Тема №3. Архиваторы

Цель: рассмотреть общее назначение программ архивации данных, базовые алгоритмы архивации, виды арживаторов, команды и опции архиваторов, характеристики, способы защиты архивов, способы автоматической установки архивации данных.

План:

1. Общее представление об архиваторах.
2. Алгоритмы архивации.

**1 Общее представление об архиваторах**

Одним из наиболее широко распространенных видов сервисных программ являются про­граммы, предназначенные для архивации, упаковки файлов путем сжатия хранимой в них информации.

Назначение. Программы-упаковщики позволяют за счёт применения специальных методов «упаковки» информации сжимать информацию на дисках, т.е. создавать копии файлов меньшего размера, а также объединять копии нескольких файлов в один архивный файл. Применение программ-упаковщиков очень полезно:

- при передаче информации по телефонным линиям (через модем);

- при создании дистрибутивов программных комплексов;

- при создании архива файлов.

Возможности программ-упаковщиков. Программы для упаковки (архивации) файлов позволяют помещать копии файлов на диск в сжатом виде в архивный файл, извлекать файлы из архива, просматривать оглавление архива и т.д. Разные программы отличаются форматом архивных файлов, скоростью работы, степенью сжатия файлов при помещении в архив, удобством использования.

**Сжатие информации** — это процесс преобразования информации, храня­щейся в файле, к виду, при котором уменьшается избыточность в ее пред­ставлении и соответственно требуется меньший объем памяти для хранения.

Сжатие информации в файлах производится за счет устранения избыточности различ­ными способами, например за счет упрощения кодов, исключения из них постоянных ботов или представления повторяющихся символов или повторяющейся последовательности сим­волов в виде коэффициента повторения и соответствующих символов. Применяются раз­личные алгоритмы подобного сжатия информации.

Сжиматься могут как один, так и несколько файлов, которые в сжатом виде помеща­ются в так называемый архивный файл или архив.

**Архивный файл** — это специальным образом организованный файл, со­держащий в себе один или несколько файлов в сжатом или несжатом виден служебную информацию об именах файлов, дате и времени их создания или модификации, размерах и т.п.

Целью упаковки файлов обычно являются обеспечение более компактного размеще­ния информации на диске, сокращение времени и соответственно стоимости передачи ин­формации по каналам связи в компьютерных сетях. Кроме того, упаковка в один архивный файл группы файлов существенно упрощает их перенос с одного компьютера на другой, со­кращает время копирования файлов на диски, позволяет защитить информацию от несан­кционированного доступа, способствует защите от заражения компьютерными вирусами.

*С тепень сжаm и я файлов* характеризуется коэффициентом *Кс,* определяе­мым как отношение объема сжатого файла *Vc* к объему исходного файла *Vо*, выраженное в процентах:

*Kc=(Vc / V0)\**100%

Степень сжатия зависит от используемой программы, метода сжатия и типа исходного файла. Наиболее хорошо сжимаются файлы графических образов, текстовые файлы и файлы данных, для которых степень сжатия может достигать 5 - 40%, меньше сжимаются файлы исполняемых программ и загрузочных модулей — 60 - 90%. Почти не сжимаются архивные файлы. Программы для архивации отличаются используемыми методами сжатия, что соответственно влияет на степень сжатия.

**Архивация (упаковка)** — помещение (загрузка) исходных файлов в архив­ный файл в сжатом или несжатом виде.

**Разархивация (распаковка)** — процесс восстановления файлов из архива точно в таком виде, какой они имели до загрузки в архив. При распаковке файлы извлекаются из архива и помешаются на диск или в оперативную память.

Программы, осуществляющие упаковку и распаковку файлов, называются *про­граммами-архиваторами.*

Большие по объему архивные файлы могут быть размещены на нескольких дисках (томах). Такие архивы называются *многотомными.* Том — это составная часть многотом­ного архива. Создавая архив из нескольких частей, можно записать его части на несколько частей.

**Виды программ-архиваторов**

В настоящее время применяется несколько десятков программ-архиваторов, которые отли­чаются перечнем функций и параметрами работы, однако лучшие из них имеют примерно одинаковые характеристики. Из числа наиболее популярных программ можно выделить: ARJ, РКРАК, LHA, ICE, HYPER, ZIP, РАК, ZOO, EXPAND, разработанные за рубежом, а также AIN и RAR, разработанные в России. Обычно упаковка и распаковка файлов выпол­няются одной и той же программой, но в некоторых случаях это осуществляется разными программами, например, программа PKZIP производит упаковку файлов, a PKUNZIP — распаковку файлов.

Программы-архиваторы позволяют создавать и такие архивы, для извлечения из кото­рых содержащихся в них файлов не требуются какие-либо программы, так как сами архив­ные файлы могут содержать программу распаковки. Такие архивные файлы называются самораспаковывающимися.

**Самораспаковывающийся архивный файл —** это загрузочный, испол­няемый модуль который способен к самостоятельной разархивации нахо­дящихся в нем файлов без использования программы-архиватора.

Самораспаковывающийся архив получил название SFX-архив (SelF-eXtracting). Архи­вы такого типа в MS DOS обычно создаются в форме ЕХЕ-файла.

Многие программы-архиваторы производят распаковку файлов, выгружая их на диск, но имеются и такие, которые предназначены для создания упакованного исполняемого мо­дуля (программы). В результате такой упаковки создается программный файл с теми же именем и расширением, который при загрузке в оперативную память самораспаковывается и сразу запускается. Вместе с тем возможно и обратное преобразование программного файла в распакованный формат. К числу таких архиваторов относятся программы PKLITE, LZEXE, UNP..

Программа EXPAND, входящая в состав утилит операционной системы MS DOS и оболочки Windows, применяется для распаковки файлов программных продуктов, постав­ляемых фирмой Microsoft.

Программы-архиваторы RAR и AIN, кроме обычного режима сжатия, имеют режим solid, в котором создаются архивы с повышенной степенью сжатия и особой структурой ор­ганизации. В таких архивах все файлы сжимаются как один поток данных, т.е. областью по­иска повторяющихся последовательностей символов является вся совокупность файлов, загруженных в архив, и поэтому распаковка каждого файла, если он не первый, связана с обработкой других. Архивы такого типа предпочтительнее использовать для архивирования большого числа однотипных файлов.

**Способы управления программой-архиватором**

Управление программой-архиватором осуществляется одним из двух способов:

* с помощью командной строки MS DOS, в которой формируется команда запуска, со­держащая имя программы-архиватора, команду управления и ключи ее настройки, а также имена архивного и исходного файлов; подобное управление характерно для ар­хиваторов ARJ, AIN, ZIP, РАК, LHA и др.;
* с помощью встроенной оболочки и диалоговых панелей, появляющихся после запуска программы и позволяющих вести управление с использованием меню и функциональ­ных клавиш, что создает для пользователя более комфортные условия работы. Такое управление имеет программа-архиватор RAR.

 Архивный файл (архив) представляет собой набор из одного или нескольких файлов, помещенных в сжатом виде в единый файл, из которого их можно при необходимости извлечь в первоначальном виде. Архив содержит оглавление, позволяющее узнать, какие файлы в нем содержатся. В оглавлении храниться следующая информация для каждого содержащегося в архиве файла:

- имя файла;

- сведения о каталоге, в котором содержится файл;

- дата и время последней модификации файла;

- размер файла на диске и в архиве;

-код циклического контроля для каждого файла, используемый для проверки целостности архива.

## Метаданные

Архив почти всегда содержит метаданные. Например:

* Имена файлов (кроме некоторых программ сжатия одиночных файлов — например, gzip, где в качестве имени файла используется имя архива без расширения, добавленного такой программой)
* Идентификаторы владельцев, групп, и т. п. файлов, и их прав
* Размеры файлов
* Даты изменения, создания файлов
* Контрольные суммы файлов для проверки правильности распаковки
* Размер и контрольные суммы архива
* Избыточные данные для восстановления данных при повреждении
* Цифровая подпись создателя архива

Архиваторы для DOS (ARJ, PKZIP, LH, RAR) достигли многого. Плот­ность.

**Классификация архиваторов**

Архиваторы делятся на упаковщики и компрессоры. Программы-упаковщики создают один большой файл из нескольких файлов и каталогов, а компрессоры сжимают этот файл в размерах. В операционных системах MS-DOS и Windows получили распространение универсальные упаковщики-компрессоры, в которых обе функции осуществляются в одной программе.

Для архивации и разархивации может использоваться как одна (преимущественно), так и две разные программы. Это условие указывается ниже особо.

Изначально появились архиваторы командной строки. Их общий индекс следующий:

- <архиватор> <команды> [{<опции>}] <файл\_архива> {<список\_файлов>}

Как правило, набирать и запоминать такие конструкции сложно, поэтому появились оболочки с графическим интерфейсом, отображающие архивные файлы в виде директорий. Такие оболочки встраиваются в менеджеры программ и файлов. К архивным оболочкам относятся: Shez, WinRar, WinZip, PowerArchiver и т.д. Но для правильного использования оболочки к ней все равно должен подключаться архиватор командной строки.

**Команды и опции архиватора**

Синтаксис вызова программ-архиваторов из командной строки операционной системы:

[<путь\_к\_архиватору>\]<архиватор> <команда> [{<опции>}]

<имя\_файла\_архива>[.<его\_расширение>] {{<список\_файлов>}]

где:

\* <путь\_к\_архиватору> -- полный путь к папке, содержащей программу-архиватор;

\* <архиватор> -- имя файла-архиватора;

\* <команда> -- команда архиватору, задающая действие, которое он должен выполнить;

\* <опции> -- одна или несколько необязательных опций, дополняющих команду архиватору;

\* <имя\_файла\_архива> -- обязательный параметр, определяющий файл-архив, с которым работает пользователь;

\* <его\_расширение> -- необязательный параметр, указывающий расширение архивного файла. Если оно пропущено, задается расширение по-умолчанию;

\* необязательные параметры, задающие имена файлов, подлежащих архивации-разархивации. Если параметр пропущен, архиватор сам определяет, какие файлы ему задействовать. Вместо имен файлов можно использовать шаблоны.

**Архиваторы командной строки MS-DOS**

Программы-архиваторы под DOS позволяют, за счет применения специальных методов «упаковки» информации на диске, создавать копии файлов меньшего размера и объединять несколько файлов в один файл-архив. Они работают по принципу: «Все в одном», то есть одновременно и упаковывают файлы в архив, и сжимают их.

Архиваторы командной строки DOS работают и на системах Windows, однако при этом не сохраняется информация о длинных именах файлов. Для сохранения этой информации используйте версии архиваторов для платформы Win32. Но при этом Вы не сможете разархивировать архивы в MS-DOS!

Среди архиваторов командной строки DOS наибольшее распространение получили архиваторы PkZip/PkUnzip (создают и распаковывают архивы с расширением \*.ZIP), Arj (с расширением \*.ARJ для основного архива и \*.A01, \*.A02 для последующих томов) и Rar (с расширением \*.RAR для основного архива и \*.R00, \*.R01 для последующих томов).

**2 Алгоритмы архивации**

**1. Сжатие способом кодирования серий (RLE)**

Наиболее известный простой подход и алгоритм сжатия информации обратимым путем - это кодирование серий последовательностей (Run Length Encoding - RLE).

Суть методов данного подхода состоит в замене цепочек или серий повторяющихся байтов или их последовательностей на один кодирующий байт и счетчик числа их повторений.

Например:

44 44 44 11 11 11 11 11 01 33 FF 22 22 - исходная последовательность

03 44 05 11 00 03 01 03 FF 02 22 - сжатая последовательность

Первый байт указывает сколько раз нужно повторить следующий байт. Если первый байт равен 00, то затем идет счетчик, показывающий сколько за ним следует неповторяющихся данных.

Данные методы, как правило, достаточно эффективны для сжатия растровых графических изображений (BMP, PCX, TIF, GIF), т.к. последние содержат достаточно много длинных серий повторяющихся последовательностей байтов.

Недостатком метода RLE является достаточно низкая степень сжатия.

**2. Алгоритм Хаффмана**

Сжимая файл по алгоритму Хаффмана первое что мы должны сделать - это необходимо прочитать файл полностью и подсчитать сколько раз встречается каждый символ из расширенного набора ASCII.

Если мы будем учитывать все 256 символов, то для нас не будет разницы в сжатии текстового и EXE файла.

После подсчета частоты вхождения каждого символа, необходимо просмотреть таблицу кодов ASCII и сформировать бинарное дерево.

**3. Арифметическое кодирование**

Совершенно иное решение предлагает т.н. арифметическое кодирование. Арифметическое кодирование является методом, позволяющим упаковывать символы входного алфавита без потерь при условии, что известно распределение частот этих символов и является наиболее оптимальным, т.к. достигается теоретическая граница степени сжатия.

Предполагаемая требуемая последовательность символов, при сжатии методом арифметического кодирования рассматривается как некоторая двоичная дробь из интервала [0, 1). Результат сжатия представляется как последовательность двоичных цифр из записи этой дроби.

Идея метода состоит в следующем: исходный текст рассматривается как запись этой дроби, где каждый входной символ является "цифрой" с весом, пропорциональным вероятности его появления. Этим объясняется интервал, соответствующий минимальной и максимальной вероятностям появления символа в потоке.

**4. Алгоритм Лемпеля-Зива-Велча (Lempel-Ziv-Welch - LZW)**

Данный алгоритм отличают высокая скорость работы как при упаковке, так и при распаковке, достаточно скромные требования к памяти и простая аппаратная реализация.

Недостаток - низкая степень сжатия по сравнению со схемой двухступенчатого кодирования.

Предположим, что у нас имеется словарь, хранящий строки текста и содержащий порядка от 2-х до 8-ми тысяч пронумерованных гнезд. Запишем в первые 256 гнезд строки, состоящие из одного символа, номер которого равен номеру гнезда.

Алгоритм просматривает входной поток, разбивая его на подстроки и добавляя новые гнезда в конец словаря. Прочитаем несколько символов в строку s и найдем в словаре строку t - самый длинный префикс s.

Пусть он найден в гнезде с номером n. Выведем число n в выходной поток, переместим указатель входного потока на length(t) символов вперед и добавим в словарь новое гнездо, содержащее строку t+c, где с - очередной символ на входе (сразу после t). Алгоритм преобразует поток символов на входе в поток индексов ячеек словаря на выходе.

При практической реализации этого алгоритма следует учесть, что любое гнездо словаря, кроме самых первых, содержащих одно-символьные цепочки, хранит копию некоторого другого гнезда, к которой в конец приписан один символ. Вследствие этого можно обойтись простой списочной структурой с одной связью.

Пример: ABCABCABCABCABCABC - 1 2 3 4 6 5 7 7 7

|  |  |
| --- | --- |
| 1 | A |
| 2 | B |
| 3 | C |
| 4 | AB |
| 5 | BC |
| 6 | CA |
| 7 | ABC |
| 8 | CAB |
| 9 | BCA |

**5. Двухступенчатое кодирование. Алгоритм Лемпеля-Зива**

Гораздо большей степени сжатия можно добиться при выделении из входного потока повторяющихся цепочек - блоков, и кодирования ссылок на эти цепочки с построением хеш таблиц от первого до n-го уровня.

Метод, о котором и пойдет речь, принадлежит Лемпелю и Зиву и обычно называется LZ-compression.

Суть его состоит в следующем: упаковщик постоянно хранит некоторое количество последних обработанных символов в буфере. По мере обработки входного потока вновь поступившие символы попадают в конец буфера, сдвигая предшествующие символы и вытесняя самые старые.

Размеры этого буфера, называемого также скользящим словарем (sliding dictionary), варьируются в разных реализациях кодирующих систем.

Экспериментальным путем установлено, что программа LHarc использует 4-килобайтный буфер, LHA и PKZIP - 8-ми, а ARJ - 16-килобайтный.

Затем, после построения хеш таблиц алгоритм выделяет (путем поиска в словаре) самую длинную начальную подстроку входного потока, совпадающую с одной из подстрок в словаре, и выдает на выход пару (length, distance), где length - длина найденной в словаре подстроки, а distance - расстояние от нее до входной подстроки (то есть фактически индекс подстроки в буфере, вычтенный из его размера).

В случае, если такая подстрока не найдена, в выходной поток просто копируется очередной символ входного потока.

**Контрольные вопросы:**

1. На каке группы делятся программы-архиваторы?
2. В чем принципиальное отличие данных групп?
3. В каких сферах полезно применение программ-упаковщиков?
4. Что называют процессом сжатия информации?
5. Чем характеризуется степень сжатия информации?
6. От чего зависит степень сжатия информации?
7. Что называют упаковкой или архивацией?
8. Что называют архивным файлом?
9. Что называют распаковкой или разархивацией?
10. Перечислите виды программархивации.
11. Дайте краткую характеричтику наиболее распространенным программам-архиватрам.
12. Какие Вы знаете алгоритмы архивации. Дайте краткую характеристику алгоритмов архивации.
13. Перечислите команды и опции архиватора.
14. Как осуществляется защита архивов?
15. Как установить автоматическую архивацию данных? На примере любой программы-архивации.

**Тема №4. Разработка моделей для написания программ (процедурных, событийных)**

Цель: ознакомить назначением, возможностями, компонентами, работой приложения MS Visio.

План:

1. Общее представление о программе Visio

2. Перечень основных тематик

3. Интерфейс Visio

4. Библиотека трафаретов

5. Меню Visio

6. Панели инструментов

7. Представление данных

8. Rational Rose - средство визуального моделирования объектно-ориентированных информационных систем

9.Этапы моделирования в Rational Rose

**1 Общее представление о прогамме Visio**

В настоящее время информационные сообщения, газеты, журналы, доклады, отчеты, учебная литература и т.п. получили новую технологию создания, тиражирования, распространения и презентации с помощью средств вычислительной техники. Поэтому актуальным становится умение специалистов оформлять полноценные электронные документы.

Для создания документов с использованием новых информационных технологий разработано уже немало различных редакторов.

Многие документы, кроме текстовой информации, часто содержат графический материал: чертежи, схемы (структурные, организационные, логические, маршрутные), фотографии, таблицы, диаграммы и т.п. Использование графики позволяет значительно повысить наглядность содержания документа и скорость его восприятия.

Различают две редакции MS Visio:

Visio Standard — средство создания различных бизнес-диаграмм, в том числе блок-схем, структурных схем, графиков предназначенных для менеджеров и бизнес-специалистов;

Visio Professional — средство моделирования и документирования бизнес-процессов, проектирования данных, построения всевозможных диаграмм, схем сетей, планов помещений, схематических чертежей, предназначенных для IT-специалистов, инженеров, технических руководителей и разработчиков программного обеспечения.

**Основные особенности Visio:**

- в Visio имеется возможность импорта изображений со сканеров и цифровых камер, получения изображений из Web, а также работы с Clip Organizer в Microsoft Office XP;

- схемы Visio можно публиковать как Web-страницы и просматривать неграфические свойства каждой фигуры внутри браузера;

- Visio содержит функции:

* настраиваемые меню,
* проверку орфографии,
* автокоррекцию текста,
* панель задач Office XP,
* при импорте схем и диаграмм Visio в Microsoft Word или PowerPoint их можно просматривать и редактировать;

- Visio поддерживает и другие продукты фирмы Microsoft, например Microsoft Project, Microsoft Excel, Exchange Server, Microsoft SQL Server, Visual Studio, а также службу каталогов Windows Active Directory;

- в Visio добавлены средства, облегчающие развертывание пакета, в частности поддержку автоматической установки и развертывания под управлением Systems Management Server (SMS).

- в Visio имеется функция самовосстановления, которая автоматически обнаруживает недостающие или поврежденные файлы и сама восстанавливает их, что упрощает техническую поддержку этого продукта;

- Visio поддерживает альтернативный формат файлов на базе XML, что позволяет корпоративным или сторонним разработчикам импортировать или экспортировать любые элементы схем Visio. Поддерживаются также надстройки и фильтры событий на основе Component Object Model (COM);

- Visio поддерживает импорт данных из CAD-приложений, таких как AutoCAD;

- средства моделирования приложений Visio поддерживают Visual Studio 6 и Visual Studio.NET, в частности Visual C++.NET и Visual Basic.NET.

В сам документ Visio можно вставить любой материал, созданный в других стандартных приложениях (текст, рисунки, фотографии, формулы, диаграммы, схемы), а затем объединять их в единую схему.

Visio позволяет с высокой скоростью создавать схемы.

Примечательно, что имеется наличие встроенной библиотеки трафаретов (stencils), ориентированной на разнообразные применения. Из библиотеки пользователь выбирает нужный трафарет, переносит из него в свой документ необходимые блоки, а затем: размножает их, масштабирует, перемещает, соединяет линиями и стрелками, снабжает надписями и т. п.

В Visio имеются средства создания авторских трафаретов, шаблонов целых документов, что дает возможность пользователю расширять библиотеки в соответствии со своими потребностями

**2 Перечень основных тематик**

Таблица - 2. Перечень основных тематик

|  |  |
| --- | --- |
| Basic | Правильные многоугольники с числом сторон от 3 до 8, круг, эллипс, овал; разнообразные по форме и направлению стрелки, в том числе криволинейные; соединители фигур. |
| Block diagram | Комплект геометрических фигур, стрелок и соединителей в большом числе вариантов, с оттенением и объемной перспективой. |
| Border | Элементы рамок и заполнителей для орнаментов (“греческий”, “египетский”, звезды, шахматная клетка, волны, зигзаг и т. п.). |
| Callout | Средства выносных пояснений: сноски, “баллоны”, ярлыки, оттиски печатей, окна, кнопки, “вспышки”. |
| Chart | Элементы диаграмм (столбчатых, объемных, секторных) и графиков, в частности координатные оси, экспоненциальные и “нормальные” кривые, вспомогательные ярлыки. |
| Clipart | Картинки: люди, средства связи, вычислительная техника, транспортные средства, чашка кофе и др. |
| Connectors | Расширенный набор соединительных элементов |
| Electrical | Компоненты электрических схем (резисторы, конденсаторы, диоды, транзисторы и т.п.), логические элементы (“И”, “ИЛИ”, “НЕ”) по стандартам IEEE. |
| Flowchart (два набора) | Элементы схем алгоритмов (все стандартизованные обозначения, внутри- и межстраничные соединители, средства разветвления и соединения потоков, базы данных). |
| Forms | Рамки для текста, шаблоны для таблиц и графиков (в частности, заготовки для логарифмической сетки), текстовые блоки для набора шрифтом Arial в 8,10 и 18 pt. |
| Maps | Группа наборов географических карт, изображающих контуры штатов США, стран, регионов и континентов, а также обозначений на картах и схемах и знаков дорожного движения |
| Network | Элементы сетей связи и компьютерных сетей: устройства ЭВМ; малые и большие ЭВМ разных типов, включая Cray; типичные сетевые структуры; антенна спутниковой связи и сам спутник |
| Office Layout | Обозначения офисного оборудования |
| Project Timeline | Средства создания проекта, графики Гантта. |
| Symbols | Денежные, дорожные, предостерегающие и информационные знаки. |

**3 Интерфейс Visio**

Основные термины в редакторе:

- Фигура (Shape) — минимальная графическая фигура, из их набора и комбинаций создаются схемы (drawing).

- Трафарет (Stencil) — набор минимальных графических фигур, объединенных по одной предметной области.

- Стиль (Style) — содержание установленных параметров объектов (размер шрифта, толщина и цвет линий, заполнение), хранится со схемой или в шаблоне документа.

- Шаблон (Template) — это особый вид документа (файла), в котором хранятся наборы стилей и установок параметров графических объектов для конкретного вида схем, по которым и создаются одного типа схемы.

После запуска Visio пользователю предоставляется автоматизированный сервис для создания, редактирования схем. Весь сервис представлен в виде меню, панелей инструментов и возможности вызова библиотеки готовых элементов будущих  схем.

В правой части его интерфейса открывается окно страницы документа с нанесенной и непечатаемой сеткой, где пользователь рисует схему.

В нижней части экрана (Status bar) отражается оперативная информация: количество страниц документа, размер и координаты фигур, угол выполняемого поворота и т. п.

В исходном положении на экране появляется лист в масштабе целой страницы. Пользователь может выбрать более удобный масштаб экранного отображения содержимого схемы.

Режим Width - показывает всю ширину страницы, Page - обеспечивает размещение в окне всей схемы, 100% — демонстрацию реального масштаба (как при печати).

**4 Библиотека трафаретов**

Одним из достоинств Visio является то, что вместе с редактором поставляется библиотека готовых элементов (трафаретов), из которых мы можем собрать схему.

 Библиотека трафаретов Visio создана в виде подборок по тематикам их возможного использования.

Меню вызова конкретной подборки трафаретов создано в двух уровнях. Сначала вызываем перечень тематик, а затем выбираем по теме набор конкретных фигур.

К библиотеке можно обратиться через меню File - Stenсils (Файл -  Трафареты) или кнопку на панели инструментов,  а затем последовательным перемещением по диалоговым окнам с выбором тематики и трафаретов   по какой- либо из них.

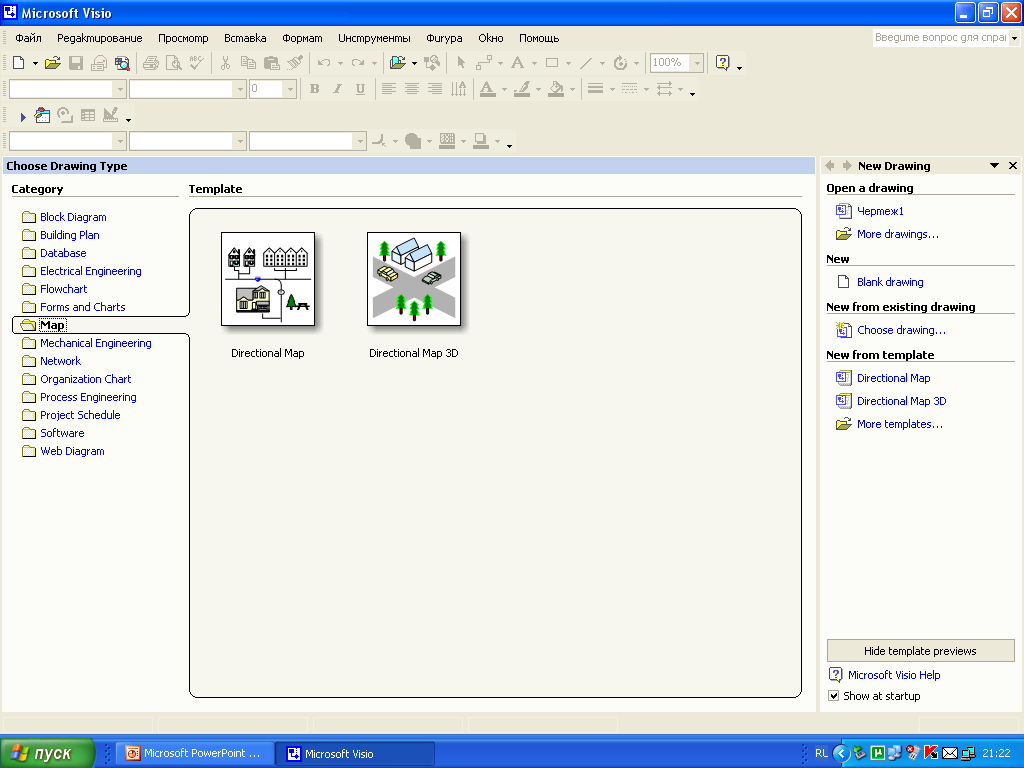


Рисунок – 29. Окно выбора трафарета

**5 Меню Visio**

Меню Visio создано в виде горизонтальной ленты с выпадающими подменю.     Чтобы использовать любой пункт подменю,  нужно сделать щелчок мышки на нем.     Технология работы с меню в этом  редакторе очень похожа на работу в Word или Excel.

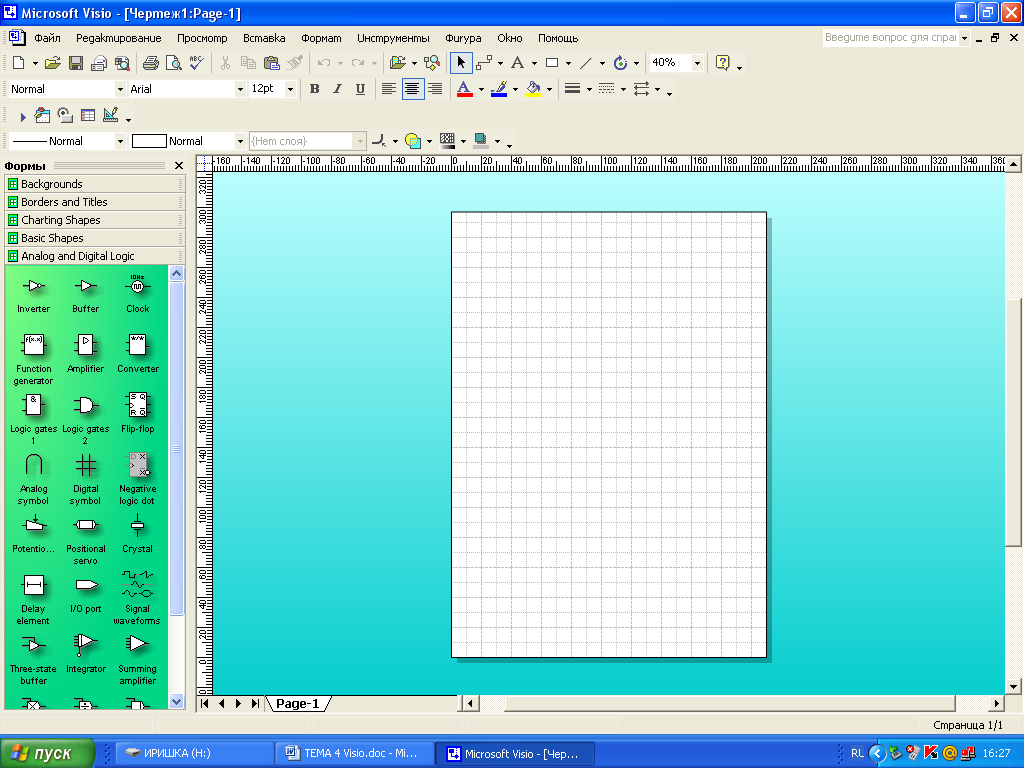
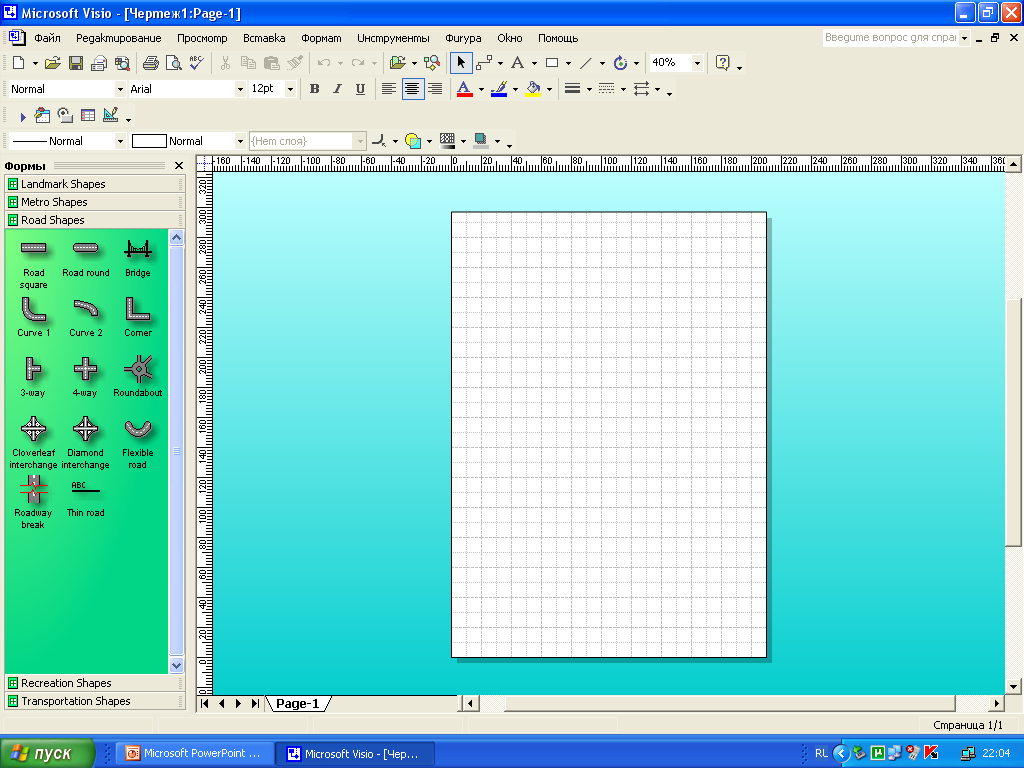
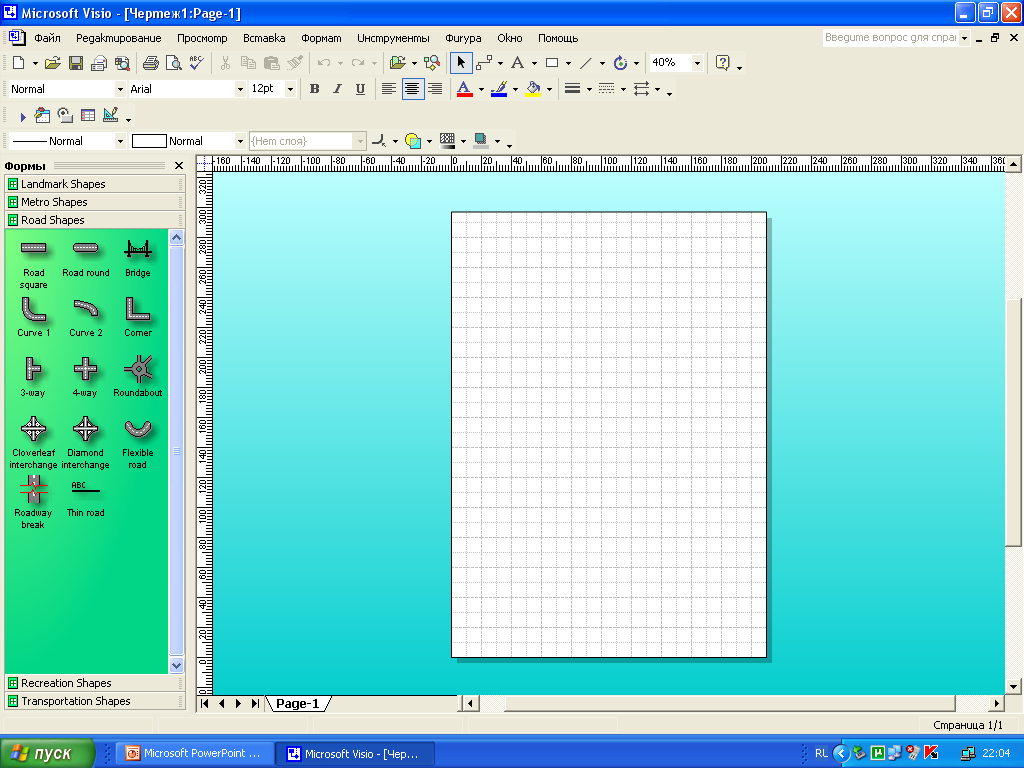


Рисунок – 30. Главное горизонтальное меню приложения

**6 Панели инструментов**

Настройка панели инструментов производится в подменю View  Toolbars (Вид  Панели инструментов).





Инструмент соединения

Открыть трафарет

Find Shape

(включает поиск)

Инструмент указателя

Рисунок – 31. Панели «Стандартная» и «Форматирования»

**Окно после выбора шаблона**

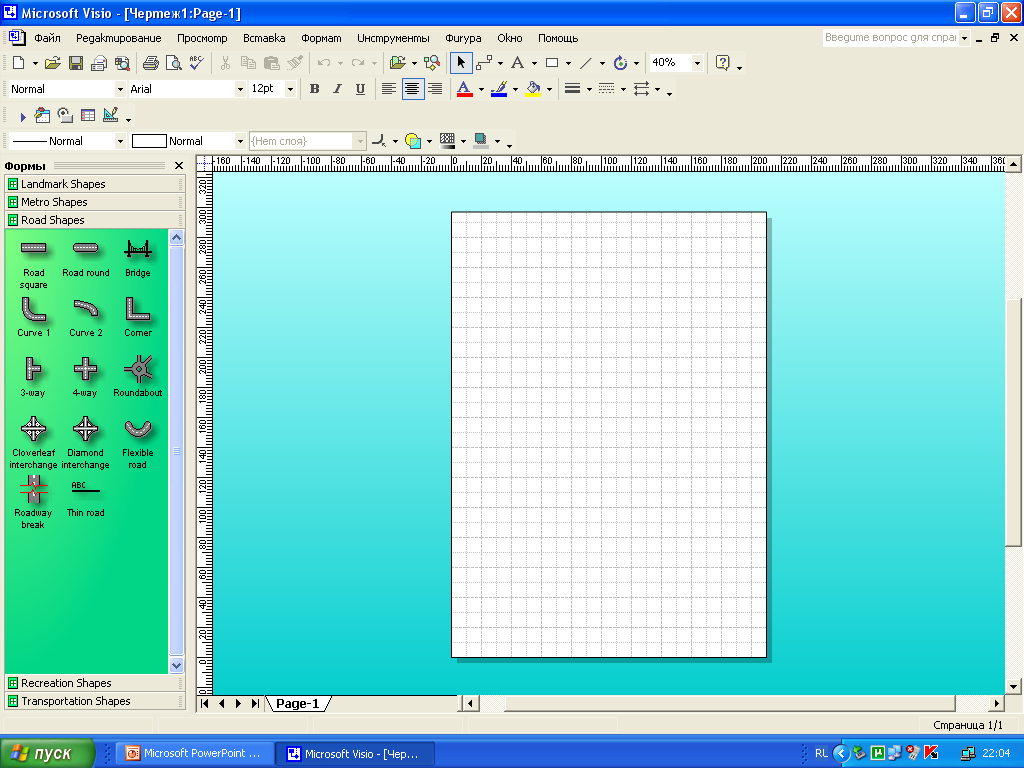


Рисунок – 32. Рабочее окно приложения

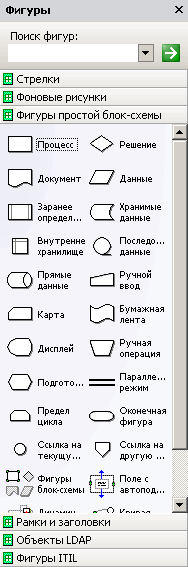


Рисунок – 33. Панель компонентов

Наиболее удобный способ начать работу с Visio - создать документ на основе шаблона. При загрузке шаблона на область задач "Фигуры" подгружаются те категории графических элементов, которые могут вам понадобиться в процессе создания диаграммы, плана или карты выбранного типа. Например, если вы хотите создать маршрутный план, то на области задач появятся такие категории фигур, как "Ориентиры", "Метро" и "Дороги".

Для добавления фигуры в проект нужно просто перетащить ее на рабочую область, после чего можно откорректировать ее размеры, задать свойства и параметры отображения.

**Настройка Visio**

После запуска Visio, как и любого программного продукта, необходимо выполнить установки глобальных параметров приложения: установить внешний вид окон, место хранения файлов системы, параметры выходных страниц документа, внешний вид отображения содержания документа на экране во время его создания, форматы и стили элементов схем и т.п.

Такие процедуры выполняются с помощью меню и вкладок. Остановимся на базовых, без которых не стоит начинать создание нового документа.

1. В главном окне с помощью меню File (Файл) или кнопок Панели инструментов выбрать один из основных режимов работы:

- создание нового документа/шаблона;

- редакция ранее созданного документа;

- создание трафаретов.

  2. По команде меню View - Toolbars (Вид) необходимо активировать вкладку (Toolbars) и установить Панели инструментов:

- стандартную (Standard);

- ввода и редактирования текста (Text);

* рисования и форматирования фигур (Shape);

-внешнего отображения и содержания страниц (View, Page).

3. По команде меню View  и его подменю необходимо установить вид окна документа и отображения в нем содержания страниц:

- масштаб отображения страницы в окне;

- сетку на странице (Grid);

- масштабные линейки (Rulers);

- отображение точек связи на фигурах схемы (Connection Point).

 4. По команде меню Tools - Options (Сервис - Параметры) необходимо активировать вкладку (Options), с помощью которой установить:

-имя пользователя, подключение сервиса проверки на наличие вирусов, условие на опрос пользователя при сохранении документа (General);

-пути доступа к системным и создаваемым файлам (File Paths);

-единицы измерения (Defalt Units - Units);

- подключение словаря и режим проверки правописания (Spelling);

-количество копий документа для восстановления поправок (Advanced - Undo Levels).

5. По команде меню File - Page Setup (Файл - начальные установки страницы) необходимо активировать вкладку (Page Setup) и с ее помощью установить основные параметры схем и страниц:

-макет и формат страниц (Page size);

-масштаб схем (Drawing Scale);

-свойство страниц (Page Properties);

- выходной формат страниц документа (Output Format);

- поля, колонтулы страниц и стиль шрифтов (Header/Footer - Choose Font).

**7 Представление данных**

Основное средство представления данных в Visio - это векторные фигуры, на основе которых строится диаграмма или план.

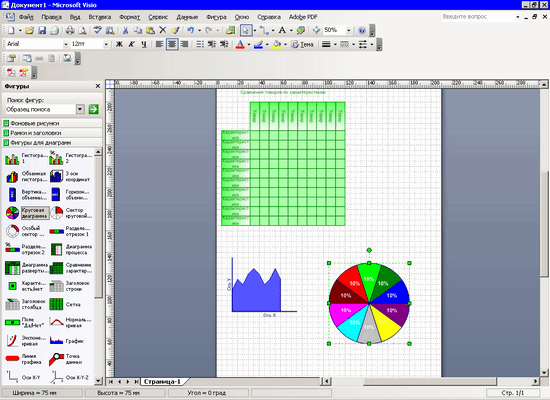


Рисунок – 34. Графическое представление данных

Фигуры являются основным, но не единственным средством для представления данных в Visio. Кроме них можно также использовать текст и числовые данные, графические элементы и форматирование цветом.

Набор надписей лучше выполнять шрифтами без засечек с кириллицей, например Arial. В одном документе следует использовать ограниченное количество шрифтов, с тем, чтобы не снижать удобство чтения и восприятия текста.

Лучше использовать стандартные шрифты Windows (Arial, Courier New, MS Sans Serif, Times New Roman) так как в противном случае при переносе файла на другой компьютер могут возникнуть трудности с отображением нестандартных или редко используемых шрифтов.

Что касается графических элементов, то они могут помочь привлечь внимание к тем или иным элементам плана или диаграммы. Такие элементы привязываются к фигурам и могут изменяться, в зависимости от указанных пользователем условий. Вы можете использовать стрелки, флажки, разноцветные светофоры и многие другие значки из набора предложенных программой. Найти применение таким средствам несложно.

**Страницы и подложkи**

Схему можно оформить на странице с подложкой или без нее (foreground / background pages). Подложки обычно содержат стандартную информацию для одной или нескольких страниц (рамку, угловой штамп, логотип организации или конкретного проекта, карту местности, план помещения и т. п.).

Каждой странице можно назначить одну подложку, в то же время, на одну подложку могут опираться несколько страниц со схемами. Подложка сама также может иметь подложку. При просмотре лицевой страницы высвечивается результат “прозрачного” наложения всех слоев. Такое оформление выводится и при печати. При передаче в другие приложения или печати подложки можно отключать.

Дополнительные страницы создаются командой New Page (Новая страница). Сразу же определяется их тип, а для лицевых страниц указывается наличие подложки; поэтому заготовку подложки необходимо выполнить первой. Любую лицевую страницу можно преобразовать в подложку с помощью меню Page (команды Page Setup или New Page).

Подложку нельзя удалить, пока существует на нее ссылка лицевой страницы. Все совместно используемые подложки и страницы должны принадлежать к одному файлу.

Единицы измерения для линеек и сетки назначаются через диалоговую вкладку в Page Setup (Установка страницы).

Частота сетки при изменении масштаба поддерживается приблизительно постоянной.

Размер схемы может быть изменен на вкладке Size - Scale (Размер - Масштаб) того же меню, если выбрать Size отличным от Same as Print Setup (Тот же, что при печати). Переформатирование лицевой страницы не затрагивает надписи и подложку.

**Основные элементы технологии**

Как вы успели заметить основную часть экрана занимают два окна: Stencils (слева, с зеленым фоном) и Drawing. Однако, чтобы разобраться с их применением, сначала нужно ознакомиться с четырьмя типами файлов, которые используются в Visio, и некоторыми другими базовыми понятиями.

**- Stencil, \*.VSS (трафарет)** - проблемно-ориентированный набор графических элементов (masters), из которых строятся все графические изображения. В состав издания Professional входят около пятидесяти готовых Stencil-файлов, а кроме того пользователь может создавать собственные наборы.

Для формирования изображений пользователь загружает один или несколько Stencils, которые располагаются в соответствующем окне в виде вкладок.

**- Master (master shape)** - отдельный элемент изображения. При перетаскивании master из окна Stencils в окно рисования создается его экземпляр, который называется shape (очертание). Master кроме графического изображения имеет целый набор свойств (графическое оформление, текст, иконка, гиперссылки и пр.). Пользователь может редактировать существующие master-компоненты или создавать новые.

**8 Rational Rose - средство визуального моделирования объектно-ориентированных информационных систем**

Работа продукта основана на языке моделирования UML (Universal Modeling Language).

Rational Rose способен решать задачи в проектировании информационных систем:

- анализ бизнес процессов;

-кодогенерация на определенном языке программирования.

Rational Rose позволяет разрабатывать как высокоуровневые, так и низкоуровневые модели, осуществляя тем самым либо абстрактное проектирование, либо логическое.

Rational Rose позволяет разрабатывать проектную документацию в виде диаграмм и спецификаций, а также генерировать программные коды на С++.

Данное Case-средство содержит средства реинженеринга, обеспечивающие повторное использование программных компонент в новых проектах.

Rational Unified Process представляет собой методологию, упорядочивающую жизненный цикл программного обеспечения с распределением ответственности между исполнителями.

Такая методология позволяет наладить производство высококачественного, отвечающего требованиям пользователя программного обеспечения и при этом полностью соблюсти установленный план и бюджет. Как и любое современное CASE средство, данная система поддерживает все стадии жизненного цикла ПО и предоставляет пользователю широкий спектр функций для анализа, проектирования, построения и сопровождения ПО. При этом используются объектно-ориентированные технологии и широко применяются графические модели. Таким образом, можно сделать вывод о том, что Rational Rose поддерживает все этапы жизненного цикла программного обеспечения от этапа анализа и, заканчивая тестированием, а также обеспечивает эволюционность его развития.

В состав Rational Suite, кроме самой технологии RUP как продукта, входят следующие компоненты:

- Rational Rose - средство визуального моделирования (анализа и проектирования), использующее язык UML;

- Rational XDE - средство анализа и проектирования, интегрируемое с платформами MS Visual Studio .NET и IBM WebSphere Studio Application Developer;

- организации совместной работы группы разработчиков. Оно позволяет команде разработчиков создавать, структурировать, устанавливать приоритеты, отслеживать, контролировать изменения требований, возникающих на любом этапе разработки компонентов приложения;

- Rational Rapid Developer - средство быстрой разработки приложений на платформе Java 2 Enterprise Edition;

- Rational ClearCase - средство управления конфигурацией ПО;

- Rational SoDA - средство автоматической генерации проектной документации;

- Rational ClearQuest - средство для управления изменениями и отслеживания дефектов в проекте на основе средств e-mail и Web;

- Rational Quantify - средство количественного определения узких мест, влияющих на общую эффективность работы программы;

- Rational Purify - средство для локализации трудно обнаруживаемых ошибок времени выполнения программы;

- Rational PureCoverage - средство идентификации участков кода, пропущенных при тестировании;

- Rational TestManager - средство планирования функционального и нагрузочного тестирования;

- Rational Robot - средство записи и воспроизведения тестовых сценариев;

- Rational TestFactory - средство тестирования надежности;

- Rational Quality Architect - средство генерации кода для тестирования.

Как известно объектное моделирование является важным инструментом при разработке сложных систем. Оно позволяет представить систему на абстрактном уровне, исключая чрезмерные подробности, и дает информацию о её структуре. А это очень полезно на начальном этапе проектирования поскольку, грамотно построенная, структура может лечь в основу программой реализации системы.

Так или иначе, разработчики всегда использовали моделирование, например, в виде рисования схем бумаге, электронной доске или документирования. Но, с развитием **UML** (Unified Modeling Language), у разработчиков появилась возможность представлять свои, под час одним им понятные модели, в общих нотациях. Что дало огромный толчок развитию объектно-ориентированного моделирования.

**UML** - это графический язык для визуализации, проектирования и документирования программных систем, призванный облегчить работу разработчика на всех её этапах.

Однако понято, что невозможно придумать язык одинаково хорошо подходящий для проектирования любых систем, поэтому UML не является замкнутым стандартом, а напротив всегда открыт для расширения.

Для того чтобы в Rose Real Time построить функционирующую систему, необходимы две составляющие:

- UML модель, которая описывает структуру и задает шаблон поведения системы.

- C++ код, который определяет конкретное поведение и помещается в стандартные места, предусмотренные Rose Real Time (таковыми местами, например, являются: действия при переходах, спецификация классов, капсул их методов и т.д.).

Для организации групповой работы в Rational Rose возможно разбиение модели на управляемые подмодели. Каждая из них независимо сохраняется на диске или загружается в модель. В качестве подмодели может выступать категория классов или подсистема. Для управляемой подмодели предусмотрены операции:

- загрузка подмодели в память;

- выгрузка подмодели из памяти;

- сохранение подмодели на диске в виде отдельного файла;

- установка защиты от модификации;

- замена подмодели в памяти на новую.

**9 Этапы моделирования в Rational Rose**

*Первоначальные шаги при моделировании предметной области включают*:

- Модель функций предметной области (Business Use case model);

- Модель объектов предметной области (Business object model);

- Спецификации, описывающие производственный процесс;

- Словарь терминов предметной области.

*Дальнейшими шагами должны быть:*

- Моделирование реализации системы (Use case model). Цель – определить функции создаваемой автоматизированной системы.

- Определение требований (Requirements). Цель - определить, что система должна делать, согласовать это с заказчиком и задокументировать.

*Основным результатом этапа определения требований к системе является:*

- Модель функций системы (Use case model), т.е., фактически описание автоматизируемых функций.

- Прототип пользовательского интерфейса (User-Interface Prototype).

- Спецификации функций системы (Supplementary Specifications).

Анализ и проектирование. Цель - преобразовать требования к системе в проект системы. Основным результатом здесь является модель стадии проектирования (Design Model). Она показывает, каким образом функции системы будут реализовываться посредством объектов и классов.

Далее: разработка, тестирование, внедрение. Rational Rose - это очень большая система и предназначена она не только для описания бизнес – процессов, но и для поэтапного создания больших автоматизированных систем, большими коллективами, их тестирования, внедрения и поддержки.

**Достоинства продукта Rational Rose**

- мощный графический язык моделирования предметной области, обладающий высоким уровнем формализации и поддерживающий объектно-ориентированную методологию;

- удобная навигация между элементами модели при помощи "инспектора проекта";

- хранение результатов проектирования в виде единой модели;

- поддержка работы над проектом группы разработчиков;

данное CASE средство может быть применено для создания разнообразного объектно-ориентированного программного обеспечения, в первую очередь для платформы Windows, а так же на языке Java;

- на всех этапах разработки применяется язык UML, и проект программного средства представляет собой единую модель;

- настройка на различные языки программирования и архитектуры программных систем, а также возможность "обратного проектирования" на основе исходных текстов, на различных языках программирования. Существует поддержка различных способов физической реализации для компонент проектируемой системы;

возможность конфигурирования системы с помощью модулей расширения;

- в наибольшей степени подходит для разработки крупных информационных систем, так как реализует большую часть функций ARIS и ERwin/BPwin.

**Недостатки продукта Rational Rose**

- слабо реализована поддержка проектирования ПО для других операционных систем, почти все стандартные рабочие среды ориентированны на построение Windows-приложений, единственным способом написания приложения для не-Windows операционной системы является использование языка Java, производительность которого, пока, оставляет желать лучшего.

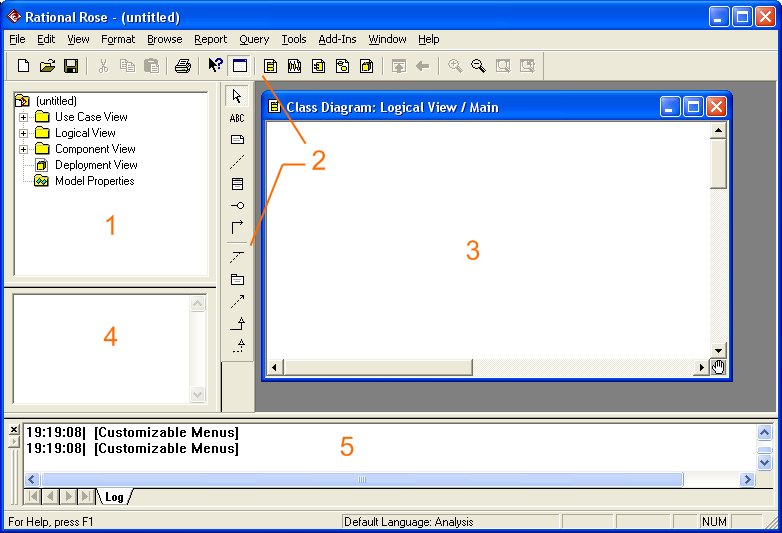
- сложность самого языка UML также накладывает определенные ограничения на привлечение к работам над проектами непрофессионалов, нельзя показать и удалить неиспользуемые объекты в отличие от BPWin;

- недостаточно функциональная графика (нельзя менять толщину линий, надписи не центрируются, текст не всегда можно поместить целиком, иногда он обрезается);

- не поддерживает функционально-стоимостной анализ;

нет возможности отобразить потоки данных между объектами или процессами.

**Интерфейс программы Rational Rose Enterprise Edition**



**Браузер (browser) -** используется для быстрой навигации по модели

**Панели инструментов (toolbars)**- обеспечивают быстрый доступ к наиболее распространенным командам

**Окно диаграммы (diagram window)**- используется для просмотра и редактирования одной или нескольких диаграмм UML

**Окно документации (documentation window)**- используется для работы с документацией элементов модели

**Журнал (log)** - применяется для просмотра ошибок и отчетов о результатах выполнения различных команд

Рисунок – 35. Главное окно приложения

Браузер - это иерархическая структура, позволяющая легко осуществлять навигацию по модели. Все добавляемые в модель элементы выводятся в окне браузера.

С помощью браузера можно:

- добавлять к модели элементы (сценарии, действующих лиц, классы, компоненты, диаграммы и т.д.);

- просматривать существующие элементы модели;

- просматривать существующие отношения между элементами модели;

- перемещать элементы модели;

- переименовывать элементы модели;

- добавлять элементы модели к диаграмме;

- связывать элемент с файлом или адресом в сети Интернет;

- группировать элементы в пакеты;

- работать с детализированной спецификацией элемента;

- открывать диаграмму.

Браузер поддерживает четыре представления (view): представление Вариантов Использования, Компонентов, Размещения и Логическое представление.

**Представление Вариантов Использования**

Это представление включает в себя всех действующих лиц, все варианты использования и их диаграммы для конкретной системы. Оно может также содержать некоторые диаграммы последовательности и диаграммы коопераций. Представление Вариантов Использования - это взгляд на систему, независимый от ее реализации. Основное внимание здесь уделяется представлению высокого уровня, отображающему, *что* система будет делать, а не *как* она будет делать это.

**Логическое представление**

Логическое представление концентрируется на том, как система будет реализовывать поведение, описанное в вариантах использования. Оно дает подробную картину составных частей системы и описывает их взаимодействие. Логическое представление включает в себя, помимо прочего, конкретные требуемые классы, диаграммы классов и диаграммы состояний. С их помощью разработчики могут сконструировать детальный проект создаваемой системы.

**Представление Компонентов**

Представление Компонентов содержит информацию о библиотеках кода, исполняемых файлах, динамических библиотеках и других компонентах моделей.

Представление Компонентов более всего используется теми участниками проекта, кто отвечает за управление кодированием, компиляцию и размещение приложения. Часть компонентов - это библиотеки кода. Остальные - динамические компоненты, например, исполняемые файлы и файлы динамических библиотек (DLL). С помощью этого представления разработчики могут понять, какие библиотеки кода были созданы и какие классы содержатся в каждом из них.

**Представление Размещения**

Представление Размещения соответствует физическому размещению системы, которое может отличаться от ее логической архитектуры. Например, система может иметь трехуровневую логическую архитектуру: интерфейс логически отделен от бизнес-логики, а бизнес-логика отделена от базы данных. Однако размещение системы может быть и двухуровневым: интерфейс находится на одном компьютере, а остальные две части - на другом.

**Приемы работы в Rational Rose для создания прецедентов и актеров**

Для создания прецедентов, актеров и отношений между ними в Rational Rose предназначена панель *Toolbox,* которая расположена вертикально между браузером и окном диаграммы. На ней по умолчанию представлены (таблица 3) следующие кнопки:

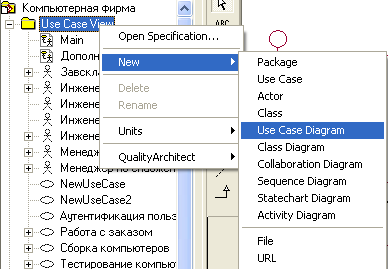
Таблица - 3. Основные кнопки панели компонентов

|  |  |  |
| --- | --- | --- |
| **Вид кнопки** | **Название кнопки** | **Назначение кнопки** |
| Стрелка выделения | Selection Tool | Превращает курсор в стрелку указателя для того, чтобы можно было выделять объекты |
| Текст на диаграмме | Text Box | Добавление к диаграмме текста |
| Примечание | Note | Добавление к диаграмме примечания |
| Связывание | Anchor Note to Item | Связывание примечания с объектом на диаграмме |
| Пакет | Package | Добавление на диаграмму нового пакета |
| Вариант использования | Use Case | Добавление на диаграмму нового варианта использования |
| Актер | Actor | Добавление на диаграмму нового актера |
|  | Unidirectional Association | Создание отношения ассоциации |
|  | Dependency or instantiaties | Создание отношения зависимости |
|  | Generalization | Создание отношения обобщения |

При желании состав кнопок на панели *Toolbox* можно изменить, добавив на нее недостающие элементы.

**Создание новой диаграммы прецедентов.**

Для создания новой диаграммы прецедентов необходимо щелкнуть правой кнопкой мыши на представлении Вариантов Использования браузера (Use Case View). В открывшемся меню выбрать пункт New > Use Case Diagram (Создать > Диаграмма прецедентов).

Рисунок – 36. Окно создания диаграммы

**Создание нового актера**

Для создания нового актера нужно щелкнуть по кнопке *Actor* на панели *Toolbox* и затем по свободному месту окна диаграммы, после чего можно ввести имя актера. По сути это единственное, что для актера нужно задавать. Но поскольку актер - это фактически класс, то для него можно задать все те свойства, что задаются для класса (см. Работа с классами в Rational Rose).

**Создание нового варианта использования**

Для создания нового варианта использования нужно щелкнуть по кнопке *Use Case* на панели *Toolbox* и затем по свободному месту окна диаграммы, после чего можно ввести название варианта использования. После создания прецедента можно определить его свойства. Для этого нужно дважды по нему щелкнуть или же вызвать для него контекстное меню и выбрать пункт *Open Specification...,* после чего откроется окно спецификации прецедента, содержащее ряд вкладок*.*

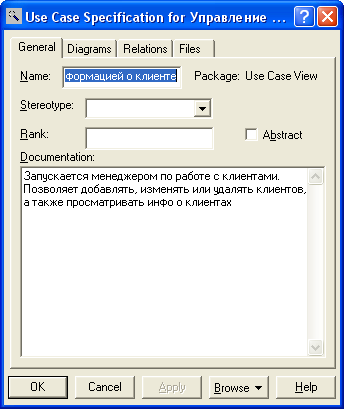
**

Рисунок – 37. Окно контекстного меню

Рассмотрим содержимое каждой вкладки:

* *General* — здесь задаются общие свойства варианта использования: имя (Name), стереотип (Stereotype), приоритет (Rank), является ли прецедент абстрактным (Abstract) и текстовое описание прецедента (Documentation);
* *Diagrams* —  здесь отображаются различные диаграммы, содержащие данный прецедент;
* *Relations* —  здесь отображаются все связи, в которых данный прецедент участвует;
* *Files —* добавление файлов, содержащих дополнительную информацию о классе.

**Создание отношений между актерами и прецедентами**

*Общее замечание.* Для любого типа отношений задание его свойств осуществляется одинаковым способом - или дважды щелкнуть по линии, или вызвать для нее контекстное меню и выбрать пункт *Specification ... .* Общими свойствами для всех типов отношений являются: имя (Name), стереотип (Stereotype) и текстовое описание отношения (Documentation). Дополнительными свойствами обладает только отношение коммуникации и обобщения, для которых часть свойств можно задать прямо в их контекстном меню.

**Отношение коммуникации**

Это единственно возможная связь между актером и прецедентом. Изображается с помощью однонаправленной ассоциации. Направление стрелки показывает кто инициирует связь.

Для создания отношения коммуникации следует выбрать кнопку *Unidirectional Association* на панели *Toolbox,* затем щелкнуть мышкой по инициатору связи (актеру или прецеденту) и не отпуская кнопки мыши перетащить стрелку на второго участника связи (прецедент или актер). Для данного типа отношения стереотип *communicate* можно не указывать, поскольку он неявно и подразумевается.

Для данного типа отношений можно задать следующие свойства (часть свойств задаются исходя из того, что актер фактически является классом):

* вкладка *General —* роль прецедентов в данном отношении (Role A и Role B);
* вкладка *Detail*  *—* является ли отношение производным (Derived); класс-ассоциация, связанный с данным отношением (Link Element); направление роли (Name Direction); ограничения - некоторое условие, которое должно выполняться (Constraints);
* вкладки *Role A General, Role B General —*задаются общие свойства ролей классов, участвующих в отношении: название роли (Role), видимость роли (Export Control) и текстовое описание роли (Documentation);
* вкладки *Role A Detail, Role B Detail —* задаются некоторые подробности ролей: имя роли (Role), ограничения для роли (Constraints), мощность роли (Multiplicity), направление отношения (Navigable), является ли отношение агрегацией (Aggregate), являются ли атрибуты классов, участвующие в отношении, статичными (Static); явлюятся ли классы отношения дружественными (Friend); каким образом будут включаться созданные атрибуты агрегации - по значению или по ссылке (Containment of Class) (при выборе значения By Value тип отношения поменяется на композицию), добавление квалификаторов (Keys/Qualifiers).

**Отношение включения**

Применяется когда один из прецедентов использует другой. Обозначается как отношение зависимости, которое направлено от базового прецедента к используемому, с указанием стереотипа - include.

Для создания отношения зависимости следует выбрать кнопку *Dependency or instantiaties* на панели *Toolbox,* затем щелкнуть мышкой по базовому прецеденту и не отпуская кнопки мыши перетащить стрелку на используемый прецедент. В спецификации отношения выбрать стереотип - include.

**Отношение расширения**

Применяется для отражения:

* дополнительных режимов;
* режимов, которые запускаются только при определенных условиях;
* альтернативных потоков, которые запускаются по выбору пользователя.

Обозначается как отношение зависимости, которое направлено от дополнительного прецедента к базовому, с указанием стереотипа - extend.

Для создания отношения зависимости следует выбрать кнопку *Dependency or instantiaties* на панели *Toolbox,* затем щелкнуть мышкой по дополнительному прецеденту и не отпуская кнопки мыши перетащить стрелку на базовый прецедент. В спецификации отношения выбрать стереотип - extend.

**Отношение обобщения**

Отношение обобщения служит для указания того факта, что некоторый вариант использования А может быть обобщен до варианта использования В. В этом случае вариант А будет являться специализацией варианта В. Фактически этим отношением показывается наследование. При этом В называется предком или родителем по отношению А, а вариант А — потомком по отношению к варианту использования В. Потомок наследует все свойства и поведение своего родителя, а также может быть дополнен новыми свойствами и особенностями поведения.

Для создания отношения зависимости следует выбрать кнопку *Generalization* на панели *Toolbox,* затем щелкнуть мышкой по прецеденту-наследнику и не отпуская кнопки мыши перетащить стрелку на прецедент-родитель. Для данного типа отношения можно задать следующие свойства: будет ли иметь класс-потомок доступ к не-public элементам класса-родителя (Friendship required), видимость отношения (Export Control), будет ли наследоваться только одна копия класса-родителя (Virtual inheritance).

# UML диаграммы в Rational Rose

Rational Rose - мощное CASE-средство для проектирования программных систем любой сложности. Одним из достоинств этого программного продукта будет возможность использования диаграмм на языке UML. Можно сказать, что Rational Rose является графическим редактором UML диаграмм.

В распоряжение проектировщика системы Rational Rose предоставляет следующие типы диаграмм, последовательное создание которых позволяет получить полное представление о всей проектируемой системе и об отдельных ее компонентах:

* Use case diagram (диаграммы прецедентов);
* Deployment diagram (диаграммы топологии);
* Statechart diagram (диаграммы состояний);
* Activity diagram (диаграммы активности);
* Interaction diagram (диаграммы взаимодействия);
* Sequence diagram (диаграммы последовательностей действий);
* Collaboration diagram (диаграммы сотрудничества);
* Class diagram (диаграммы классов);
* Component diagram (диаграммы компонент).

### 

### **Use case diagram (диаграммы прецедентов)**

Этот вид диаграмм позволяет создать список операций, которые выполняет система. Часто этот вид диаграмм называют диаграммой функций, потому что на основе набора таких диаграмм создается список требований к системе и определяется множество выполняемых системой функций.

Каждая такая диаграмма или, как ее обычно называют, каждый Use case – это описание сценария поведения, которому следуют действующие лица (Actors).

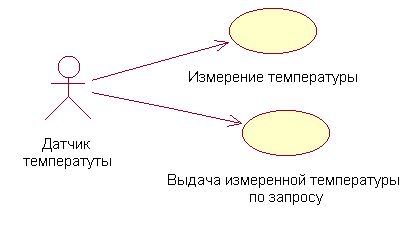


Рисунок – 38. Диаграммы прецедентов

Данный тип диаграмм используется при описании бизнес процессов автоматизируемой предметной области, определении требований к будущей программной системе. Отражает объекты как системы, так и предметной области и задачи, ими выполняемые.

### 

### **Deployment diagram (диаграммы топологии)**

Этот вид диаграмм предназначен для анализа аппаратной части системы, то есть «железа», а не программ. В прямом переводе с английского Deployment означает «развертывание», но термин «топология» точнее отражает сущность этого типа диаграмм.

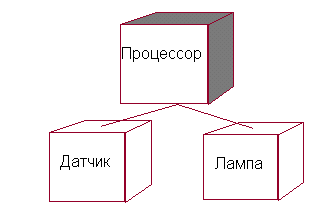


Рисунок – 39. Диаграммы топологии

Для каждой модели создается только одна такая диаграмма, отображающая процессоры (Processor), устройства (Device) и их соединения.

 Обычно этот тип диаграмм используется в самом начале проектирования системы для анализа аппаратных средств, на которых она будет эксплуатироваться.

### 

### **State Maсhine diagram (диаграммы состояний)**

Каждый объект системы, обладающий определенным поведением, может находится в определенных состояниях, переходить из состояния в состояние, совершая определенные действия в процессе реализации сценария поведения объекта. Поведение большинства объектов реальных систем можно представить с точки зрения теории конечных автоматов, то есть поведение объекта отражается в его состояниях, и данный тип диаграмм позволяет отразить это графически. Для этого используется два вида диаграмм: Statechart diagram (дмаграмма состояний) и Activity diagram (диаграмма активности)

#### Activity diagram (диаграммы активности)

Это дальнейшее развитие диаграммы состояний. Фактически данный тип диаграмм может использоваться и для отражения состояний моделируемого объекта, однако, основное назначение Activity diagram в том, чтобы отражать бизнес-процессы объекта. Этот тип диаграмм позволяет показать не только последовательность процессов, но и ветвление и даже синхронизацию процессов.

Этот тип диаграмм позволяет проектировать алгоритмы поведения объектов любой сложности, в том числе может использоваться для составления блок-схем.

#### Пример UML диаграммы Statechart

#### **Рисунок – 40. Диаграмма активности**

#### Statechart diagram (диаграмма состояний)

Диаграмма состояний (Statechart) предназначена для отображения состояний объектов системы, имеющих сложную модель поведения. Это одна из двух диаграмм State Machine, доступ, к которой осуществляется из одного пункта меню.

#### 

#### Пример UML Activity диаграммы

#### **Рисунок - 41. Диаграмма состояний**

### 

### **Interaction diagram (диаграммы взаимодействия)**

Этот тип диаграмм включает в себя диаграммы Sequence diagram (диаграммы последовательностей действий) и Collaboration diagram (диаграммы сотрудничества). Эти диаграммы позволяют с разных точек зрения рассмотреть взаимодействие объектов в создаваемой системе.

#### 

#### Sequence diagram (диаграммы последовательностей действий)

Взаимодействие объектов в системе происходит посредством приема и передачи сообщений объектами-клиентами и обработки этих сообщений объектами-серверами. При этом в разных ситуациях одни и те же объекты могут выступать и в качестве клиентов, и в качестве серверов.

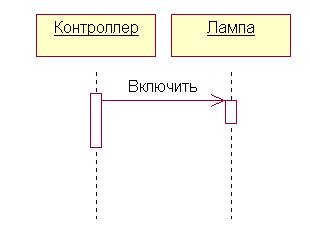


Рисунок – 43. Диаграмма последовательности действий

Данный тип диаграмм позволяет отразить последовательность передачи сообщений между объектами.

Этот тип диаграммы не акцентирует внимание на конкретном взаимодействии, главный акцент уделяется последовательности приема/передачи сообщений. Для того чтобы окинуть взглядом все взаимосвязи объектов, служит Collaboration diagram.

#### 

#### Collaboration diagram (диаграммы сотрудничества)

Этот тип диаграмм позволяет описать взаимодействия объектов, абстрагируясь от последовательности передачи сообщений. На этом типе диаграмм в компактном виде отражаются все принимаемые и передаваемые сообщения конкретного объекта, и типы этих сообщений.

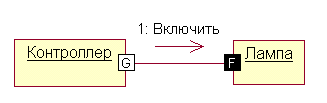


Рисунок – 44. Диаграммы сотрудничества

 По причине того, что диаграммы Sequence и Collaboration являются разными взглядами на одни и те же процессы, Rational Rose позволяет создавать из Sequence диаграммы диаграмму Collaboration и наоборот, а также производит автоматическую синхронизацию этих диаграмм.

### 

### **Class diagram (диаграммы классов)**

Этот тип диаграмм позволяет создавать логическое представление системы, на основе которого создается исходный код описанных классов.

Значки диаграммы позволяют отображать сложную иерархию систем, взаимосвязи классов (Classes) и интерфейсов (Interfaces).

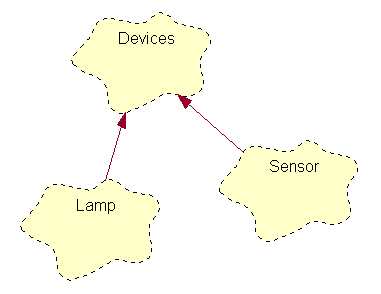


Рисунок – 45. Диаграммы классов

Данный тип диаграмм противоположен по содержанию диаграмме Collaboration, на котором отображаются объекты системы. Rational Rose позволяет создавать классы при помощи данного типа диаграмм в различных нотациях. В нотации, предложенной Г. Бучем, которая так и называется Booch, классы изображаются в виде чего-то нечеткого, похожего на облако. Таким образом Г.Буч пытается показать, что класс – это лишь шаблон, по которому в дальнейшем будет создан конкретный объект.

### **Component diagram (диаграммы компонентов)**

Этот тип диаграмм предназначен для распределения классов и объектов по компонентам при физическом проектировании системы. Часто данный тип диаграмм называют диаграммами модулей.

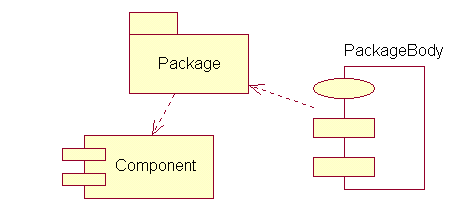


Рисунок – 46. Диаграммы компонентов

При проектировании больших систем может оказаться, что система должна быть разложена на несколько сотен или даже тысяч компонентов, и этот тип диаграмм позволяет не потеряться в обилии модулей и их связей.

## Контрольные вопросы:

Назначение MS Visio.

В какой редакции можно встретить MS Visio?

Перечислите основные особенности MS Visio.

Какие функции содержит MS Visio?

Что можно моделировать в MS Visio?

Перечислите основные термины в редакторе MS Visio?

Как осуществляется представление данных в MS Visio?

## **Назначение Rational Rose.**

## **Какие задачи способен решать Rational Rose в проектировании информационных систем?**

## **Назначения языка программирования UML.**

## **Перечислите этапы моделирования в Rational Rose.**

## **Назовите достоинства Rational Rose.**

## **Перечислите недостатки Rational Rose.**

## **Перечислите представления поддерживаемые браузером.**