класса. В языке программирования C++ спецификации пакета соответствует отдельный файл с расширением «cpp»

	Task Specification	Спецификация задачи. Может содержать определение класса, его атрибутов и операций, которые предполагается использовать в независимом потоке управления
	Task Body	Тело задачи. Может содержать реализацию операций класса, которые имеют независимый поток управления.
	Generic Subprogram	Типовая подпрограмма. Содержит описание переменных, процедур и функций, которые могут быть использованы в нескольких программных приложениях. При этом типовая подпрограмма не содержит определений классов
	Generic Package	Типовой пакет. Содержит определение класса, его атрибутов и операций, которое может быть использовано в нескольких программных приложениях
	Database	База данных. Содержит определение одного или нескольких классов, их атрибутов и, возможно, операций. При этом соответствующие классы могут быть реализованы в форме одной или нескольких таблиц базы данных

2.1.1 Добавление компонента на диаграмму компонентов и редактирование его свойств

Для добавления *компонента* на диаграмму *компонентов* нужно с помощью левой кнопки мыши нажать кнопку с изображением пиктограммы *компонента* на специальной панели инструментов, отпустить левую кнопку мыши и щелкнуть левой кнопкой мыши на свободном месте рабочего листа диаграммы. Добавить *компонент* на диаграмму можно также с помощью операции главного меню: Tools→ Create→Component или с помощью операции контекстного меню: New→Component, предварительно выделив представление *компонентов* в браузере проекта.

В результате этих действий на диаграмме появится изображение *компонента* с маркерами изменения его геометрических размеров и предложенным средой именем по умолчанию, которое разработчику следует изменить (рис. 6.1).

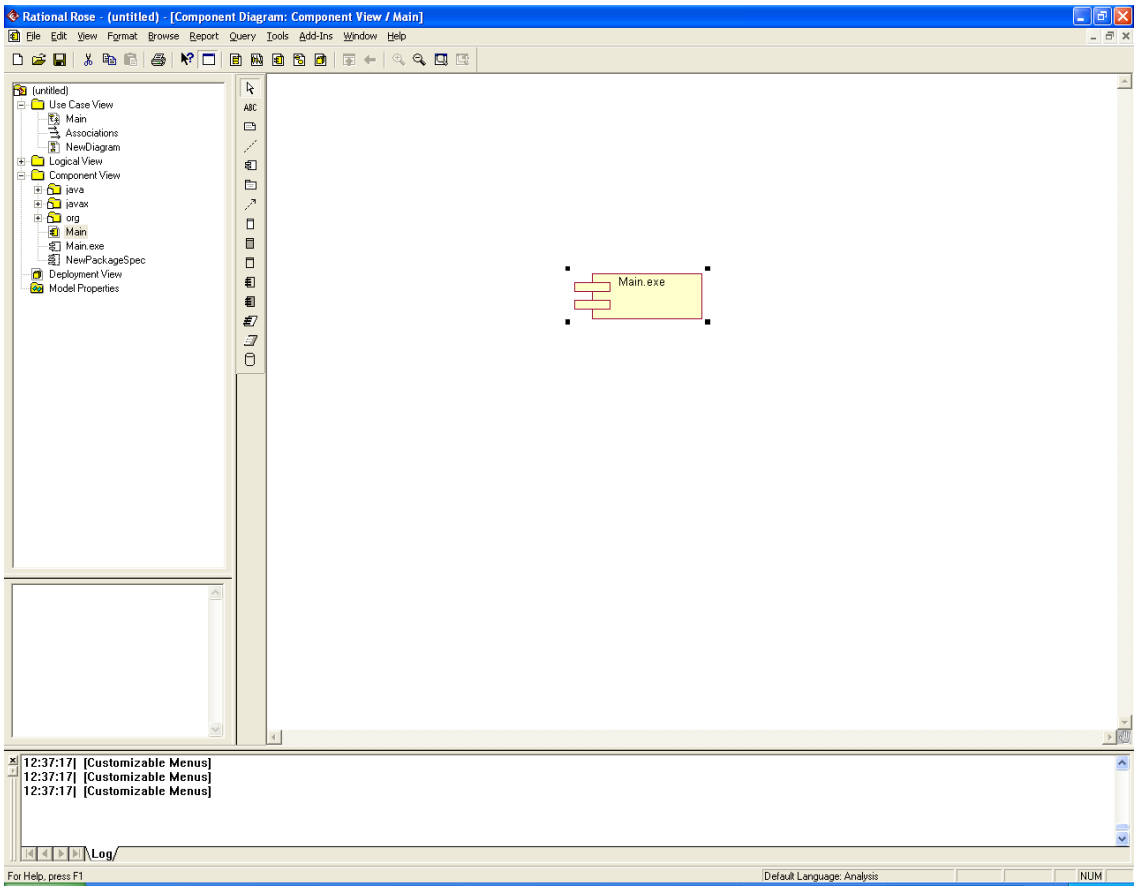


Рисунок 6.1 - Диаграмма компонентов после добавления компонента Main.exe

Для каждого *компонента* можно определить различные свойства, такие как стереотип, язык программирования, декларации, реализуемые классы. Редактирование этих свойств для произвольного *компонента* осуществляется с помощью диалогового окна спецификации свойств (рис. 6.2).

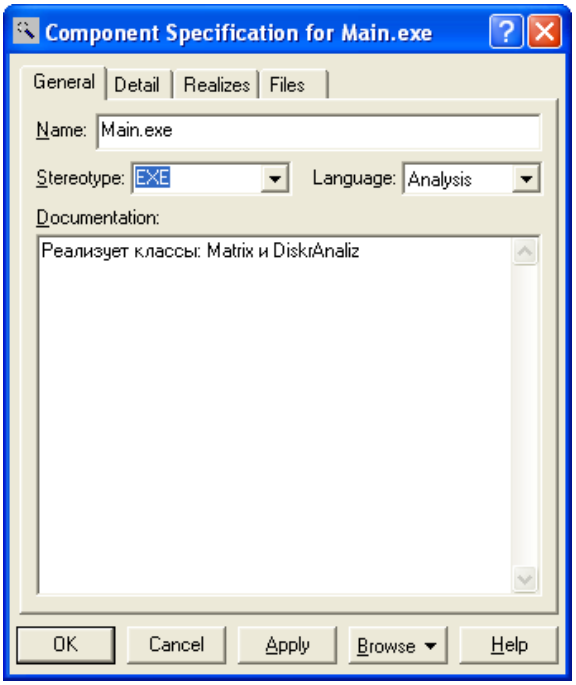


Рисунок 6.2 - Диалоговое окно спецификации свойств компонента Main.exe

В частности, для *компонента* Main.exe можно выбрать стереотип <<EXE>> из предлагаемого вложенного списка, поскольку применительно к разрабатываемой модели предполагается реализация этого *компонента* в форме исполнимого файла. При этом на вкладке Realizes (Реализует) содержатся все классы, включая и актеров, которые на данный момент присутствуют в модели.

По умолчанию в среде IBM Rational Rose 2003 для всех добавляемых на диаграмму компонентов в качестве языка реализации используется язык анализа, который в последствии следует изменить на тот язык программирования, который предполагается использовать для написания программного кода. В дальнейшем при генерации программного кода необходимо будет дополнительно выбрать те классы, которые реализует тот или иной компонент модели.

2.1.2 Добавление отношения зависимости и редактирование его свойств

Добавление отношения зависимости на диаграмму компонентов аналогично добавлению соответствующего отношения на диаграмму вариантов использования. Для добавления зависимости между двумя компонентами нужно с помощью левой кнопки мыши нажать кнопку с изображением зависимости на специальной панели инструментов, отпустить левую кнопку мыши, щелкнуть левой кнопкой мыши на изображении исходного компонента на диаграмме и отпустить ее на изображении целевого компонента. В результате этих действий на диаграмме появится изображение отношения зависимости в форме пунктирной линии со стрелкой, соединяющей два выбранных компонента.

2.1.3 Пример построения диаграммы компонентов

Ниже представлена диаграмма компонентов для программного средства, реализующего алгоритм дискриминантного анализа (рис. 6.3).

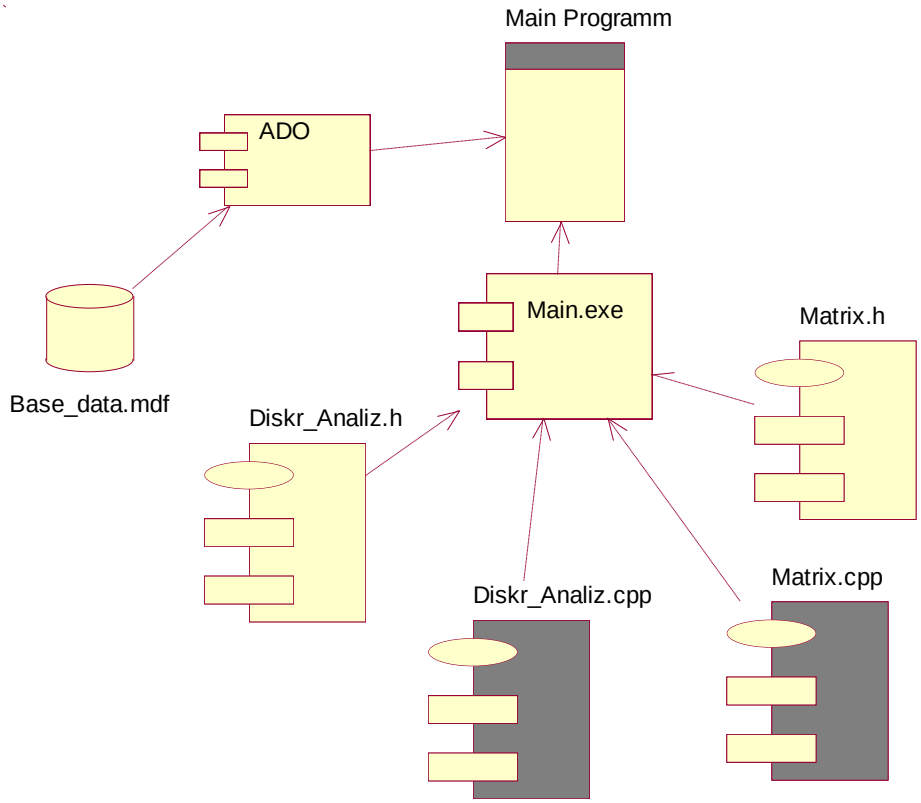


Рисунок 6.3 – Диаграмма компонентов рассматриваемого ПС

2.2 Тестирование программного средства

Тестирование – процесс многократного повторения программы с целью обнаружения ошибок. Существуют следующие методы тестирования ПС:

- статическое тестирование (ручная проверка программы за столом);
- детерминированное тестирование (при различных комбинациях исходных данных);
- стохастическое (исходные данные выбираются произвольно, на выходе определяется качественное совпадение результатов или примерная оценка).

При тестировании разработанного ПС необходимо использовать подходящий по функциональности пакет прикладного математического программного обеспечения.

В выбранной среде необходимо произвести расчеты всех параметров, реализованных в ПС, затем следует сравнить результаты и сделать вывод о качестве данного программного продукта.

Задание на лабораторное занятие

1. Изучить теоретический материал
2. Выполнить тестирование и отладку информационной системы, разработанной в лабораторной работе 14-15, то есть разработать диаграмму компонентов рассматриваемого ПС.

Содержание отчета

- титульный лист;
- постановка задачи;

- тестовый набор данных и результаты тестирования;
- вывод.

Варианты заданий

2_1. Автоматизированная информационная система «Индивидуальный план преподавателя»

Описание предметной области.

Для каждого преподавателя (ФИО, Год рождения, Домашний адрес, Контактные телефоны) высшего учебного заведения (Код, Название, Краткое название) на каждый учебный год (Год начала учебного года, Год окончания учебного года) формируется индивидуальный план. В индивидуальном плане отражается общий объем работ преподавателя, который он должен выполнить в течение учебного года. Учет работ ведется по следующей форме:

№	Наименование работы	План		Факт	
		Осенний семестр	Весенний семестр	Осенний семестр	Весенний семестр

В течение учебного года преподаватель выполняет следующие виды работ (Код, Название Краткое название): «Учебная работа», «Учебно-методическая работа», «Научно-методическая работа», «Научно-исследовательская работа», «Организационно-методическая работа», «Внеучебная работа со студентами», «Прочие виды работ». Необходимо вести учет в часах (целых и долях часов) объем запланированных и фактически выполненных объемов работ для каждого преподавателя по семестрам. Для каждого преподавателя также необходимо фиксировать место работы – факультет (Код, Название, Краткое название), кафедра (Код, Название, Краткое название), занимаемую должность (Код, Название, Краткое название), время работы в этой должности (Дата начала, Дата окончания, Ставка, Дата избрания на должность), кем является преподаватель – штатным сотрудником или совместителем. Также для преподавателя фиксируются:

- ученая степень (Код, Название, Краткое название) – доктор, кандидат; каких наук (Код, Название, Краткое название) – технических, экономических и т.п.; год присуждения;
- ученое звание (Код, Название, Краткое название) – профессор, доцент, с.н.с. и т.п.; год присуждения звания.

Необходимо осуществлять следующую обработку данных:

- формирование для каждого преподавателя итоговой суммы (в часах) запланированных и выполненных объемов работ по семестрам;
- список преподавателей, у которых фактическое значение выполненных работ превышает плановое (факультет, кафедра, ФИО, уч.степень, уч.звание, должность, семестр, кол-во перевыполненных объемов работ);
- список преподавателей заданной кафедры, имеющих заданную ученую степень на заданную дату.

2_2. Автоматизированная информационная система «Обслуживание заказов клиентов»

Описание предметной области.

Предприятие (Код, Название, Краткое название) осуществляет доставку разных

товаров (Код, Название, Краткое название) населению. Прием заказов от населения осуществляет специальная служба (Код, Название, Краткое название) предприятия.

Для того чтобы стать потребителем услуг предприятия каждый абонент должен зарегистрироваться, при этом фиксируются его ФИО, адрес, телефон и паспортные данные (Серия, Номер, Дата выдачи, Кем выдан). Каждый абонент в течение дня может сделать несколько заказов (Дата, Время), заказу присваивается номер.

В каждом заказе может содержаться несколько товаров, для каждого указывается количество товара, единица измерения (Код, Название, Краткое Название), цена за единицу товара, общая стоимость товара. Заказ также имеет итоговую сумму. При формировании бланка заказа, который будет подписан абонентом при получении товара фиксируется, оплачен заказ, или абонент получает товар в кредит. Также на бланке заказа указывается: реквизиты предприятия (название, адрес, контактные телефоны); ФИО и должность оператора, приявшего заказ; ФИО, должность сотрудника, доставившего заказ.

Необходимо осуществлять следующую обработку данных:

- список товаров (код, наименование), пользующихся наибольшим спросом (максимальное количество позиций заказов) у населения за заданный период;
- динамика изменения стоимости заданного товара за заданный период по месяцам;
- список наименований улиц, на которых проживают абоненты предприятия по убыванию числа абонентов.

2_3. Автоматизированная информационная система «Прохождение преддипломной практики студентами вуза»

Описание предметной области.

Студенты высшего учебного заведения (Код, Название, Краткое название) в период подготовки дипломной работы (проекта) проходят преддипломную практику. Для каждого студента (Номер зачетной книжки, ФИО), обучающегося на определенной специальности (Код, Название, Краткое название), факультете (Код, Название, Краткое название), форме обучения (Код, Название, Краткое название) фиксируется место прохождения преддипломной практики – предприятие (Код, Название, Краткое название), адрес предприятия, ФИО, должность руководителя от вуза, ФИО, должность руководителя от предприятия, срок прохождения практики (Дата начала, Дата окончания). В базе данных также необходимо вести данные о сроках защиты практики для каждой группы, оценке, полученной студентом за практику. При вводе данных о месте прохождения практики для каждого студента необходимо помечать – планирует ли студент в дальнейшем работать на данном предприятии, варианты ответов - да, нет, не знаю.

Необходимо осуществлять следующую обработку данных:

- количество студентов, проходивших практику на заданном предприятии в заданный период;
- перечень предприятий (название, адрес) по алфавиту, на которых проходили преддипломную практику студенты заданной специальности за заданный период;
- на заданную дату список студентов заданной специальности и потока (год обучения), не имеющих оценку за практику.

2_4 Автоматизированная информационная система «Лицензионное

программное обеспечение организации»

Описание предметной области.

Необходимо вести учет и анализ информации о лицензионном программном обеспечении (ПО), установленном на компьютерах организации (Код, Название, Краткое название). Для каждого компьютера фиксируется инвентарный номер, тип (рабочая станция или сервер), местоположение – в каком подразделении (Код, Название, Краткое название) организации компьютер установлен. Компьютеры могут передаваться из подразделения в подразделение, при этом необходимо знать сроки (Дата начала, Дата окончания) нахождения компьютера в подразделении и на основании какого документа он перемещается (Номер документа, Дата документа), тип этого документа (приказ, распоряжение и т.п.). При установке лицензионного ПО фиксируется, куда установлено ПО – на какой компьютер, название продукта, его тип (среда разработки прикладных программ, среда администрирования БД, операционная система, антивирусная программа и т.п.), фирма производитель, срок действия лицензии (Дата начала, Дата окончания), дата установки, цена за единицу ПО. При этом также необходимо фиксировать информацию об организации, продавшей программное обеспечение – название, адрес, контактные телефоны, адрес сайта.

Необходимо осуществлять следующую обработку данных:

- на заданную дату список подразделений, на компьютерах которых установлено не лицензионное ПО;
- список лицензионного ПО, количество лицензий на это ПО (по убыванию) на заданную дату;
- список подразделений, количество компьютеров у подразделения (по убыванию) на заданную дату.

2_5 Автоматизированная информационная система «Арендная плата за нежилые помещения»

Описание предметной области.

Организация (Код, Название, Краткое название, Адрес, Контактные телефоны, электронный адрес) сдает в аренду помещения. Каждое помещение характеризуется следующими показателями:

- адрес;
- площадь – кв.м.;
- площадь подвала – кв.м. (при наличии);
- коэффициент подвала – значение от 0 до 1;
- коэффициент технического обустройства помещения (КТ) – значение от 1 до 2.

Арендная плата зависит от базовой ставки за 1 кв.м. (в рублях), которая утверждается документом (Номер, Дата) агентства Госкомимущества России.

Формула расчета месячной арендной платы (МАП):

МАП = (базовая ставка/12 * площадь помещения + базовая ставка/12 * площадь подвала * коэффициент подвала) * КТ.

При изменении базовой ставки МАП изменяется со следующего месяца после даты изменения ставки. Оплата производится ежемесячно.

Договор об аренде может заключаться как с организациями (Юридическими лицами), так и с физическими лицами. В договоре об аренде помещения, имеющего

номер, дату фиксируется дата начала аренды, дата заключения аренды. Для юридического лица в БД заносятся название, адрес, ИНН, номер и дата лицензии о деятельности. Для физического лица – ФИО, паспортные данные (Серия, Номер, Дата выдачи, Кем выдан), ИНН и адрес.

Необходимо осуществлять следующую обработку данных:

- итоговая сумма оплат за текущий месяц (на заданную дату);
- список арендаторов (тип, название, адрес и другие характеристики арендуемого помещения) на текущую дату;
- список помещений, не сданных в аренду на текущую дату.

2_6 Автоматизированная информационная система «Списание основных средств»

Описание предметной области.

Основные средства - это имущество организации, предприятия со сроком полезного использования. На предприятии (Код, Название, Краткое название) имеется перечень основных средств разного типа (мебель, вычислительная техника, оборудование, инструменты и т.п.), закрепленных за подразделениями предприятия. Закрепление осуществляется на основании определенного документа, имеющего номер, дату, в нем указан срок закрепления средства за подразделением. При списании имущества предприятия создается комиссия, в которую входят руководитель предприятия, главный бухгалтер, главный инженер, главный энергетик, главный механик, руководитель подразделения, где находится средство, материально ответственный в подразделении. При списании средства формируется документ, имеющий номер, название, дату и подписи членов комиссии. В каждом документе может быть указано сразу несколько списываемых средств, для каждого указывается:

- инвентарный номер;
- название;
- принадлежностью к типу;
- дата постановки на учет в подразделении;
- плановый срок эксплуатации (год, месяц);
- балансовая стоимость (в рублях), определяемая при постановке средства на учет.

Для каждого средства также указывается дефект, ставший причиной списания (Код, Название) – износ, поломка, не имеющая восстановления, утрата и др.

Необходимо осуществлять следующую обработку данных:

- на заданную дату список (наименование) средств, закрепленных за каждым подразделением, балансовая стоимость средства;
- динамика списания средств заданного наименования (количество) за заданный период по месяцам;
- на заданную дату список комиссии по списанию.

2_7. Автоматизированная информационная система «Аттестация сотрудников предприятия»

Описание предметной области.

Предприятие (Код, Название, Краткое название) периодически проводит аттестацию сотрудников на соответствие ими занимаемой должности. Каждый

сотрудник за время работы может проходить несколько аттестаций.

Для проведения аттестации (Дата) необходима следующая информация: ФИО сотрудника, дата рождения, место работы (Код, Название, Краткое название) подразделения, занимаемая должность (Код, Название, Краткое название), ставка, дата начала работы, дата окончания работы контракта), название, номер и дата приказа о назначении на должность. Необходимы также следующие сведения:

- сведения об образовании – какое заведение окончил, документ об образовании, квалификация по образованию (инженер, учитель, экономист);
- дата начала трудового стажа;
- дата начала стажа по специальности;
- сведения о повышении квалификации – в каком заведении проходил, дата начала, дата окончания прохождения.

У каждого сотрудника может быть несколько документов об образовании и повышении квалификации.

Каждому аттестуемому могут задать несколько вопросов, необходимо хранить количество заданных вопросов и количество правильных ответов. Также необходимо хранить оценку деятельности работника – соответствует или не соответствует занимаемой должности.

Каждую аттестацию проводит комиссия, необходимо фиксировать ФИО, место работы и должность члена комиссии. Максимальное число – 5 человек.

Необходимо осуществлять следующую обработку данных:

- на заданную дату список сотрудников (ФИО, место работы), не прошедших аттестацию – не соответствующих занимаемой должности;
- на заданную дату количество сотрудников, работающих на предприятии в заданной должности;
- список учебных заведений, предприятий, их адреса, на которых сотрудники предприятия повышали свою квалификацию.

2_8. Автоматизированная информационная система «Трудоустройство»

Описание предметной области.

Организация (Код, Название, Краткое название Адрес, Контактные телефоны, электронный адрес) предоставляет услуги по трудоустройству. Организацией ведется банк данных о существующих вакансиях. По каждой вакансии поддерживается следующая информация:

- предприятие (Код, Название, Краткое название Адрес, Контактные телефоны, электронный адрес);
- название вакансии (должность);
- требования к соискателю: пол, возраст (Верхняя граница, Нижняя граница), образование (высшее, среднее, не имеет значение и т.п.), знание определенных видов деятельности (выбор из перечня - знание электронного документооборота, определенных прикладных программ и т.п.), коммуникабельность (да, нет);
- обязанности (выбор из перечня – заключение договоров, распространение агитационного материала, работа с клиентами и т.п.);
- предполагаемая оплата (Нижняя граница, Верхняя граница), единицы измерения оплаты - рубли;
- оформление трудовой книжки (да, нет);
- наличие социального пакета (да, нет);

- срок начала открытия вакансии;
- срок закрытия вакансии (вакансия занята).

Необходимо осуществлять следующую обработку данных:

- на заданную дату список предприятий, имеющих вакансии по заданной должности;
- название должности, на которую за заданный период было предложено максимальное количество вакансий;
- на заданную дату список предприятий, предлагающих вакансии, не требующих образования.

2_9. Автоматизированная информационная система «Спортивные сооружения области»

Описание предметной области.

Областная организация (Код, Название, Краткое название Адрес, Контактные телефоны, электронный адрес) ведет и предоставляет на сайте информацию о спортивных сооружениях области. По каждому сооружению ведется информация:

- место – населенный пункт, городского или сельского типа, адрес;
- номер, название, краткое название;
- тип сооружения (игровые виды спорта, легкоатлетический манеж, каток, ипподром и др.);
- площадь спортивной арены, кв.м.;
- вместимость зрителей, чел., тыс. чел.;
- организация (Код, Название, Краткое название Адрес, Контактные телефоны, электронный адрес), принявшая сооружение на баланс;
- дата принятия на баланс.

Каждое сооружение за время функционирования может находиться на балансе у разных организаций в разные периоды времени.

Необходимо также фиксировать мероприятия, проводимые в спортивных сооружениях:

- тип мероприятия – тренировочный процесс, соревнования, сдача в аренду, концерт и т.п.;
- название мероприятия;
- дата начала, дата окончания мероприятия;
- количество человек, посетивших мероприятие.

Необходимо осуществлять следующую обработку данных:

- на заданную дату список спортивных сооружений заданного типа;
- за заданный период динамика занятости спортивного сооружения в мероприятиях заданного типа по месяцам;
- на заданную дату список организаций, на балансе у которых находятся спортивные сооружения, их количество.

2_10 Автоматизированная информационная система «Справочник предприятия»

Описание предметной области.

Для формирования контактов организации, имеющей большой контингент клиентов, и представления их на сайте, необходимо хранить следующую информацию:

- код, название краткое название предприятия, каждого его подразделения, взаимодействующего с клиентами;

- вид деятельности предприятия, подразделения – работа с абонентами, изготовление продукции; изучение рынка спроса; IT-подразделение и др.;

- местоположение предприятия, подразделения – адрес, вплоть до номера комнаты. Местоположение может меняться, необходимо отслеживать все данные, для этого фиксируется дата начала закрепления адреса за предприятием, подразделением;

- контактные телефоны – их может быть несколько, и они могут меняться, необходимо хранить историю закрепления телефонов;

- электронный адрес предприятия. Подразделения;

- ФИО, должность руководителя. Руководители также могут меняться, необходимо отслеживать историю их изменения и поддерживать исторические данные.

Необходимо осуществлять следующую обработку данных:

- на заданную дату список контактных телефонов подразделений предприятия;

- на заданную дату количество подразделений, не имеющих электронные адреса;

- название подразделения, у которого за заданный период сменилось наибольшее число руководителей.

2_11 Автоматизированная информационная система «Паспорт здоровья сотрудника»

Описание предметной области.

Организация придает большое значение здоровью сотрудников и имеет в своей структуре подразделение, занимающееся профилактикой здоровья сотрудников. Для учета состояния здоровья для каждого сотрудника ведется «Паспорт здоровья», в котором сохраняется следующая информация:

- ФИО сотрудника, пол, дата рождения;

- образование (высшее, среднее, без образования). Если человек за время работы на предприятии повышал своё образование – необходимо фиксировать все соответствующие данные, привязывая их к дате получения соответствующего документа;

- история всех перемещений сотрудника на предприятии – подразделение, должность, категория должности (инженерно-технический работник, рабочий, управленческий персонал, IT-специалист и др.), должность, ставка, дата начала работы, дата окончания;

- история семейного положения – состояние (холост, в браке, разведен и др.), дата начала семейной жизни, дата окончания;

- история антропологических измерений – на дату – рост, вес;

- история прививок – дата, название прививки;

- история заболеваний – название, дата постановки на учет, дата снятия с учета.

Необходимо осуществлять следующую обработку данных:

- на заданную дату название заболевания, зафиксированного у сотрудников за все время наблюдения максимальное число раз;

- на заданный период список сотрудников, не сделавших прививку заданного вида;

- за заданный период динамика количества заболеваний в организации – по месяцам, количество заболевших с высшим, средним образованием и без образования.

2_12 Автоматизированная информационная система «Справочник абитуриента»

Описание предметной области.

Высшее учебное заведения для предоставления на сайте информации абитуриентам ведет банк данных со следующей информацией:

- список специальностей (Код, Название, Краткое название), на которых осуществляется обучение в вузе. Специальности привязаны к учебным подразделениям – факультетам, кафедрам (Код, Название, Краткое название), и распределены по формам обучения (очная, очно-заочная, заочная);
- адрес учебных подразделений;
- телефоны учебных подразделений;
- если есть – адрес сайта учебного подразделения;
- ФИО, ученая степень, ученое звание руководителя учебного заведения (декан факультета, заведующий кафедрой). При этом необходимо вести историю всех руководителей – дата начала работы, дата окончания;
- по каждой форме обучения:
  - план приема на специальность на каждый год;
  - перечень предметов, по которым необходимо сдавать вступительные экзамены (ЕГЭ);
  - проходной балл на специальность по годам с разбивкой по предметам.

Необходимо осуществлять следующую обработку данных:

- на заданный год – список специальностей заданной формы обучения и планы приема;
- на заданный год наименование специальности, на которую был максимальный проходной балл по математике;
- на заданный год список руководителей учебных подразделений, имеющих ученую степень «доктор наук» и ученое звание «профессор».

2_13 Автоматизированная информационная система «Платные образовательные услуги населению»

Описание предметной области.

Организация (Код, Название, Краткое название) оказывает платные образовательные услуги населению. Услуги оказываются в виде проведения курсов обучения, по которым необходимо хранить следующую информацию:

- тип проведения – групповые, индивидуальные;
- вид проведения – очные, заочные;
- дата начала, дата окончания курсов;
- срок обучения (дни, месяцы, годы);
- количество часов обучения;
- на базе какого образования (среднее, высшее);
- темы, входящие в курс, для каждой темы:
  название;
  количество часов;

- время проведения занятий – дни недели, часы;

- вид выпускного контроля (квалификационная работа, экзамен, собеседование и прочее);

- вид выдаваемого документа (документ государственного образца, документ установленного образца);

- стоимость обучения

Для предоставления информации на сайте необходимо хранить адрес организации, контактные телефоны, электронный адрес, адрес сайта, серия, номер и вид документа о предоставлении образовательных услуг.

Необходимо осуществлять следующую обработку данных:

- список курсов, на которых можно прослушать заданную темы, например, «1С Бухгалтерия»;

- список курсов, на которых можно пройти заочное обучение и имеющих минимальную стоимость;

- список самых длительных курсов.

2_14 Автоматизированная информационная система «Новостная лента организации»

Описание предметной области.

Для предоставления новостных событий организации на её сайте необходимо вести следующие данные:

- название, краткое название организации, контактные телефоны, адрес, электронный адрес, адрес сайта;

- название и координаты подразделений организации, информация о которых будет предоставляться на сайте;

- список работающих сотрудников подразделений организации. Которым предоставляется возможность размещать информацию на сайте – ФИО, подразделение, должность, логин, пароль. При изменении статуса сотрудника – увольнение, перевод – информация должна соответствующим образом изменяться, например, сотрудник переводится в статус неработающего, логин и пароль д.б. заблокированы;

- описание новостной информации, размещаемой на сайте:

тип (новость, объявление, сообщение и др.);

название информации;

дата создания;

текст;

дата размещения;

дата перевода информации в архив;

размер информации в Кб;

наличие прикрепляемых к информации файлов – для каждого название, размер, тип, краткое описание;

ответственный за информацию – сотрудник подразделения, имеющий соответствующий доступ.

Необходимо осуществлять следующую обработку данных:

- на заданную дату список ответственных за информацию на сайте от подразделений, не имеющих логин и пароль;

- на заданную дату название информации, размещенной на сайте (не в архиве) и

имеющей самый большой размер.

- динамика предоставления информации для сайта заданным подразделением за заданный период – количество по месяцам.

2_15 Автоматизированная информационная система «Анализ продаж»

Описание предметной области.

Магазин (Код, Название, Краткое название) ведет учет продаж товаров и анализ работы с постоянными клиентами. Каждая единица товара учитывается при поступлении в магазин из накладной (Номер, Дата накладной), которая может иметь несколько позиций. В каждой позиции есть её номер, наименование товара, количество единиц поступившего товара, единица измерения, цена за единицу. Товары учитываются по виду - одежда, кожгалантерея, чулочно-носочные изделия, обувь и т.п. Каждый товар также имеет определенный артикул.

Ведет учет и продаж товаров – фиксируется дата продажи конкретного товара, количество проданных единиц.

Магазин ведет учет постоянных клиентов – фиксируется ФИО клиента, его паспортные данные (Серия, Номер, Дата выдачи, Кем выдан), дата рождения, контактный телефон. Покупателю, сделавшему покупку на сумму свыше 3000 тыс. рублей выдается дисконтная карта, имеющая 5-ти значный номер. Карта дает покупателю скидку 3%. При накоплении сумм покупок покупателем более чем на 10000 тыс. рублей, процент скидки увеличивается до 5%, более 20000 – максимальный процент скидки достигает размера 10%.

Необходимо осуществлять следующую обработку данных:

- на заданную дату количество и список покупателей (ФИО, контактный телефон), имеющих 10% скидку;

- за заданный период - динамика продажи заданного товара – количество по месяцам – поступление/ продажа;

- на заданную дату список покупателей (ФИО, контактный телефон), у которых в ближайшие 10 дней будет день рождения.

2_16 Автоматизированная информационная система «Электронный реестр помещений»

Описание предметной области.

Предприятие (Код, Название, Краткое название) имеет иерархическую организационную структуру, отражающая подчиненность большого количества подразделений. Для каждого подразделения необходимо хранить:

- код, полное название, краткое название;

- родительные и дательные падежи названий для автоматизированного формирования ряда документов и отчетов.

Каждое подразделение может занимать несколько помещений. Каждое помещение имеет номер, в который входит номер корпуса (предприятие может иметь много зданий – 1 или 2 цифры) и номер этажа – 1 или 2 цифры. В пределах одного этажа каждое помещение имеет свой номер 1 или 3 цифры. Помещение относится к определенному типу, о котором также необходимо иметь сведения, например, кабинет руководителя, приемная руководителя, лаборатория, цех, столовая и т.п. Необходимо также хранить данные о площади каждого помещения (кв. м).

Закрепление помещений за подразделениями может изменяться. Это осуществляется на основе определенного документа, имеющего название (приказ, распоряжение) и дату. В каждом документе м.б. несколько позиций, отображающих следующую информацию: номер позиции документа; действие, осуществляемое с помещением (передать, закрепить) дата действия; название подразделения; перечень помещений, возможное наименование другого подразделения. Например – «передать с 20.06.2007 г. отделу № 3 лабораторные помещения 14105 и 14106, закрепленные за лабораторией № 5»; «закрепить за медпунктом с 15.09.2007 г. складской помещение 3109» .

Необходимо осуществлять следующую обработку данных:

- на заданную дату список подразделений предприятия (наименование) и перечень занимаемых им помещений – номер, тип;
- список, отображающий иерархию (дерево) подчинения подразделений предприятия;
- динамика изменения количества площадей помещений у заданного подразделения за заданный период – количество по годам.

2_17 Автоматизированная информационная система «Скорая помощь»

Описание предметной области.

Лечебное учреждение (Код, Название, Краткое название, Адрес, Контактные телефоны) оказывает скорую медицинскую помощь населению. В учреждении имеется штат сотрудников, о которых необходимо хранит следующие сведения:

- табельный номер;
- ФИО; дата рождения, пол;
- должность, дата начала работы в данной должности, дата окончания, ставка.

Работа в учреждении круглосуточная – сотрудники работают по 24 часа с последующими выходными днями. Необходимо знать, в какой смене и бригаде работает тот или иной сотрудник. Закрепление в бригаду осуществляется на основании внутреннего приказа, имеющего номер и дату. В каждой позиции приказа указывается, что конкретный сотрудник с даты 1 по дату 2 работает в бригаде с заданным номером.

Необходимо вести учет сведений о выездах бригад на вызовы. Каждый вызов определяется датой, временем выезда и адресом. Пациент, которому оказывается помощь, может быть описан следующими данными ФИО, возраст (примерный), первоначальный диагноз. Необходимо также знать ФИО и должности сотрудников выехавшей на вызов бригады (включая водителя и диспетчера). Необходимо также хранить небольшое текстовое описание принятых бригадой мер. Если больной был госпитализирован, либо получил направление на госпитализацию, также необходимо знать в какое учреждение он был направлен (название, адрес). При возвращении бригады фиксируется время прибытия.

Необходимо осуществлять следующую обработку данных:

- на заданную дату список выездов всех бригад учреждения (номер выезда, время, номер бригады, принятые меры);
- на заданную дату описание самого длительного выезда;
- на заданную дату список заданной бригады (табельный номер, ФИО, должность).

Критерии оценки работы:
1. Полный ответ – 5 баллов.
2. Дополнительный уточняющий вопрос – 4 балла.
3. Краткий ответ – 3 балла.

ПРАКТИЧЕСКАЯ РАБОТА № 28

Работа с CASE – средствами кодирования программного обеспечения

Цель работы: Подготовка модели к генерации программного кода; генерация программного кода.

Исходные данные (задание):

Теоретическая часть

Одно из самых мощных свойств Rational Rose – возможность генерации программного кода, представляющего модель. Варианты генерации программ меняются в зависимости от установленной версии Rose. В настоящее время применяются следующие три различные версии:

- Rose Modeler – позволяет создавать модель системы, но не поддерживает генерацию программного кода и обратное проектирование.
- Rose Professional – позволяет генерировать программный код на одном языке.
- Rose Enterprise – позволяет генерировать программный код на Ada 83, Ada 95, ANSI C++, CORBA, Java, COM, Visual Basic, Visual C++, C++ и XML. Кроме того, поддерживается генерация кода и обратное проектирование баз данных.

Лабораторный практикум построен на изучении Case-средства Rational Rose версии - Enterprise Edition.

Подготовка к генерации программного кода

Процесс генерации программного кода состоит из пяти основных этапов:

6. Проверка модели.
7. Создание компонентов.
8. Отображение классов на компоненты.
9. Установка свойств генерации программного кода.
10. Генерация программного кода.

В разных языках не все этапы обязательны. Так, программы, разработанные на языке C++, генерируются и без предварительного создания компонентов. Генерировать код программ на любом языке можно, не выполняя проверки модели, хотя во время генерации это порой приводит к различным ошибкам.

Хотя не все этапы обязательны, первые пять из них рекомендуется выполнять до начала процесса генерации. Проверка модели поможет выявить ее неточности и недостатки, нежелательные с точки зрения последующей генерации программ. Этапы, на которых рассматриваются компоненты, - это способ отобразить логическую схему системы на ее физическую реализацию, причем на этих этапах собирается большой объем полезной информации. Если пропустить эти этапы, то для создания компонентов Rose воспользуется пакетной структурой логического представления.

3.1 Проверка модели

В Rose существует не зависящее от языка средство проверки моделей,

применяемое для обеспечения корректности модели перед генерацией программного кода. Такая проверка помогает выявить в модели неточности и ошибки, не позволяющие генерировать программы надлежащим образом.

В общем случае проверка модели может выполняться на любом этапе работы над проектом. Однако после завершения разработки графических диаграмм она является обязательной, поскольку позволяет выявить целый ряд ошибок разработчика. К числу таких ошибок и предупреждений относятся, например, не используемые ассоциации и классы, оставшиеся после удаления отдельных графических элементов с диаграмм, а также операции, не являющиеся именами сообщений на диаграммах взаимодействия.

Для проверки модели следует выполнить операцию главного меню: Tools→ Check Model (Инструменты→Проверить модель). Результаты проверки разработанной модели на наличие ошибок отображаются в окне журнала. Прежде чем приступить к генерации текста программного кода разработчику следует добиться устранения всех ошибок и предупреждений, о чем должно свидетельствовать чистое окно журнала (рисунок 3.1).

Рисунок 3.1- Вид журнала при отсутствии ошибок по результатам проверки модели

К наиболее распространенным ошибкам относятся, например, сообщения на диаграмме Последовательности или Кооперативной диаграмме, не отображенные на операцию, либо объекты этих диаграмм, не отображенные на класс.

3.2 Создание компонентов

Второй этап процесса генерации программного кода - создание компонентов для классов. Существуют компоненты самых разных типов: файлы исходного программного кода, исполняемые файлы, библиотеки исполнения в реальном времени, компоненты ActiveX, апплеты и т.д. Перед генерацией программ можно отобразить каждый из классов на соответствующий компонент исходного программного кода.

После создания компонентов можно добавить зависимости между ними на диаграмме Компонентов. Зависимости между компонентами - это зависимости во время компиляции системы.

При генерации программ на C++, Java или Visual Basic выполнять подобный шаг не обязательно. В Java и Visual Basic Rose создаст автоматически соответствующий компонент для каждого из классов.

Для создания компонента:

1. Откройте диаграмму Компонентов (Component).
2. С помощью значка Component панели инструментов Diagram введите новый

компонент в диаграмму.

3.3 Отображение классов на компоненты

Каждый компонент исходного кода - это файл с исходным программным кодом для одного или нескольких классов. В C++ каждый класс отображается на два компонента с исходным кодом: файл заголовка и основной файл (тело). В PowerBuilder на один компонент отображается несколько классов. Компонентом с исходным программным кодом в PowerBuilder является файл библиотеки PowerBuilder (.pbl). В Java каждый компонент - это один файл .java. Компоненты также создаются для элементов управления ActiveX, апплетов, файлов DDL, исполняемых файлов, а также других исходных и скомпилированных файлов.

Третий этап процесса генерации программного кода - отображение каждого из классов на соответствующие компоненты. В PowerBuilder необходимо отобразить каждый класс на компонент перед генерацией программы, в то время как в C++, Java и Visual Basic этот шаг не является обязательным. Rose может генерировать программный код самостоятельно. При генерации в Rose программ Java и Visual Basic производится еще и генерация нужных компонентов и отображение классов. Однако для C++ компоненты не создаются автоматически, а кроме того, ни для одного из языков не генерируются зависимости. Поэтому рекомендуется выполнять этот шаг независимо от применяемого языка программирования. Для отображения класса на компонент:

1) Щелкните правой кнопкой мыши на компоненте, на диаграмме Компонентов или в браузере.
2) Выберите Open Specification в контекстном меню.
3) Выберите вкладку Realizes (Реализует).
4) Во вкладке Realizes щелкните правой кнопкой мыши на нужном классе (классах) и выберите Assign (Присвоить) в контекстном меню.
5) В браузере имя компонента будет показано в круглых скобках вслед за именем класса в Логическом представлении (Logical view).

ИЛИ

1) Найдите класс в окне Logical view браузера.
2) Перетащите класс на нужный компонент в окне Component view.
3) В окне Logical view имя компонента будет показано в круглых скобках за именем класса.

3.4 Установка свойств генерации программного кода

Для каждого языка в Rose предусмотрен ряд определенных свойств генерации программного кода. Можно установить несколько параметров генерации программного кода для классов, атрибутов, компонентов и других элементов модели. Этими свойствами определяется способ генерации программ. В Rose предлагаются общепринятые параметры по умолчанию.

3.4.1 Настройка свойств C++

Для того чтобы пользоваться возможностями C++, необходимо описать назначение этих свойств, список которых доступен во вкладке C++ спецификаций класса. (рисунок 3.2)

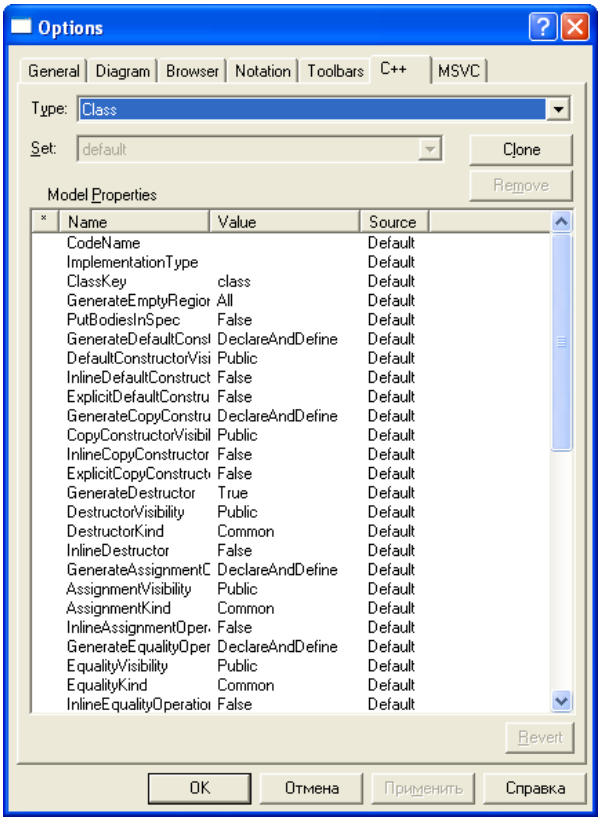


Рисунок 3.2 – Окно свойств генерации кода на C++

Перед генерацией программного кода рекомендуется анализировать эти свойства и вносить необходимые изменения.

3.4.2Назначение свойств:

30) **CodeName** - устанавливает имя класса в создаваемом коде. Данное свойство необходимо устанавливать только в том случае, если имя класса должно быть отлично от имени заданного в модели Rational Rose. Данное свойство необходимо использовать для создания работоспособного кода C++, если для классов в модели используются русские имена.

31) **ImplementationType** - позволяет использовать простые типы вместо определения класса, устанавливаемого Rational Rose по умолчанию. При задании этого параметра создается директива typedef.

32) **ClassKey** - используется для задания типа класса, такого как class, struct, или union. Если тип не указан, то создается класс.

33) **GenerateEmptyRegion** - свойство указывает, как будет создаваться пустой раздел protected: None - пустой раздел не будут создан; Preserved - пустой раздел будет создан, если будет установлено свойство «preserve=yes»; Unpreserved — пустой раздел будет создан, если будет установлено свойство «preserve=no»; All — всегда будет создаваться.

34) **PutBodiesInSpec** - если установлено как true, то в заголовочный файл попадет и описание тела класса. Используется для компиляторов, которым необходимо определение шаблона класса в каждом компилируемом файле.

35) **GenerateDefaultConstructor** - позволяет установить, необходимо ли создавать конструктор для класса по умолчанию. Может принимать следующие значения: DeclareAndDefine - создается определение для конструктора и скелет конструктора в теле класса; Declare Only - создается только определение;

DoNotDeclare - не создается ни определения, ни скелета конструктора.

36) **DefaultConstructorVisibility** - устанавливает раздел, в котором будет определен конструктор по умолчанию: public, protected, private, implementation.

37) **InlineDefaultConstructor** - устанавливает, будет ли конструктор по умолчанию создаваться как inline подстановка. Если конструктора по умолчанию нет, то данное свойство не оказывает на код никакого эффекта.

38) **ExplicitDefaultConstructor** - устанавливает конструктор по умолчанию как explicit (явно заданный).

39) **InlineRelationalOperations** - определяет, будут ли функции операторов сравнения создаваться как inline подстановка.

40) **GenerateStorageMgmtOperations** - определяет, будут ли переопределяться операторы new и delete в классе.

41) **StorageMgmtVisibility** - определяет раздел, в который будут помещены операторы new и delete.

42) **InlineStorageMgmtOperations** - определяет, будут ли операторы new и delete определены как inline подстановка.

43) **GenerateSubscriptOperation** - определяет, будет ли переопределен оператор [].

44) **Subscript Visibility** определяет - раздел, в который будет помещен оператор [].

45) **SubscriptKind** - определяет вид функций оператора []: Common - обычная, Virtual - виртуальная, Abstract - абстрактная.

46) **SubscriptResultType** - определяет тип возвращаемого выражения для оператора [].

47) **InlineSubscriptOperation** - определяет, будет ли оператор [] определен как inline подстановка.

48) **GenerateDereferenceOperation** - определяет, будет ли переопределен оператор *.

49) **Dereference Visibility** - определяет раздел, в который будет помещен оператор *.

50) **DereferenceKind** - определяет вид функций оператора *: Common - обычная, Virtual - виртуальная, Abstract - абстрактная.

51) **DereferenceResultType** - определяет тип возвращаемого выражения для оператора *.

52) **InlineDereferenceOperation** - определяет, будет ли оператор * определен, как inline подстановка.

53) **GenerateIndirectionOperation** - определяет, будет ли переопределен оператор ->.

54) **IndirectionVisibility** - определяет раздел, в который будет помещен оператор ->.

55) **IndirectionKind** - определяет вид функций оператора ->: Common - обычная, Virtual - виртуальная, Abstract - абстрактная.

56) **IndirectionResultType** - определяет тип возвращаемого выражения для оператора ->.

57) **InlineIndirectionOperation** - определяет, будет ли оператор -> определен как inline подстановка.

58) **GenerateStreamOperations** - определяет, будут ли переопределены

операторы потоков (« и »).

3.5 Выбор класса, компонента или пакета

При генерации программного кода за один раз можно создать класс, компонент или целый пакет. Программный код генерируется с помощью диаграммы или браузера. При генерации кода из пакета можно выбрать либо пакет Логического представления на диаграмме Классов, либо пакет представления Компонентов на диаграмме Компонентов. При выборе пакета Логического представления генерируются все его классы. При выборе пакета представления Компонентов генерируются все его компоненты.

Программный код можно генерировать одновременно для нескольких классов, компонентов или пакетов. На диаграмме с помощью клавиши Ctrl выберите классы, компоненты или пакеты, для которых нужно сгенерировать программный код, а затем - соответствующую команду генерации в меню.

3.6 Генерация программного кода

Если у вас установлены Rose Professional или Rose Enterprise, то в меню Tools предлагается несколько вариантов, специфичных для конкретного языка программирования (рисунок 3.3).

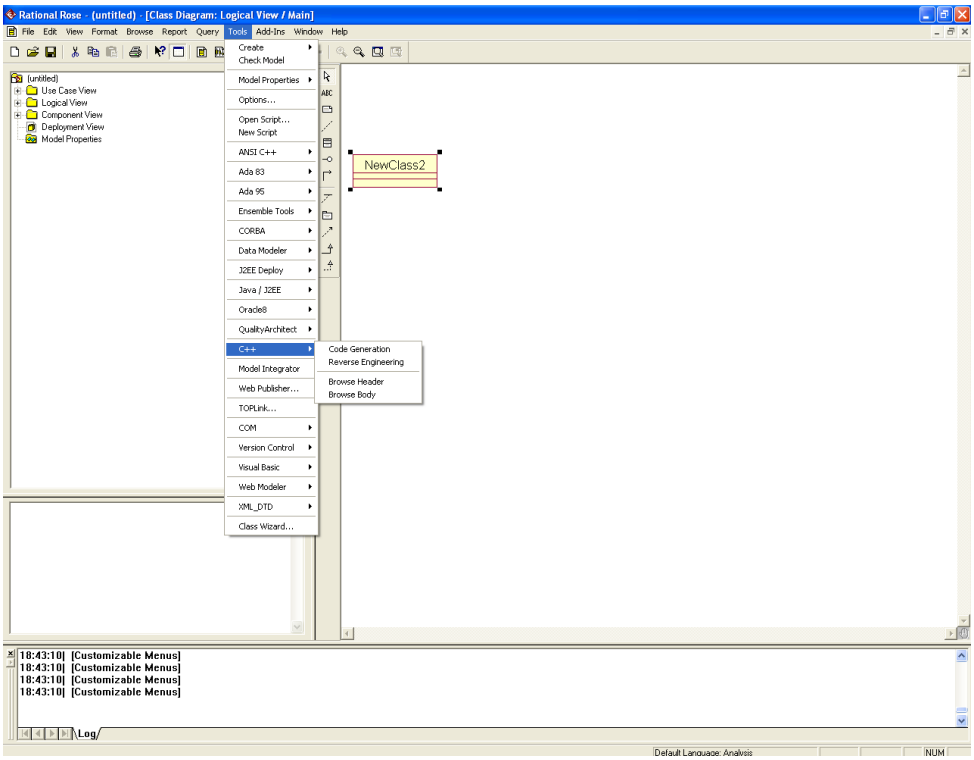


Рисунок 3.3 - Пункты меню генерации кода

Чтобы показать или скрыть эти пункты меню, выберите пункт Add-Ins → Add-Ins (Надстройки → Менеджер надстроек). В диалоговом окне Add-In Manager (рисунок 3.4) с помощью флажков покажите или скройте нужные варианты для различных языков.

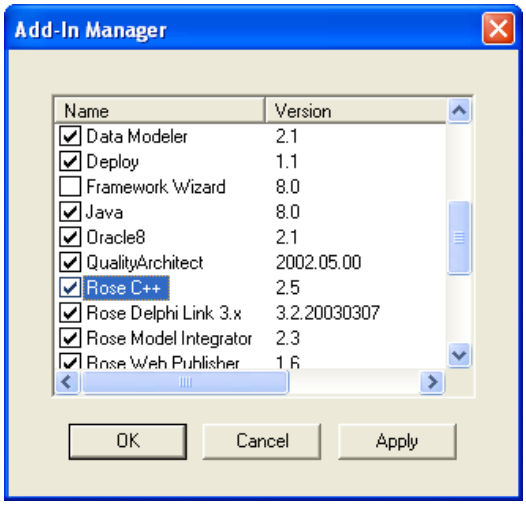


Рисунок 3.4 - Менеджер надстроек Add-Ins

Генерация программного кода в среде IBM Rational Rose 2003 возможна для отдельного класса или компонента. Для этого нужный элемент модели предварительно следует выделить в браузере проекта и выполнить операцию контекстного меню: Tools→C++→Code Generation - (Язык C++→Генерировать код). В результате этого будет открыто диалоговое окно с предложением выбора классов для генерации программного кода на выбранном языке программирования (рисунок 3.5). После выбора соответствующих классов и нажатия кнопки OK программа IBM Rational Rose 2003 выполняет кодогенерацию.

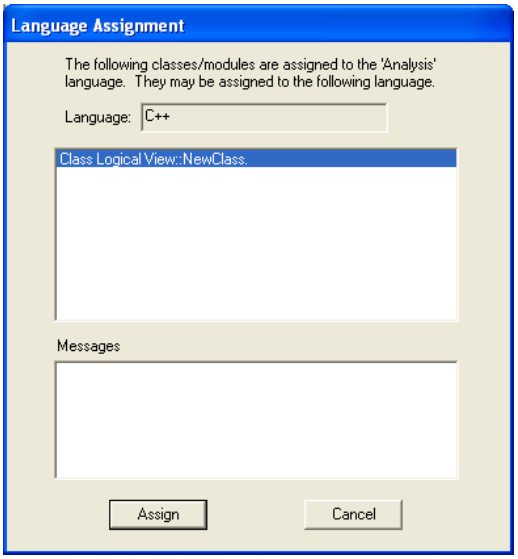


Рисунок 3.5 - Окно выбора классов для генерации программного кода

Затем происходит компиляция и выдается окно статуса (Code Generation Status). Здесь можно увидеть информацию о том, какой класс был закодирован и количество ошибок и предупреждений (рисунок 3.6). Если у вас произошла, какая-либо ошибка или же предупреждение, то их можно увидеть на рабочем поле в Rational Rose, для этого и существует самое нижнее окно, в нем передаются все ваши действия и ошибки, произошедшие в ходе кодогенерации.

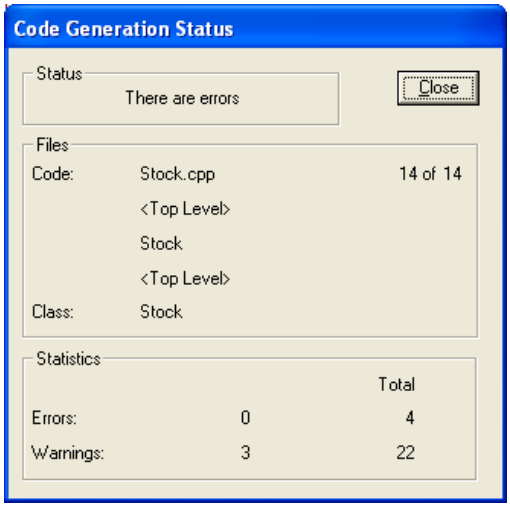


Рисунок 3.6 – Окно статуса компиляции

3.7 Результаты генерации

В результате кодогенерации Rational Rose создает два файла с расширением “.h” и “.cpp”, названия у них те же, что и название класса. Итак, выполнив эти действия, нажимаем правой клавишей на класс, появляется окошко, в нем ищем “C++”, и видим два пункта Browse Header и Browse Body, и в зависимости от того какой из файлов нам нужен “.h” (заголовочный) или “.cpp” (непосредственно реализация), выбираем их. Эти файлы открываются с помощью блокнота и теперь легко можно увидеть скелет класса, с различными комментариями, которые писали вы на диаграммах, и комментарии которые вставляет сама Rose. Теперь можно открыть один из файлов в C++ и доработать класс, описать работу функций, добавить различные нововведения.

Следует заметить, что при установленной на компьютер разработчика интегрированной среды сгенерированные файлы с текстом программного кода автоматически открываются в этой среде после двойного щелчка на пиктограмме этих файлов. Тем не менее, лучше копировать содержимое этих файлов в предварительно созданные программные проекты для полного контроля в этих средах процесса программирования и отладки приложений.

Сгенерированные программой IBM Rational Rose 2003 файлы с текстом программного кода содержат минимум информации. Для включения дополнительных элементов в программный код следует изменить свойства генерации программного кода, установленные по умолчанию.

В заключение следует отметить, что эффект от использования средства IBM Rational Rose 2003 проявляется при разработке *масштабных проектов* в составе команды или проектной группы. Однако ситуация покажется не столь тривиальной, когда станет необходимо выполнить проект с несколькими десятками вариантов использования и сотней *классов*. Именно для подобных проектов явно выявляется преимущество использования средства IBM Rational Rose 2003 и нотации языка UML для документирования и реализации соответствующих моделей.

Задание на лабораторное занятие

3. Изучить теоретический материал

4. Сгенерировать программный код на C++ для диаграммы классов, разработанной вами в предыдущей лабораторной работе.

Содержание отчета

- титульный лист;
- постановка задачи;
- листинг сгенерированного кода;
- вывод.

Контрольные вопросы

1. Какие диаграммы необходимо предварительно разработать, чтобы выполнить кодогенерацию?
2. Как посмотреть исходный код?
3. Какие установки свойств доступны на вкладке C++?
4. Какова структура создаваемого кода?
5. Что необходимо добавить в шаблоны классов для получения работоспособного приложения?
6. Какие шаги нужно предпринять для обновления модели по исходному коду?
7. Какие основные этапы кодогенерации вы знаете? Расскажите кратко о каждом из них?

Критерии оценки работы:

1. Полный ответ – 5 баллов.
2. Дополнительный уточняющий вопрос – 4 балла.
3. Краткий ответ – 3 балла.

ПРАКТИЧЕСКАЯ РАБОТА № 29

Работа с CASE – средствами тестирования программного обеспечения

Цель работы: отладка разработанного программного средства; тестирование разработанного программного средства.

Исходные данные (задание):

Теоретическая часть

Отладка ПС – это деятельность, направленная на обнаружение и исправление ошибок в ПС с использованием процессов выполнения его программ. *Тестирование* ПС – это процесс выполнения его программ на некотором наборе данных, для которого заранее известен результат применения или известны правила поведения этих программ. Указанный набор данных называется тестовым или просто тестом. Таким образом, отладку можно представить в виде многократного повторения трех процессов: тестирования, в результате которого может быть констатировано наличие в ПС ошибки, поиска места ошибки в программах и документации ПС и редактирования программ и документации с целью устранения обнаруженной ошибки.

Отладка программного средства невозможна без представления физической структуры. На этом этапе проектирования необходимо разработать *диаграмму компонентов*.

Диаграммой компонентов (Component diagram) называется диаграмма UML, на которой показаны компоненты системы и зависимости между ними.

Компонентом называется физический модуль кода. Компонентами бывают как библиотеки исходного кода, так и исполняемые файлы. Например, .h и .cpp и .exe - будут отдельными компонентами.

Особенности разработки диаграмм компонентов в среде IBM Rational Rose 2003

Диаграмма *компонентов* служит частью физического представления модели, играет важную роль в процессе ООАП и является необходимой для генерации программного кода. Для разработки диаграмм *компонентов* в браузере проекта предназначено отдельное представление *компонентов* (Component View), в котором уже содержится диаграмма *компонентов* с пустым содержанием и именем по умолчанию Main (Главная).

Активизация диаграммы *компонентов* может быть выполнена одним из следующих способов:

Щелкнуть на кнопке с изображением диаграммы *компонентов* на стандартной панели инструментов.

Раскрыть представление *компонентов* в браузере (Component View) и дважды щелкнуть на пиктограмме Main (Главная).

Через пункт меню Browse→Component Diagram (Браузер→Диаграмма *компонентов*).

В результате выполнения этих действий появляется новое окно с чистым рабочим листом диаграммы *компонентов* и специальная панель инструментов, содержащая кнопки с изображением графических примитивов, необходимых для разработки диаграммы *компонентов* (таблица 4.1).

Программа IBM Rational Rose 2003 не поддерживает следующие графические *стереотипы*. Графическое изображение этих *стереотипов* и их краткая характеристика приводятся в следующей таблице (таблица 4.2). При этом каждому из *компонентов*, как правило, соответствует отдельный файл исходной сборки программного приложения.

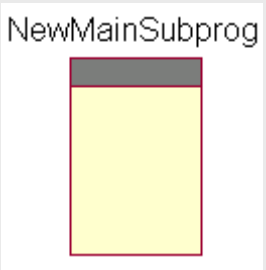
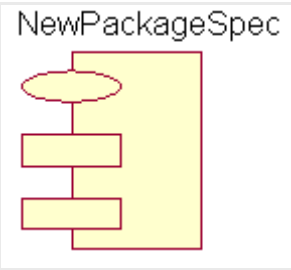
Использование *стереотипов* существенно увеличивают наглядность графического представления диаграммы *компонентов* и позволяют архитектору уточнить характер реализации модели программистом на выбранном языке программирования.

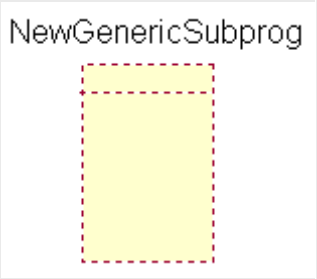
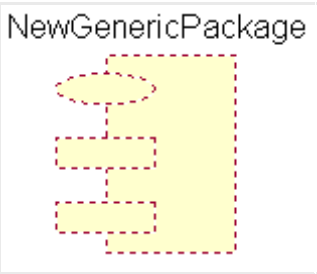
Таблица 4.1 - Назначение кнопок специальной панели инструментов диаграммы компонентов

Графическое изображение	**Всплывающая подсказка**	**Назначение кнопки**
	Selection Tool	Превращает изображение курсора в форму стрелки для последующего выделения элементов на диаграмме
	Text Box	Добавляет на диаграмму текстовую область
	Note	Добавляет на диаграмму примечание
	Anchor Note to Item	Добавляет на диаграмму связь примечания с соответствующим графическим элементом диаграммы
	Component	Добавляет на диаграмму *компонент*
	Package	Добавляет на диаграмму пакет
	Dependency	Добавляет на диаграмму отношение зависимости
	Subprogram Specification	Добавляет на диаграмму спецификацию подпрограммы
	Subprogram Body	Добавляет на диаграмму тело подпрограммы
	Main Program	Добавляет на диаграмму главную программу
	Package Specification	Добавляет на диаграмму спецификацию пакета
	Package Body	Добавляет на диаграмму тело пакета

	Task Specification	Добавляет на диаграмму спецификацию задачи
	Task Body	Добавляет на диаграмму тело задачи
	Generic Subprogram	Добавляет на диаграмму типовую подпрограммы(по умолчанию отсутствует)
	Generic Package	Добавляет на диаграмму типовой пакет (по умолчанию отсутствует)
	Database	Добавляет на диаграмму базу данных (по умолчанию отсутствует)

Таблица 4.2. Графическое изображение стереотипов компонентов и их характеристика

Графическое изображение и имя по умолчанию	**Названи е стереотипа**	**Характеристика стереотипа *компонента***
	Subprogram Specification	Спецификация подпрограммы. Содержит описание переменных, процедур и функций и не содержит определений классов
	Subprogram Body	Тело подпрограммы. Содержит реализацию процедур и функций, не относящихся к каким-то классам, при этом не содержит определений классов или реализаций операций других классов
	Main Program	Главная программа. Реализует базовую логику работы программного приложения и содержит ссылки на другие *компоненты* модели
	Package Specification	Спецификация пакета. Содержит определение класса, его атрибутов и операций. В языке программирования C++ спецификации пакета соответствует отдельный файл с расширением «h»

	Package Body	Тело пакета. Содержит код реализации операций класса. В языке программирования C++ спецификации пакета соответствует отдельный файл с расширением «cpp»
	Task Specification	Спецификация задачи. Может содержать определение класса, его атрибутов и операций, которые предполагается использовать в независимом потоке управления
	Task Body	Тело задачи. Может содержать реализацию операций класса, которые имеют независимый поток управления.
	Generic Subprogram	Типовая подпрограмма. Содержит описание переменных, процедур и функций, которые могут быть использованы в нескольких программных приложениях. При этом типовая подпрограмма не содержит определений классов
	Generic Package	Типовой пакет. Содержит определение класса, его атрибутов и операций, которое может быть использовано в нескольких программных приложениях
	Database	База данных. Содержит определение одного или нескольких классов, их атрибутов и, возможно, операций. При этом соответствующие классы могут быть реализованы в форме одной или нескольких таблиц базы данных

4.1 Добавление компонента на диаграмму компонентов и редактирование его свойств

Для добавления *компонента* на диаграмму *компонентов* нужно с помощью левой кнопки мыши нажать кнопку с изображением пиктограммы *компонента* на специальной панели инструментов, отпустить левую кнопку мыши и щелкнуть левой кнопкой мыши на свободном месте рабочего листа диаграммы. Добавить *компонент* на диаграмму можно также с помощью операции главного меню: Tools→ Create→Component или с помощью операции контекстного меню: New→Component, предварительно выделив представление *компонентов* в браузере проекта.

В результате этих действий на диаграмме появится изображение *компонента* с маркерами изменения его геометрических размеров и предложенным средой именем по умолчанию, которое разработчику следует изменить (рисунок 4.1).

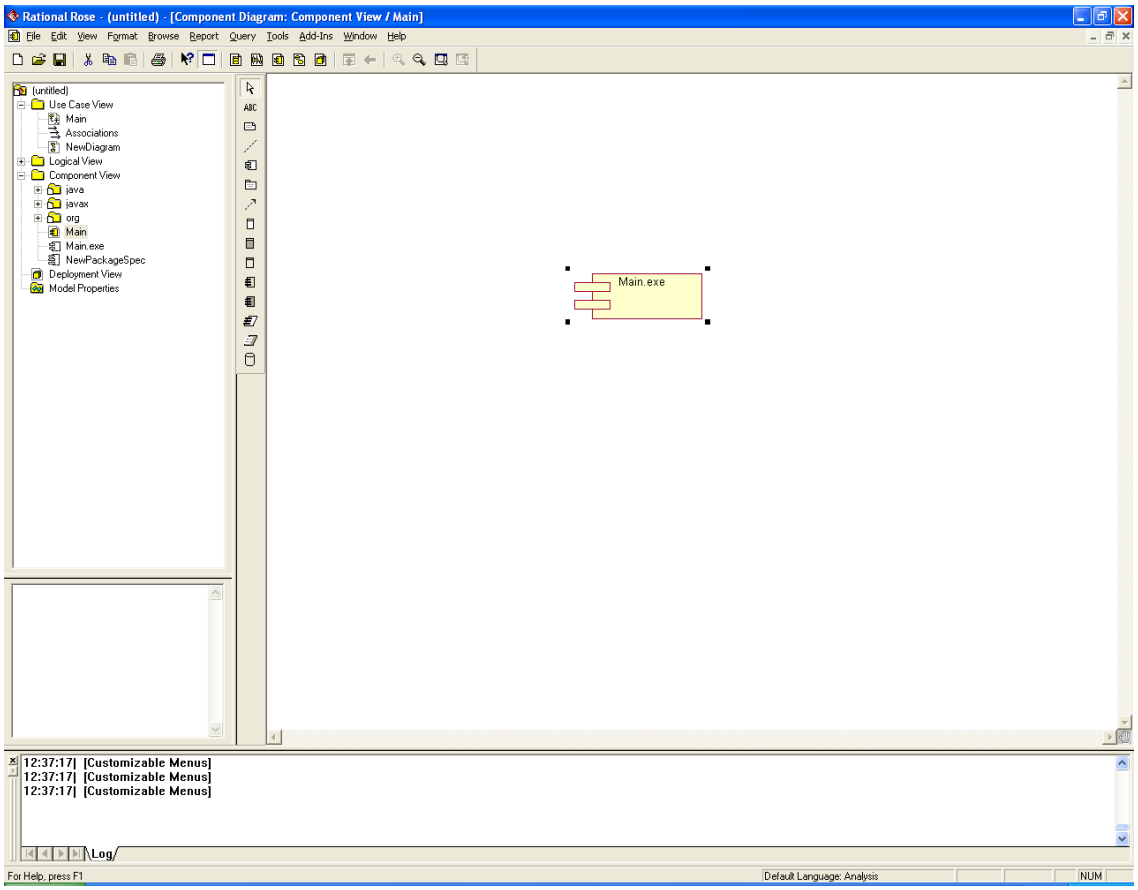


Рисунок 4.1 - Диаграмма компонентов после добавления компонента Main.exe

Для каждого *компонента* можно определить различные свойства, такие как стереотип, язык программирования, декларации, реализуемые классы. Редактирование этих свойств для произвольного *компонента* осуществляется с помощью диалогового окна спецификации свойств (рисунок 4.2).

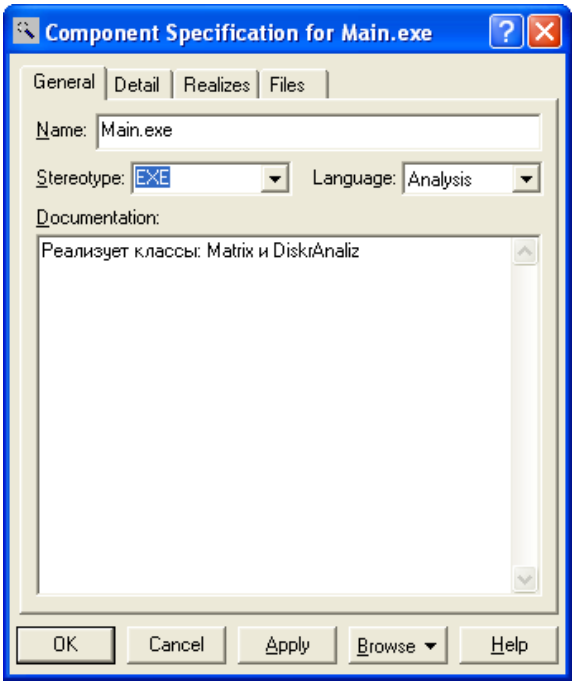


Рисунок 4.2 - Диалоговое окно спецификации свойств компонента Main.exe

В частности, для *компонента* Main.exe можно выбрать стереотип <<EXE>> из предлагаемого вложенного списка, поскольку применительно к разрабатываемой модели предполагается реализация этого *компонента* в форме исполнимого файла. При этом на вкладке Realizes (Реализует) содержатся все классы, включая и актеров, которые на данный момент присутствуют в модели.

По умолчанию в среде IBM Rational Rose 2003 для всех добавляемых на диаграмму компонентов в качестве языка реализации используется язык анализа, который в последствии следует изменить на тот язык программирования, который предполагается использовать для написания программного кода. В дальнейшем при генерации программного кода необходимо будет дополнительно выбрать те классы, которые реализует тот или иной компонент модели.

4.2 Добавление отношения зависимости и редактирование его свойств

Добавление отношения зависимости на диаграмму компонентов аналогично добавлению соответствующего отношения на диаграмму вариантов использования. Для добавления зависимости между двумя компонентами нужно с помощью левой кнопки мыши нажать кнопку с изображением зависимости на специальной панели инструментов, отпустить левую кнопку мыши, щелкнуть левой кнопкой мыши на изображении исходного компонента на диаграмме и отпустить ее на изображении целевого компонента. В результате этих действий на диаграмме появится изображение отношения зависимости в форме пунктирной линии со стрелкой, соединяющей два выбранных компонента.

4.3 Пример построения диаграммы компонентов

Ниже представлена диаграмма компонентов для программного средства, реализующего алгоритм дискриминантного анализа (рисунок 4.3).

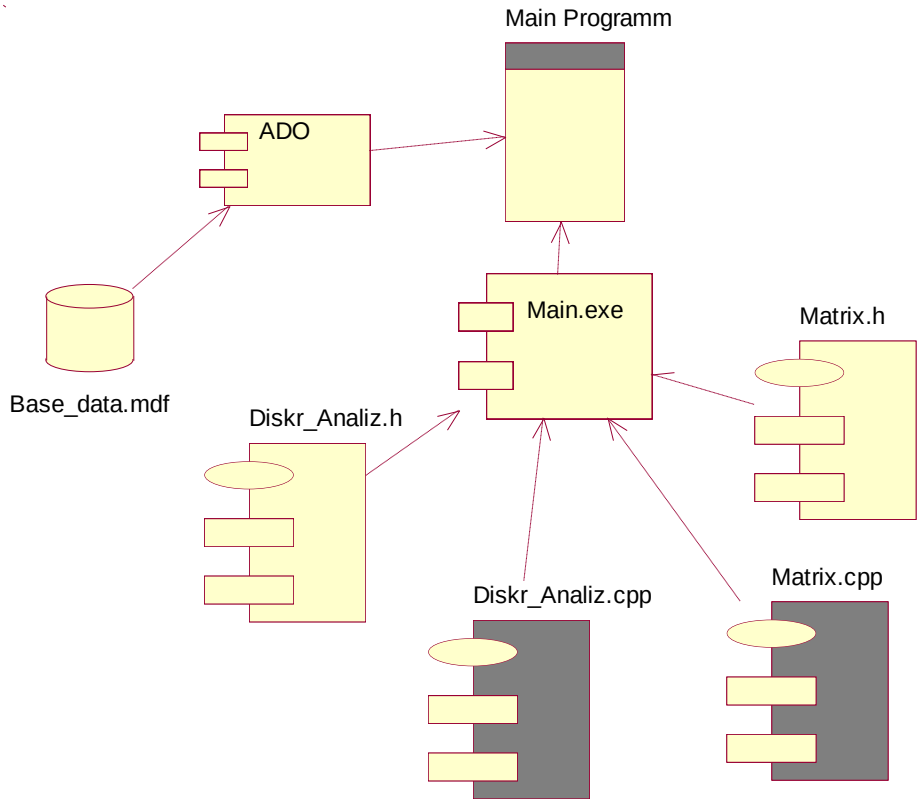


Рисунок 4.3 – Диаграмма компонентов рассматриваемого ПС

5 Тестирование программного средства

Тестирование – процесс многократного повторения программы с целью обнаружения ошибок. Существуют следующие методы тестирования ПС:

- статическое тестирование (ручная проверка программы за столом);
- детерминированное тестирование (при различных комбинациях исходных данных);
- стохастическое (исходные данные выбираются произвольно, на выходе определяется качественное совпадение результатов или примерная оценка).

При тестировании разработанного ПС необходимо использовать подходящий по функциональности пакет прикладного математического программного обеспечения.

В выбранной среде необходимо произвести расчеты всех параметров, реализованных в ПС, затем следует сравнить результаты и сделать вывод о качестве данного программного продукта.

Задание на лабораторное занятие

3. Изучить теоретический материал
4. Выполнить тестирование и отладку информационной системы, разработанной в лабораторной работе, то есть разработать диаграмму компонентов рассматриваемого ПС.

Содержание отчета

- титульный лист;
- постановка задачи;
- тестовый набор данных и результаты тестирования;
- вывод.

Критерии оценки работы:
1. Полный ответ – 5 баллов.
2. Дополнительный уточняющий вопрос – 4 балла.
3. Краткий ответ – 3 балла.

ПРАКТИЧЕСКАЯ РАБОТА № 30

Название

Цель работы: ознакомиться с теоретическим материалом, ответить на вопросы.

Исходные данные (задание):

Критерии оценки работы:
1. Полный ответ – 5 баллов.
2. Дополнительный уточняющий вопрос – 4 балла.
3. Краткий ответ – 3 балла.

1. АИС «Индивидуальный план преподавателя»

Описание предметной области.

Для каждого преподавателя (ФИО, Год рождения, Домашний адрес, Контактные телефоны) высшего учебного заведения (Код, Название, Краткое название) на каждый учебный год (Год начала учебного года, Год окончания учебного года) формируется индивидуальный план. В индивидуальном плане отражается общий объем работ преподавателя, который он должен выполнить в течение учебного года. Учет работ ведется по следующей форме:

№	Наименование работы	План		Факт	
		Осенний семестр	Весенний семестр	Осенний семестр	Весенний семестр

В течение учебного года преподаватель выполняет следующие виды работ (Код, Название Краткое название): «Учебная работа», «Учебно-методическая работа», «Научно-методическая работа», «Научно-исследовательская работа», «Организационно-методическая работа», «Внеучебная работа со студентами», «Прочие виды работ». Необходимо вести учет в часах (целых и долях часов) объем запланированных и фактически выполненных объемов работ для каждого преподавателя по семестрам. Для каждого преподавателя также необходимо фиксировать место работы – факультет (Код, Название, Краткое название), кафедра (Код, Название, Краткое название), занимаемую должность (Код, Название, Краткое название), время работы в этой должности (Дата начала, Дата окончания, Ставка, Дата избрания на должность), кем является преподаватель – штатным сотрудником или совместителем. Также для преподавателя фиксируются:

- ученая степень (Код, Название, Краткое название) – доктор, кандидат; каких наук (Код, Название, Краткое название) – технических, экономических и т.п.; год присуждения;
- ученое звание (Код, Название, Краткое название) – профессор, доцент, с.н.с. и т.п.; год присуждения звания.

Необходимо осуществлять следующую обработку данных:

- формирование для каждого преподавателя итоговой суммы (в часах) запланированных и выполненных объемов работ по семестрам;
- список преподавателей, у которых фактическое значение выполненных работ превышает плановое (факультет, кафедра, ФИО, учёная степень, учёное звание, должность, семестр, количество перевыполненных объемов работ);
- список преподавателей заданной кафедры, имеющих заданную ученую степень на заданную дату.

2. АИС «Обслуживание заказов клиентов»

Описание предметной области.

Предприятие (Код, Название, Краткое название) осуществляет доставку разных товаров (Код, Название, Краткое название) населению. Прием заказов от населения осуществляет специальная служба (Код, Название, Краткое название) предприятия.

Для того чтобы стать потребителем услуг предприятия каждый абонент должен зарегистрироваться, при этом фиксируются его ФИО, адрес, телефон и паспортные данные (Серия, Номер, Дата выдачи, Кем выдан). Каждый абонент в течение дня может сделать несколько заказов (Дата, Время), заказу присваивается номер.

В каждом заказе может содержаться несколько товаров, для каждого указывается количество товара, единица измерения (Код, Название, Краткое Название), цена за единицу товара, общая стоимость товара. Заказ также имеет итоговую сумму. При формировании бланка заказа, который будет подписан абонентом при получении товара фиксируется, оплачен заказ, или абонент получает товар в кредит. Также на бланке заказа указывается: реквизиты предприятия (название, адрес, контактные телефоны); ФИО и должность оператора, приявшего заказ; ФИО, должность сотрудника, доставившего заказ.

Необходимо осуществлять следующую обработку данных:

- список товаров (код, наименование), пользующихся наибольшим спросом (максимальное количество позиций заказов) у населения за заданный период;
- динамика изменения стоимости заданного товара за заданный период по месяцам;
- список наименований улиц, на которых проживают абоненты предприятия по убыванию числа абонентов.

3. АИС «Прохождение преддипломной практики студентами вуза»

Описание предметной области.

Студенты высшего учебного заведения (Код, Название, Краткое название) в период подготовки дипломной работы (проекта) проходят преддипломную практику. Для каждого студента (Номер зачетной книжки, ФИО), обучающегося на определенной специальности (Код, Название, Краткое название), факультете (Код, Название, Краткое название), форме обучения (Код, Название, Краткое название) фиксируется место прохождения преддипломной практики – предприятие (Код, Название, Краткое название), адрес предприятия, ФИО, должность руководителя от вуза, ФИО, должность руководителя от предприятия, срок прохождения практики (Дата начала, Дата окончания). В базе данных также необходимо вести данные о сроках защиты практики для каждой группы, оценке, полученной студентом за практику. При вводе данных о месте прохождения практики для каждого студента необходимо помечать – планирует ли студент в дальнейшем работать на данном предприятии, варианты ответов - да, нет, не знаю.

Необходимо осуществлять следующую обработку данных:

- количество студентов, проходивших практику на заданном предприятии в заданный период;
- перечень предприятий (название, адрес) по алфавиту, на которых проходили преддипломную практику студенты заданной специальности за заданный период;
- на заданную дату список студентов заданной специальности и потока (год обучения), не имеющих оценку за практику.

4 АИС «Лицензионное программное обеспечение организации»

Описание предметной области.

Необходимо вести учет и анализ информации о лицензионном программном обеспечении (ПО), установленном на компьютерах организации (Код, Название, Краткое название). Для каждого компьютера фиксируется инвентарный номер, тип (рабочая станция или сервер), местоположение – в каком подразделении (Код, Название, Краткое название) организации компьютер установлен. Компьютеры могут передаваться из подразделения в подразделение, при этом необходимо знать сроки (Дата начала, Дата окончания) нахождения компьютера в подразделении и на основании какого документа он перемещается (Номер документа, Дата документа), тип этого документа (приказ, распоряжение и т.п.). При установке лицензионного ПО фиксируется, куда установлено ПО – на какой компьютер, название продукта, его тип (среда разработки прикладных программ, среда администрирования БД, операционная система, антивирусная программа и т.п.), фирма производитель, срок действия лицензии (Дата начала, Дата окончания), дата установки, цена за единицу ПО. При этом также необходимо фиксировать информацию об организации, продавшей программное обеспечение – название, адрес, контактные телефоны, адрес сайта.

Необходимо осуществлять следующую обработку данных:

- на заданную дату список подразделений, на компьютерах которых установлено не лицензионное ПО;
- список лицензионного ПО, количество лицензий на это ПО (по убыванию) на заданную дату;
- список подразделений, количество компьютеров у подразделения (по убыванию) на заданную дату.

5 АИС «Арендная плата за нежилые помещения»

Описание предметной области.

Организация (Код, Название, Краткое название, Адрес, Контактные телефоны, электронный адрес) сдает в аренду помещения. Каждое помещение характеризуется следующими показателями:

- адрес;
- площадь – кв.м.;
- площадь подвала – кв.м. (при наличии);
- коэффициент подвала – значение от 0 до 1;
- коэффициент технического обустройства помещения (КТ) – значение от 1 до 2.

Арендная плата зависит от базовой ставки за 1 кв.м. (в рублях), которая утверждается документом (Номер, Дата) агентства Госкомимущества России.

Формула расчета месячной арендной платы (МАП):

МАП = (базовая ставка/12 * площадь помещения + базовая ставка/12 * площадь подвала * коэффициент подвала) * КТ.

При изменении базовой ставки МАП изменяется со следующего месяца после даты изменения ставки. Оплата производится ежемесячно.

Договор об аренде может заключаться как с организациями (Юридическими лицами), так и с физическими лицами. В договоре об аренде помещения, имеющего номер, дату фиксируется дата начала аренды, дата заключения аренды. Для юридического лица в БД заносятся название, адрес, ИНН, номер и дата лицензии о деятельности. Для физического лица – ФИО, паспортные данные (Серия, Номер, Дата выдачи, Кем выдан), ИНН и адрес.

Необходимо осуществлять следующую обработку данных:

- итоговая сумма оплат за текущий месяц (на заданную дату);
- список арендаторов (тип, название, адрес и другие характеристики арендуемого помещения) на текущую дату;
- список помещений, не сданных в аренду на текущую дату.

6 АИС «Списание основных средств»

Описание предметной области.

Основные средства - это имущество организации, предприятия со сроком полезного использования. На предприятии (Код, Название, Краткое название) имеется перечень основных средств разного типа (мебель, вычислительная техника, оборудование, инструменты и т.п.), закрепленных за подразделениями предприятия. Закрепление осуществляется на основании определенного документа, имеющего номер, дату, в нем указан срок закрепления средства за подразделением. При списании имущества предприятия создается комиссия, в которую входят руководитель предприятия, главный бухгалтер, главный инженер, главный энергетик, главный механик, руководитель подразделения, где находится средство, материально ответственный в подразделении. При списании средства формируется документ, имеющий номер, название, дату и подписи членов комиссии. В каждом документе может быть указано сразу несколько списываемых средств, для каждого указывается:

- инвентарный номер;
- название;
- принадлежностью к типу;
- дата постановки на учет в подразделении;
- плановый срок эксплуатации (год, месяц);
- балансовая стоимость (в рублях), определяемая при постановке средства на учет.

Для каждого средства также указывается дефект, ставший причиной списания (Код, Название) – износ, поломка, не имеющая восстановления, утрата и др.

Необходимо осуществлять следующую обработку данных:

- на заданную дату список (наименование) средств, закрепленных за каждым подразделением, балансовая стоимость средства;
- динамика списания средств заданного наименования (количество) за заданный период по месяцам;
- на заданную дату список комиссии по списанию.

7. АИС «Аттестация сотрудников предприятия»

Описание предметной области.

Предприятие (Код, Название, Краткое название) периодически проводит аттестацию сотрудников на соответствие ими занимаемой должности. Каждый сотрудник за время работы может проходить несколько аттестаций.

Для проведения аттестации (Дата) необходима следующая информация: ФИО сотрудника, дата рождения, место работы (Код, Название, Краткое название) подразделения, занимаемая должность (Код, Название, Краткое название), ставка, дата начала работы, дата окончания работы контракта), название, номер и дата приказа о назначении на должность. Необходимы также следующие сведения:

- сведения об образовании – какое заведение окончил, документ об образовании, квалификация по образованию (инженер, учитель, экономист);
- дата начала трудового стажа;
- дата начала стажа по специальности;
- сведения о повышении квалификации – в каком заведении проходил, дата начала, дата окончания прохождения.

У каждого сотрудника может быть несколько документов об образовании и повышении квалификации.

Каждому аттестуемому могут задать несколько вопросов, необходимо хранить количество заданных вопросов и количество правильных ответов. Также необходимо хранить оценку деятельности работника – соответствует или не соответствует занимаемой должности.

Каждую аттестацию проводит комиссия, необходимо фиксировать ФИО, место работы и должность члена комиссии. Максимальное число – 5 человек.

Необходимо осуществлять следующую обработку данных:

- на заданную дату список сотрудников (ФИО, место работы), не прошедших аттестацию – не соответствующих занимаемой должности;
- на заданную дату количество сотрудников, работающих на предприятии в заданной должности;
- список учебных заведений, предприятий, их адреса, на которых сотрудники предприятия повышали свою квалификацию.

8. АИС «Трудоустройство»

Описание предметной области.

Организация (Код, Название, Краткое название Адрес, Контактные телефоны, электронный адрес) предоставляет услуги по трудоустройству. Организацией ведется банк данных о существующих вакансиях. По каждой вакансии поддерживается следующая информация:

- предприятие (Код, Название, Краткое название Адрес, Контактные телефоны, электронный адрес);
- название вакансии (должность);
- требования к соискателю: пол, возраст (Верхняя граница, Нижняя граница), образование (высшее, среднее, не имеет значение и т.п.), знание определенных видов деятельности (выбор из перечня - знание электронного документооборота, определенных прикладных программ и т.п.), коммуникабельность (да, нет);
- обязанности (выбор из перечня – заключение договоров, распространение агитационного материала, работа с клиентами и т.п.);
- предполагаемая оплата (Нижняя граница, Верхняя граница), единицы измерения оплаты - рубли;
- оформление трудовой книжки (да, нет);
- наличие социального пакета (да, нет);
- срок начала открытия вакансии;
- срок закрытия вакансии (вакансия занята).

Необходимо осуществлять следующую обработку данных:

- на заданную дату список предприятий, имеющих вакансии по заданной должности;
- название должности, на которую за заданный период было предложено максимальное количество вакансий;
- на заданную дату список предприятий, предлагающих вакансии, не требующих образования.

9. АИС «Спортивные сооружения области»

Описание предметной области.

Областная организация (Код, Название, Краткое название Адрес, Контактные телефоны, электронный адрес) ведет и предоставляет на сайте информацию о спортивных сооружениях области. По каждому сооружению ведется информация:

- место – населенный пункт, городского или сельского типа, адрес;
- номер, название, краткое название;
- тип сооружения (игровые виды спорта, легкоатлетический манеж, каток, ипподром и др.);
- площадь спортивной арены, кв.м.;
- вместимость зрителей, чел., тыс. чел.;
- организация (Код, Название, Краткое название Адрес, Контактные телефоны, электронный адрес), принявшая сооружение на баланс;
- дата принятия на баланс.

Каждое сооружение за время функционирования может находиться на балансе у разных организаций в разные периоды времени.

Необходимо также фиксировать мероприятия, проводимые в спортивных сооружениях:

- тип мероприятия – тренировочный процесс, соревнования, сдача в аренду, концерт и т.п.;
- название мероприятия;
- дата начала, дата окончания мероприятия;
- количество человек, посетивших мероприятие.

Необходимо осуществлять следующую обработку данных:

- на заданную дату список спортивных сооружений заданного типа;
- за заданный период динамика занятости спортивного сооружения в мероприятиях заданного типа по месяцам;
- на заданную дату список организаций, на балансе у которых находятся спортивные сооружения, их количество.

10 АИС «Справочник предприятия»

Описание предметной области.

Для формирования контактов организации, имеющей большой контингент клиентов, и представления их на сайте, необходимо хранить следующую информацию:

- код, название краткое название предприятия, каждого его подразделения, взаимодействующего с клиентами;
- вид деятельности предприятия, подразделения – работа с абонентами, изготовление продукции; изучение рынка спроса; IT-подразделение и др.;
- местоположение предприятия, подразделения – адрес, вплоть до номера комнаты. Местоположение может меняться, необходимо отслеживать все данные, для этого фиксируется дата начала закрепления адреса за предприятием, подразделением;
- контактные телефоны – их может быть несколько, и они могут меняться, необходимо хранить историю закрепления телефонов;
- электронный адрес предприятия. Подразделения;
- ФИО, должность руководителя. Руководители также могут меняться, необходимо отслеживать историю их изменения и поддерживать исторические данные.

Необходимо осуществлять следующую обработку данных:

- на заданную дату список контактных телефонов подразделений предприятия;
- на заданную дату количество подразделений, не имеющих электронные адреса;
- название подразделения, у которого за заданный период сменилось наибольшее число руководителей.

11 АИС «Паспорт здоровья сотрудника»

Описание предметной области.

Организация придает большое значение здоровью сотрудников и имеет в своей структуре подразделение, занимающееся профилактикой здоровья сотрудников. Для учета состояния здоровья для каждого сотрудника ведется «Паспорт здоровья», в котором сохраняется следующая информация:

- ФИО сотрудника, пол, дата рождения;
- образование (высшее, среднее, без образования). Если человек за время работы на предприятии повышал своё образование – необходимо фиксировать все соответствующие данные, привязывая их к дате получения соответствующего документа;
- история всех перемещений сотрудника на предприятии – подразделение, должность, категория должности (инженерно-технический работник, рабочий, управленческий персонал, IT-специалист и др.), должность, ставка, дата начала работы, дата окончания;
- история семейного положения – состояние (холост, в браке, разведен и др.), дата начала семейной жизни, дата окончания;
- история антропологических измерений – на дату – рост, вес;
- история прививок – дата, название прививки;
- история заболеваний – название, дата постановки на учет, дата снятия с учета.

Необходимо осуществлять следующую обработку данных:

- на заданную дату название заболевания, зафиксированного у сотрудников за все время наблюдения максимальное число раз;
- на заданный период список сотрудников, не сделавших прививку заданного вида;
- за заданный период динамика количества заболеваний в организации – по месяцам, количество заболевших с высшим, средним образованием и без образования.

12 АИС «Справочник абитуриента»

Описание предметной области.

Высшее учебное заведения для предоставления на сайте информации абитуриентам ведет банк данных со следующей информацией:

- список специальностей (Код, Название, Краткое название), на которых осуществляется обучение в вузе. Специальности привязаны к учебным подразделениям – факультетам, кафедрам (Код, Название, Краткое название), и распределены по формам обучения (очная, очно-заочная, заочная);
- адрес учебных подразделений;
- телефоны учебных подразделений;
- если есть – адрес сайта учебного подразделения;
- ФИО, ученая степень, ученое звание руководителя учебного заведения (декан факультета, заведующий кафедрой). При этом необходимо вести историю всех руководителей – дата начала работы, дата окончания;
- по каждой форме обучения:
    - план приема на специальность на каждый год;
    - перечень предметов, по которым необходимо сдавать вступительные экзамены (ЕГЭ);
    - проходной балл на специальность по годам с разбивкой по предметам.

Необходимо осуществлять следующую обработку данных:

- на заданный год – список специальностей заданной формы обучения и планы приема;
- на заданный год наименование специальности, на которую был максимальный проходной балл по математике;
- на заданный год список руководителей учебных подразделений, имеющих ученую степень «доктор наук» и ученое звание «профессор».

13 АИС «Платные образовательные услуги населению»

Описание предметной области.

Организация (Код, Название, Краткое название) оказывает платные образовательные услуги населению. Услуги оказываются в виде проведения курсов обучения, по которым необходимо хранить следующую информацию:

- тип проведения – групповые, индивидуальные;
- вид проведения – очные, заочные;
- дата начала, дата окончания курсов;
- срок обучения (дни, месяцы, годы);
- количество часов обучения;
- на базе какого образования (среднее, высшее);
- темы, входящие в курс, для каждой темы:
  название;
  количество часов;
- время проведения занятий – дни недели, часы;
- вид выпускного контроля (квалификационная работа, экзамен, собеседование и прочее);
- вид выдаваемого документа (документ государственного образца, документ установленного образца);
- стоимость обучения

Для предоставления информации на сайте необходимо хранить адрес организации, контактные телефоны, электронный адрес, адрес сайта, серия, номер и вид документа о предоставлении образовательных услуг.

Необходимо осуществлять следующую обработку данных:

- список курсов, на которых можно прослушать заданную темы, например, «1С Бухгалтерия»;
- список курсов, на которых можно пройти заочное обучение и имеющих минимальную стоимость;
- список самых длительных курсов.

14 АИС «Новостная лента организации»

Описание предметной области.

Для предоставления новостных событий организации на её сайте необходимо вести следующие данные:

- название, краткое название организации, контактные телефоны, адрес, электронный адрес, адрес сайта;
- название и координаты подразделений организации, информация о которых будет предоставляться на сайте;
- список работающих сотрудников подразделений организации. Которым предоставляется возможность размещать информацию на сайте – ФИО, подразделение, должность, логин, пароль. При изменении статуса сотрудника – увольнение, перевод – информация должна соответствующим образом изменяться, например, сотрудник переводится в статус неработающего, логин и пароль должны быть заблокированы;
- описание новостной информации, размещаемой на сайте:

  тип (новость, объявление, сообщение и др.);

  название информации;

  дата создания;

  текст;

  дата размещения;

  дата перевода информации в архив;

  размер информации в Кб;

  наличие прикрепляемых к информации файлов – для каждого название, размер, тип, краткое описание;

  ответственный за информацию – сотрудник подразделения, имеющий соответствующий доступ.

Необходимо осуществлять следующую обработку данных:

- на заданную дату список ответственных за информацию на сайте от подразделений, не имеющих логин и пароль;
- на заданную дату название информации, размещенной на сайте (не в архиве) и имеющей самый большой размер.
- динамика предоставления информации для сайта заданным подразделением за заданный период – количество по месяцам.

15 АИС «Анализ продаж»

Описание предметной области.

Магазин (Код, Название, Краткое название) ведет учет продаж товаров и анализ работы с постоянными клиентами. Каждая единица товара учитывается при поступлении в магазин из накладной (Номер, Дата накладной), которая может иметь несколько позиций. В каждой позиции есть её номер, наименование товара, количество единиц поступившего товара, единица измерения, цена за единицу. Товары учитываются по виду - одежда, кожгалантерея, чулочно-носочные изделия, обувь и т.п. Каждый товар также имеет определенный артикул.

Ведет учет и продаж товаров – фиксируется дата продажи конкретного товара, количество проданных единиц.

Магазин ведет учет постоянных клиентов – фиксируется ФИО клиента, его паспортные данные (Серия, Номер, Дата выдачи, Кем выдан), дата рождения, контактный телефон. Покупателю, сделавшему покупку на сумму свыше 3000 тыс. рублей выдается дисконтная карта, имеющая 5-ти значный номер. Карта дает покупателю скидку 3%. При накоплении сумм покупок покупателем более чем на 10000 тыс. рублей, процент скидки увеличивается до 5%, более 20000 – максимальный процент скидки достигает размера 10%.

Необходимо осуществлять следующую обработку данных:

- на заданную дату количество и список покупателей (ФИО, контактный телефон), имеющих 10% скидку;
- за заданный период - динамика продажи заданного товара – количество по месяцам – поступление/ продажа;
- на заданную дату список покупателей (ФИО, контактный телефон), у которых в ближайшие 10 дней будет день рождения.

16 АИС «Электронный реестр помещений»

Описание предметной области.

Предприятие (Код, Название, Краткое название) имеет иерархическую организационную структуру, отражающая подчиненность большого количества подразделений. Для каждого подразделения необходимо хранить:

- код, полное название, краткое название;
- родительные и дательные падежи названий для автоматизированного формирования ряда документов и отчетов.

Каждое подразделение может занимать несколько помещений. Каждое помещение имеет номер, в который входит номер корпуса (предприятие может иметь много зданий – 1 или 2 цифры) и номер этажа – 1 или 2 цифры. В пределах одного этажа каждое помещение имеет свой номер 1 или 3 цифры. Помещение относится к определенному типу, о котором также необходимо иметь сведения, например, кабинет руководителя, приемная руководителя, лаборатория, цех, столовая и т.п. Необходимо также хранить данные о площади каждого помещения (кв. м).

Закрепление помещений за подразделениями может изменяться. Это осуществляется на основе определенного документа, имеющего название (приказ, распоряжение) и дату. В каждом документе м.б. несколько позиций, отображающих следующую информацию: номер позиции документа; действие, осуществляемое с помещением (передать, закрепить) дата действия; название подразделения; перечень помещений, возможное наименование другого подразделения. Например – «передать с 20.06.2007 г. отделу № 3 лабораторные помещения 14105 и 14106, закрепленные за лабораторией № 5»; «закрепить за медпунктом с 15.09.2007 г. складской помещение 3109» .

Необходимо осуществлять следующую обработку данных:

- на заданную дату список подразделений предприятия (наименование) и перечень занимаемых им помещений – номер, тип;
- список, отображающий иерархию (дерево) подчинения подразделений предприятия;
- динамика изменения количества площадей помещений у заданного подразделения за заданный период – количество по годам.

17 АИС «Скорая помощь»

Описание предметной области.

Лечебное учреждение (Код, Название, Краткое название, Адрес, Контактные телефоны) оказывает скорую медицинскую помощь населению. В учреждении имеется штат сотрудников, о которых необходимо хранит следующие сведения:

- табельный номер;
- ФИО; дата рождения, пол;
- должность, дата начала работы в данной должности, дата окончания, ставка.

Работа в учреждении круглосуточная – сотрудники работают по 24 часа с последующими выходными днями. Необходимо знать, в какой смене и бригаде работает тот или иной сотрудник. Закрепление в бригаду осуществляется на основании внутреннего приказа, имеющего номер и дату. В каждой позиции приказа указывается, что конкретный сотрудник с даты 1 по дату 2 работает в бригаде с заданным номером.

Необходимо вести учет сведений о выездах бригад на вызовы. Каждый вызов определяется датой, временем выезда и адресом. Пациент, которому оказывается помощь, может быть описан следующими данными ФИО, возраст (примерный), первоначальный диагноз. Необходимо также знать ФИО и должности сотрудников выехавшей на вызов бригады (включая водителя и диспетчера). Необходимо также хранить небольшое текстовое описание принятых бригадой мер. Если больной был госпитализирован, либо получил направление на госпитализацию, также необходимо знать в какое учреждение он был направлен (название, адрес). При возвращении бригады фиксируется время прибытия.

Необходимо осуществлять следующую обработку данных:

- на заданную дату список выездов всех бригад учреждения (номер выезда, время, номер бригады, принятые меры);
- на заданную дату описание самого длительного выезда;
- на заданную дату список заданной бригады (табельный номер, ФИО, должность).