



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФГБОУ ВО «Брянский государственный технический университет»
(БГТУ)

Политехнический колледж (ПК БГТУ)

УТВЕРЖДАЮ

Ректор ФГБОУ ВО БГТУ

_____ О.Н. Федонин

«__30__» __04__ 2021г.

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ
по изучению профессионального модуля
ПМ.02 Осуществление интеграции программных модулей

Специальность:	09.02.07 Информационные системы и программирование
Уровень образования выпускника:	среднее профессиональное образование (СПО)
Присваиваемая квалификация:	Программист
Форма обучения:	очная
Срок получения СПО по ППССЗ:	3 года 10 месяцев
Уровень образования, необходимый для приема на обучение по ППССЗ:	основное общее образование

Брянск 2021

Методические рекомендации по изучению профессионального модуля

ПМ.02. Осуществление интеграции программных модулей

(далее — МР)

для специальности **09.02.07 Информационные системы и программирование**

Разработал(и):

– преподаватель ПК БГТУ
– преподаватель ПК БГТУ

С.А.Горбарчук
Е.В.Симонян

МР рассмотрены и одобрены на заседании предметно-цикловой комиссии «Программирование в компьютерных системах» ПК БГТУ (далее — ПЦК)

от «30»04.2021 г., протокол № 10

Председатель ПЦК

Е.С. Левшакова

Согласовано:

Заместитель директора ПК БГТУ
по учебно-методической работе,

Т.Е.Балашова

© Горбарчук С.А., Симонян Е.В.
© ФГБОУ ВО «Брянский государственный
технический университет»

Оглавление

Введение	4
Общие указания к выполнению работ	7
Практические работы по МДК 02.01 Технология разработки программного обеспечения	8
Практические работы по МДК 02.02 Инструментальные средства разработки программного обеспечения	63
Практические работы по МДК 02.03 Анализ и моделирование программного обеспечения	98
Информационные источники	194

Введение

Главной задачей среднего профессионального образования является подготовка компетентных специалистов. В процессе формирования профессиональных и общих компетенций практические занятия занимают промежуточное положение между теоретическим и производственным обучением и служат одним из важнейших средств осуществления связи теории и практики.

Практические занятия являются неотъемлемым этапом изучения по профессиональному модулю ПМ02. «Осуществление интеграции программных модулей» и проводятся с целью:

- формирования практических умений в соответствии с требованиями к уровню подготовки обучающихся, установленными рабочей программой;
- обобщения, систематизации, углубления, закрепления полученных теоретических знаний;
- готовности использовать теоретические знания на практике.

Выполнение практических работ призвано способствовать закреплению теоретических знаний, формированию умений и способов действий через самостоятельную деятельность студентов. Ведущей дидактической целью практических занятий является формирование практических (профессиональных) умений – выполнение определённых действий, операций, необходимых в последующей профессиональной деятельности. Основная задача практических работ - научить студентов применять теоретические знания в практических ситуациях.

Выполнению практических заданий на уроке предшествует проверка знаний студентов, их теоретической готовности к выполнению практической работы.

Структура и содержание практических работ включает в себя следующие элементы: тема, цель выполнения работы, оборудование, программное обеспечение, методические указания по выполнению работы, контрольные вопросы. По каждой работе необходимо оформить отчет в соответствии с требованиями, сделать выводы, ответить на контрольные вопросы. Отчет о выполненной работе представляется студентом преподавателю для проверки, в том числе с защитой результатов, и оценивания. Отчет может быть представлен как в письменном, так и печатном виде.

Практические занятия по профессиональному модулю ПМ02 «Осуществление интеграции программных модулей» способствуют формированию следующих общих и профессиональных компетенций:

Код	Наименование результата обучения
ОК 1.	Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам.
ОК 2.	Осуществлять поиск, анализ и интерпретацию информации, необходимой для выполнения задач профессиональной деятельности.
ОК 3.	Планировать и реализовывать собственное профессиональное и личностное развитие.
ОК 4.	Работать в коллективе и команде, эффективно взаимодействовать с коллегами, руководством, клиентами.
ОК 5.	Осуществлять устную и письменную коммуникацию на государственном языке с учетом особенностей социального и культурного контекста.
ОК 6.	Проявлять гражданско-патриотическую позицию, демонстрировать осознанное поведение на основе традиционных общечеловеческих ценностей, применять стандарты антикоррупционного поведения.
ОК 7.	Содействовать сохранению окружающей среды, ресурсосбережению, эффективно действовать в чрезвычайных ситуациях.
ОК 8.	Использовать средства физической культуры для сохранения и укрепления здоровья в процессе профессиональной деятельности и поддержания необходимого уровня физической подготовленности.
ОК 9.	Использовать информационные технологии в профессиональной деятельности.
ОК 10.	Пользоваться профессиональной документацией на государственном и иностранном языке.
ОК 11.	Использовать знания по финансовой грамотности, планировать предпринимательскую деятельность в профессиональной сфере.
ПК 2.1	Разрабатывать требования к программным модулям на основе анализа проектной и технической документации на предмет взаимодействия компонент.
ПК 2.2	Выполнять интеграцию модулей в программное обеспечение.
ПК 2.3	Выполнять отладку программного модуля с использованием специализированных программных средств.
ПК 2.4	Осуществлять разработку тестовых наборов и тестовых сценариев для программного обеспечения.
ПК 2.5	Производить инспектирование компонент программного обеспечения на предмет соответствия стандартам кодирования.

В результате освоения учебной дисциплины обучающийся должен

иметь практический опыт в:

- интеграции модулей в программное обеспечение;
- отладке программных модулей. уметь:
- использовать выбранную систему контроля версий;
- использовать методы для получения кода с заданной функциональностью и степенью качества.

знать:

- модели процесса разработки программного обеспечения;
- основные принципы процесса разработки программного обеспечения;
- основные подходы к интегрированию программных модулей;
- основы верификации и аттестации программного обеспечения.

Для успешного выполнения практических работ обучающиеся обязаны ознакомиться с порядком их проведения и изучить соответствующие разделы теоретического курса.

Обучающиеся должны четко представлять задачу, уметь проводить необходимые расчеты.

Общие указания к выполнению работ

Перед началом выполнения работы внимательно ознакомьтесь с инструкцией, заданием к практической работе.

Отчет оформляется на листах со штампом. В отчет впишите тему, цель работы, оборудование, программное обеспечение. При выполнении работы следуйте приведенным пунктам плана. Номер пунктов и их названия должны соответствовать друг другу.

По мере выполнения работы необходимо сформулировать вывод.

Отчеты оформляются в журнале. На титульном листе должны быть указаны: название предмета, группа и фамилия студента. Титульный лист оформляется на формате А4

Пример оформления титульного листа:

Министерство науки и высшего образования РФ
ФГБОУ ВО «Брянский государственный технический университет»
«Политехнический колледж»

ЖУРНАЛ
практических работ по профессиональному модулю
ПМ02 Осуществление интеграции программных модулей

Студент _____

Группа: _____

Вариант № _____

Преподаватель _____

Брянск-2021

Практические по МДК 02.01 «Технология разработки программного обеспечения»

Практическая работа №1

Тема: «Моделирование программного проекта и определение состава разработчиков».

Цель: Научиться моделировать программный продукт, тестирование программ, выполнять организацию работ по проекту.

Оборудование: ПК

Программное обеспечение: MS Word

Порядок выполнения работы:

1. Разработать ассоциативную модель проекта.
2. Составить примерный сценарий работы с программой.
3. Разработать структуру исходных данных и продумать варианты их сохранения в программе.
4. Смоделировать метод организации работ над проектом и определить состав разработчиков.
5. Сделать вывод.

Варианты заданий:

1. Библиотеки - автоматизированный каталог.
2. Склад оптовой торговли крупными партиями товаров.
3. Производство продукции из компонентов (учёт сырья и материалов).
4. Жилищно-коммунальное хозяйство - учёт квартплаты.
5. Архив личных дел для сотрудников предприятия.
6. Учёт вкладов в Сбербанке.
7. Учебная часть - учёт успеваемости.
8. Учебная часть - учёт посещаемости.
9. Разработка программного обеспечения для службы занятости.
10. Военкомат - учёт призывников.
11. Поликлиника - автоматизированная регистратура.
12. Электронный телефонный справочник.
13. Разработка аналитической системы сбора и обработки информации в сфере индивидуального предпринимательства.
14. Разработка программного обеспечения учёта товарно-материальных ценностей в магазине.

Пример выполнения работы:

Проект: Автоматизированная система «Агентство недвижимости»

1. Ассоциативная модель - «Конвейер»

Поступление товара на конвейер - Поступление информации о квартире в агентство.

Упаковка товара - Занесение квартиры в общую базу данных.

Сход товара с конвейера - продажа квартиры и ее удаление из базы данных.

2. План работы с программой:

1. Вводятся некоторые критерии, по которым будет выбираться квартира из БД.

2. Выбираются наиболее подходящие квартиры.

3. Если покупатель определился с выбором, то составляется договор о продаже, то вводятся данные по покупателю, данные по квартире (добавляются автоматически), квартира удаляется из общей БД и добавляется в БД проданных квартир (заносятся данные по квартире, покупателю и времени составления договора).

4. Если возникает необходимость в составлении отчетных данных по проданным квартирам за определенный период, то формируется отчет, содержащий эти данные.

3. Структура данных:

Данные хранятся в виде таблицы. Сохранение может осуществляться оператором, автоматически через определенное (указанное в настройках) время, при закрытии программы.

4. Метод разработки ПП -

«Экстремальное программирование»

Используемые принципы:

1. Спагетти
2. Итеративности
3. Инкрементности
4. Сотрудничества

- Разработка ведется двумя программистами (ведущий программист - написание основных процедур и функции, второй

программист составление документации и написание дополнительных функций) совместно с представителем заказчика

5. Вывод: научились моделировать программный продукт,
тестированию программ, выполнять организацию работ по проекту.

Практическая работа №2

Тема: «Разработка архитектуры программного проекта»

Цель: Научиться разрабатывать архитектуру проекта и его модули.

Оборудование: ПК.

Программное обеспечение: MS Word

Порядок выполнения работы:

1. Определить функциональный состав и основные характеристики программы.
2. Описать архитектуру проекта (разработать схему условий использования, сценарий, модуль логического представления, схему модулей (приложить таблицу с описанием функций каждого модуля)).
3. Внести изменения в архитектуру.
4. Составить план разработки.
5. Сделать вывод.

Варианты заданий:

1. Библиотеки - автоматизированный каталог.
2. Склад оптовой торговли крупными партиями товаров.
3. Производство продукции из компонентов (учёт сырья и материалов).
4. Жилищно-коммунальное хозяйство - учёт квартплаты.
5. Архив личных дел для сотрудников предприятия.
6. Учёт- вкладов в Сбербанке.
7. Учебная часть - учёт успеваемости.
8. Учебная часть - учёт посещаемости.
9. Разработка программного обеспечения для службы занятости.
10. Военкомат - учёт призывников.
11. Поликлиника - автоматизированная регистратура.
12. Электронный телефонный справочник.
13. Разработка аналитической системы сбора и обработки информации в сфере индивидуального предпринимательства.

14. Разработка программного обеспечения учёта товарно-материальных ценностей в магазине.

Пример выполнения практической работы.

Практическая работа №1

Тема: «Разработка архитектуры программного проекта»

Цель: Научиться разрабатывать архитектуру проекта и его модули

Оборудование: ПК

Программное обеспечение: MS Word

Порядок выполнения работы:

1. Определить функциональный состав и основные характеристики программы.
2. Описать архитектуру проекта (разработать схему условий использования, сценарий, модель логического представления, схему модулей (приложить таблицу с описанием функций каждого модуля)).
3. Внести изменения в архитектуру.
4. Составить план разработки.
5. Вывод.

Выполнение работы:

Проект: Автоматизированная система «Агентство недвижимости»

1. Функциональный состав:

1. Поиск в базе данных наиболее подходящей по заданным параметрам квартиры.
2. Формирование договора при продаже квартиры.
3. Формирование отчетности по проданным квартирам за определенный период.

2. Схема условий использования:

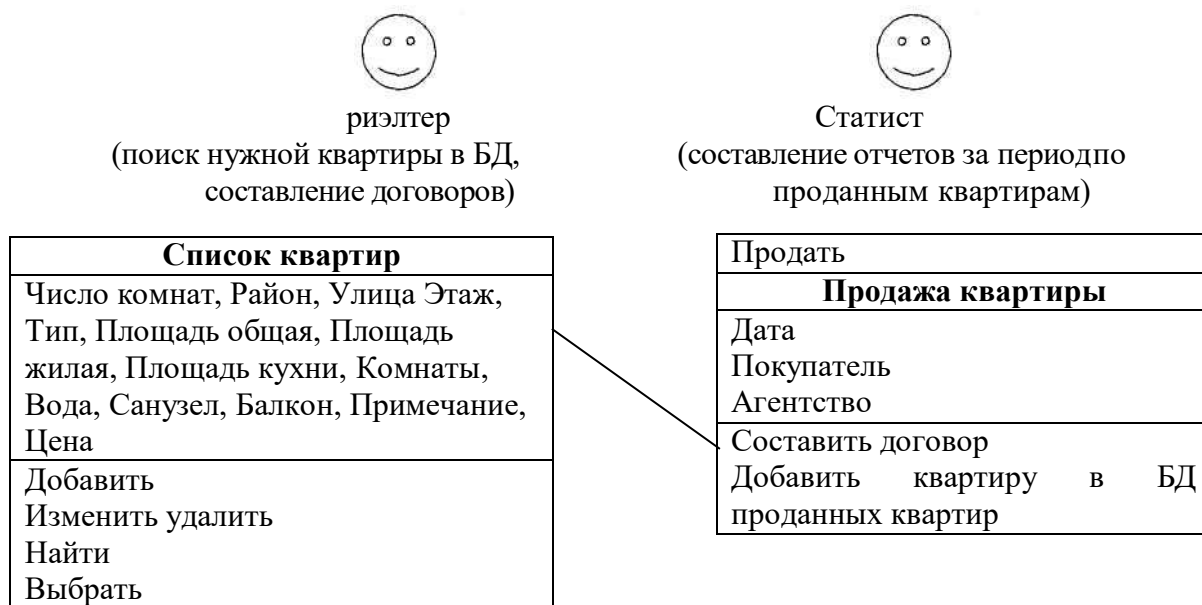


Схема модулей:



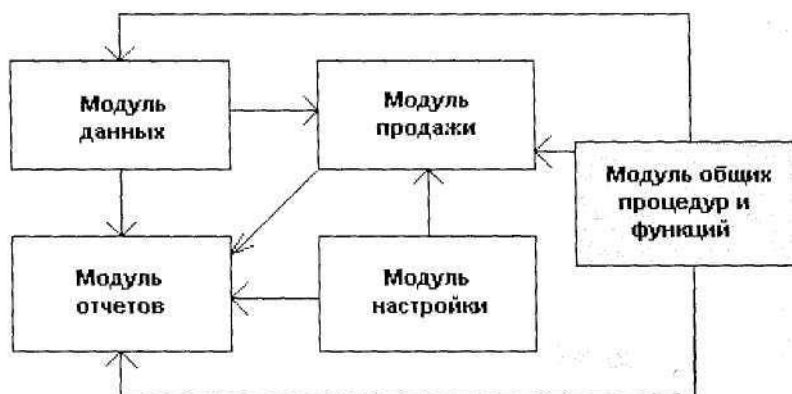
Модуль данных - содержит данные о квартирах, имеются функции редактирования данных.

Модуль продажи - вводятся данные о покупателе, автоматически заносятся данные по продаваемой квартире из модуля данных, составляется договор.

Модуль отчетов - выбор вида отчета и отчетного периода по проданным за этот период квартирам.

Модуль настройки - предоставляет другим модулям дополнительную информацию.

3. Внести изменения в архитектуру проекта



Модуль общих процедур и функций - содержит общие процедуры и функции, которые используются в нескольких модулях.

4. План разработки:

1. Модуль данных

2. Модуль продажи

3. Модуль отчетов

4. Модуль настройки

- **Модуль общих процедур и функций** дополняется новыми процедурами по мере необходимости

5. **Вывод:** научились разрабатывать архитектуру проекта и его модули.

Практическая работа №3

Тема: Написание исходного кода программы и повышение его надежности.

Цель: Получить практические навыки написания исходного кода программы и повышение его надежности.

Порядок выполнения работы.

1. Написать исходный код программы.
2. Предусмотреть меры по защите программы от несанкционированного ввода исходных данных.
3. Обеспечить сохранение информации через определенный промежуток времени.
4. Обеспечить мероприятия по выводу контрольных сообщений в случае неадекватного поведения программы.
5. Сделать вывод.

Пример выполнения работы

1. Добавление/Изменение записи:

```
procedure TMWin.AddCHck(Sender:TObject);

var rc:longint;

begin
  with MTable
  do begin
    with Sender as TButton do
    begin
      if Application.MessageBox(pchar(AnsiUpperCase(Caption)+'
Запись?'),'Сообщение...',mb_YesNo+mb_IconQuestion)=idNo then
        exit;
      if Name='Add then
      begin
        if Cells[0,RowCounM]<>' then
          RowCount:=RowCount+1; rc:=RowCount-
          1;
        end else
          rc:=Row;
        end;
        Row:=rc;
        Ceils[0,rc]:=NumF.Text; Cells[1,rc]:=Dist.Text;
        Cells[2,rc]:=Street.Text; Cells[3,rc]:=Floor.Text;
        Cells[4,rc]:=House.Text; Cells[5,rc]:=Scom.Text;
        Cells[6,rc]:=Sdwel.Text; Cells[7,rc]:=Skit.Text;
```

```

Cells[8,rc]:=Flats.Text; Cells[9,rc]:=Water.Text;
Cells[10,rc]:=San.Text; Cells[H,rc]:=Balc.Text;
Cells[12,rc]:=Note.Text;
Cells[13,rc]:=Price.Text;
end;
SortNumR(MTable,0,1,MTable.RowCoim);
SortColChange(Sender); FindClick(Sender);
MTableMouseDown(Sender,mbLeft,[ssLeft], 1,1);

```

1. Правильность ввода данных обеспечивается за счет запрета ввода символов, несоответствующих числовому типу.

```
with Sender as TEdit do
```

```
if not(Key in ['O'-.V]) and (ord(Key)<>8) and (Key<>^M) then Key:=#0;
```

2. Сохранение таблицы с данными через определенный промежуток времени.

```
procedure TMWin.Timer1Timer(Sender:TObject);
```

```
begin
```

```
SaveTable(MTable,'Table.dbf'); //Вызов процедуры сохранения
таблицы
```

```
end;
```

3. Проверка числового поля таблицы на соответствие типа при сортировке по данному полю.

```
procedure SortNumR(Table:TStringGrid;scol,crow,lrow:longint); // Процедура
сортировки
```

```
var ij,k:longint;s:string; begin
```

```
with Table do
```

```
for i:=crow to lrow-2
```

```
do
```

```
for j:=i+1 to Irow-1 do
```

```
begin
```

```
try // Начало обработки исключительных ситуаций
```

```
if strtoint(Cells[scol,i])>strtoint(Cells[scol,j])then
```

```
for k:=0 to ColCount-1 do
```

```
begin
```

```
s:=cells[kj]; Cells[kj]:=Cells[k,i]; Cells[k,i]:=-s;
```

```
end;
```

```
except // Команды, выполняемые в случае возникновения
ошибок
```

```
showmessage('Не числовой тип: строка - '+inttostr(k)+T);
```

```
end; // Конец обработки исключительных ситуаций
```

```
end;
```

```
end;
```

4. **Вывод;** получили практические навыки написания исходного кода програм

Практическая работа №4

Тема: «Разработка пользовательского интерфейса»

Цель: получить практические навыки в разработке многооконного или многокомпонентного интерфейса

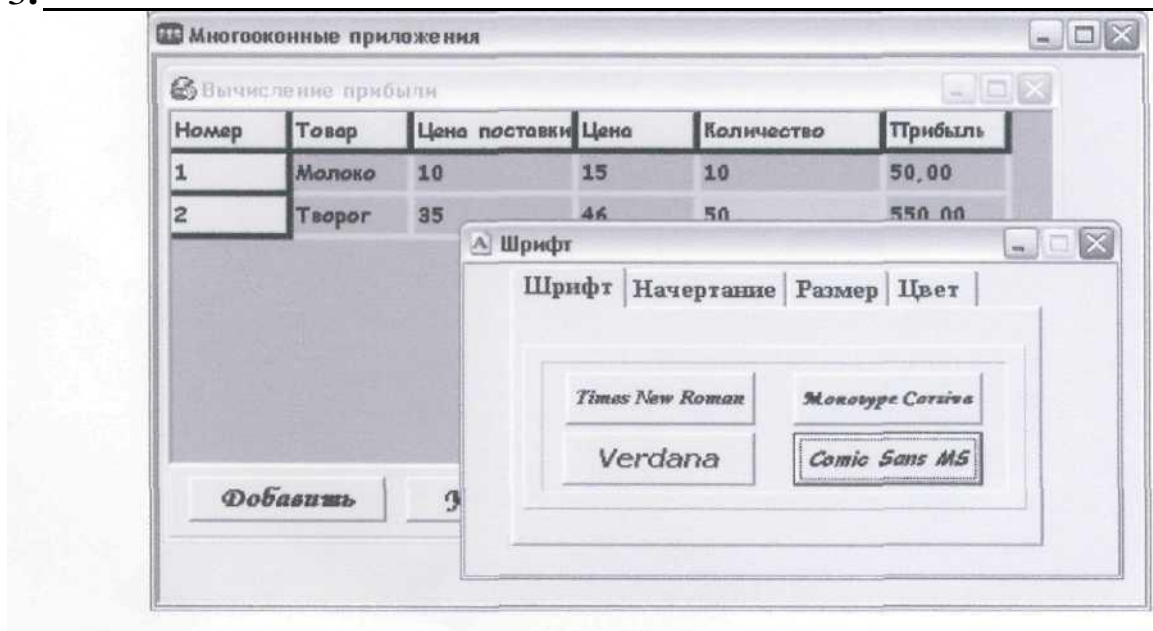
Оборудование: ПК, Borland Delphi

План

1. Выбрать вид интерфейса и обосновать свой выбор.
2. Разработать способы размещения компонентов и обосновать.
3. Разработать дизайн интерфейса.
4. Смоделировать работу программы.
5. Сделать вывод.

Пример выполнения работы

1. Программа к данной практической работе представляет собой пример создания приложения, как с многооконным интерфейсом, так и с многокомпонентным.
2. Пример работы со шрифтом использует многостраничный компонент **TabbedNotebook**. Каждый параметр шрифта имеет отдельную закладку, что позволяет не загромождать экран различными окнами.
- 3.



4. Расчет прибыли. Установка шрифта для таблицы вынесено в отдельное окно. Входные данные: **Цена поставки, Цена продажная, Количество**. Выходные данные: **Прибыль**.

Некоторые процедуры из модуля установки шрифта:

```
procedure TForm2.Button8Click(Sender: TObject);
```

```
begin
```

```
form3.sgl.Font.Style:=form3.sgl.Font.Style+[fsBold3;
```

```
end;
```

```
procedure TForm2.Button12Click(Sender: TObject);  
begin  
form3.sgl.Font.Style:=Q;  
end;
```

```
procedure TForm2.Button3Click(Sender: TObject);  
begin  
form3.sgl.Font.Name:='Times New Roman';  
end;
```

```
procedure TForm2.Button7CHck(Sender: TObject);  
begin  
if colordialog1.Execute then  
form3.sgl.font.Color:=colordialog1.Color;  
end;
```

```
procedure TForm2.Button1Click(Sender: TObject);  
begin  
if form3.sgl.font.Size>8 then  
begin  
form3.sgl.font.Size:=form3.sgl.font.Size-2;  
labell.Caption:=mttostr(form3.sgl.font.Size);  
end;  
end;
```

```
procedure TForm2.Button2Click(Sender: TObject);  
begin  
if form3.sgl .font.Size<14 then  
begin  
form3.sgl.font.Size:=form3.sgl.font.Size+2;  
labell.Caption~inttostr(form3.sgl.font.Size);  
end;  
end;
```

5. Вывод: получили практические навыки в разработке многооконного и многокомпонентного интерфейса

Практическая работа №5

Тема Разработка справочной системы

Цель: Получить практические навыки в разработке справочной системы.

Оборудование: IBM PC.

Программное обеспечение: Borland Delphi 7.0

Порядок выполнения работы:

1. К объектам, размещенным на форме разработать подсказки и время их появления.
2. В файле *.txt разработать файл справки.
3. Написать программный код для запуска справки.
4. Написать программный код для запуска справки по нажатию F1.
5. Сделать вывод.

Пример выполнения работы:

Программный код:

```
procedure TForm1.Button1MouseMove(Sender: TObject;
ShiftTShiftState; X,
Y:integer);
begin
  if CheckBox1.Checked=true then
    Button1.Hint:='В данный момент курсор находится над
кнопкой!!!'+
      #13+'Для этого используем "Hint¹"
  else Button1.Hint:="";

  if CheckBox2.Checked=true then
    Form1.StatusBar1.Panels[0].Text:='Это тоже подсказка...!!!'
  else Form1.StatusBar1.Panels[0].Text:="";
end;
procedure TForm1.FormMouseMove(Sender: TObject; Shift:
TShiftState; X,
Y: Integer);
begin
  Form1.StatusBar1.Panels[0].Text:="";
end;
procedure TForm1.Button1Click(Sender: TObject);
begin
  if CheckBox3.Checked=true then
    ShellExecute(handle, 'open', 'help.html', nil, nil, sw_shownormal);
end;
procedure TForm1.FormKeyDown(Sender: TObject; var Key:
Word;
```

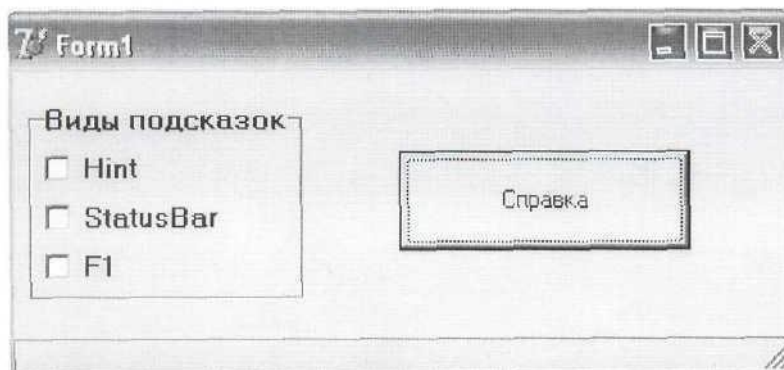
```

        Shift: TShiftState);
begin
    Button1.Click;
end;

procedure TForm1.Button1KeyDown(Sender: TObject; var Key:
Word;
    Shift: TShiftState);
begin
    if key=vk_F1 then
        ButtonLClick;
end;
procedure TForm1.CheckBox1KeyDown(Sender: TObject; var Key:
Word;
    Shift: TShiftState);
begin
    if key=vk_F1 then
        Button1Click;
end;
procedure TForm1.CheckBox2KeyDown(Sender: TObject; var Key:
Word;
    Shift: TShiftState);
begin
    if key=vk_F1 then
        Button1.Click;
end;
procedure TForm1.CheckBox3KeyDown(Sender: TObject; var Key:
Word;
    Shift: TShiftState);
begin
    if key=vk_F1 then
        Button1Click;
end;

end.

```



Вывод: получил практические навыки в разработке справки

Практическая работа 6

Тема: Модернизация программного обеспечения.

Цель работы: Получить практические навыки в модернизации программного обеспечения.

Оборудование: Среда программирования Delphi.

Теоретическая часть:

Современная информационная система предприятия представляет собой сложный механизм. Внедрение нового либо модернизация существующего элемента ставит ряд задач по совершенствованию функциональности и улучшению характеристик системы, а также тесным образом связано с вопросами миграции приложений и данных и обеспечением взаимодействия между подсистемами.

Бывает, что с течением времени происходит утрата контактов с производителем программного обеспечения, в результате чего получение услуг по доработке и техническому сопровождению становится невозможным.

Необходимость модернизации программного обеспечения может быть продиктована следующими причинами:

устареванием программного обеспечения и снижением его эффективности
невозможностью поддержки существующего ПО
необходимостью усовершенствования ПО и приведения его к более современным формам (переработка графического интерфейса, внедрение более современных технологий и пр.)
потерей контроля над информационной системой вследствие многочисленных доработок и изменений
изменениями бизнес-процессов и невозможностью адаптации ПО старого к ним
наличием архитектурных недостатков, затрудняющих изменение ПО и снижающих его гибкость

Модернизация ПО преследует следующие задачи:

расширение функциональных возможностей
перенос приложений на новые платформы и технологии
миграция и адаптация данных
оптимизация производительности ПО
системная интеграция

Для обеспечения успешного функционирования бизнеса ситуация требует быстрого разрешения. В нашей компании Вам помогут модернизировать Ваше ПО, в соответствии с Вашими требованиями. Специалисты нашей компании вместе с Заказчиком согласуют техническое задание и приступят к его выполнению. Кроме этого, после выполнения всех пунктов тех. задания мы можем заключить с Заказчиком специальный договор, по которому компания будет оказывать поддержку программного обеспечения в будущем.

2. Порядок выполнения работы:

2.1 К разработанному программному продукту доработать процедуру или функцию, расширяющую возможности программы.

- 2.2 Подключить разработанные программы к программному проекту.
- 2.3 Провести отладку и тестирование программного продукта.
- 3. Контрольные вопросы:
 - 1. Что такое модернизация программ?
 - 2. С какой целью проводится модернизация?
 - 3. На каком этапе жизненного цикла выполняется модернизация программы?

Практическая работа №7

Тема: «Измерение производительности алгоритмов и программы»

Цель: Получить практические навыки в определении производительности программ.

Оборудование: IBM PC.

Программное обеспечение: Borland Delphi 7.0

Порядок выполнения работы:

- 1. Заполнить строки одного из компонентов в Delphi.
- 2. Исследовать влияние объема данных на производительность программ.
- 3. Построить график зависимости скорости работы программы от объема данных.
- 4. Сделать вывод.

Пример выполнения работы.

1. Программный код:

```
unit Unit1;
...
var
  Form1: TForm1;
  a:array [1..10000] of integer;
  dt,dt1:tdatetime;
  Hour, Min, Sec, MSec: Word;
  time1,time2:string;
implementation

{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
var l,i,j, s, k:integer;
begin
  Edit1.text:=' ';
  dt:=now;
  DecodeTime(dt, Hour, Min, Sec, MSec);
  time1:=timetostr(dt);
  for l:=0 to 10000 do
    a[l]:=random(10000);
  for j:=1 to 9999 do
    for i:=j+1 to 10000 do
      begin
        if a[j]>a[i] then
          begin
            s:=a[j];
```

```

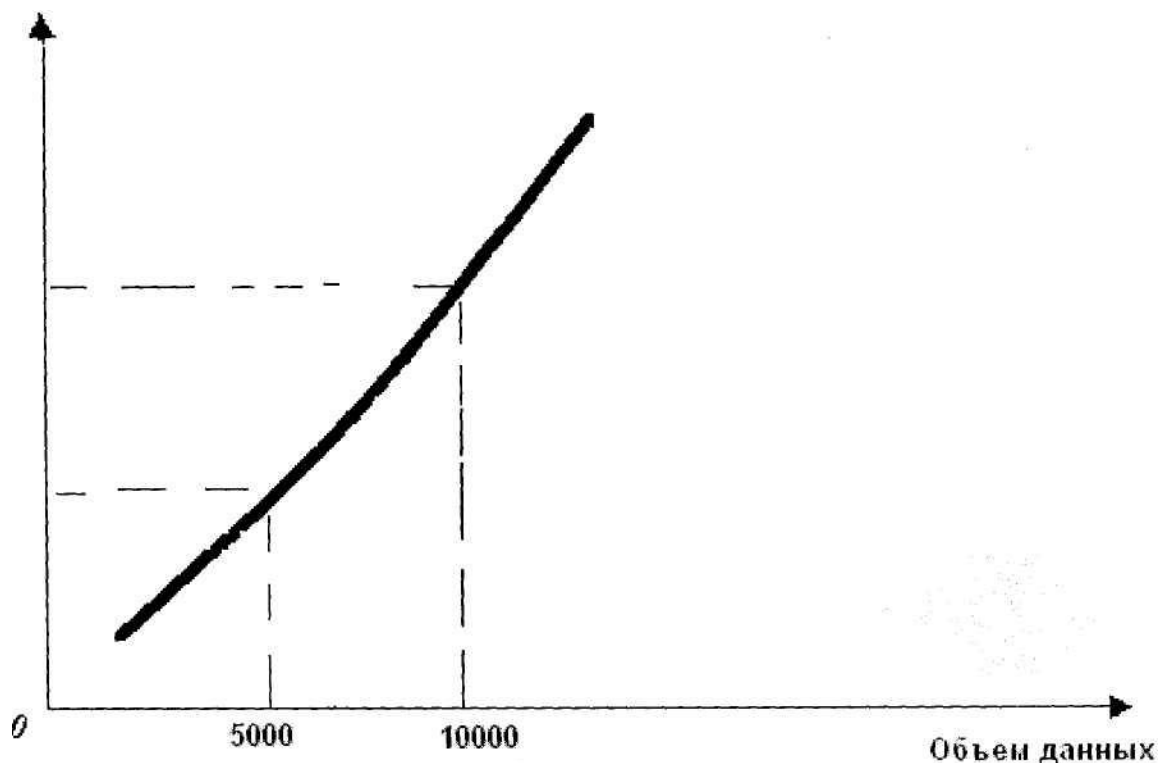
a[j]:=a[i];
a[i]:=s;
end;
end;
for k:=1 to 10000 do
Memo1.Lines.Add(inttostr(a[k]));
dt1:=now;
DecodeTime(dt1, Hour, Min, Sec, MSec);
time2:=timetostr(dt1);
Edit1.Text:=timetostr(((dt1)-(dt)));
end;

procedure TForm1.Button2Click(Sender: TObject);
var l,i,j, s, k:integer;
begin
Edit1.text:=' ';
dt:=now;
DecodeTime(dt, Hour, Min, Sec, MSec);
time1:=timetostr(dt);
for l:=0 to 10000 do
Memo1.Lines.Add(inttostr(random(10000)));
{for j:=1 to Memo1.Lines.Count-2 do
for i:=j+1 to Memo1.Lines.Count-1 do
begin
if Memo1.Lines.Strings[j]>Memo1.Lines.Strings[i] then
begin
s:=strtoint(Memo1.Lines.Strings[j]);
Memo1.Lines.Strings[j]:=Memo1.Lines.Strings[i];
Memo1.Lines.Strings[i]:=inttostr(s);
end;
end; }
dt1:=now;
DecodeTime(dt1, Hour, Min, Sec, MSec);
time2:=timetostr(dt1);
Edit1.Text:=timetostr(((dt1)-(dt)));
end;
end.

```

2. Величина объема данных прямым образом сказывается на производительности программ. Чем больше объем данных мы возьмем тем дольше будет работать программа.

3. График зависимости скорости работы программы от объема данных. Время, мсек



Вывод: получил практические навыки в определении производительности программ.

Практическая работа №8

Тема: «Отладка программного продукта».

Цель: Закрепить практические навыки в отладке программ.

Программное обеспечение: Delphi

Оборудование: ПК

Порядок выполнения работы:

1. Для своей программы выбрать метод отладки.
2. Разбить программный код на этапы и выполнить отладку каждого этапа.
3. Выполнить отладку программы в комплексе
4. Сделать вывод.

Пример выполнения работы:

1. Метод отладки – метод флажков.
2. Так как выбран метод с помощью флажков, то отладка будет

производиться за счет установки программных флажков. Правильность ввода данных обеспечивается за счет запрета ввода символов, несоответствующих числовому типу.

```
with Sender as TEdit do  
if not(Key in ['0'..'9']) and (ord(Key)<>8) and (Key<>'.') then Key:=#0;
```

Дополнительная обработка клавиш для избежания возможных ошибочных ситуаций:

```
procedure TMWin.HouseKeyDown(Sender:TObject; var Key: Word;  
Shift: TShiftState);  
begin  
with Sender as TComboBox do  
if (Key<>vk_Up) and (Key<>vk_Down) then Key:=ord('0');  
end;
```

```
procedure TMWin.HouseKeyPress(Sender: TObject; var Key: Char);  
begin  
with Sender as TComboBox do  
Key:=#0;  
end;
```

3. В программе предусмотрена обработка клавиш при вводе данных, чтобы исключить ввод некорректной информации, однако, например, при вводе вещественного числа разрешено

вводить запятую, и пользователь может дважды её ввести, поэтому предусмотрена выдача сообщения об ошибочных действиях пользователя:

```
procedure TMWin.ScomChange(Sender: TObject);  
var a:double;  
begin  
with Sender as TEdit do  
begin  
try  
a:=strtofloat(text);  
except
```



```
if Text<>'' then  
  ShowMessage('Неправильный формат данных!');  
  Text:='';  
end;  
end;  
end;
```

4. **Вывод:** закрепили практические навыки в отладке программ.

Практическая работа №9

Тема: «Тестирование программ методом «Белого ящика»»

Цель: Получить практические навыки в тестировании программ методом «Белого ящика».

Оборудование: IBM PC.

Программное обеспечение: Borland Delphi 7.0

Теоретическая часть

Стратегия «белого ящика», или стратегия тестирования, тестирования, управляемого логикой программы, позволяет исследовать внутреннюю структуру программы. Тестирующий получает тестовые данные путем анализа логики программы.

Исчерпывающему входному тестированию может быть поставлено в соответствие исчерпывающее тестирование маршрутов – программа проверена полностью, если с помощью тестов удастся осуществить выполнение программы по всем возможным маршрутам её графа.

Слабые места:

- Число не повторяющих друг друга маршрутов в программе – астрономическое.
- Даже если каждый маршрут в программе будет проверен, сама программа может содержать ошибки.

Принципы тестирования

1. Хорош тот тест, для которого высока вероятность обнаружить ошибку.
2. Одна из самых сложных проблем тестирования – решить, когда нужно его закончить: как выбрать конечное число тестов, которое дает максимальную отдачу (вероятность обнаружения ошибок) для данных затрат.
3. Не нужно тестировать свою собственную программу. Тестирование всегда должна выполнять внешняя группа, которая стоит особняком от

программиста и проекта.

4. Необходимая часть всякого теста – описание ожидаемых выходных данных и результатов. Одна из самых распространенных ошибок при тестировании состоит в том, что результаты каждого теста не прогнозируются до его выполнения.

5. Избегайте невоспроизводимых тестов, не тестируйте «с лету». Основной недостаток в том, что такие тесты мимолетны; они исчезают по окончании их выполнения. Всякий раз, когда программу понадобится тестировать повторно, придется придумывать тесты заново.

6. Готовьте тесты как для правильных, так и для неправильных входных данных. Многие ошибки, которые потом неожиданно обнаруживаются в работающих программах, проявляются в следствии непредусмотренных действий пользователей. Тесты, представляющие неожиданные и неправильные входные данные, часто лучше обнаруживают ошибки, чем правильные тесты.

7. Детально изучите результаты каждого теста.

8. Поручайте тестирование самым способным программистам. Тестирование и проектирование тестов – этап в разработке программного обеспечения, особенно требующий творческого подхода.

9. Проект системы должен быть таким, чтобы каждый модуль подключался к системе только один раз.

10. Никогда не изменяйте программу, чтобы облегчить её тестирование.

ГОСТ Р ИСО/МЭК 12119-2000 «Информационная технология. Пакеты программ. Требования к качеству и тестирование». Содержит указания, которые определяют порядок тестирования продукта на соответствие его требованиям и качеству.

2 Ход работы

2.1 Разработать программу по заполнению структуры данных

2.2 Разработать процедуру по обработке структуры данных и вызвать ее из головной программы

2.3 Тестировать программы методом белого ящика

Контрольные вопросы

1. В чем состоит тестирование программ методом «белого ящика»
2. Назовите основные принципы тестирования
3. Перечислите аксиомы тестирования
4. Опишите последовательность разработки тестов.

Практическая работа №10

Тема: «Тестирование программ методом «Черного ящика»»

Цель: Получить практические навыки в тестировании программных продуктов методом «Черного ящика».

Оборудование: IBM PC.

Программное обеспечение: Среда программирования Delphi

1. Теоретическая часть

Тестирование программного обеспечения

Тестирование (testing) – процесс выполнения программы (или части программы) с целью найти ошибки.

Доказательство (proof) – попытка найти ошибки в программе безотносительно к внешней для программы среде. Предполагает формулировки утверждений о поведении программы и затем вывод и доказательство математических теорем о правильности программы. Доказательства могут рассматриваться как форма тестирования, хотя они и не предполагают прямого выполнения программы.

Контроль (verification) – попытка найти ошибки, выполняя программу в тестовой или моделируемой среде.

Испытание (validation) – попытка найти ошибки, выполняя программу в заданной реальной среде.

Аттестация (certification) – авторитетное подтверждение правильности программы. При тестировании с целью аттестации выполняется сравнение с некоторым заранее определенным стандартом.

Отладка (debugging) не является разновидностью тестирования. Направлена на установление точной природы известной ошибки, а затем на её исправление. Эти два вида деятельности связаны – результаты тестирования являются исходными данными для отладки.

Тестирование модуля, или автономное тестирование (module testing, unit testing) – контроль отдельного программного модуля изолированно от всех остальных модулей.

Тестирование сопряжений (integration testing) – контроль сопряжений между частями системы (модулями, компонентами, подсистемами).

Тестирование внешних функций (external function testing) – внешнего поведения системы, определенного внешними спецификациями.

Комплексное тестирование (system testing) – контроль и/или испытание системы по отношению к исходным целям. Является процессом контроля, если оно выполняется в моделируемой среде, и процессом испытания, если выполняется в среде реальной.

Тестирование приемлемости (acceptance testing) – проверка соответствия каждого конкретного варианта установки системы с целью выявить любые ошибки, возникшие в процессе настройки системы.

Тестирование настройки (installation testing) – проверка соответствия каждого конкретного варианта установки системы с целью выявить любые ошибки, возникшие в процессе настройки системы.

Тестирование программы методом черного ящика

Стратегия «черного ящика» - тестирование с управлением по данным или тестирование с управлением по входу-выходу. Программа рассматривается как «черный ящик». Ее цель – выяснение обстоятельств, в которых поведение программы не соответствует спецификации. Тестовые данные используются только в соответствии со спецификацией программы (т.е. без учета значений о её внутренней структуре).

При таком подходе обнаружение всех ошибок в программе является критерием исчерпывающего входного тестирования. Последнее может быть достигнуто, если в качестве тестовых наборов использовать все возможные наборы входных данных.

Построение исчерпывающего исходного теста невозможно. Это подтверждается двумя аргументами: нельзя создать тест, гарантирующий отсутствие ошибок; создание таких тестов противоречит экономическим требованиям (целью является максимизация числа ошибок, обнаруживаемых одним тестом).

Порядок разработки тестов

1. По внешней спецификации разрабатываемые тесты:
 - а. Для каждого класса входных данных;
 - б. Для граничных и особых значений входных данных.

Контролируется, все ли классы выходных данных при этом проверяются, и добавляются при необходимости нужные тесты.

2. Разрабатываются тесты для всех функций, которые не проверяются в п.1.
3. По тексту программы проверяется, все ли условные переходы выполнены в каждом направлении (C1). При необходимости добавляются новые тесты.
4. Аналогично проверяется, проходятся ли пути для каждого цикла: без выполнения тела, с однократным или максимальным числом повторений.
5. Готовятся тесты, проверяющие исключительные ситуации, недопустимые входные данные, аварийные ситуации.

Функциональное тестирование дополняется здесь структурным. Классы входных/выходных данных должны быть определены в плане тестирования уже во внешней спецификации. Согласно статистике п.1. и 2. Обеспечивают степень

охвата C1 в среднем 40-50%. Проверка по C1 (п. 3) обычно выявляет 90% всех ошибок, найденных при тестировании. (Все программное обеспечение ВВС США принимается с проверкой по C1).

Систематическое тестирование предполагает так же ведение журнала отладки (Bug Book), в котором фиксируется ошибка (описание, дата обнаружения, автор модуля) и в дальнейшем – исправление (дата, автор).

Приведем так называемые аксиомы тестирования.

1. Тест должен быть направлен на обнаружение ошибки, а не на подтверждение правильности программы.
2. Автор теста – не автор программы.
3. Тесты разрабатываются одновременно или до разработки программы.
4. Необходимо предсказать ожидаемые результаты теста до его выполнения и анализировать причины расхождения результатов.
5. Предыдущее тестирование необходимо повторять после каждого внесения исправления в программу.
6. Следует повторять полное тестирование после внесения изменений в программу или после переноса её в другую среду.
7. Для тех программ, в которых обнаружено много ошибок, необходимо дополнить первоначальный набор тестов.

2. Порядок выполнения работы

- 2.1. По вариантам своего проекта составить исходный код процедуры или функций.
- 2.2. Составить тесты для тестирования процедур с использованием процедуры тестирования методом «черного ящика»
- 2.3. Разработать тест для тестирования программы методом «черного ящика».
- 2.4. Сравнить результаты тестирования и сделать вывод.

Варианты заданий

1. Библиотеки – автоматизированный каталог.
2. Склад оптовой торговли крупными партиями товаров.
3. Производство продукции из компонентов (учет сырья и материалов).
4. Жилищно-коммунальное хозяйство – учет квартплаты.
5. Архив личных дел для сотрудников предприятия.
6. Учет вкладов в Сбербанке.
7. Учебная часть – учет успеваемости.
8. Учебная часть – учет посещаемости.
9. Разработка программного обеспечения для службы занятости.
10. Военкомат – учет призывников.
11. Поликлиника – автоматизированная регистратура.

12. Электронный телефонный справочник.
13. Разработка аналитической системы сбора и обработки информации о сфере индивидуального предпринимательства.
14. Разработка программного обеспечения учета товарно-материальных ценностей в магазине.

3. Контрольные вопросы

- 3.1. В чем состоит тестирование программ методом «черного ящика»
- 3.2. С какой целью проводят тестирование программ?
- 3.3. Какие виды ошибок существуют?
- 3.4. Что такое тест? Какими свойствами он обладает?

Практическая работа №11

Тема: Разработка программ со структурами данных.

Цель: Получить практические навыки в разработке программ со структурами данных.

Оборудование: IBM PC.

Программное обеспечение: Borland Delphi 7.0

Порядок выполнения работы:

1. По данным своего варианта разработать структуру данных в виде массива записей.
2. Охарактеризовать структуру данных.
3. Реализовать программу ввода структуры данных и поиска записей в ней.

Порядок выполнения работы:

1. Разработать структуру данных по книгам.
2. Структура данных содержит следующие данные: номер книги, название, автор, год издания.
3. Составить и отладить программный код:

Пример выполнения работы

type

```
    rec=record  
        name,author:string;  
        nom,str:integer;  
    end;  
    ...
```

var

```
    Form1: TForm1;  
    a:array[0..5] of rec;  
    krit:string;
```

```

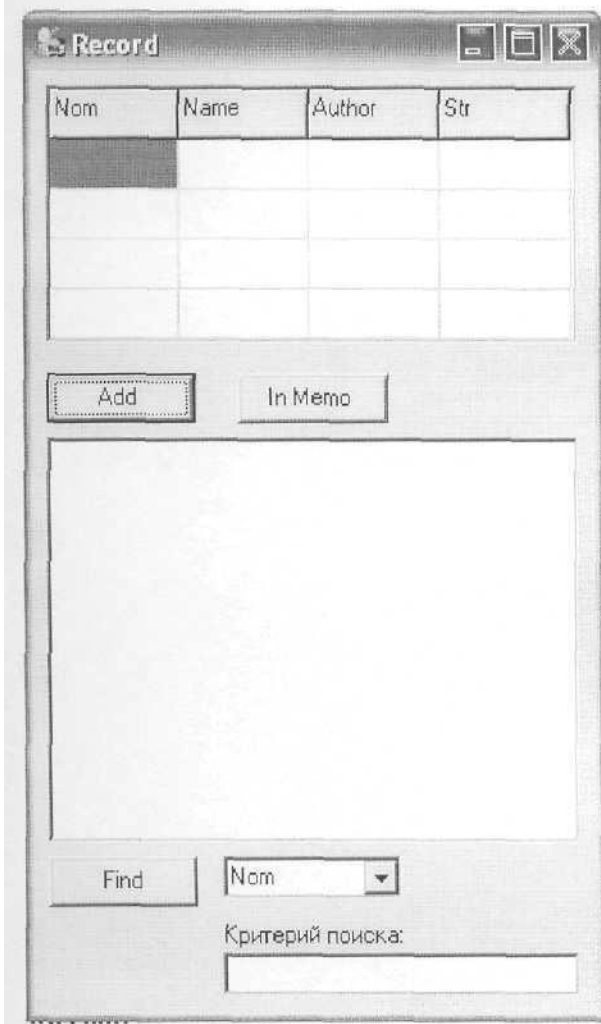
...
procedure TForm1.Button1Click(Sender: TObject);
var
i:integer;
begin
//i:=0;
for i:=1 to SG1.RowCount-1 do
begin
//ShowMessage(SG1.Cells[0,i]);
a[i-1].nom:=strtoint(SG1.Cells[0,i]);
a[i-1].name:=SG1.Cells[1,i];
a[i-1].author:=SG1.Cells[2,i];
a[M].str:=strtoint(SG1.Cells[3,i]);
end;
end;
procedure TForm1.FormCreate(Sender: TObject);
begin
SG1.Cells[0,0]:='Nom';
SG1.Cells[1,0]:='Name';
SG1.Cells[2,0]:='Author';
SG1.Cells[3,0]:='Str';
end;
procedure TForm1.Button2Click(Sender: TObject);
var i:integer;
begin
//i:=0;
for i:=1 to SG1.RowCount-1 do
Memoi.Lines.Add('Nom-'+' '+inttostr(a[i-1].nom)+' '+Name-' '+' '+a[i-1].name+' '+
  'Author-' +a[i-1].author+' '+'Str-' +inttostr(a[M].str));
end;
procedure TForm1.CB1Change(Sender: TObject);
begin
case CBLItemIndex of 0: krit:='Nom'; 1:krit:='Name'; 2: krit:='Author'; 3: krit:='Str';
end; end;
procedure TForm1.Button3Click(Sender: TObject); var i,j,k,l:integer;
begin
Memo!. Clear; if krit^Nom then begin
for i:=1 to SG1.RowCount-1 do
if strtoint(Edit1 .text)=a[i-1].nom then
Memoi.Lines.Add('Nom-'+' '+inttostr(a[i-1].nom)+' '+'Name-' '+' '+a[i-1].name+' '+
  'Author-' +a[i-1].author+' '+'Str-' +inttostr(a[i-1].str))
end else
if krit='Name' then
begin
for j:=1 to SG1.RowCount-1 do

```

```

if (Edit1.text)=a[j-1].name then
Memo1.Lines.Add('Nom-'+' '+inttostr(a[j-1].nom)+' '+'Name-'+' '+a[j-1].name+' '+'
Author-' +a[j-1].author+' '+'Str-' +inttostr(a[j-1].str)) end else
if knt='Author' then begin
for k:=1 to SG1.RowCount-1 do if (Edit1.text)=a[k-1].author then
Memo1.Lines.Add('Nom-'+' '+inttostr(a[k-1].nom)+' '+'Name-'+' '+a[k-1].name+'
'Author-' +a[k-1].author+' '+'Str-' +inttostr(a[k-1].str))
end else
if krit='Str' then
begin for l:=1 to SG1.RowCount-1 do
if strtoint(Edit1.text)=a[l-1].str then
Memo1.Lines.Add('Nom-'+' '+inttostr(a[M].nom)+' '+'Name-'+' '+
a[l-1].name+' '+'Author-' +a[l-1].author+' '+'Str-' +inttostr(a[l-1].str))
end;
end; end.

```



Вывод: получил практические навыки в разработке структур данных.

Практическая работа №12

Тема: Разработка динамических структур данных.

Цель: Получить практические навыки в разработке программ с динамическими структурами данных.

Оборудование: IBM PC.

Программное обеспечение: Borland Delphi 7.0

Порядок выполнения работы

1. На основе предыдущей практической работы разработать динамическую запись.
2. Написать программный код с использованием динамической записи
3. Отладить программу, сохранить.
4. Сделать вывод.

Пример выполнения работы

Программный код:

```
type
  prec=^trec;
  trec=record
    name,author:string;
    nom,str:integer;
  end;
  .....

var
  Form1: TForm1;
  a:array[0..5] of prec;
  krit:string;
  .....

procedure TForm1.Button1Click(Sender: TObject);
var i:integer;
begin
  //i:=0;
  for i:=1 to SG1.RowCount-1 do
    begin
      new(a[i-1]);
      //ShowMessage(SG1.Cells[0,i]);
      a[i-1]^nom:=strtoint(SG1.Cells[0,i]);
      a[i-1]^name:=SG1.Cells[1,i];
      a[i-1]^author:=SG1.Cells[2,i];
      a[i-1]^str:=strtoint(SG1.Cells[3,i]);
    end;
  end;
```

```

procedure TForm1.FormCreate(Sender: TObject);
begin
  SG1.CellsICOI[0,0]:='Nome';
  SG1.Cells[1,0]:='Name';
  SG1.Cells[2,0]:='Author';
  SG1.Cells[3,0]:='Str';
end;

```

```

procedure TForm1.Button2Click(Sender: TObject);
var i:integer;
begin
  //i:=0;
  for i:=1 to SG1.RowCount-1 do
  begin
    Memo1.Lines.Add('Nom-'+' '+inttostr(a[i-1].nom)+' '+'Name-'+' '+a[i-1].name+' '+'Author-'+' '+a[i-1].author+' '+'Str-'+' '+inttostr(a[i-1].str)); dispose(a[i-1]);
  end;
end;

```

```

procedure TForm1.CB1Change(Sender: TObject);
begin
  case CBL1.ItemIndex of
    0: krit:='Nom';
    1: krit:='Name';
    2: krit:='Author';
    3: krit:='Str';
  end;
end;

```

```

procedure TForm1.Button3Click(Sender: TObject);
var i,j,k,l:integer;
begin
  Memo1.Clear;

```

```

if krit='Nom' then
begin
for i:=1 to SG1.RowCount-1 do
if strtoint(Edit1.text)=a[i-1].nom then
Memo1.Lines.Add('Nom'+',''+inttostr(a[i-1].nom)+' '+Name-+' '+
+a[i-1].name+' '+Author-+a[M].author+' '+Str-+inttostr(a[i-1].str));
end else
if krit:='Name' then begin
for j:=1 to SG1.RowCount-1 do
if (Edit1.text)=a[j-1].name then
Memo1.Lines.Add('Nom-+' '+inttostr(a[j-1].nom)+' '+Name-+' '+a[j-1].name+' '+Author-+a[j-1].author+' '+Str-+inttostr(a[j-1].str));
end else
if krit='Author' then begin
for k:=1 to SG1.RowCount-1 do
if (Edit1.text)=a[k-1].author then
Memo1.Lines.Add('Nom-+' '+inttostr(a[k-1].nom)+' '+name-+' '+a[k-1].name+' '+Author-+a[k-1].author+' '+Str-+inttostr(a[k-1].str))
end else
if krit='Str' then
begin
for l:=1 to SGI.RowCount-1 do if strtoint(Edit1.text)=a[l-1].str then
Memo1.Lines.Add('Nom-+' '+inttostr(a[l-1].nom)+' '+Name-+' '+a[l-1].name+' '+Author-+a[l-1].author+' '+Str-+inttostr(a[l-1].str))

end;
end;
end.

```

The image shows a software window titled "Record". At the top, there is a table with four columns: "Nom", "Name", "Author", and "Str". The first row of the table is highlighted in grey. Below the table, there are two buttons: "Add" and "In Memo". Underneath these buttons is a large, empty rectangular area. At the bottom of the window, there is a "Find" button, a dropdown menu currently showing "Nom", and a text input field labeled "Критерий поиска:" (Search criterion:).

Вывод: получил практические навыки в разработке динамических структур

Практическая работа №13

Тема: Доработка структур данных.

Цель: Получить практические навыки в разработке пользовательских типов данных.

Оборудование: IBM PC.

Программное обеспечение: Borland Delphi 7.0

Порядок выполнения работы:

1. Используя структуру данных разработать для нее пользовательский тип данных.
2. Используя этот тип составить программу.
3. Составить процедуру по расшифровке пользовательского типа.
4. Отладить программу, привести пример, сделать вывод.

Пример выполнения работы:

Программный код:

```
type
  mytype=(drama, fantasy);
  trec=record anrmytype;
  name,author:string;
  strinteger; end;
var
  Form1: TForm1;
  book:trec;
  stroka:string;
procedure TForm1.Button1Click(Sender: TObject);
begin
  book.name:=Edit1.Text;
  book.author:=Edit2.Text;
```

```

book.str:=StrToInt(Edit3.Text);
if ComboBox1.ItemIndex=0 then
book.anr:=drama else book.anr:=fantasy;
if book.anr=drama then
stroka:='drama' else stroka:='fantasy';
Memo1.Lines.Add('Название-'+' '+book.name'b';'+ ' '+
  'Автор-V '+book.author+';'+ ' '+
  'Страниц-'+' '+inttostr(book.str)+';'+ ' '+ ^aHp-' '+stroka+');
end;
end.

```

Вывод: Получил практические навыки в разработке пользовательских типов данных.

Практическая работа №14

Тема: «Разработка программ сортировки данных».

Цель: Закрепление навыков работы с данными.

Оборудование: ПК.

Программное обеспечение: Borland Delphi 6.

Порядок выполнения работы:

1. Выполнить реализацию алгоритма заполнения данных.
2. Разработать подпрограмму сортировки данных.
3. Привести пример выполнения.
4. Оформить отчет, сохранить проект, сделать вывод.
- 5.

Пример выполнения работы:

```
procedure TForm1.Button1Click(Sender: TObject);
  var k,n,si:Longint;
  s:string;
  begin
    ListBox1.Clear;
    for k:=1 to 10000 do
      begin
        s:='';
        for n:=1 to 25 do
          s:=s+chr(random(25)+33);
        listBox1.items.add(s);
      end;
    end;
  end.
```

Разработать подпрограмму сортировки по алфавиту.

```
Procedure SortLst(Ls:tStrings);
  var n,m:longint;
  begin
    for m:=Pred(Ls.Count) downto 1 do
      for n:=1 to m do
        if Ls[Pred(n)]>Ls [n]
        then Ls.Exchange (Pred(n),n);
      end;
    end.
```

Разработать программу поиска данных.

```
procedure TForm1.Button2.Click(Sender: TObject);
  var x, y, p: longint;
```

```

st: string;
beg, wrd: Boolean;
begin
Edit2.setfocus;
st:=AnsiLowerCase(edit1.text{st});
if beg then
begin
x:=0; y:=1;
end
else
begin
x:=stringgrid1.col+1;
y:=stringgrid1.row;
if x>=stringgrid1.colcount then
begin
x:=0; y:=y+1;
end;
end;
while y<stringgrid1.rowcount do
begin
if wrd then
p:= ord(st=AnsiLowerCase(stringgrid1.cells[x,y]))
else p:=pos(st, AnsiLowerCase(stringgrid1.cells[x,y]));
if (p>0) and (stringgrid1.rowheights[y]>0) then
begin
if x<stringgrid1.fixedcols
then stringgrid1.col:=stringgrid1.fixedcols
else stringgrid1.col:=x;
stringgrid1.row:=y;
exit;
end;
x:=x+1;
if x>=stringgrid1.colcount then
begin

```



```
x:=0;  
y:=y+1;  
end;  
end;  
y:=0;  
x:=x div y;  
end;  
end.
```

Вывод: Закрепил навыки работы с данными.

Практическая работа №15

Тема: «Разработка программы фильтрации данных»

Цель: Закрепить теоретические знания по структурам данных.

Оборудование: IBM PC.

Программное обеспечение: Borland Delphi 6.

Порядок выполнения работы:

1. Сформировать таблицу в базе данных.
2. Разработать программу фильтрации данных.
3. Привести пример выполнения.
4. Сохранить проект, оформить отчет и сделать вывод.

Пример выполнения работы.

```

procedure Zagr_Tabl;
var Det_Num: extended; i: integer;
begin
  With Form1 do
  begin
    try
      Det_Num:=StrToDate(CBox1.Text+'.'+IntToStr(CBox2.ItemIndex+1)+'.'+CBox3.Text);
      Table1.Filter := '(KDate = ''' + DateToStr(Det_Num) +
        ''')and('+'KTime = ''' +
          CBox4.Text + ''')';
      Table1.Filtered:=True;
      Memo1.Text:=Table1.Fields[5].Text;
      Edit2.Text:=Table1.Fields[6].Text;
      Edit4.Text:=IntToStr(Table1.RecordCount);
      Memo2.Text:=FloatToStr(StrToFloat(Edit4.Text)*StrToFloat(Edit2.Text)
    );
      Table1.First;
      while not Table1.EOF do
      begin
        StringGrid1.Cells[ StrToInt( Table1.Fields[3].Text ),
StringGrid1.RowCount -
          StrToInt( Table1.Fields[4].Text ) ] := #9;
        Table1.Next;
      end;
    except
      Table1.Filter:='';
      Table1.Filtered:=False;
    end;
    Table1.Filter:='';
    Table1.Filtered:=False;
  end;
end;

```

Выполнить сортировку данных. Сортировку выполняем следующим образом в окне Object Inspector в поле IndexFieldName устанавливаем Kpom.

Вывод: Закрепил теоретические знания по структурам данных.

Практическая работа №16

Тема: Разработка программ для решения экономических задач.

Цель работы: Получить практические навыки в разработке программ для решения экономических задач.

Оборудование: IBM PC.

Программное обеспечение: Borland Delphi 7.0

Порядок выполнения работы:

1. Выбрать задачу одного из классов.
2. Создать ведомость по одному из классов экономических задач (учетный номер, ФИО, должность, стаж, тарифная сетка, кол-во отработанных часов, надбавка).
3. Сделать добавление и сохранение информации.
4. Отладить программу, привести пример, сделать вывод.

Пример выполнения работы:

Программный код:

unit Unit1;

var

Form1: TForm1;

f:textfile;

implementation

uses Unit2;

{ \$R *.dfm }

```

procedure TForm1.FormCreate(Sender: TObject);
begin
    StringGrid1.Cells[0,0]:='Номер';
    StringGrid1.Cells[1,0]:='Фамилия';
    StringGrid1.Cells[2,0]:='Должность';
    StringGrid1.Cells[3,0]:='Стаж';
    StringGrid1.Cells[4,0]:='Тариф. сетка';
    StringGrid1.Cells[5,0]:='ОТраб. часы';
    StringGrid1.Cells[6,0]:='Наплата';
end;
procedure TForm1.Button1Click(Sender: TObject);
begin
    Form2.Visible:=true;
end;

procedure TForm1.Button2Click(Sender: TObject);
var i,j:integer; S,s1,s2,s3,s4,s5,s6:string;
begin
    i:=0;
    for j:=1 to StringGrid1.RowCount-1 do
    begin
        s:=StringGrid1.Cells[i,j];
        s1:=StringGrid1.Cells[i+1,j];
        s2:=StringGrid1.Cells[i+2,j];
        s3:=StringGrid1.Cells[i+3,j];
        s4:=StringGrid1.Cells[i+4,j];
        s5:=StringGrid1.Cells[i+5,j];
        s6:=StringGrid1.Cells[i+6,j];
        Memo1.Lines.Add(s+' '+s1+' '+s2+' '+s3+' '+s4+' '+s5+' '+s6);
    end;
    AssignFile(f,ExtractFilePath(Application.ExeName)+'\'+Save.prj');
    Rewrite(f);

```

```

Write(f,Memo1.text);
CloseFile(f);
end;
end.

unit Unit2;
var
    Form2: TForm2;
implementation

uses Unit1;

{$R *.dfm}

procedure TForm2.Button1Click(Sender: TObject);
begin
    Form1 .StringGrid1 .RowCount:=Form1 .StringGrid1 .RowCount+1;
    Form1.StringGrid1.Height:=(Form1.StringGrid1.RowCount)*22;
    Form1.Height:=Form1.StringGrid1.Height+100;
    Form1.StringGrid1.Cells[0,Form1.StringGrid1.RowCount-2]:=Edit2.Text;
    Form1 .StringGrid1 .Cells[1,Form1 .StringGrid1 .RowCount-2]:=Edit1 .Text;
    Form1 .StringGrid1 .Cells[2,Form1 .StringGrid1 .RowCount-2]:=ComboBox1.Items[ComboBox1.ItemIndex];
    Form1.StringGrid1.Cells[3,Form1.StringGrid1.RowCount-2]:=Edit3.Text;
    Form1 .StringGrid1 .Cells[4, Form1 .StringGrid1 .RowCount-
2]:=ComboBox2.Items[ComboBox2.ItemIndex];
    Form1.StringGrid1.Cells[5,Form1.StringGrid1.RowCount-2]:=Edit4.Text;
    Form1 .StringGrid1 .Cells[6,Form1 .StringGrid1 .RowCount-2]:=Edit5.Text;
    Edit1.Text:=""; Edit4.Text:="";
    Edit2.Text:=""; Edit5.Text:="";
    Edit3.Text:="";
    Form2.Visible:=False;
end;
procedure TForm2.FormCreate(Sender: TObject);

```

```

begin
Label3.Caption:='Тарифная'+#13+'сетка'
end;
end.

```

The 'Write' dialog box contains the following fields and controls:

- Номер**: Text box with value 3
- Фамилия**: Text box with value Сидоров
- Должность**: Dropdown menu with value Инженер програм
- Стаж**: Text box with value 5
- Тарифная сетка**: Dropdown menu with value 5
- Часы**: Text box with value 78
- Надбавка**: Text box with value 1000
- Сформировать**: Button at the bottom

The 'Pr №16' window displays a table with the following data:

Номер	Фамилия	Должность	Стаж	Тариф. сетка	Отраб. часы	Надбавка
1	Петров	Оператор П 1	2	30	500	
2	Иванов	Программи 2	3	56	750	

Buttons: Запись, Сохранение

Вывод: Получил практические навыки в разработке программ для решения финансово- учетных задач.

Практическая работа №17

Тема: Программирование инженерных задач.

Цель: "Получить практические навыки в разработке инженерных программ. **Оборудование:** IBM PC.

Программное обеспечение: Borland Delphi 7.0

Порядок выполнения работы:

1. Разработать форму по вводу данных для определения межосевого расстояния зубчатых передач.
2. Составить программу по определению межосевого расстояния.
3. Предусмотреть в программе округление межосевого расстояния до стандартного значения.
4. Отладить программу, привести пример, сделать вывод.

Пример выполнение работы:

Программный код:

```
unit Unit1;
```

```
interface
```

```
uses
```

```
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms, Dialogs, ExtCtrls, StdCtrls;
```

```
type
```

```
TForm1 = class(TForm)    Label2: TLabel;  Shape1:  
TShape; Shape2: TShape; Label3: TLabel;  Label1:  
TLabel; GroupBox1: TGroupBox; Edit1: TEdit;  
Label4: TLabel;  Label5: TLabel;    Edit2: TEdit;
```



```
Label6: TLabel;      Edit3: TEdit;      Label7: TLabel;  
Edit4: TEdit;      Label8: TLabel;    Edit5: TEdit;  
Button1: TButton;   Edit6: TEdit;
```

```

Label9: TLabel;
ComboBox1: TComboBox;
procedure Button1Click(Sender: TObject);
procedure FormCreate(Sender: TObject); private
{ Private declarations } public
{ Public declarations } end;

var Form1: TForm1;

implementation {$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
var m, k, u, f, j: real; begin
m:=StrToFloat(Edit1.Text);
k:=StrToFloat(Edit2.Text);
u:=StrToFloat(Edit3.Text);
f:=StrToFloat(Edit4.Text);
j:=StrToFloat(Edit5.Text);

Edit6Text:=FloatToStr((StrToFloat(ComboBox1.Text)+1)*sqrt((m*k)/(u*n*f))); end;

procedure TForm1.FormCreate(Sender: TObject); var i: integer; mas: array
[1..50] of real; begin

for i:=1 to 50 do begin
mas[i]:=1/(random(50)+0.01);
ComboBox1.Items.Add(FloatToStr(mas[i])); end; end;

end.

```

Form1

Межосевое расстояние зубчатых передач

$$Q_w = (K_a + 1) \sqrt{(M_2 * K_{H\beta}) / (U * (J_n) * F_b a)}$$

Ввод данных

M2 "Вращающий момент"	2
K _{Hβ} "Коэффициент нагрузки"	0,55
U "Передаточное отношение"	10
F _{βa} "Коэффициент ширины"	5
J "Допустимое напряжение"	0,47
K _a "Коэффициент"	0,0666222518321119

Расчет

0,336607758737867

Вывод: Я получил практические навыки в разработке инженерных программ.

Практическая работа №18

Тема: Разработка программ для решения технологических задач.

Цель: Получить практические навыки в разработке технологических программ.

Оборудование: IBM PC.

Программное обеспечение: Borland Delphi 7.0

Порядок выполнения работы:

1. Для данного типа детали назначить режущий инструмент и станок для технологической операции.
2. Назначить припуски на механическую обработку.
3. Исходя из размера детали назначить соответствующий инструмент.
4. Отладить программу, привести пример, сделать вывод.

Пример выполнения работы:

Программный код:

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls;

Type

TForm1 = class(TForm)

```

ListBox1:TListBox;
ComboBox1: TComboBox;
Edit1: TEdit;
Button1:TButton;
Label1: TLabel;
Label2: TLabel;
Edit2: TEdit;
Label3: TLabel;
Edit3: TEdit;
Label4: TLabel;
ListBox2: TListBox;
Memo1: TMemo;
Button2: TButton;
SD:TSaveDialog;
  procedure Button1Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
private
  {Private declarations}
public
  {Public declarations }
end;

var Form1: TForm1;

implementation {$R*.dfm}

procedure TForm1.Button1Click(Sender: TObject);

```

```

var n:integer;
begin
n:=strtoint(combobox1 .Text);
case n of
1: begin Edit1.Text:=ListBox1.Item[2];
    Edit2.Text:=ListBox2.Items[0];
    Edit3.Text:='1,5cm';end;
2: begin Edit1.Text:=ListBox1.Items[4];
    Edit2.Text:=ListBox2.Items[2];
    Edit3.Text:='2,0 cm'; end;
3: begin Edit1.Text:=ListBox1.Items[3];
    Edit2.Text:=ListBox2.Items[4];
    Edit3.Text:='1,2cm';end; end;
Memo1.Lines.Add('№ '+ComboBox1 .Text+'; Поверхность '+Edit1. Text+'; Станок '+Edit2.Text+';
Припуски '+Edit3.Text);
end;

procedure TForm1 .Button2Click(Sender: TObject); begin
if sd.Execute then
memo1.Lines.SaveToFile(sd.FileName); end;

end.

```

№	Поверхность	Станок	Припуски
1	Отверстие	Сверлильный	1,5 см

Action

- Цилиндрическая
- Коническая
- Отверстие
- Отверстие с резьбой
- Плоская

Save

- Сверлильный
- Точильный
- Шлифовальный
- Фрейзерный
- Сверлильно-токарный

№ 2; Поверхность Плоская; Станок Шлифовальный; Припуски 2,0 см
 № 1; Поверхность Отверстие; Станок Сверлильный; Припуски 1,5 см

Вывод: получить практические навыки в разработке технологических программ.

Практическая работа №19

Тема: Разработка программного обеспечения для решения банковских задач

Цель работы: Получить практические навыки в разработке программного обеспечения для решения банковских задач.

Теоретическая часть: Внедрение электронных систем обработки и передачи информации приобретают универсальный и всеобщий характер, охватив все направления банковской деятельности. Современные информационные технологии позволяют координировать деятельность подразделений банков, расширить межбанковские связи, действовать

однократно на финансовых рынках ряда стран. Новые возможности автоматизации банковских операций рабочих мест специалистов, информационных технологий банковских услуг позволяют комплексно решать проблемы анализа банковской деятельности, разработки и создания региональных, межрегиональных и международных банковских систем.

Автоматизация информационных и других технологий банка содействует решению стратегических задач. Главными направлениями, по которым автоматизация обслуживания клиентов воздействует на конкурентную позицию банков, являются уменьшение издержек и увеличение качества обслуживания.

Достижения электронной техники и технологии предложили принципиально новый вид услуг - систему автоматизированного управления наличностью денежной массы. Эта система дает операционную информацию о состоянии всех банковских счетов, позволяет регулировать и прогнозировать движение денежных средств, уменьшить трудоемкость работ с наличными деньгами, переходить на безбумажную технологию.

Положительные аспекты безбумажной технологии:

1. практически мгновенная пересылка данных;
2. уникальность хранения;
3. улучшенная защищенность;
4. резкое уменьшение трудоемкости обработки документов.

Решение аналитических задач банковской деятельности диктуется необходимостью определения финансовых результатов, прогнозирования направления развития, оценкой экономической обоснованности и целесообразности деятельности каждого банка. В отечественной практике автоматизация аналитических расчетов воплощается пока в отдельных программных продуктах и еще не получили широкого распространения.

На отечественном рынке сформировались классы АБС, каждая из которых имеет определенных потребителей от начинающих банков, осуществляющих лишь ограниченный спектр рублевых операций, до ведения банков, вышедших на зарубежный уровень объема и услуг. АБС содержит необходимый потребителю набор функций.

Основная масса предлагаемых на отечественном рынке АБС по стоимости и требованиям к вычислительным средствам рассчитана на средние и небольшие банки. Расширение систем происходит, как правило, в том случае, если возрастающие запросы потребителя носят потенциально массовый характер для российских условий. Поэтому по мере роста финансовых возможностей банков можно ожидать увеличение спроса на более совершенные (многоплатформенные) системы, техническое и программное обеспечение которое потребует все возрастающей квалификации специалистов.

Почти все фирмы - разработчики содержат новые версии АБС, обобщающие предыдущий опыт, ориентированные на те же программно-активные средства, но с улучшенной архитектурой и большим спектром возможностей. На рынке АБС нет единого магистрального направления их развития, а появление новых классов систем в значительной мере определяется интенсивностью и особенностями развития банковского дела в стране.

Информационное обеспечение АБС - информационная модель банка бывает двух видов.

1) Внемашинное ИО - вся совокупность информации в банке, включая системы показателей, методы классификации и кодирования элементов информации, документов, документооборота информационных потоков.

2) Внутримашинное ИО - предоставление данных на машинных носителях в виде разнообразных по созданию, по назначению и специфическим обработкам органических массивов (файлов), БД и их информационных связей.

Современные системы банковских показателей складываются из показателей видов банковских услуг и банковской деятельности, которая отражает расчетно-кассовый, кредитный, депозитный, бухгалтерских, нормативный, законный, фондовый, инвестиционный и другие аспекты функционирования банка. Показатели банковской деятельности характеризуют соотношения депозитов, кредитов, собственных и привлеченных средств, долю межбанковских операций в общем объеме ресурсов и вложений, определяют удельный вес и значимость тех или иных операций, их использование позволяет выявить возможности увеличения прибыльности и кредитоспособности банка. Спецификой деятельности банков является широкий диапазон их клиентуры - это предприятия и организации всех отраслей экономики, в т.ч. страховые органы, бюджетные и внебюджетные структуры, а только население и, следовательно, большое разнообразие показателей.

Автоматизация банковских операций требует приведения всего множества показателей в единую целостную систему, установления четкой взаимосвязи между ними. Значительная роль при этом игровая классификация и кодирование, позволяющее обеспечить сжатие прозрачной части показателей, уменьшить объем и время на поиск информации, облегчить обработку информации.

В банковской деятельности для систематизации информации используются различного рода классификаторы: как ЕСКК, так и отраслевые (банковские): коды ценностей - банкнот, монет, чеков, акций и т.д.; коды валют, банков и т.д., так и локальные (в рамках отдельного банка), например, классификаторы банковских счетов, регистрационных номеров, внутрибанковских номеров клиентов и т.д.

Использование кодов и идентификаторов (компактное обозначение элементов данных) значительно уменьшит трудоемкость поиска, хранения, передвижения, обработки информации, увеличит эффективность автоматизации, экономит дорогостоящие ресурсы памяти и технических средств и увеличит степень безопасности и защиты данных.

Значительную долю немашинного ИО занимает документация. Унифицированные типовые документы в банковской системе увеличивают эффективность автоматизации (платежные поручения, чеки, приходные и расходные кассовые ордера и др.).

2. Порядок выполнения работы:

2.1 Разработать форму для программирования банковской задачи.

2.2 Разработать исходный код.

2.3 Выполнить отладку и тестирование программы.

2.4 Привести пример выполнения.

3. Контрольные вопросы

3.1 Какие средства программирования используют в банковских программах?

3.2 По каким принципам формируются базы данных в банковских программах?

3.3 Назовите основные классы банковских задач

Практическая работа №20

Тема: «Разработка программного обеспечения для решения графических задач».

Цель работы: Закрепить навыки применения графики в приложениях.

Оборудование: IBM PC.

Программное обеспечение: Borland Delphi6.

Порядок выполнения работы:

1. Написать исходный код программы по построению столбчатой диаграммы для финансово-учетных задач
2. Отладить программу
3. Привести пример выполнения
4. Сделать вывод, сохранить проект
5. Оформить отчет

Пример выполнения работы:

```
procedure TForm1.Button2Click(Sender: TObject);
var
i: integer;
d: String;
s: real;
begin
PaintBox1.Canvas.Brush.Color:=clBtnFace; PaintBox1.Canvas.Rectangle(-5, -5, 900, 900);
PaintBox1.Canvas.Pen.Color:=clBlack; PaintBox1.Canvas.Pen.Width:=4;
PaintBox1.Canvas.MoveTo(50, 50);
PaintBox1.Canvas.LineTo(50, 420) ;
PaintBox1.Canvas.LineTo (720, 420) ;
d:=DateToStr(StrToDate('1.'+IntToStr(ComboBox1.ItemIndex+1) + '. ' + ComboBox2.Text)+31);
d:=copy(DateToStr(StrToDate('1.'+copy(d, 4, length(d))))-1,1,2);
for i:=0 to StrToInt(d)-1 do
begin
s:=-0;
try
Table1.Filter := ' KDate = '' + DateToStr (StrToDate
(IntToStr (i+1) +'.'+ IntToStr
(ComboBox1.ItemIndex +1) + 'T'+ComboBox2.Text)) + ''';
Table1.Filtered:=-True;
Table1.First;
while not Table1.EOF do

begin
s:=s+StrToFloat(Copy(Table1.Fields[6].Text, 1,
```

```

    length(Table1.Fields[6].Text)-2));
    Table1.Next;
end;
except
end ;
Table1.Filtered:=False;
PaintBox1.Canvas.Brush.Color:=clBtnFace;
PaintBox1.Canvas.TextOut (i*21+53, 425, IntToStr(i+1));PaintBox1.Canvas.Brush.Color:=clWhite;
PaintBox1.Canvas.Rectangle(i*21+5Q, round(420-s/15), i*21+71, 420) ;
if s>0 then begin
    PaintBox1.Canvas.Brush.Color:-clBtnFace;
    PaintBox1.Canvas.TextOut(i*21+50, round(420-s/15)-20, FloatToStrfs));
end;
end;
end;
end.

```

Вывод: Закрепил навыки работы с графикой в приложениях.

Практические работы по МДК 02.02 Инструментальные средства разработки ПО

Практическая работа №1

Тема: Инструментальные средства для разработки динамических структур данных.

Цель: Получить практические навыки в использовании программных средств для разработки динамических структур данных.

Оборудование: Персональный компьютер, среда программирования Delphi.

1. Теоретическая часть

В IBM PC-совместимых компьютерах память условно разделена на сегменты. Компилятор формирует сегменты кода, данных и стека, а остальная доступная программе память называется динамической (хипом). Динамические переменные создаются в хипе во время выполнения программы. Обращение к ним осуществляется через указатели. С помощью этих переменных можно обрабатывать данные, объем которых до начала выполнения программы не известен. Память под такие данные выделяется блоками, которые связываются друг с другом. Этот способ хранения данных называется динамическими структурами.

Указателем называется переменная, предназначенная для хранения адресов областей памяти. В указателе можно хранить адрес данных или программного кода. Адрес занимает четыре байта и хранится в виде двух слов, одно из которых определяет сегмент, второе — смещение. Указатели делятся на стандартные и определяемые программистом. Для указателей определены только операции проверки на равенство и неравенство и присваивания. Правила присваивания указателей: Любому указателю можно присвоить стандартную константу nil, которая означает, что указатель не ссылается на какую-либо конкретную ячейку памяти. Указатели стандартного типа pointer совместимы с указателями любого типа. Указателю на конкретный тип данных можно присвоить только значение указателя того же или стандартного типа.

Динамические переменные

Динамические переменные создаются в хипе во время выполнения программы с помощью подпрограмм `new` или `getmem`. Динамические переменные не имеют собственных имен — к ним обращаются через указатели.

Процедура `new(var p : тип_указателя)` выделяет в динамической памяти участок размера, достаточного для размещения переменной того типа, на который ссылается указатель `p`, и адрес начала этого участка заносит в этот указатель. Функция `new(тип_указателя) : pointer` выделяет в динамической памяти участок размера, достаточного для размещения переменной базового типа для заданного типа указателя, и возвращает адрес начала этого участка. `new` применяется для типизированных указателей. Процедура `getmem(var p : pointer; size : word)` выделяет в динамической памяти участок размером в `size` байт и присваивает адрес его начала указателю `p`. Если выделить требуемый объем памяти не удалось, программа аварийно завершается. Указатель может быть любого типа. Если до начала работы с данными невозможно определить, сколько памяти потребуется для их хранения, память следует распределять во время выполнения программы по мере необходимости отдельными блоками. Блоки связываются друг с другом с помощью указателей. Такой способ организации данных называется динамической структурой данных, поскольку она размещается в динамической памяти и ее размер изменяется во время выполнения программы. Из динамических структур в программах чаще всего используются линейные списки, стеки, очереди и бинарные деревья. Они различаются способами связи отдельных элементов и допустимыми операциями. Динамическая структура, в отличие от массива или записи, может занимать несмежные участки оперативной памяти. Динамические структуры широко применяют и для более эффективной работы с данными, размер которых известен, особенно для решения задач сортировки. Элемент любой динамической структуры состоит из двух частей: информационной, ради хранения которой и создается структура, и указателей, обеспечивающих связь элементов друг с другом.

Стеки

Стек является простейшей динамической структурой. Добавление элементов в стек и выборка из него выполняются из одного конца, называемого вершиной стека. Другие операции со стеком не определены. При выборке элемент исключается из стека.

Очереди

Очередь — это динамическая структура данных, добавление элементов в которую выполняется в один конец, а выборка — из другого конца. Другие операции с очередью не определены. При выборке элемент исключается из очереди. Говорят, что очередь реализует принцип обслуживания FIFO (first in — first out, первым пришел — первым обслужен). В программировании очереди применяются очень широко — например, при моделировании, буферизованном вводе-выводе или диспетчеризации задач в операционной системе.

2. Ход работы:

- 2.1 Создать новый тип данных в разделе type-динамическую запись.
- 2.2 Поместить на форму компоненты Edit и заполнить через них динамическую запись.
- 2.3 В процессе работы с динамической записью взять память оператором New и очистить память по окончании работы оператором dispose.
- 2.4 Отобразить заполненные поля динамической записью в редакторе Memo1.

3. Контрольные вопросы:

- 3.1 Что такое указатели.
- 3.2 Что такое типизированные и нетипизированный указатель.
- 3.3 С какой целью выполняют управление оперативной памятью компьютера.

Практическая работа №2

Тема: Использование среды Delphi для исследования числовых типов данных.

Цель: Получить практические навыки в исследовании числовых типов данных.

Оборудование: Персональный компьютер, среда программирования Delphi.

1. Теоретическая часть

С помощью типов данных программист указывает компилятору, как хранить информацию в программе. При объявлении переменной необходимо указать ее тип. Одни типы уже определены в языке, другие программисту приходится задавать самому. В ранних языках программирования допускалось ограниченное число типов данных, и Pascal оказался одним из первых языков, допускающих определение в программе новых типов.

2. Ход работы:

- 2.1 Для заданного типа данных определить размер типа в байтах используя функцию Sizeof.
- 2.2 Определить верхнее значение и нижнее значение типа данных используя функции high и Low.
- 2.3 Определить возможность присваивания отрицательных значений, обработав исключительные ситуации try->except.
- 2.4 Определить возможность присваивания вещественному типу целого типа и наоборот, обработав исключительные ситуации try->except. Присвоить целому типу вещественный воспользовавшись функциями trunc и round.
- 2.5 Определить предыдущее и последующее значение, используя функцию succ и pred, обработав исключительные ситуации.
- 2.6 Вывести результаты исследования на форму в текстовые редакторы Edit.

3. Контрольные вопросы:

- 3.1 Назовите числовые типы данных в среде Delphi.

- 3.2 Что отличает целочисленный тип данных от вещественного.
- 3.3 Охарактеризуйте тип Byte.

Практическая работа №3

Тема: Инструментальные средства для отладки программ.

Цель: Получить практические навыки работы с интегрированным отладчиком в среде Delphi.

Оборудование: Персональный компьютер, среда программирования Delphi.

1. Теоретическая часть

Помимо интеллектуального редактора текста профессиональная среда программирования обычно содержит инструментальные средства отладки. Инструментальные средства отладки призваны дать разработчику максимально ясное представление о том, как работает его программа. И уже искусство разработчика состоит в том, чтобы, используя все имеющиеся в его распоряжении средства, быстро выявить ошибки. Набор средств отладки в Access широк: это и специальное меню Debug (Отладка), и во многом дублирующие его кнопки на панели инструментов, и специальные окна отладки. Далее кратко дается описание каждого средства.

Меню Debug

Меню Debug (Отладка) и специальная панель инструментов Debug (Отладка) представлены на рис. 13.25.

Назначение команд меню описано в табл. 13.11.



Рис. 13.25. Меню и панель инструментов Debug

Таблица 13.11. Команды меню Debug.

Команда	Назначение
Compile (Компиляция)	Компилирует все модули в текущей базе данных
Step Into (Шаг с за ХОДОМ)	Исполняет очередную строку кода с заходом в процедуры

Команда	Назначение
Step Over (Шаг с обходом)	Исполняет очередную строку кода без захода в процедуры, т. е. функции и процедуры выполняются за один шаг
Step Out (Шаг с выходом)	Выполняет остаток текущей процедуры и останавливается в вызывающей программе на следующей строке после вызова этой процедуры
Run to Cursor (Запуск до курсора)	Выполняются все строки кода от текущей строки до строки, в которой установлен курсор. Останавливается перед этой строкой
Add Watch (Добавление контрольного значения)	Открывает окно Добавление контрольного значения
Edit Watch (Изменение контрольного значения)	Открывает окно Изменение контрольного значения
Quick Watch (Быстрый просмотр)	Выводит в специальном окне текущее значение выражения в точке останова
Toggle Breakpoint (Установка/сброс точек останова)	Устанавливает/снимает точку останова на строку, в которой находится курсор
Clear All Breakpoints (Сброс всех точек останова)	Снимает все точки останова, установленные в данном модуле
Set Next Statement (Установка следующего предложения)	Устанавливает очередную выполняемую команду на строку, в которой находится курсор
Show Next Statement (Показ	Отображает в окне редактора очередную команду для выполнения

Команда	Назначение
следующего предложения)	

Кнопки на инструментальной панели в основном повторяют описанные команды. Это видно из значков, находящихся рядом с командой и на кнопках. Кроме того, здесь существуют кнопки, соответствующие меню Run:

Run Sub/UserForm – продолжает выполнение процедуры после точки останова, снимая при этом пошаговый режим, если он был установлен;

Break – прекращает выполнение процедуры;

Reset (Сброс) – прекращает выполнение процедуры и присваивает переменным начальные значения по умолчанию.

Еще несколько кнопок инструментальной панели позволяют открыть специальные окна отладки.

2. Ход работы:

- 2.1 Взять разработанный код программ из практической работы №3 и задать в ней точки остановки программы.
- 2.2 Наблюдать значение переменных в программе. Для чего добавить имя переменной в список наблюдаемых элементов (Watch List). Из меню Run выбрать команду AddWatch и в поле Expression появившегося диалогового окна Watch Propertie ввести имя переменной. Или после того, после того как программа достигла точки остановки, в раскрывшемся окне редактора кода установить курсор мыши на имени переменной, которую нужно проверить. В окне редактора кода появится окно подсказки, в котором будет выведено значение переменной.
- 2.3 Оформить отчет, сделать вывод.

3. Контрольные вопросы:

- 3.1 Дайте классификация программных ошибок.

- 3.2 Что такой отладка программы? С какой целью она проводится.
- 3.3 Назовите основные функции интегрированного отладчика.

Практическая работа №4

Тема: «Инструментальные средства для дистанционного обучения».

Цель работы: Изучить назначение и основные возможности программы Adobe Acrobat Connect

Оборудование: ПК, загрузочный файл

1. Теоретическая часть

Формы организации дистанционных занятий

Чат-занятия — учебные занятия, осуществляемые с использованием чат-технологий. Чат-занятия проводятся синхронно, то есть все участники имеют одновременный доступ к чату. В рамках многих дистанционных учебных заведений действует чат-школа, в которой с помощью чат-кабинетов организуется деятельность дистанционных педагогов и учеников.

Веб-занятия — дистанционные уроки, конференции, семинары, деловые игры, лабораторные работы, практикумы и другие формы учебных занятий, проводимых с помощью средств телекоммуникаций и других возможностей «Всемирной паутины».

Для веб-занятий используются специализированные образовательные веб-форумы — форма работы пользователей по определённой теме или проблеме с помощью записей, оставляемых на одном из сайтов с установленной на нём соответствующей программой.

От чат-занятий веб-форумы отличаются возможностью более длительной (многодневной) работы и асинхронным характером взаимодействия учеников и педагогов.

Телеконференция — проводится, как правило, на основе списков рассылки с использованием электронной почты. Для учебных телеконференций характерно достижение образовательных задач.

Также существуют формы дистанционного обучения, при котором учебные материалы высылаются почтой в регионы.

В основе такой системы заложен метод обучения, который получил название «Природный процесс обучения» (англ. natural learning manner). Дистанционное обучение — это демократичная простая и свободная система обучения. Сейчас активно используется жителями Европы для получения дополнительного образования. Студент, постоянно выполняя практические задания, приобретает устойчивые автоматизированные навыки. Теоретические знания усваиваются без дополнительных усилий, органично вплетаясь в тренировочные упражнения. Формирование теоретических и практических навыков достигается в процессе систематического изучения материалов и прослушивания и повторения за диктором упражнений на аудио и видеоносителях (при наличии)...

Телеприсутствие. Существует много различных способов дистанционного обучения. Например дистанционное присутствие с помощью робота R.Bot 100. Сейчас в Москве в одной из школ, идёт эксперимент по такому виду дистанционного обучения. Мальчик-инвалид, находясь дома за компьютером, слышит, видит, разговаривает при помощи робота. Учитель задаёт ему вопросы, он отвечает. При этом и учитель видит ученика, потому что на роботе находится монитор. При этом у мальчика создаётся почти полное впечатление, что он находится в классе вместе со своими сверстниками на уроке. На переменах он может также общаться со своими одноклассниками. Если эксперимент станет удачным, он может открыть дорогу большому проекту по внедрению такого метода дистанционного обучения по всей России.

Дистанционное обучение (ДО) — взаимодействие учителя и учащихся между собой на расстоянии, отражающее все присущие учебному процессу компоненты (цели, содержание, методы, организационные формы, средства обучения) и реализуемое специфичными средствами Интернет-технологий или другими средствами, предусматривающими интерактивность.

Дистанционное обучение — это самостоятельная форма обучения, информационные технологии в дистанционном обучении являются ведущим средством

Adobe Acrobat Connect — программное обеспечение для веб-конференций, которое позволяет отдельным лицам и малым предприятиям мгновенно общаться и сотрудничать через простой в использовании онлайн-доступ. Adobe Acrobat Connect является частью семейства Adobe и состоит из набора модулей:

- **Adobe Connect Pro Meeting** – Средство организации совещаний и семинаров по сети в реальном времени. Позволяет пользователям проводить презентации, обмениваться файлами, потоковым аудио, видео, а также служит средством для организации многопользовательских видеоконференций. Вы можете сохранять уже созданные виртуальные переговорные комнаты и их содержимое для последующего быстрого доступа к ним – такая возможность значительно сокращает время подготовки к семинарам, переговорам и проведению презентаций.
- **Adobe Connect Pro Training** – Средство, позволяющее создавать, управлять, проводить и отслеживать курсы дистанционного обучения. Позволяет разрабатывать учебные программы, которые могут сочетать в себе как индивидуальные учебные планы на основе курсов, созданных с помощью Adobe Presenter, так и материалы сторонних производителей, и интерактивное обучение под руководством преподавателя.
- **Adobe Connect Pro Events** – Средство управления жизненным циклом всех событий, относящихся к участию во встречах и тренингах, таких как оценка обучающихся, регистрация на курсы, уведомления и отчетность.

2. Ход работы:

- 2.1 Изучить и описать назначение и составляющие программы Adobe Acrobat Connect
- 2.2 Описать возможности проведения конференций и презентаций
- 2.3 Описать возможности дистанционного обучения

3. Ответить на контрольные вопросы:

- 3.1 Основные преимущества дистанционного обучения
- 3.2 Как организовать WEB-конференцию?
- 3.3 Как организовать WEB-семинар?
- 3.4

Практическая работа №5

Тема: «Инструментальные средства для создания анимаций».

Цель работы: Получить практические навыки в создании анимированных изображений.

Оборудование: ПК, программа для создания анимаций GIMP 2.8

4.

Теоретическая часть

Компьютерная анимация

Компьютерная анимация - это получение движущихся изображений на экране дисплея. Художник создает на экране рисунки начального и конечного положения движущихся объектов, все промежуточные состояния рассчитывает и изображает компьютер, выполняя расчеты, опирающиеся на математическое описание данного вида движения. Полученные рисунки, выводимые последовательно на экран с определенной частотой, создают иллюзию движения. Мультимедиа - это объединение

высококачественного изображения на экране компьютера со звуковым сопровождением. Наибольшее распространение системы мультимедиа получили в области обучения, рекламы, развлечений.

Для того, чтобы рисованный или объемный персонаж на экране ожил, его движение разбивают на отдельные фазы, а затем снимают на киноплёнку. Если внимательно посмотреть на отснятую плёнку, видно, что в каждом кадре положение персонажа чуть-чуть отличается от предыдущего и последующего кадра, это и создает при проекции на экран иллюзию движения, основанную на способности сетчатки человеческого глаза удерживать изображение в течение некоторого времени, пока на него не накладывается следующее изображение.

Компьютерная анимация - вид анимации, создаваемый при помощи компьютера. На сегодня она получила широкое применение как в области развлечений, так и в производственной, научной и деловой сферах. Являясь производной от компьютерной графики, анимация наследует те же способы создания изображений: векторная графика, растровая графика, фрактальная графика, трехмерная графика (3D).

По принципу анимирования можно выделить несколько видов компьютерной анимации :

Анимация по ключевым кадрам. Расстановка ключевых кадров производится аниматором. Промежуточные же кадры генерирует специальная программа. Этот способ наиболее близок к традиционной рисованной анимации, только роль фазовщика берет на себя компьютер, а не человек.

Запись движения. Данные анимации записываются специальным оборудованием с реально двигающихся объектов и переносятся на их имитацию в компьютере. Распространённый пример такой техники – Motion capture (захват движений). Актеры в специальных костюмах с датчиками совершают движения, которые записываются камерами и анализируются специальным программным обеспечением. Итоговые данные о перемещении суставов и конечностей актеров применяют к трёхмерным скелетам виртуальных персонажей, чем добиваются высокого уровня достоверности их движения.

Такой же метод используют для переноса мимики живого актера на его трёхмерный аналог в компьютере. Процедурная анимация полностью или частично рассчитывается компьютером. Сюда можно включить следующие её виды: симуляция физического взаимодействия твёрдых тел; имитация

движения систем частиц, жидкостей и газов; имитация взаимодействия мягких тел (ткани, волос); расчёт движения иерархической структуры связей (скелета персонажа) под внешним воздействием; имитация автономного (самостоятельного) движения персонажа.

Программы для создания анимации с помощью цифрового фотоаппарата. Сегодня программное обеспечение, позволяющее задействовать цифровой фотоаппарат для съёмки анимации, применяется также часто, как и ставшие привычными 3D- или 2D-пакеты. Любая программа такого типа обеспечивает управление цифровым фотоаппаратом через компьютер и работу с полученными кадрами.

Хранение. Компьютерная анимация может храниться в универсальных графических файлах (например, в формате GIF) в виде набора независимых изображений, либо в специализированных файлах соответствующих пакетов анимации (3ds Max, Blender, Maya и т. п.) в виде текстур и отдельных элементов, либо в форматах, предназначенных для просмотра (FLIC (англ.)) и применения В играх (Bink). Также, анимация может сохраняться в форматах, предназначенных для хранения видео (например, MPEG-4).

5. Ход работы:

1. Установить программу GIMP 2.8
2. Создать рисунок первого слоя (файл -> создать -> Ок) с помощью графического редактора, имеющегося в программе.
3. Создать рисунки последующих слоев, для чего создать новый слой (***), задав типы заливки слоя -> белый.
4. Сохранить последующие слои в файле на рабочем столе, задав время смены слоев. (файл -> export as) выбрав тип файла изображений GIF -> экспортировать установив соответствующие флажки:
 - Сохранить как анимацию
 - Задать бесконечный цикл, нажав кнопку экспорт

6. Контрольные вопросы:

- 6.1 Как создаются движущиеся изображения на экране компьютера?
- 6.2 В каких файлах хранится компьютерная информация?
- 6.3 Какие бывают виды компьютерной анимации?

Практическая работа №6

Тема: Инструментальные средства для разработки сайтов.

Цель: Изучить основные правила Web-дизайна и получить практические навыки в разработке сайтов

Оборудование: Персональный компьютер, файл WEB-дизайна.

1. Теоретическая часть

Информация, доступная пользователям Internet, располагается на компьютерах (Web-серверах), на которых установлено специальное программное обеспечение. Значительная часть этой информации организована в виде Web-сайтов. Web-сайт – это информация, представленная в определенном виде, которая располагается на Web-сервере и имеет свое имя (адрес).

Этапы разработки Web-сайта

Выделяют следующие этапы разработки Web-сайта: планирование, реализация, публикация, продвижение, поддержка. На стадии планирования определяется следующее: • цели создания Web-сайта (Зачем? Какие задачи он должен выполнять и на какую аудиторию он рассчитан?); • характер содержимого; • структура (Юзабилити – удобство пользования); • особенности оформления (определяется структура каждой страницы и разрабатывается графика).

Любую страницу можно оценить по трем параметрам: контенту, внешнему виду и навигации. Если сайт рассчитан на долгое вдумчивое чтение, то он должен иметь хорошую читабельность, меньше отвлекающих динамических эффектов, не утомляющее цветовое сочетание фона и текста. И наоборот: сократите текстовые блоки до минимума, если Вы создаете сайт, дающий посетителю, прежде всего визуальную и другую мультимедийную информацию. При создании достаточно больших документов надо помнить о том, что не все из посетителей имеют высокоскоростной доступ в Internet.

Реализация

Это и есть работа по созданию сайта. На этом этапе проводится подготовка текстового и графического материала (печать, сканирование). Материал разбивается по файлам в соответствии со структурой. Организуются ссылки между файлами сайта. Рекомендуется создать шаблон-заготовку страницы с основными структурными областями и стилевым оформлением и использовать ее для создания всех страниц узла. Меняйте в каждой новой страницетолько содержимое и адресацию ссылок, такая организация работы сократит время, потраченное на каждую из них.

Тестирование

Завершив работу по размещению страниц на Web-сайте, необходимо выполнить тестирование. Оно состоит из двух этапов: тестирование на работоспособность и тестирование на удобство пользования интерфейсом.

На этапе тестирования на работоспособность проверяют, как функционирует Web-сайт, используя те же условия, при которых с ним будет работать пользователь. Поработайте с Web-сайтом в различных браузерах и посмотрите, как выглядит Ваш сайт в каждом из них. Постарайтесь оценить время загрузки страниц, что очень важно.

Публикация

Готовый Web-сайт необходимо опубликовать на Web-сервере, чтобы он был доступен через Internet. Если ваш сайт создан посредством редактора FrontPage, то на сервере должны быть установлены серверные

расширения FrontPage, что обеспечит полную поддержку доступных в FrontPage компонентов, которые были помещены на странице в процессе создания сайта. Если у вас нет собственного сервера, то в Сети можно найти огромное количество ссылок на free web pages, где некоторые провайдеры предоставляют своим клиентам бесплатное место под страницу.

Рекламирование сайта

Существует множество приемов рекламирования сайта: размещение информации о нем на поисковом Web-сайте, организация взаимных ссылок с другими сайтами и т.д. Как привлечь пользователя? Красиво оформленные страницы Web-сайта – это только половина дела. В первую очередь страницы должны быть содержательными. Основное требование к содержимому Web-сайта – полнота и достоверность. Информация должна быть представлена таким образом, чтобы пользователь, однажды посетивший Web-сайт, еще ни раз обратился к нему.

Сопровождение сайта

Содержимое Web-сайта может подвергаться неоднократным изменениям. Важно, чтобы предоставляемая на Web-сайте информация всегда была актуальной, поэтому как можно чаще обновляйте информацию на своем Web-сайте, по возможности расширяйте материал, улучшайте дизайн. Обязательное правило. Web-сайт должен обновляться не реже одного раза в месяц. В противном случае вы потеряете не только потенциальных, но и уже состоявшихся посетителей. Рекомендуется создать на своем компьютере копию Web-сайта, вносить в нее изменения и новую версию передавать для размещения на сервере в завершенном виде.

2. Ход работы:

Изучить сайт по WEB дизайну и составить отчет, в котором описать следующие разделы:

- 2.1 Основные аспекты WEB-дизайна.
- 2.2 Юзабилити сайта
- 2.3 Макетирование сайта.
- 2.4 Система навигации по сайту.
- 2.5 Дизайн в интернете.
- 2.6 Этапы разработки WEB-сайта.

- 2.7 Подготовка иллюстраций и логических элементов.
- 2.8 Обзор форматов.
- 2.9 Обзор HTML редакторов.

3. Контрольные вопросы:

- 3.1 Назовите основные этапы разработки WEB-сайта.
- 3.2 Что включают в себя основные аспекты WEB-дизайна.
- 3.3 Дайте сравнительную характеристику HTML редакторов.

4.

Практическая работа №7

Тема: Инструментальные средства для IP телефонии.

Цель: Ознакомиться с основными сведениями по IP телефонии.

Оборудование: Персональный компьютер, загрузочный файл с презентацией программы Skype.

1. Теоретическая часть

Skype — бесплатное программное обеспечение с

закрытым кодом, обеспечивающее шифрованную голосовую связь через Интернет между компьютерами (VoIP), а также платные услуги для звонков на мобильные и стационарные телефоны. Программа также позволяет совершать конференц - звонки (до 25

голосовых абонентов, включая инициатора), видеозвонки (в том числе видеоконференции до 10 абонентов), а также обеспечивает передачу текстовых сообщений (чат) и передачу файлов. Есть возможность вместо изображения с веб-камеры передавать изображение с экрана монитора. Компания Skype была основана двумя предпринимателями — Никласом Зеннстрёмом из Швеции и Янусом Фриисом из Дании. Авторами программного

обеспечения стали эстонские программисты Ахти Хейнла, Прийт Казесалу и Яан Таллинн. В отличие от многих других программ IP-телефонии, для передачи данных Skype

использует P2P-архитектуру. Каталог пользователей Skype распределён по компьютерам пользователей сети Skype, что позволяет сети легко масштабироваться до очень больших размеров (в данный момент более 100

миллионов пользователей, 15—25 миллионов онлайн) без дорогой инфраструктуры централизованных серверов.

Единственным центральным элементом для Skype является сервер идентификации, на котором хранятся учётные записи пользователей и резервные копии их списков контактов. Центральный сервер нужен только для установки связи. После того как связь установлена, компьютеры пересылают голосовые данные напрямую друг другу (если между ними есть прямая связь). В частности, если два компьютера, находящиеся внутри одной локальной сети, установили между собой Skype-соединение, то связь с Интернетом можно прервать и разговор будет продолжаться вплоть до его завершения пользователями или какого-либо сбоя связи внутри локальной сети.

При установке соединения между ПК данные шифруются при помощи AES-256, для передачи ключа которого, в свою очередь, используется 1024-битный ключ RSA. Skype Me - связав пользователей по всему миру с помощью голоса, Skype

дал возможность людям из разных стран общаться друг с другом. Для облегчения этой задачи, Skype предлагает устанавливать статус Skype Me, который указывает, что данный абонент открыт для звонков со всего мира. Установка данного статуса привлекает пользователей, желающих попрактиковаться в иностранном языке, а также мошенников и спамеров, поэтому начиная с 4 версии статус SkypeMe удален из программы. Чат Skype позволяет пользователям общаться с помощью голоса и более традиционным способом с помощью текстовых сообщений (IM-чата). Голосовой чат позволяет как разговаривать с одним пользователем, так и устраивать конференц-связь. Он использует собственные кодеки. Skype-чат позволяет устраивать групповые чаты, посылать смайлики, хранить

историю. SkypeOut (звонки на телефоны) Позволяет совершать исходящие звонки на стационарные и мобильные телефоны в большинстве стран мира. Оплата поминутная, дифференцированная.

SkypeIn (онлайн-номер) Позволяет получать телефонные звонки от пользователей традиционных телефонных сетей. Все входящие звонки на данный номер будут приходить на учётную запись Skype, а при положительном счёте возможна переадресация звонков на любой телефонный номер.

Skype Voicemail (голосовая почта) Услуга запущена 10 марта 2005 года. Позволяет записывать входящие

сообщения, когда пользователь не в сети, и работает как автоответчик.

Номер Skype To Go Это специальный номер доступа, на который можно позвонить с любого телефона (стационарного или мобильного) для того, чтобы связаться с другим номером по выгодным тарифам, денежные средства при этом снимаются со счета Skype.

Отправка SMS Возможность отправлять SMS.

2. Ход работы:

Изучить презентацию программы Skype и описать в ответе:

- 2.1 Обзор программы.
- 2.2 История развития.
- 2.3 Технологии.
- 2.4 Бесплатные услуги.
- 2.5 Платные услуги.

3. Контрольные вопросы:

- 3.1 Назначение программы Skype?
- 3.2 Какие технологии использует Skype?
- 3.3 Какие услуги предлагает программа Skype?

Практическая работа №8

Тема: Централизованная служба мгновенного обмена сообщениями сети интернет-ICQ.

Цель: Ознакомиться с работой службы мгновенного обмена сообщениями - ICQ.

Оборудование: Персональный компьютер, загрузочный файл.

4. Теоретическая часть

ICQ — (I seek You — Я ищу тебя) централизованная служба мгновенного обмена сообщениями сети Интернет.

ICQ и её логотип — достаточно известный и узнаваемый бренд. Логотип представляет из себя стилизованное изображение цветка

ромашки с диском жёлтого цвета и восемью лепестками, семь из которых окрашены в зелёный цвет, а один — в красный.

Это изображение используется не только в качестве логотипа службы, но и в интерфейсе официального клиента для визуализации процесса подключения клиента к серверу, а также как идеограмма статусов присутствия.

Для использования службы требуется зарегистрировать учётную запись, что может быть сделано через интерфейс клиента, а также интернет-портал. Для идентификации пользователей служба использует UIN (Universal Identification Number) — уникальный для каждой учётной записи номер, состоящий из 5-9 арабских цифр. Этот номер присваивается учётной записи при первичной регистрации пользователя в системе, после чего, в паре с паролем, может использоваться для аутентификации в системе.

Для каждой учётной записи служба хранит следующие данные:

никнейм — отображаемое имя пользователя, которое, в отличие от UIN, можно изменять; e-mail, дающий возможность восстановления доступа к аккаунту в случае утраты пароля; публичную информацию, которая может включать имя, фамилию, список увлечений, и т. д.; аватар в формате BMP, JPEG или GIF; список контактов — набор UIN-номеров собеседников; статус присутствия; дополнительный информационный статус. После успешной авторизации клиент ICQ загружает с сервера список контактов пользователя. Контакты в списке могут быть разделены на группы, имена и количество которых изменяются пользователем. При добавлении контакта может потребоваться авторизация — разрешение видеть его статус присутствия и отправлять ему файлы. Максимально можно иметь 1000 контактов

В ICQ реализована передача файлов по технологии Peer-to-peer, то есть при непосредственном интернет-соединении двух компьютеров, минуя сервер. Передача файлов возможна только тогда, когда статус у получателя «В сети». Подобный способ передачи файлов может быть опасен тем, что отправитель узнает IP получателя (или наоборот) или отправит ему вредоносное программное обеспечение. С каждым из контактов можно вести личную переписку. Если отправитель не отключил эту возможность, то, в зависимости от клиента, получатель информируется о наборе сообщения, что создаёт эффект присутствия отправителя. Длина отправляемого сообщения ограничена. В случае, если в момент отправки сообщений адресат не находился в сети, они будут сохранены службой и доставлены адресату, как только

тот подключится к сети. Владельцами службы с момента её появления разрабатывается и предоставляется пользователям бесплатная компьютерная программа-клиент. В настоящее время она представлена в двух версиях: ICQ Lite и ICQ 7.0 (для России программа ICQ 7.0 имеет кастомизацию от компаний Rambler и Яндекс). В графическом интерфейсе клиента присутствует баннерная реклама, исходный код программы закрыт.

5. Ход работы:

Изучить организацию и работу службы ICQ и описать в отчете:

- 5.1 Организация учетной записи.
- 5.2 Статус приветствия.
- 5.3 Список контактов
- 5.4 Обмен сообщениями.
- 5.5 Отправку файлов.
- 5.6 Программу клиент.

6. Контрольные вопросы:

- 6.1 Как организована учетная запись в службе ICQ?
- 6.2 Какие бывают статусы приветствия?
- 6.3 Какие существуют списки контактов?

Практическая работа №9

Тема: «Использование пакетов среды Delphi для отображения хода выполнения длительных операций».

Цель работы: Получить практические навыки в программировании компонентов для отображения длительных операций.

Оборудование: ПК, среда программирования Delphi

7. Теоретическая часть

Отображение хода выполнения длительных операций — компоненты ProgressBar и Gauge

Рассмотрим компоненты ProgressBar со страницы библиотеки Win32 и Gauge со страницы Samples, предназначенные для отображения в стиле Windows 95/98 хода процессов, занимающих заметное время,

например, копирования больших файлов, настройку приложения, установку приложения на компьютере и т.п. Пример возможных вариантов отображения хода процесса компонентами ProgressBar и Gauge



Рисунок 1- отображение хода операций

Основные свойства этих компонентов очень схожи, различаясь только именами:

Свойство ProgressBar	Свойство Gauge	Описание
Max	MaxValue	Максимальное значение позиции (Position, Progress), которое соответствует завершению отображаемого процесса. По умолчанию задается в процентах — 100
Min	MinValue	Начальное значение позиции (Position, Progress), которое соответствует началу отображаемого процесса
Position	Progress	Позиция, которую можно задавать по мере протекания процесса, начиная со значения Min или MinValue в начале процесса, и кончая значением Max или MaxValue в конце. Если минимальное и максимальное значения выражены в процентах, то позиция — это процент завершенной части процесса
Smooth	-	Непрерывное (при значении true) или дискретное отображение процесса. На рис. 1 в горизонтальном компоненте ProgressBar задано Smooth = true, а в вертикальном — false
Step	—	Шаг приращения позиции, используемый в методе StepBy. Значение по умолчанию — 10
Orientation	-	Ориентация шкалы компонента: pbHorizontal — горизонтальная, pbVertical — вертикальная. Если задана ориентация pbVertical, то компонент надо вытянуть по вертикали (на рис. 1 компонент слева)
-	ForeColor	Цвет заполнения
-	ShowText	Текстовое отображение процента выполнения на фоне диаграммы

-	Kind	Тип диаграммы: gkHorizontalBar — горизонтальная полоса, gkVerticalBar — вертикальная полоса, gkPie — круговая диаграмма, gkNeedle — секторная диаграмма, gkText — отображение текстом
---	------	---

Отображение хода процесса можно осуществлять, задавая значение позиции — Position в ProgressBar или Progress в Gauge. Например, если полная длительность процесса характеризуется значением целой переменной Count (объем всех копируемых файлов, число настроек, количество циклов какого-то процесса), K выполненная часть — целой переменной Current, то задавать ■позицию диаграммы в случае, если используются значения минимальной и максимальной позиции по умолчанию (т.е. 0 и 100), можно операторами

ProgressBar.Position := 100 * Current div Count; или

Gauge.Progress := 100 * Current div Count;

соответственно для ProgressBar и Gauge.

Можно поступать иначе: задать сначала значение максимальной величины равным Count, а затем в ходе процесса задавать позицию равной Current. Например:

Gauge.MaxValue := Count;

Gauge.Progress := Current;

Компонент ProgressBar имеет два метода, которыми тоже можно воспользоваться для отображения процесса: StepBy(Delta: Integer) — увеличение позиции на заданную величину Delta, и StepIt — увеличение позиции на один шаг, величина которого задается свойством Step.

8. Ход работы:

5. Изучить назначение и свойства компонентов ProgressBar и Gauge.

6. Поместить на форму компонент `TIMER`, задать свойство `INTERVAL` и запрограммировать процедуру `Procedure TForm1.Timer1 TIMER` по значению компонентов `PROGRESS BAR` И `GAUGE`
Begin
progressBar1.Position:=ProgressBar1.position+1
Gouge1.Progress:=Gouge1.Progress+1
End
7. Наблюдать работу компонентов `ProgressBar` и `Gauge` при отображении длительных процессов.

9. Контрольные вопросы:

- 9.1 С какой целью применяются компоненты `ProgressBar` и `Gauge`?
- 9.2 Назовите основное программируемое свойство этих компонентов.
- 9.3 Как можно изменить скорость работы компонентов `ProgressBar` и `Gauge`.

Практическая работа №10

Тема: «Использование компонентов среды Delphi для воспроизведения немых видеоклипов»

Цель работы: Получить практические навыки при работе с немymi видеоклипами.

Оборудование: ПК, среда программирования Delphi

10. Теоретическая часть

Рассмотрим способ воспроизведения в приложении Delphi' стандартных мультимедий Windows и файлов .avi — клипов, без звукового сопровождения. Это позволяет сделать компонент `Animate`, расположенный на странице Win32 библиотеки.

Компонент `Animate` позволяет воспроизводить на форме стандартные видео клипы Windows (типа копирования файлов, поиска файлов и т.п.) и немые видео файлы .avi — Audio Video Interleaved. Эти

файлы представляют собой последовательность кадров битовых матриц. Они могут содержать и звуковую дорожку, но компонент `Animate` воспроизводит только немые клипы `AVI`.

Откройте новое приложение, перенесите на форму компонент `Animate` и познакомьтесь с ним.

Воспроизводимое им изображение задается одним из двух свойств: `FileName` или `CommonAVI`. Первое из этих свойств, как ясно из его названия, позволяет в процессе проектирования или программно задать имя воспроизводимого файла. А свойство `CommonAVI` позволяет воспроизводить стандартные мультипликации `Windows`. Это свойство объявлено следующим образом:

```
type TCommonAVI = (aviNone, aviFindFolder, aviFindFile,  
aviFindComputer, aviCopyFiles, aviCopyFile, aviRecycleFile, aviEmptyRecycle, aviDeleteFile);  
property CommonAVI: TCommonAVI;
```

Тип `TCommonAVI`, определяет множество predefined в `Windows` мультипликаций типа копирования файлов, поиска файлов, удаления файлов и т.п. Что означает каждое значение `VI` увидите из тестового приложения, которое постройте чуть позже.

А пока установите значение `CommonAVI`, например, равным `HviCopyFile`. Это соответствует стандартному изображению копирования файла. Соответствующий начальный рисунок немедленно появится на вашей форме. Свойство `Repetitions` компонента `Animate` задает число повторений воспроизведения клипа. Если оно равно 0 (значение по умолчанию), то воспроизведение повторяется вновь и вновь до тех пор, пока не будет выполнен метод `Stop`. При выполнении этого метода генерируется событие `OnStop`, которое можно использовать, например, чтобы стереть Изображение — сделать его невидимым.

Если же свойство `Repetitions` задать большим нуля, оно определит число повторений клипа. Задайте его, например, равным 3, А теперь установите свойство `Active` компонента `Animate` в `True`. Вы увидите (рис. 1), что еще в процессе проектирования ваше приложение заработает. Изображение оживет и клип будет повторен 3 раза.

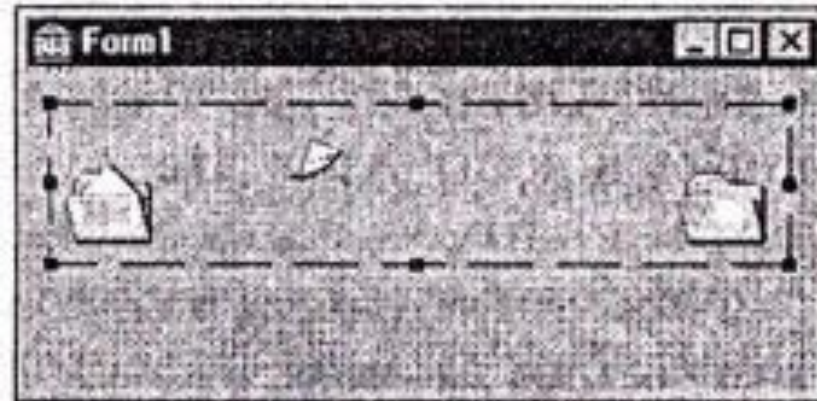


Рисунок 2- анимация копирования файла

Вы можете посмотреть воспроизводимое изображение по кадрам. Для этого щелкните на компоненте правой кнопкой мыши и из всплывшего меню выберите разделы Next Frame (следующий кадр) или Previous Frame (предыдущий кадр). Это позволит вам выбрать фрагмент клипа, если вы не хотите воспроизводить клип полностью. Воспроизвести фрагмент клипа можно, установив соответствующие значения свойств StartFrame — начальный кадр воспроизведения, и StopFrame — последний кадр воспроизведения.

Воспроизводить фрагмент клипа можно и методом Play, который определен следующим образом:

```
procedure Play(FromFrame, ToFrame: Word; Count: Integer);
```

Метод воспроизводит заданную последовательность кадров клипа от FromFrame до ToFrame включительно и воспроизведение повторяется Count раз. Если FromFrame = 1, то воспроизведение начинается с первого кадра. Значение ToFrame должно быть не меньше FromFrame и не больше значения, определяемого свойством FrameCount (свойство только для чтения), указывающим полное число кадров в клипе. Если Count = 0, то воспроизведение повторяется до тех пор, пока не будет выполнен метод Stop.

Выполнение Play идентично заданию StartFrame равным FromFrame, StopFrame равным ToFrame, Repetitions равным Count и последующей установке Active в true.

11. Ход работы:

8. Поместите на форму компонент Animate, расположенный на странице WIN 32 библиотеки WIN32
9. Установите значение AVI равным avi Copy File
10. Задайте свойство Repetitions равным 0. Поменяйте это свойство на значение 3 и число повторений клипа
11. Поменяйте свойство AVI и наблюдайте работу видеоклипа.

12. Контрольные вопросы:

- 12.1 Какой компонент используется в среде Delphi для создания видео клипов, где он находится?
- 12.2 Для чего предназначено свойство FileName?
- 12.3 Как используется свойство AVI?

Практические работы по МДК 02.03 Моделирование и анализ ПО

ПРАКТИЧЕСКАЯ

РАБОТА

№

1

Тема: “Модели IDEF0, синтаксис, действия, границы и связи”

Цель: Изучить модели IDEF0, синтаксис, действия, границы и связи

Оборудование: IBM PC

- кратко описать выбранную предметную область (чем занимается предприятие, какие основные процессы в нем происходят)
- определить контекст моделирования

Перед началом построения диаграмм необходимо изучить выбранную предметную область. В этой и последующих работах в качестве предметной области будет выступать вымышленное предприятие по сборке и продаже настольных компьютеров и ноутбуков. Компания не производит комплектующие самостоятельно, а только собирает и тестирует компьютеры. Основные процедуры в компании:

- продавцы принимают заказы клиентов;
- сотрудники группируют заказы по типам компьютеров;
- сотрудники собирают и тестируют компьютеры;
- сотрудники упаковывают компьютеры согласно заказам;
- кладовщик отгружает клиентам заказы
- снабженцы заказывают и доставляют комплектующие, необходимые для сборки.

Компания использует купленную бухгалтерскую информационную систему, которая позволяет оформить заказ, счет и отследить платежи по счетам.

Построение модели какой-либо системы в методологии IDEF0 начинается с определения **контекста моделирования**, который включает в себя субъекта моделирования, цель моделирования и точку зрения на модель. Под **субъектом** понимается сама система, при этом необходимо точно установить, что входит в систему, а что

лежит за ее пределами, другими словами, необходимо определить, что в дальнейшем будет рассматривать как компоненты системы, а что как внешнее воздействие. **Цель моделирования** . Модель не может быть построена без четко сформулированной цели. Цель должна отвечать на следующие вопросы:

- Почему этот процесс должен быть смоделирован?
- Что должна показывать модель?
- Что может получить читатель?

Точка зрения. Не смотря на то, что при построении модели учитываются мнения различных людей, модель должна строиться с единой точки зрения. Точку зрения можно представить как взгляд человека, который видит систему в нужном для моделирования аспекте. Точка зрения должна соответствовать цели моделирования. В течении моделирования важно оставаться на выбранной точке зрения. В данной работе субъектом будет выступать само предприятие, а именно процессы, происходящие внутри него; цель моделирования - воспроизвести бизнес-процессы, происходящие на предприятии (модель AS-IS); точка зрения - с позиции директора как лица, знающего структуру предприятия в целом.

Практическая работа №2

Тема: "Построение моделей IDEF0. Цикл моделирования"

Цель: Изучить построение моделей IDEF0.

Оборудование: IBM PC

Теоретические указания к выполнению работы:

После определения контекста моделирования можно приступать к построению контекстной диаграммы (называемой еще "черным ящиком"). Данный тип диаграммы позволяет показать, что подается на вход работы и что является результатом работы, без детализации ее составляющих. Данная диаграмма содержит только одну работу, которая будет представлять всю деятельность предприятия в целом (рис.1).

USED AT:	AUTHOR: Fastowsky G. Eduard	DATE: 12.09.2007	WORKING	READER	DATE	CONTEXT: TOP
	PROJECT: Computer Firm	REV: 01.02.2010	DRAFT			
			RECOMMENDED			
			PUBLICATION			
NOTES: 1 2 3 4 5 6 7 8 9 10						
<div style="border: 1px solid black; padding: 10px; margin: 20px auto; width: 80%;"> <p>Деятельность предприятия по сборке и продаже компьютеров и ноутбуков</p> <p>0р. A0</p> </div>						
NODE: A-0	TITLE: Деятельность предприятия по сборке и продаже ноутбуков		NUMBER:			

Рисунок 1. Контекстная диаграмма

Любая IDEF0 диаграмма состоит из прямоугольников, называемых работами (activity), и стрелок (arrow). Работа представляет собой некоторую конкретную функцию в рамках рассматриваемой системы. По требованиям стандарта название каждой работы должно быть выражено отглагольным существительным (например, "Изготовление детали", "Оформление заказа" и т.д.). Каждая из четырех сторон прямоугольника имеет свое определенное значение (рис.2):

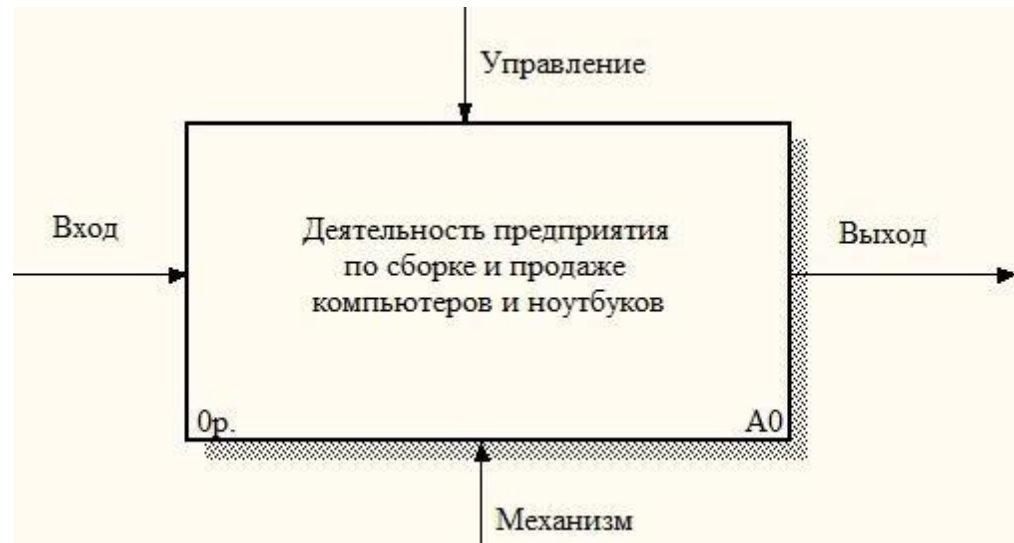


Рисунок 2. Работа в IDEF0

- Вход – это потребляемая или изменяемая работой информация или материал
- Выход – информация или материал, которые производятся работой
- Управление – процедуры, правила, стратегии или стандарты, которыми руководствуется работа
- Механизмы – ресурсы, которые выполняют работу (например, сотрудники, оборудование, устройства ит.д.)

Для рассматриваемого предприятия *входными стрелками* будут:

- Заказы клиентов - список компьютеров и их конфигурация, которые клиент желает приобрести
- Комплектующие от поставщиков - комплектующие, полученные от поставщиков, из которых собираются

компьютеры и ноутбуки

Выходные стрелки:

- Готовая продукция - собранные компьютеры и ноутбуки
- Заказы поставщикам - список комплектующих, которые предприятие закупает у поставщиков

- Оплата за комплектующие - деньги поставщикам за комплектующие
- Маркетинговые материалы - прайс-листы, рекламки и т.п.

Стрелки управления:

- Законодательство - различные законодательные документы, которыми руководствуется предприятие в процессе своей деятельности
- Правила и процедуры - различные правила и процедуры, которыми руководствуется предприятие в процессе своей деятельности (например, правила сборки и тестирования компьютеров, процедура общения с клиентами и т.п.)

Стрелки механизмов:

- Бухгалтерская система
- Персонал

Итоговая контекстная диаграмма имеет вид (рис.3):

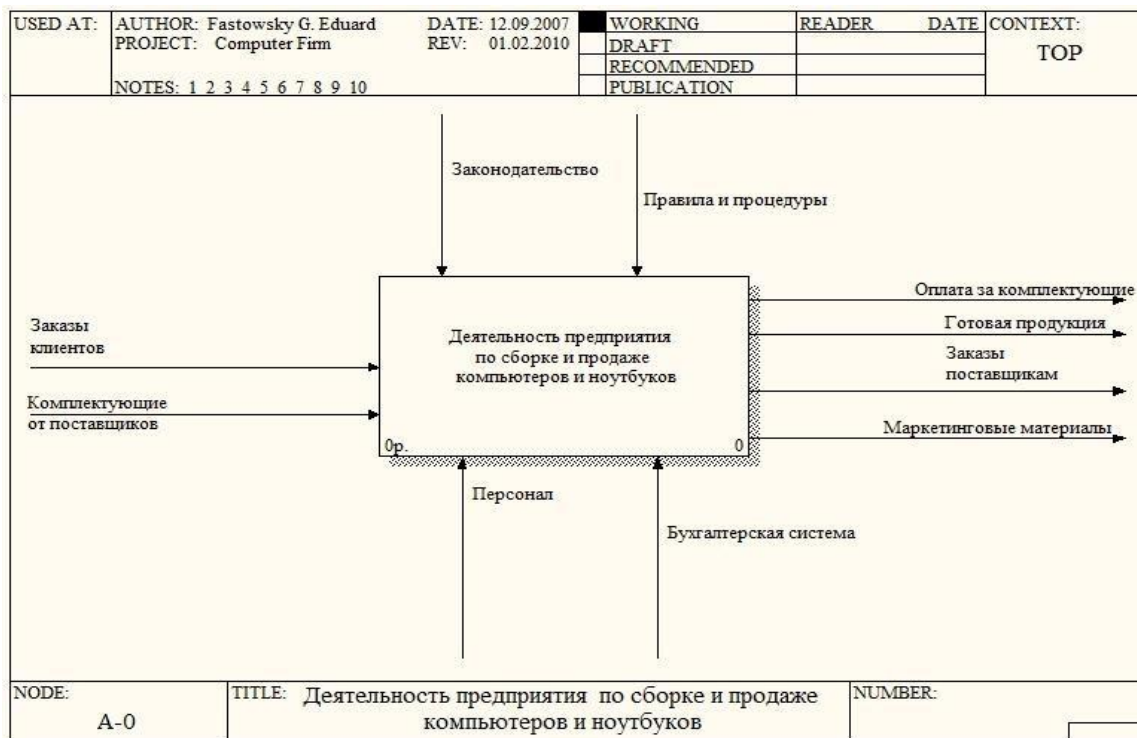


Рисунок 3. Итоговая контекстная диаграмма

ПРАКТИЧЕСКАЯ РАБОТА №3

Тема: “Построение диаграмм, контекст диаграмм, блоки и связи. Свойство диаграмм”

Цель: Научиться строить диаграммы, контекст диаграммы, блоки и связи. Устанавливать свойство диаграмм.

Оборудование: IBM PC

Создание диаграммы A-0

Построение модели бизнес-процесса рассмотрим на примере работы мебельной фабрики. Во время проведения обследования предприятия были выявлены её целевые задачи, функциональные деятельности каждого из подразделений предприятия и функциональные взаимодействия между ними; информационные потоки внутри подразделений и между ними; внешние по отношению к предприятию объекты и внешние информационные воздействия, а так же нормативно-справочная документация, данные по имеющимся на предприятии средствам и системам автоматизации.

Целевые функции мебельной фабрики:

- переработка сырья;
- изготовление деталей для мебели;
- сборка изделия;
- контроль качества.

Нормативные документы мебельной фабрики:

- чертежи (деталей, сборочный);
- нормы по переработке сырья;
- стандарты качества;
- производственные инструкции;
- инструкции по технике безопасности.

Подразделения предприятия:

- цех по обработке сырья и бракованных изделий;

- цех по изготовлению деталей;
- сборочный цех;
- отдел проверки качества изделия.

Основным сырьем для изготовления мебели является *дерево*.

Для контекстного процесса ИЗГОТОВЛЕНИЕ МЕБЕЛИ определим необходимую информацию:

- ВХОД - сырьё;
- УПРАВЛЕНИЕ – чертежи, производственные инструкции, инструкции по технике безопасности

(нормативные документы);

- МЕХАНИЗМЫ – персонал, производственное оборудование;
- ВЫХОД – готовая мебель.

Стандартное окно BPwin (рис. 1.4.):

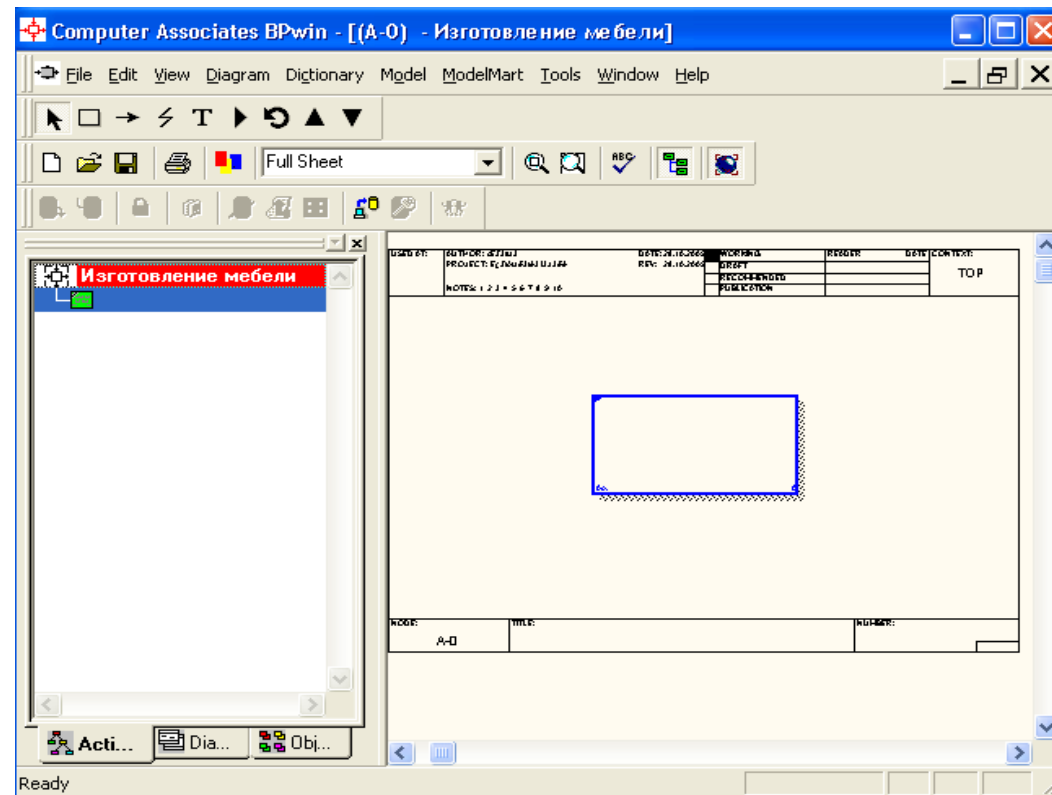


Рис. 1.4. Стандартное окно BPwin

Основные инструменты BPwin











На основной панели инструментов расположены элементы управления, в основном знакомые по другим Windows-интерфейсам (рис. 1.5.):



Рис. 1.5. Элементы управления BPwin

Функциональность панели инструментов доступна из основного меню BPwin (табл. 1).

Таблица 1. 1. Элементы управления основной панели инструментов BPwin

Элемент управления	Описание	Соответствующий пункт меню
	Создать новую модель	File/New
	Открыть модель	File/Open
	Сохранить модель	File/Save
	Напечатать модель	File/Print
	Вызвать генератор отчетов Report Builder	Tools/Report Builder
	Выбор масштаба	View/Zoom
	Масштабирование	View/Zoom
	Проверка правописания	Tools/Spelling
	Включение и выключение навигатора модели Model Explorer	View/Model Explorer
	Включение и выключение дополнительной панели инструментов работы с ModelMart	ModelMart

Упражнение 2. Работа с блоком.

Построение контекстной диаграммы процесса ИЗГОТОВЛЕНИЕ МЕБЕЛИ.

Для ввода имени блока необходимо:

1. Щелкнуть правой клавишей мыши по блоку.
2. Выбрать команду **Name**.
3. В диалоговом окне ввести название «Изготовление мебели»

4. Для того чтобы текст стал понятен, в контекстном меню выберите пункт **Font** (рис. 1.8.):
5. В диалоговом окне **Activity Properties** в нижней части вкладки **Font** установите флажки в опциях **Apply setting to**, позволяющих изменить шрифт для всех работ на текущей диаграмме, в модели, и в группе **Global**, позволяющей изменить шрифт одновременно для всех объектов модели, в опции **Script** выберите «кириллический».
6. Установите шрифт **Arial Unicode MS**, курсив, 16 пт (рис. 1.9.).

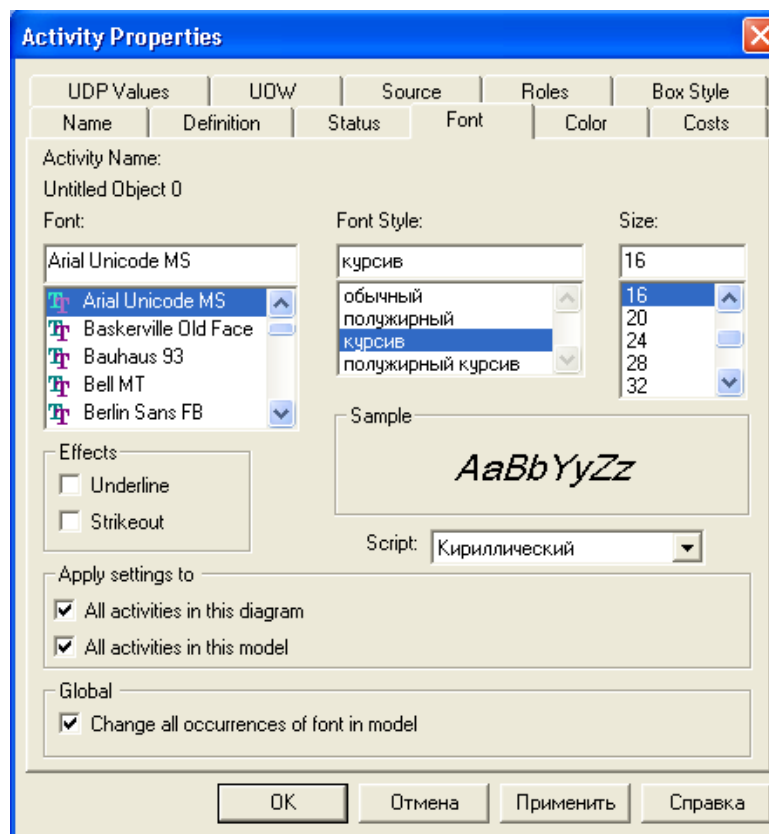



Рис. 1.9. Вкладка Font диалога Activity Properties

Упражнение 3. Построение дуг.

Для построения дуги **управления** необходимо:

1. Выбрать на панели инструментов кнопку .
2. Подвести курсор к верхнему краю окна построения диаграммы до появления черной полосы и произвести щелчок левой кнопкой мыши по этой полосе (рис. 1.10.).
3. Подвести курсор мыши к верхней стороне блока до образования темного треугольника и щелкнуть левой кнопкой мыши (рис. 1.11.).


Построение дуг **входа** и **механизмов** производится аналогичным образом.

4. Постройте дуги **входа** и **механизмов**.

Для построения дуги **выхода** выполняются те же действия, но в противоположном порядке: от правой стороны блока к правой стороне окна построения диаграмм.

5. Постройте дугу **выхода**.

Упражнение 4. Идентификация дуги управления.

1. Выберите на панели редактирования кнопку .
2. Щелкните правой кнопкой мыши по дуге.
3. Выберите команду **Name** (рис. 1.12.).

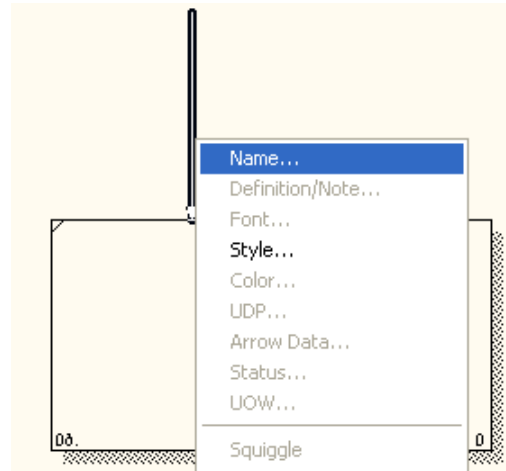


Рис. 1.12. Контекстно-зависимое меню

4. В диалоговом окне введите название дуги: «Нормативная документация» (рис. 1.13.).

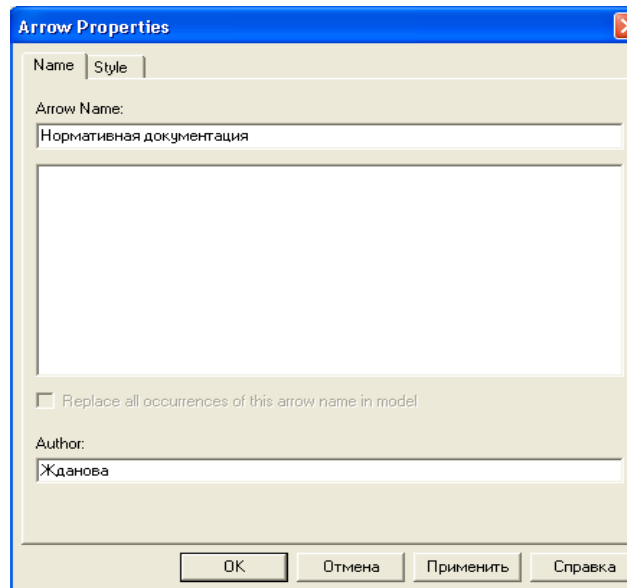


Рис. 1.13. Диалоговое окно Arrow Properties

5. Для того чтобы текст для дуги стал понятен (рис 1.14.), выберите меню **Model - Default Fonts**.

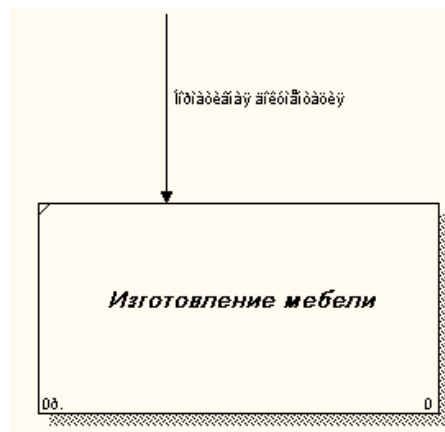


Рис. 1.14. Фрагмент диаграммы

6. В диалоговом окне **Default Context Arrow Name Text Font** в нижней части установите флажок в опции **Change all occurrences**, позволяющей изменить шрифт для названий всех дуг на текущей диаграмме, в опции **Script** выберите «кириллический».
7. Установите шрифт **Arial Unicode MS**, курсив, 14 пт. (рис. 1.16.).

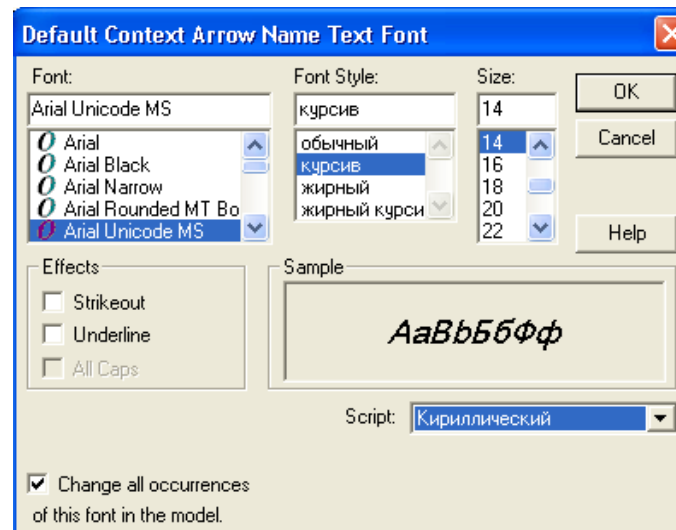


Рис. 1.16. Диалоговое окно *Default Context Arrow Name Text Font*

Вот что должно у вас получиться (рис. 1.17.).



Рис. 1.17. Фрагмент диаграммы

Упражнение 5. Работа с блоком.


Самостоятельно постройте дуги:

- ВХОДА: «Сырьё»;
- МЕХАНИЗМА: «Персонал», «Производственное оборудование»;
- ВЫХОДА: «Готовая мебель».

Упражнение 6. Установление тильды.

Название дуги является независимым объектом, который можно перемещать относительно дуги. Текст может располагаться по отношению к дуге в свободной форме, либо соединяться с дугой символом тильды.

Чтобы установить тильду следует:

1. На панели инструментов нажать кнопку  ;
2. Щелкнуть левой кнопкой мыши по тексту, а затем по дуге (рис.
3. Можно также использовать команду контекстно-зависимого меню **Squiggle** (рис. 1.19.).

Дуга представляет собой совокупность отдельных графических объектов: прямые участки, изогнутые участки, изображение наконечника. Отдельные элементы можно передвигать независимо друг от друга, меняя форму дуги, также дугу можно перемещать как единый неделимый элемент.

4. Установите тильду к остальным дугам и их названиям.



Упражнение 7. Изменение цвета текста, фона блока, цвета и стиля дуг.

1. Для изменения цвета текста выполните команду контекстно-зависимого меню **Color** (рис. 1.20.):

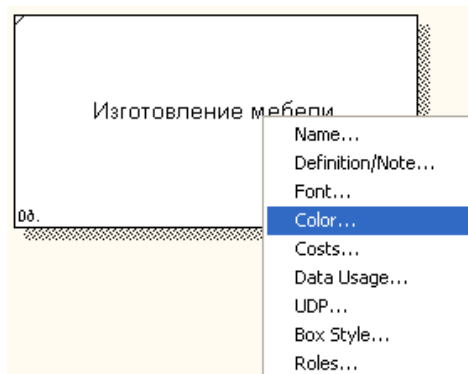


Рис. 1.20. Контекстно-зависимое меню

2. Выберите цвет и нажмите кнопку  (рис. 1.21.).

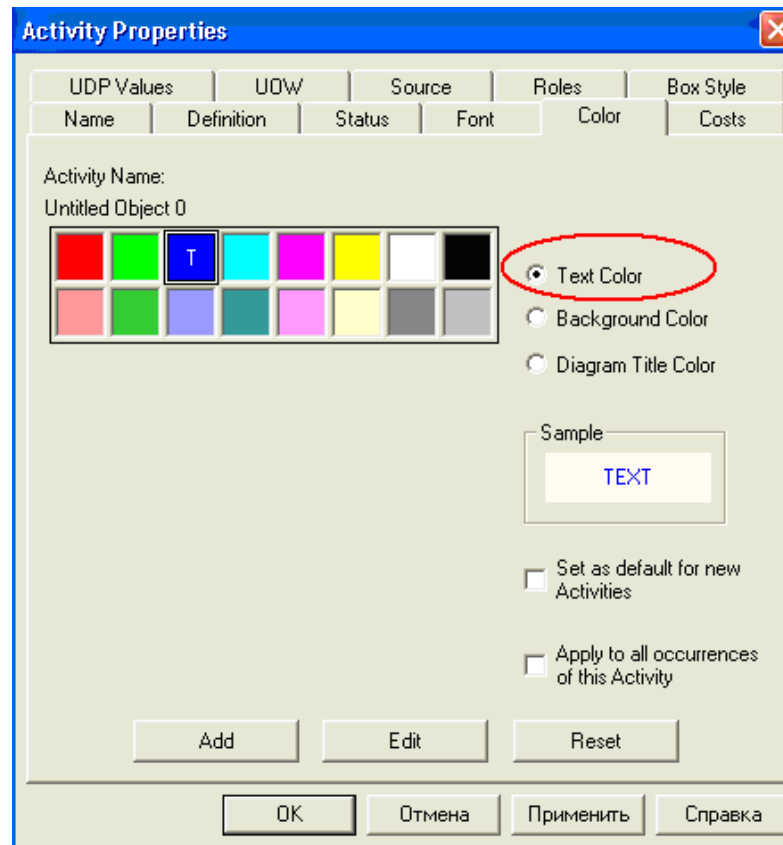


Рис. 1.21. Диалоговое окно выбора цвета текста и стрелок

3. Для изменения фона блока выберите **Background Color** и цвет (рис. 1.22.):

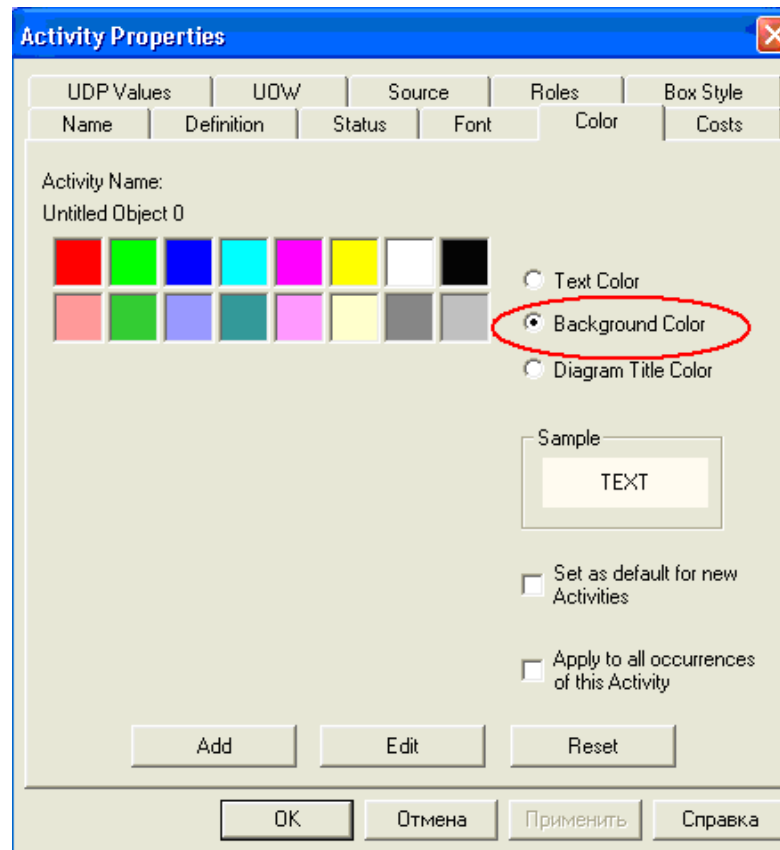


Рис. 1.22. Вкладка Color диалога Activity Properties

4. Для изменения стиля дуги выберите в контекстно-зависимом меню команду **Style** (рис. 1.23.):

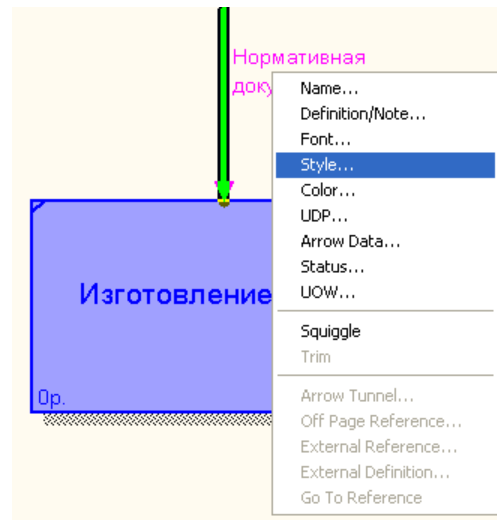


Рис. 1.23. Контекстно-зависимое меню

5. В диалоговом окне укажите тип и стиль дуги, нажмите на кнопку ОК (рис. 1.24.).

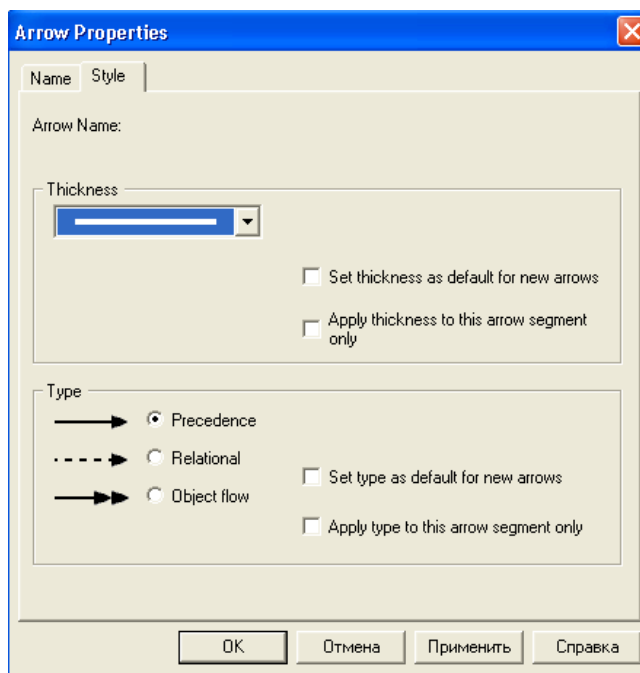


Рис. 1.24. Диалоговое окно *Arrow Properties*

Вот что должно у вас получиться (рис. 1.25.).

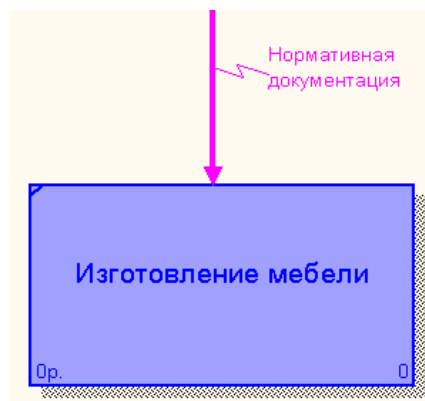


Рис. 1.25. Пример форматирования элементов диаграммы

Удаление блоков, дуг или текста.

Для удаления блока и дуги или текста необходимо их активизировать щелчком левой кнопки мыши и нажать клавишу Delete, а затем подтвердить намерения по поводу удаления.

Упражнение 8. Форматирование диаграммы.

1. Самостоятельно произведите форматирование всех элементов диаграммы, опираясь на данные, приведенные в таблице 1.3.

После выполнения задания у вас должна получиться следующая контекстная диаграмма (рис. 1.26.):

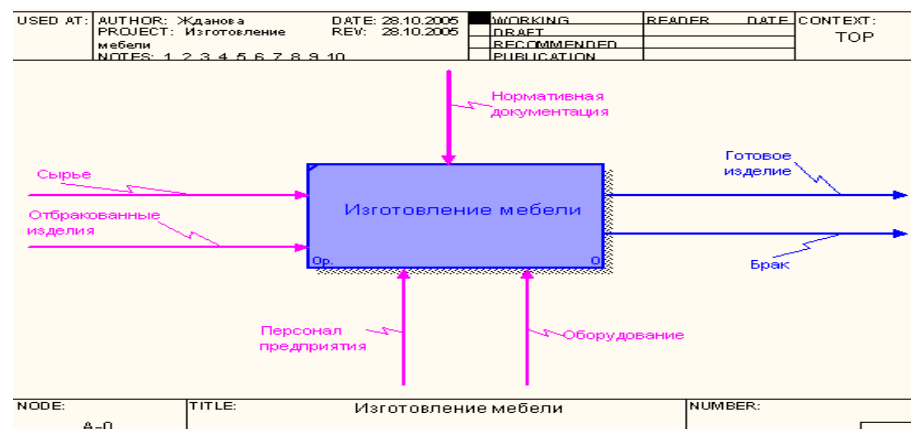


Рис. 1.26. Контекстная диаграмма процесса «Изготовление мебели»



Упражнение 9. Сохранение полученной диаграммы.

Сохраните полученную диаграмму.

1. Создайте папку, назовите её своей фамилией и в неё сохраняйте свои работы.
2. В меню **File** выберите команду **Save as**.
3. Укажите путь к своей папке и имя файла **Lab1.bp1** (рис. 1.27.).
4. Нажмите на кнопку СОХРАНИТЬ.

Контрольные вопросы

1. Перечислите основные возможности BPwin.
2. Охарактеризуйте основные элементы рабочего интерфейса BPwin.
3. Какую методологию поддерживает BPwin?
4. Укажите назначение каждой из дуг изображенных на рисунке.



5. Назовите основные этапы построения модели.
6. Какой процесс можно назвать функциональной декомпозицией?
7. Перечислите элементы контекстной диаграммы.
8. При помощи какого инструмента строятся дуги на диаграмме?

ПРАКТИЧЕСКАЯ РАБОТА № 4

Тема: “Общие принципы и положения. Классификация функций, моделируемых блоками IDEF0”

Цель: Изучить принципы и положения, моделируемых блоками IDEF0


Оборудование: IBM PC

С помощью этой практической работы Вы сможете:

- научиться производить декомпозицию контекстной диаграммы;
- освоить правила построения дуг и тоннелирования стрелок.

Теоретические сведения

Декомпозиция

 После создания контекстной диаграммы, которая представляет собой описание контекста моделируемой системы, проводится **функциональная декомпозиция** – система разбивается на подсистемы и каждая подсистема описывается в том же синтаксисе, что и система в целом. Затем каждая подсистема разбивается на более мелкие и так до достижения нужного уровня подробности. В результате такого разбиения, каждый фрагмент системы изображается на отдельной диаграмме декомпозиции. *Диаграмма декомпозиции предназначена для детализации работы.*

Все *работы* модели нумеруются. Номер состоит из префикса и числа. Может быть использован префикс любой длины, но обычно используют префикс А. Контекстная (корневая) *работа* дерева имеет номер А0. *Работы* i декомпозиции А0 имеют номера А1, А2, А3 и т. д. *Работы* декомпозиции нижнего уровня имеют номер родительской *работы* и очередной порядковый номер, например *работы* декомпозиции А3 будут иметь номера А31, А32, А33, А34 и т. д. *Работы* образуют иерархию, где каждая *работа* может иметь одну родительскую и несколько дочерних *работ*, образуя дерево. Такое дерево называют деревом узлов, а вышеописанную нумерацию — нумерацией по узлам.

Диаграммы IDEF0 имеют двойную нумерацию. Во-первых, диаграммы имеют номера по узлу. *Контекстная диаграмма* всегда имеет номер A-0, декомпозиция *контекстной диаграммы* — номер A0, остальные диаграммы декомпозиции — номера по соответствующему узлу (например, A1, A2A21A213, и т. д.). BРwin автоматически поддерживает нумерацию по узлам, т. е. при проведении декомпозиции создается новая диаграмма и ей автоматически присваивается соответствующий номер узлам.

При декомпозиции процесса все стрелки, входящие или исходящие из него, должны быть перенесены на диаграмму нижнего уровня и использованы при ее построении. При этом запрещены всякие новые стрелки, выходящие за пределы новой диаграммы, кроме специальных, так называемых "**тоннелированных**" стрелок.

Диаграмма верхнего уровня создается путем декомпозиции основной функции контекстной диаграммы. На диаграмме декомпозиции функции нумеруются автоматически слева направо. Номер функции показывается в правом нижнем углу. В левом верхнем исчезает небольшая диагональная черта, которая показывает, что данная функция была декомпозирована.

Практическое задание «Разработка диаграмм декомпозиции 1 уровня»


Детализация процесса «Изготовление мебели».

Откройте файл **Lab1.bp1**, сохраненный на предыдущем уроке.

Следующим шагом является детализация контекстного процесса с помощью диаграммы верхнего уровня. Эта диаграмма содержит в себе четыре процесса:

- 1) Процесс 1.1 – ПЕРЕРАБОТКА СЫРЬЯ.
- 2) Процесс 1.2 – ИЗГОТОВЛЕНИЕ ДЕТАЛЕЙ.
- 3) Процесс 1.3 – СБОРКА ИЗДЕЛИЯ.
- 4) Процесс 1.4 – КОНТРОЛЬ КАЧЕСТВА.

Произведите детализацию процесса «Изготовление мебели», задав нужное количество новых блоков. Для этого:

1. Щелкните по блоку «Изготовление мебели» и выберите инструмент .
2. В диалоговом окне введите число, на которое будет произведена декомпозиция – 4.
3. Укажите тип диаграммы **IDEF0** (Рисунок 1.) и нажмите ОК.

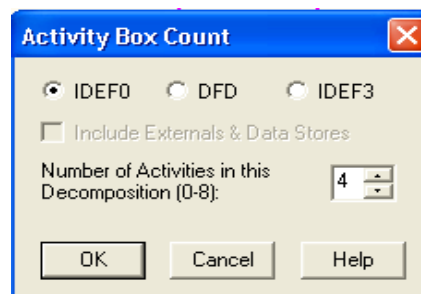


Рисунок 1 . Диалоговое окно декомпозиции блока

4. Укажите названия новых блоков («Переработка сырья», «Изготовление деталей», «Сборка изделия», «Контроль качества»).

При декомпозиции функции входящие в нее и исходящие из нее дуги автоматически появляются на диаграмме декомпозиции (миграция дуг), но при этом не касаются блоков. Такие стрелки называются **несвязанными** и воспринимаются в ВРwin как синтаксическая ошибка (Рисунок 2.).

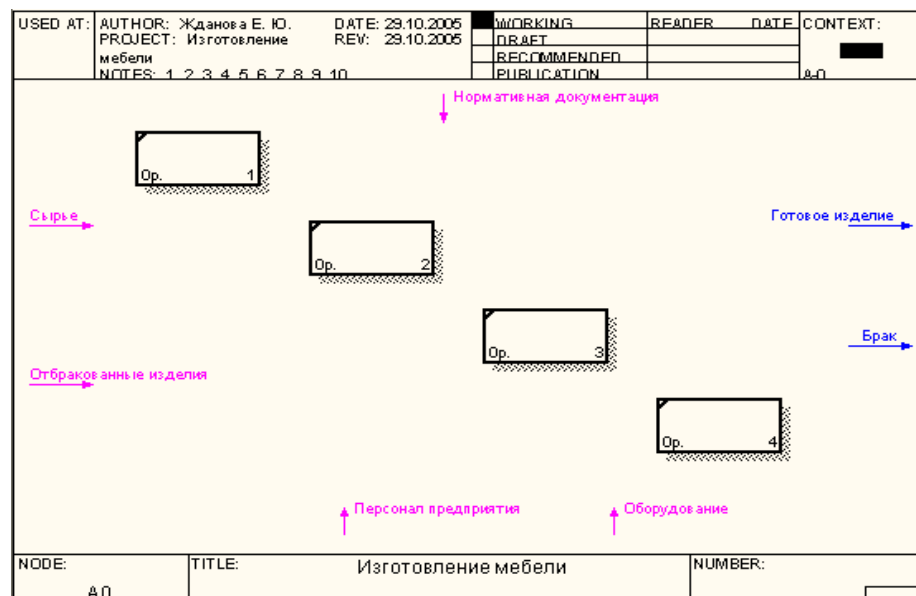



Рисунок 2. Декомпозиция верхнего уровня

Определим входные и выходные потоки для новых процессов.

Процесс 1.1. ПЕРЕРАБОТКА СЫРЬЯ:



- 1) Вход – СЫРЬЁ.
- 2) Вход – ОТБРАКОВАННЫЕ ИЗДЕЛИЯ.
- 3) Выход – ЗАГОТОВКИ.

Произведем процесс связывания мигрирующих дуг:

5. Выберите инструмент  рисования дуг.
6. Щелкните мышью по наконечнику входного потока СЫРЬЁ.
7. Щелкните по входной стороне блока ПЕРЕРАБОТКА СЫРЬЯ.

Вход – ОТБРАКОВАННЫЕ ИЗДЕЛИЯ построим немного позже.

Для построения выходного потока ЗАГОТОВКИ выполните действия:

8. Выберите инструмент  рисования дуг.
9. Щелкните левой кнопкой мышки по выходной стороне блока ПЕРЕРАБОТКА СЫРЬЯ.
10. Затем щелкните по входной стороне блока ИЗГОТОВЛЕНИЕ ДЕТАЛЕЙ.
11. Выберите инструмент  текст, в контекстном меню – команду **Name**, укажите название дуги ЗАГОТОВКИ.
12. Проверьте себя (Рисунок 3.).

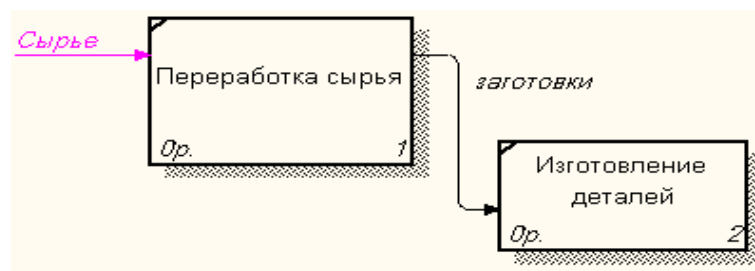


Рисунок 3. Фрагмент диаграммы

Детализация процесса «Изготовление мебели».

1. Самостоятельно выполните детализацию процессов:

Процесс 1.2. ИЗГОТОВЛЕНИЕ ДЕТАЛЕЙ:

- 1) Вход – ЗАГОТОВКИ.
- 2) Выход – ГОТОВЫЕ ДЕТАЛИ.

Процесс 1.3. СБОРКА ИЗДЕЛИЯ:


- 1) Вход – ГОТОВЫЕ ДЕТАЛИ.
- 2) Выход – СОБРАННОЕ ИЗДЕЛИЕ.

Процесс 1.4. КОНТРОЛЬ КАЧЕСТВА:

- 1) Вход – СОБРАННОЕ ИЗДЕЛИЕ.
- 2) Выход – ГОТОВОЕ ИЗДЕЛИЕ.
- 3) Выход – БРАК.
- 4) Выход – ПРОИЗВОДСТВЕННЫЕ ОТХОДЫ

Смена направления дуги.

На Выходе БРАК не выходит за границу модели, а возвращается в процесс ПЕРЕРАБОТКА СЫРЬЯ:

1. Удалите дуги ОТБРАКОВАННЫЕ ИЗДЕЛИЯ и БРАК.
2. Выберите инструмент  рисование дуг.
3. Щелкните левой кнопкой мыши на Выходе блока КОНТРОЛЬ КАЧЕСТВА.
4. Щелкните левой кнопкой на Входе блока ПЕРЕРАБОТКА СЫРЬЯ.

5. Назовите новую дугу – БРАК (Рисунок 4.).

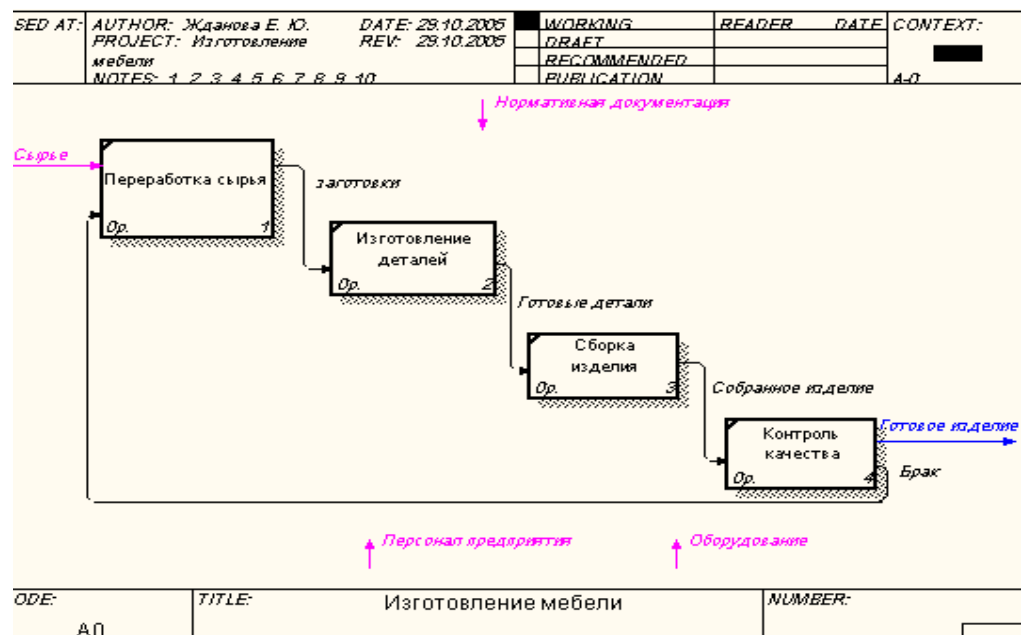


Рисунок 4. Процесс декомпозиции

Построение ответвлений дуг.

Переработка сырья, изготовление деталей, сборка изделия, контроль качества осуществляются согласно Нормативным документам, поэтому у управляющей стрелки НОРМАТИВНАЯ ДОКУМЕНТАЦИЯ появятся ответвления: НОРМЫ ПЕРЕРАБОТКИ СЫРЬЯ, ЧЕРТЕЖ ДЕТАЛИ, СБОРОЧНЫЙ ЧЕРТЕЖ, СТАНДАРТ КАЧЕСТВА.

1. Выберите инструмент рисование дуг.

2. Щелкните мышью по наконечнику входного потока НОРМАТИВНАЯ ДОКУМЕНТАЦИЯ.
3. Щелкните по входной стороне блока ПЕРЕРАБОТКА СЫРЬЯ.
4. Самостоятельно выполните ответвления дуги НОРМАТИВНАЯ ДОКУМЕНТАЦИЯ на блоки ИЗГОТОВЛЕНИЕ ДЕТАЛЕЙ, СБОРКА ИЗДЕЛИЯ, КОНТРОЛЬ КАЧЕСТВА.
5. Проверьте себя (Рисунок 5.).

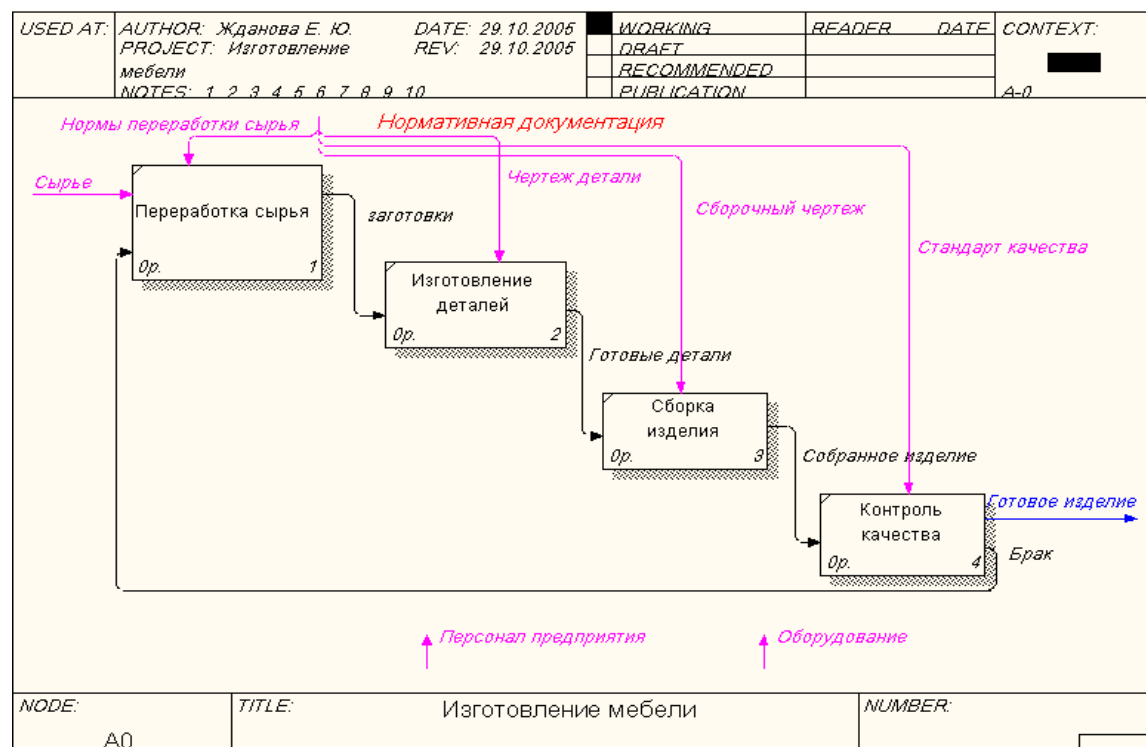


Рисунок 5. Ответвления дуги НОРМАТИВНАЯ ДОКУМЕНТАЦИЯ

Построение дуг Персонал предприятия, Оборудование.

Дуги ПЕРСОНАЛ ПРЕДПРИЯТИЯ и ОБОРУДОВАНИЕ для всех процессов будут одинаковые.

1. Самостоятельно соедините каждую дугу с каждым блоком, укажите ее имя.
2. Проверьте себя (Рисунок 6.).

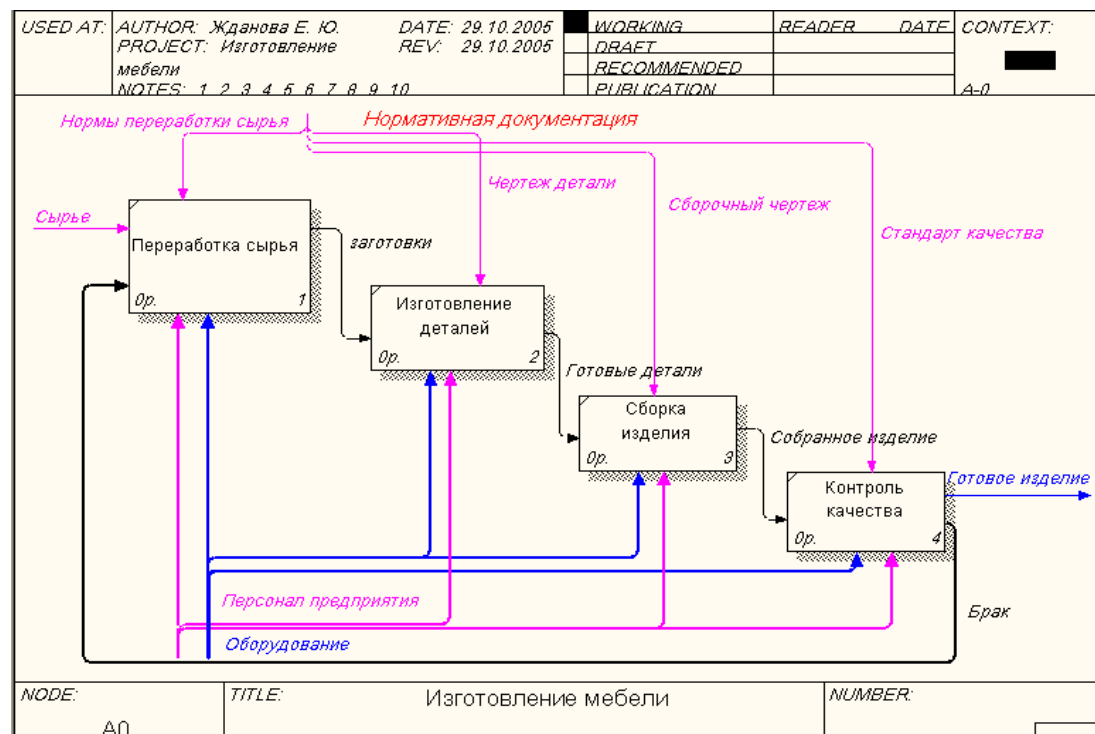



Рисунок 6. Построение дуг ПЕРСОНАЛ ПРЕДПРИЯТИЯ и ОБОРУДОВАНИЕ

«Тоннелирование» стрелок.

1. В Процессе 1.2. ИЗГОТОВЛЕНИЕ ДЕТАЛЕЙ постройте новую граничную дугу, которой обозначьте Выход – ПРОИЗВОДСТВЕННЫЕ ОТХОДЫ.

Вновь внесенные граничные дуги на диаграмме декомпозиции нижнего уровня изображаются в квадратных скобках и автоматически не появляются на диаграмме верхнего уровня.

Для их «перетаскивания» наверх нужно:

2. Выбрать инструмент  редактирования.
3. Кликнуть правой кнопкой мыши по квадратным скобкам.
4. Выбрать в контекстном меню пункт **Arrow Tunnel**.
5. В появившемся диалоге **Border Arrow Editor** (Рисунок 7.) щелкнуть по кнопке **Resolve it to border arrow** для миграции стрелки на диаграмму верхнего уровня или по кнопке **Change it to resolved rounded tunnel** для «тоннелирования» дуги.

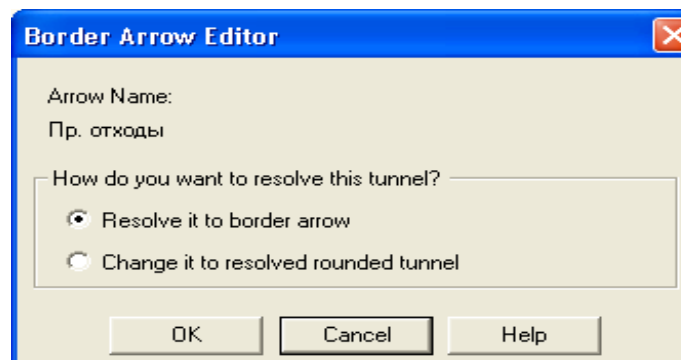


Рисунок 7. Диалог Border Arrow Editor

Тоннельная дуга изображается с круглыми скобками на конце и не попадет на другую диаграмму (Рисунок 8.). Такое тоннелирование может быть применено для изображения малозначимых стрелок.

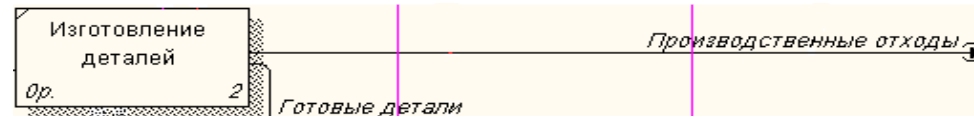




Рисунок 8. Граничная дуга

6. Отправьте созданную дугу "Производственные отходы" в тоннель.

Создание обратной связи по управлению.

Качество изделия может быть повышено путем непосредственного регулирования процессами изготовления деталей и сборки мебели в зависимости от результата (выхода) работы КОНТРОЛЬ КАЧЕСТВА. Обратная связь по управлению свидетельствует об эффективности бизнес-процесса и создается следующим образом:

1. Выберите инструмент  рисование дуг.
2. Щелкните мышью по выходу КОНТРОЛЬ КАЧЕСТВА.
3. Щелкните по управлению блоков ИЗГОТОВЛЕНИЕ ДЕТАЛЕЙ, СБОРКА ИЗДЕЛИЯ.
4. Выберите инструмент  текст.
5. Назовите обратную связь РЕКОМЕНДАЦИИ.

После выполнения работы у вас должна получиться следующая диаграмма (Рисунок 9.):

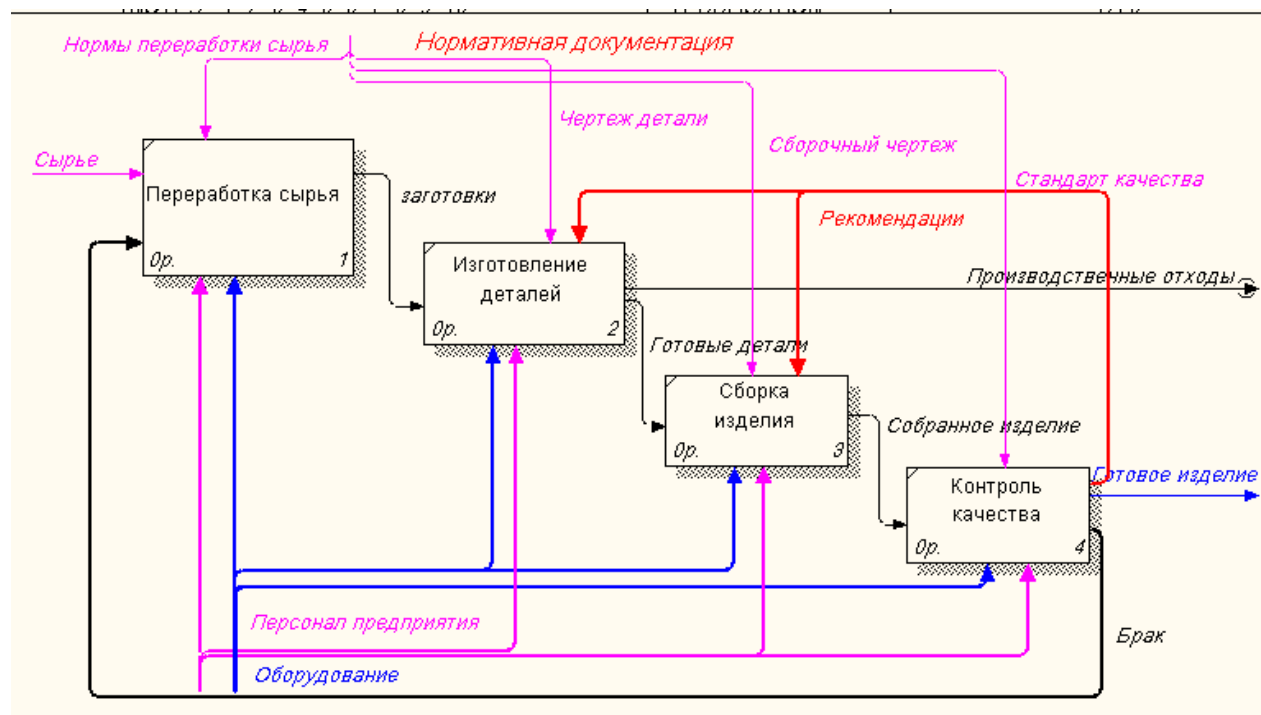


Рисунок 9. Диаграмма декомпозиции блока ИЗГОТОВЛЕНИЕ МЕБЕЛИ

Сохранение полученной диаграммы.

1. В меню **File** выберите **Save as**.
2. Укажите путь к своей папке и имя файла **Lab2.bp1**.
3. Нажмите **ОК**.

Контрольные вопросы

1. Как создается диаграмма верхнего уровня?
2. Как на диаграмме отображается декомпозиция?
3. Каким инструментом задается количество блоков для декомпозиции?
4. Какие стрелки VPwin воспринимает как синтаксическую ошибку?
5. Для чего создается обратная связь по управлению?
6. Для чего служит тоннелирование дуг?

ПРАКТИЧЕСКАЯ РАБОТА № 5

Тема: “Дерево модели. Построение контекстных диаграмм”

Цель: Изучить построение контекстных диаграмм

Оборудование: IBM PC

С помощью этой практической работы Вы сможете:

- научиться детализировать процессы;
- освоить правила описания свойств модели;
- научиться составлять отчет о свойствах модели.

Теоретические сведения



Последним шагом построения модели является **функциональная декомпозиция**. Построенная диаграмма верхнего уровня также имеет множество процессов, которые в свою очередь могут быть детализированы в диаграмме нижнего уровня. Таким образом строится иерархия IDEF0 с контекстной диаграммой в вершине иерархии.

Этот процесс декомпозиции продолжается до достижения нужного уровня подробности. При таком построении иерархии IDEF0 каждый процесс более низкого уровня необходимо соотнести с процессом верхнего уровня. Обычно для этой цели все работы модели нумеруются. Номер состоит из префикса и числа. Может быть использован префикс любой длины, но обычно используют префикс А.

Контекстная работа дерева имеет номер А0. Работы декомпозиции А0 имеют номера А1, А2, А3 и т.д. Работы декомпозиции нижнего уровня имеют номер родительской работы и очередной порядковый номер, например работы декомпозиции А3 будут иметь номера А31, А32, А33, А34 и т. д.

Работы образуют иерархию, где каждая работа может иметь одну родительскую и несколько дочерних работ, образуя дерево. Такое дерево называют **деревом узлов**, а вышеописанную нумерацию - **нумерацией по узлам**.

Имеются незначительные варианты нумерации, которые можно настроить во вкладке **Numbering** (Рисунок 1.) диалога **Model Properties** (меню **Model – Model Properties**).

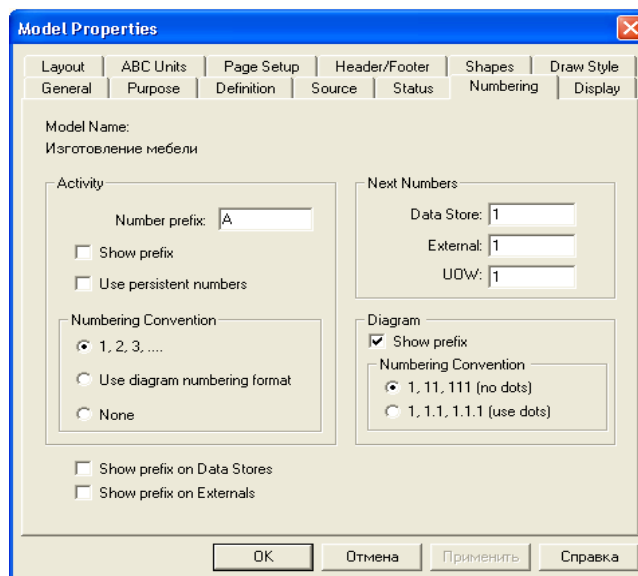


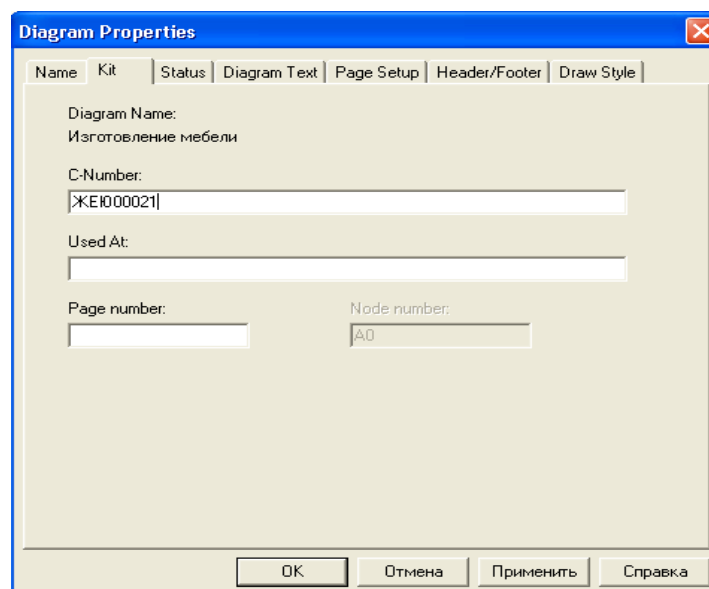
Рисунок 1. Диалоговое окно настройки нумерации работ в диаграмме

Диаграммы IDEF0 имеют двойную нумерацию. Во-первых, диаграммы имеют номера по узлу. Контекстная диаграмма всегда имеет номер A-0, декомпозиция контекстной диаграммы - номер A0, остальные диаграммы декомпозиции - номера по соответствующему узлу (например, A1, A2, A21, A213 и т.д.).

ВРwin *автоматически поддерживает нумерацию по узлам*, т. е. при проведении декомпозиции создается новая диаграмма и ей автоматически присваивается соответствующий номер. В результате проведения экспертизы диаграммы могут уточняться и изменяться, следовательно, могут быть созданы различные версии одной и той же (с точки зрения ее расположения в дереве узлов) диаграммы декомпозиции. ВРwin позволяет иметь в модели только одну диаграмму декомпозиции в данном узле.

Прежние версии диаграммы можно хранить в виде бумажной копии либо как FEO-диаграмму. (К сожалению, при создании FEO-диаграмм отсутствует возможность отката, т. е. можно получить из диаграммы декомпозиции FEO, но не наоборот.)

В любом случае следует отличать различные версии одной и той же диаграммы. Для этого существует специальный номер - C-number, который должен присваиваться автором модели вручную. C-number - это произвольная строка, но рекомендуется придерживаться стандарта, когда номер состоит из буквенного префикса и порядкового номера, причем в качестве префикса используются инициалы автора диаграммы, а порядковый номер отслеживается автором вручную, например **ЖЕЮ000021** (Рисунок 2.).



The image shows a 'Diagram Properties' dialog box with the following fields and values:

- Diagram Name:** Изготовление мебели
- C-Number:** ЖЕЮ000021
- Used At:** (empty)
- Page number:** (empty)
- Node number:** A0

Buttons at the bottom: OK, Отмена, Применить, Справка.

Рисунок 2. Диалоговое окно присваивания номера данной версии диаграммы

Практическое задание «Разработка диаграммы функциональной декомпозиции»

На предыдущих лабораторных работах вы построили контекстную диаграмму процесса "Изготовление мебели" и провели его детализацию с помощью диаграммы верхнего уровня. Последним шагом построения модели является **функциональная декомпозиция**, т.е. разбиение сложных процессов на более простые. Этот процесс декомпозиции продолжается до достижения нужного уровня подробности.

Детализация процесса «Изготовление деталей».

1. Откройте файл **Lab2.bp1**, сохраненный на предыдущем уроке.
2. Проведите детализацию процесса **1.2. ИЗГОТОВЛЕНИЕ ДЕТАЛЕЙ** с помощью диаграммы нижнего уровня. Данные представлены в таблице 1:

Таблица 1. Детализирование процесса «Изготовление деталей»

Процесс	Вход	Выход
1.2.1 – Переработка заготовки в деталь	Заготовки	Готовые детали
1.2.2 – Проверка качества деталей	Готовые детали	Готовые детали, брак
Управляющие стрелки и стрелки механизмов, указанные на диаграмме верхнего уровня должны быть и в диаграмме детализации.		

3. Выберите инструмент  и щелкните по блоку ИЗГОТОВЛЕНИЕ ДЕТАЛЕЙ;

4. В диалоговом окне введите число, на которое будет произведена декомпозиция - 2;
5. Укажите тип диаграммы **IDEF0** (Рисунок 3.) и нажмите OK.

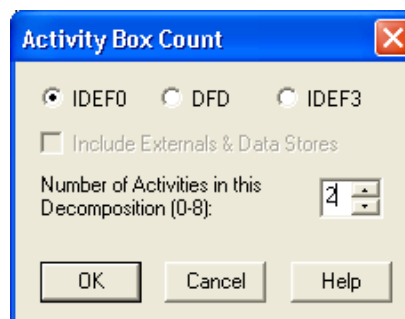


Рисунок 3. Диалоговое окно декомпозиции блока

Вы получите диаграмму декомпозиции уровня **A2** (Рисунок 4.).

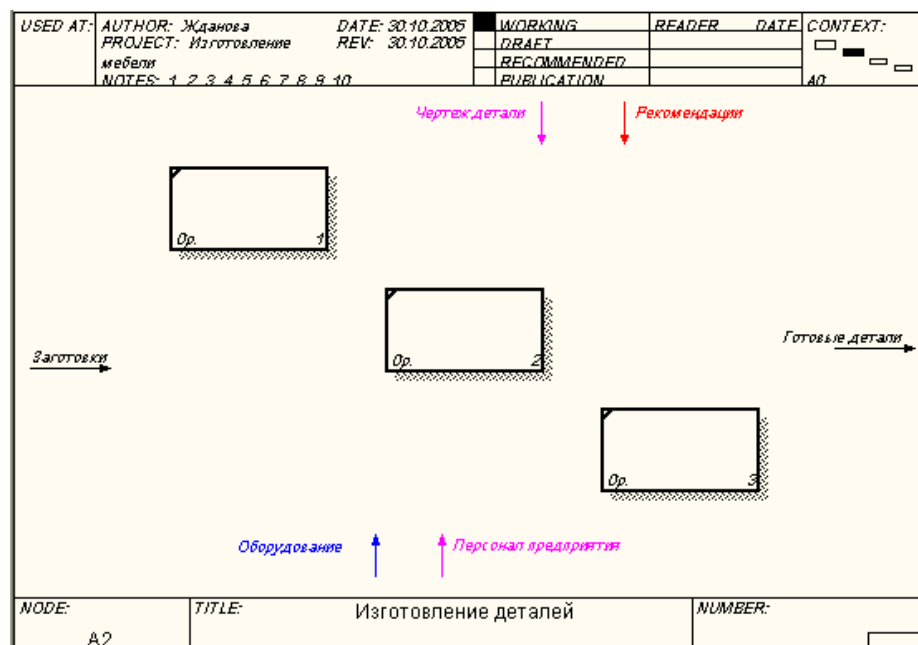


Рисунок 4. Декомпозиция уровня А2

6. Укажите названия процессов;
7. Соедините дугами обозначенные процессы, используя данные из таблицы 3.1;
8. Проверьте себя (Рисунок 5.).

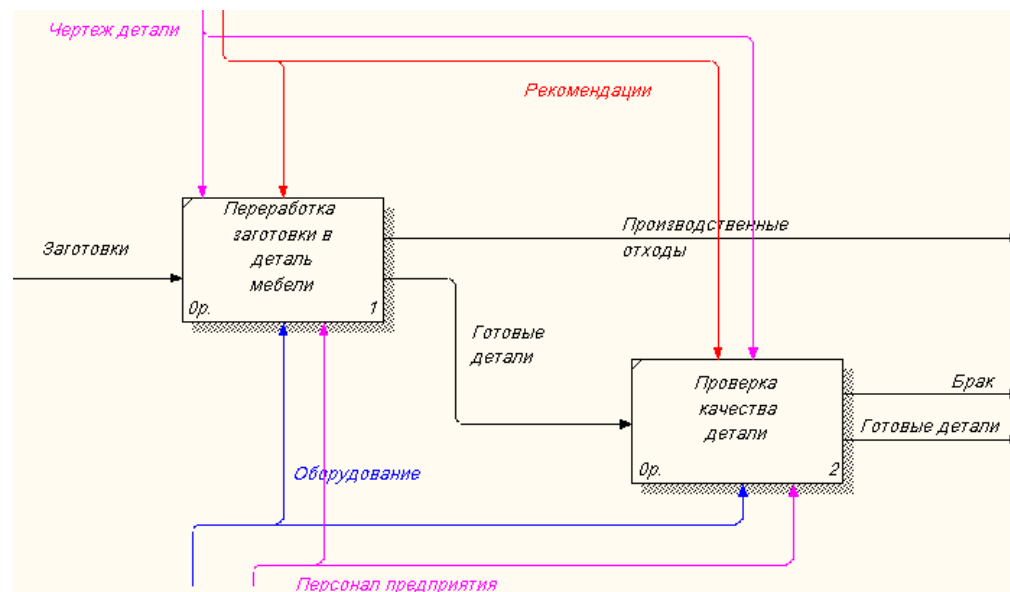


Рисунок 5. Детализация процесса ИЗГОТОВЛЕНИЕ ДЕТАЛЕЙ

Детализация процесса «Контроль качества».

1. Самостоятельно выполните детализацию процесса КОНТРОЛЬ КАЧЕСТВА.

После выполнения работы у вас должна получиться следующая диаграмма (Рисунок 6.):

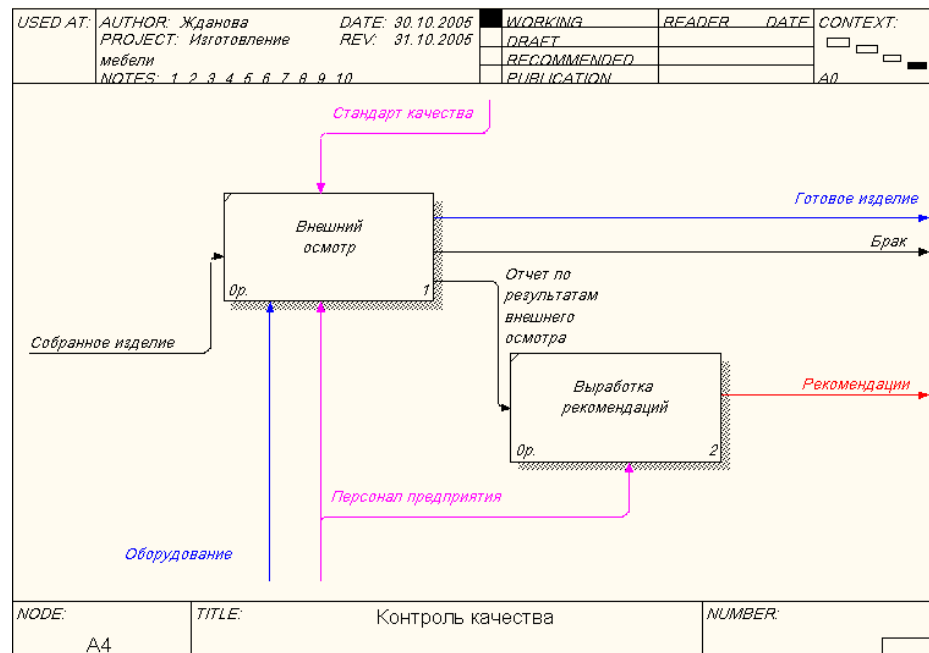


Рисунок 6. Детализация процесса КОНТРОЛЬ КАЧЕСТВА

Описание свойств модели.

IDEF0-модель предполагает наличие четко сформулированной цели, единственного субъекта моделирования и одной точки зрения. Для внесения области, цели и точки зрения в модели IDEF0 в BPwin следует:

1. Выбрать пункт меню **Model - Model Properties**, вызывающий диалог **Model Properties** (Рисунок 7.);

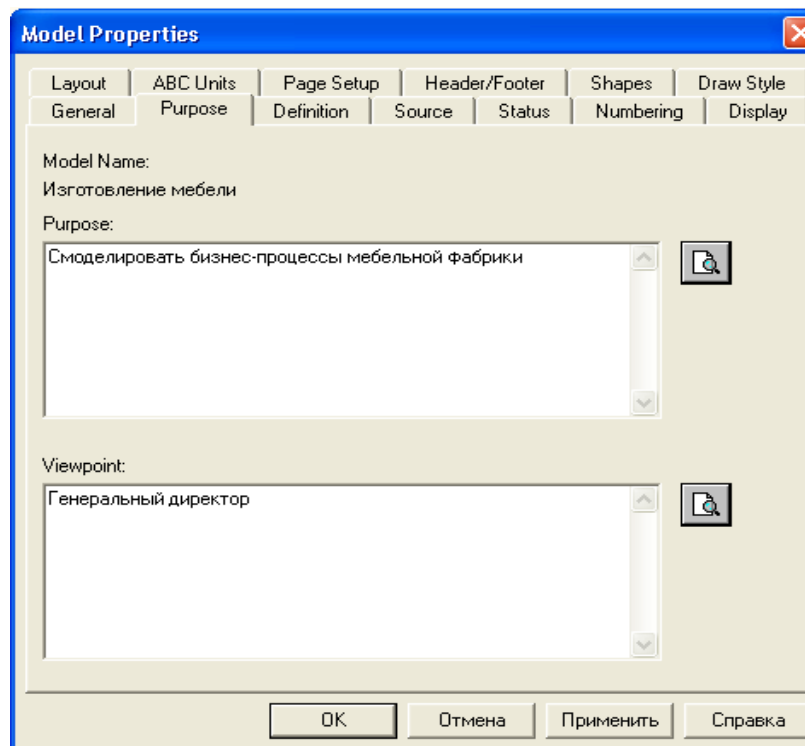


Рисунок 7. Диалог задания свойств модели

2. Во вкладку **Purpose** внести цель и точку зрения, а во вкладку **Definition** – определение модели;

Цель и точку зрения принято выносить на контекстную диаграмму **A-0** в виде текстового блока. После описания они появятся на контекстной диаграмме в виде текстового блока. Описание производится на уровне контекстной диаграммы.

Для описания цели и точки зрения следует:

3. Перейти на уровень диаграммы **A-0**;

4. Выбрать кнопку текста **T** на палитре инструментов;
5. Щелкнуть мышью в позиции предполагаемого ввода текста;
6. В диалоговом окне набрать нужный текст и установить опцию значимости (обычный текст, цель или точка зрения) (Рисунок 8.).

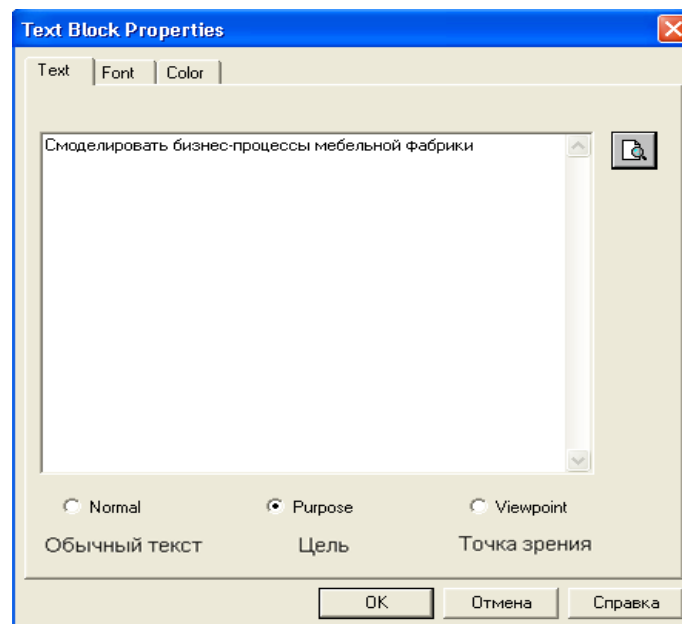


Рисунок .8. Установка опции Text

7. Во вкладке **Status** того же диалога опишите статус модели (черновой вариант, рабочий, окончательный и т.д.), время создания и последнего редактирования (отслеживается в дальнейшем автоматически по системной дате);

8. Во вкладке **Source** опишите источники информации для построения модели (например, «Опрос экспертов предметной области и анализ документации»);

9. Вкладка **General** служит для внесения имени проекта и модели, имени и инициалов автора и временных рамок модели.

Составление отчета.

Результат описания модели можно получить в отчете **Model Report**.

1. Диалоговое окно настройки отчета по модели вызовите из пункта меню **Tools – Reports - Model Report**.

2. Выберите необходимые поля, при этом автоматически отображается очередность вывода информации в отчете (рис. 3.9.);

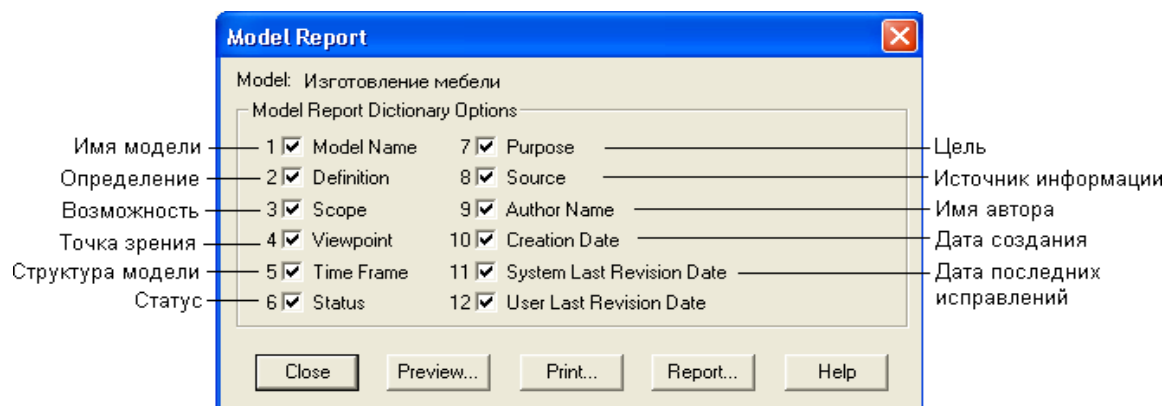


Рисунок 9. Диалоговое окно выбора информации для отчета

3. Нажмите на кнопку **Preview**, чтобы просмотреть отчет (Рисунок 10).

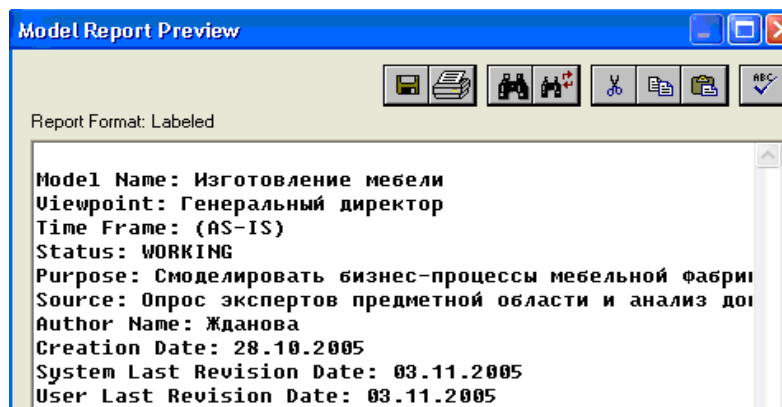


Рисунок 10. Отчет о модели

Сохранение полученной диаграммы.

1. В меню **File** выберите **Save As**.
2. Укажите путь к своей папке и имя файла **Lab3.bp1**.
3. Нажмите ОК.

Контрольные вопросы

1. Как нумеруются модели в иерархии **IDEF0**?
2. Дайте понятие определению **Дерево узлов**.
3. Какой процесс в разработке модели называют функциональной декомпозицией?
4. Как можно вынести цель и точку зрения проекта на диаграмму?
5. Для чего необходимо составление отчета?

ПРАКТИЧЕСКАЯ РАБОТА № 6

Тема: “Открытие древовидных и FEO-диаграмм”

Цель: Научиться производить открытие древовидных и FEO диаграмм.

Оборудование: IBM PC

С помощью этой лабораторной работы Вы сможете:

- освоить принципы построения диаграммы дерева узлов;
- научиться задавать свойства и стиль диаграмме дерева узлов;
- освоить правила построения диаграммы FEO.

Теоретические сведения

Диаграммы дерева узлов и FEO.

📖 Диаграмма дерева узлов показывает иерархию работ в модели и позволяет рассмотреть всю модель целиком, 📖 показывает взаимосвязи между работами (рис. 4.1.).

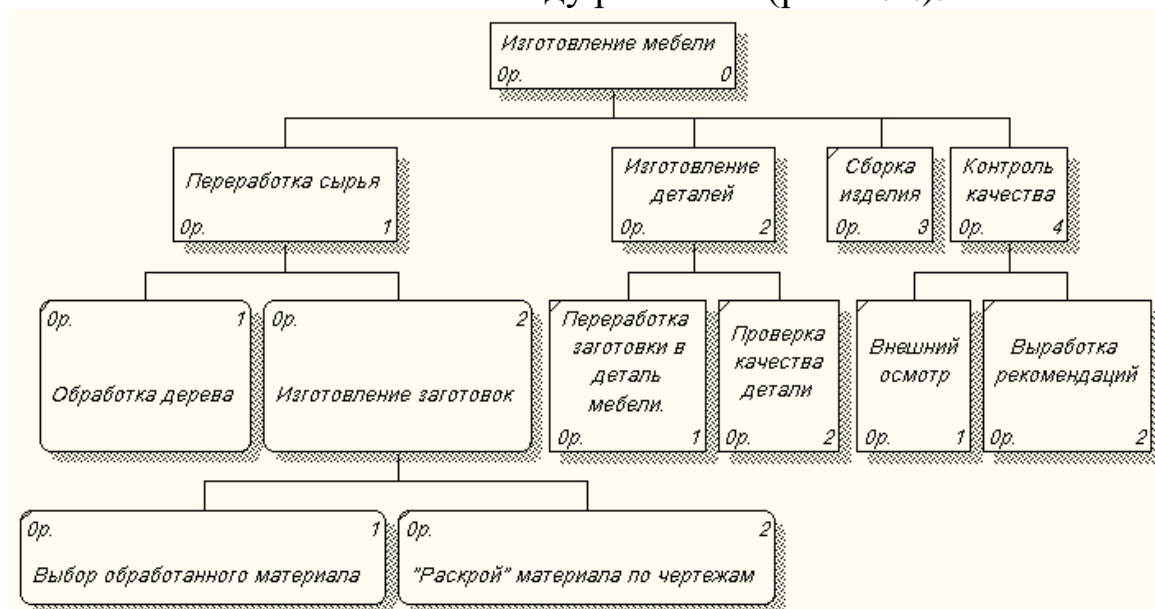


Рис. 4.1. Диаграмма дерева узлов

Процесс создания модели работ является итерационным (повторяющимся, многократно меняющимся), следовательно, работы могут менять свое расположение в дереве узлов многократно. Чтобы не запутаться и проверить способ декомпозиции, следует после каждого изменения создавать диаграмму дерева узлов. Впрочем, ВРwin имеет мощный инструмент навигации по модели - **Model Explorer** (рис. 4.2.), который позволяет представить иерархию работ и диаграмм в удобном и компактном виде, однако этот инструмент не является составляющей стандарта **IDEF0**.

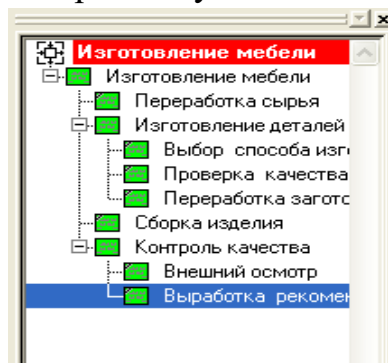


Рис. 4.2. Навигатор модели Model Explorer

Практическое задание «Построение диаграммы дерева узлов (целей)»



Упражнение 23. Создание диаграммы дерева узлов.

1. Откройте файл **Lab3.bp1**, сохраненный на предыдущем уроке.

Для создания диаграммы дерева узлов следует:

2. Выбрать в меню пункт **Diagram - Add Node Tree**.

Появится диалог создания диаграммы дерева узлов **Node Tree Wizard** (рис. 4.3.).

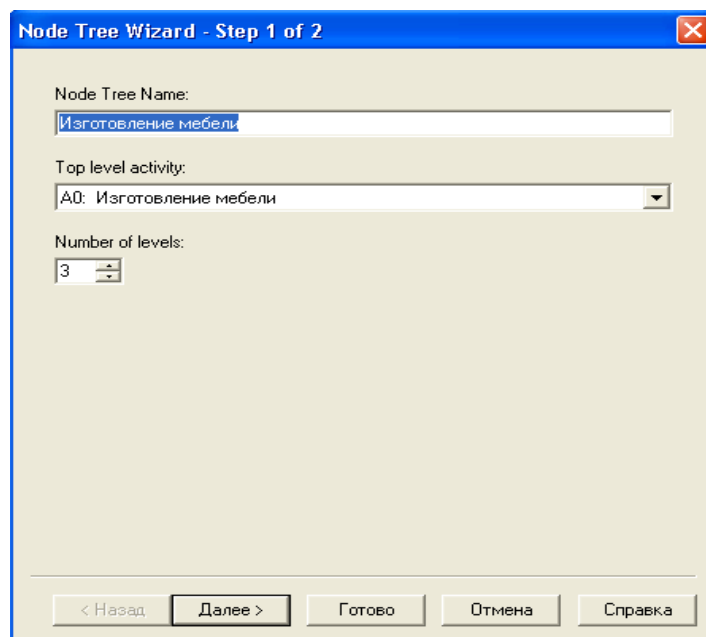


Рис. 4.3. Диалог создания диаграммы дерева узлов Node Tree Wizard

3. В первом диалоге эксперта введите 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 может иметь произвольную глубину. В одной модели можно создавать множество диаграмм деревьев узлов.

Имя дерева узлов по умолчанию совпадает с именем работы верхнего уровня, а номер диаграммы автоматически генерируется как номер узла верхнего уровня плюс литера "N", например, A0N.

Второй диалог эксперта **Node Tree Wizard** (рис. 4.4.) позволяет задать свойства диаграммы дерева узлов.

По умолчанию нижний уровень декомпозиции показывается в виде списка, остальные работы - в виде прямоугольников (рис. 4.5.).

Для отображения всего дерева в виде прямоугольников следует убрать опцию **Bullet Last Level**. Группа **Connection Style** позволяет выбрать стиль соединительных линий - диагональные (по умолчанию) или ортогональные.

4. Щелкните правой кнопкой мыши по свободному месту, не занятому объектами, выберите меню **Node tree Diagram Properties** (рис. 4.6.).

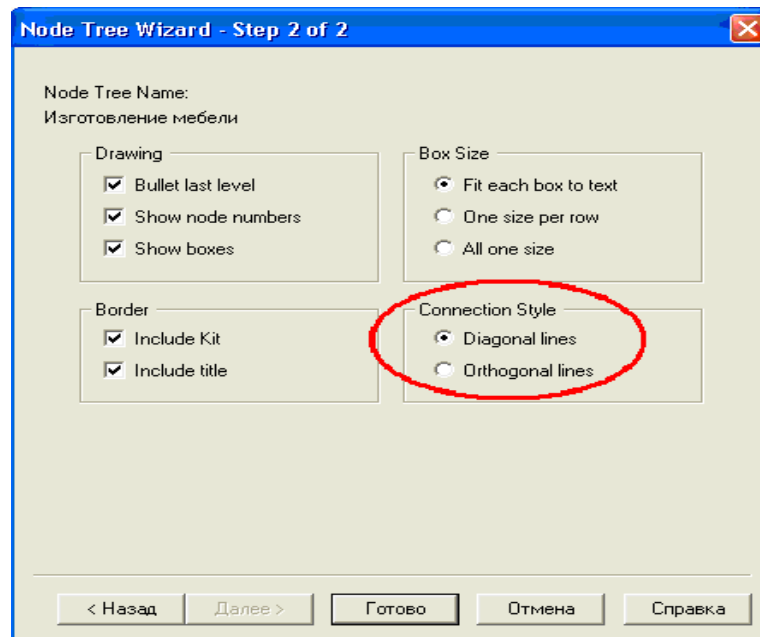


Рис. 4.4. Диалог эксперта Node Tree Wizard

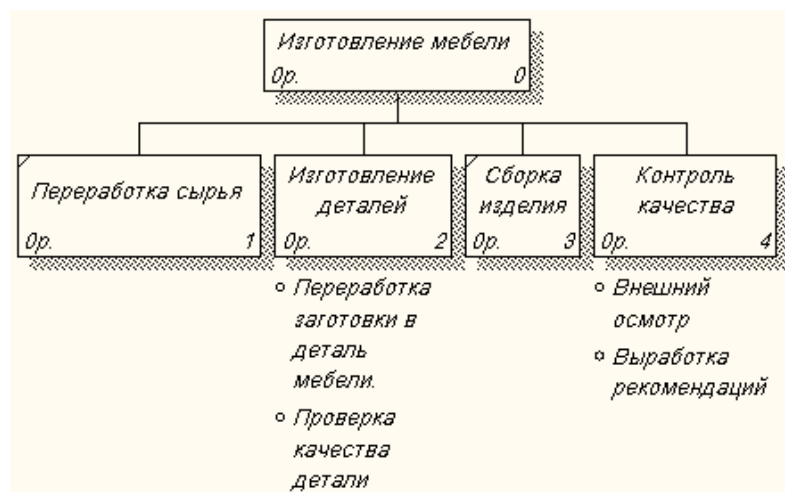


Рис. 4.5. Дерево узлов с ортогональными линиями

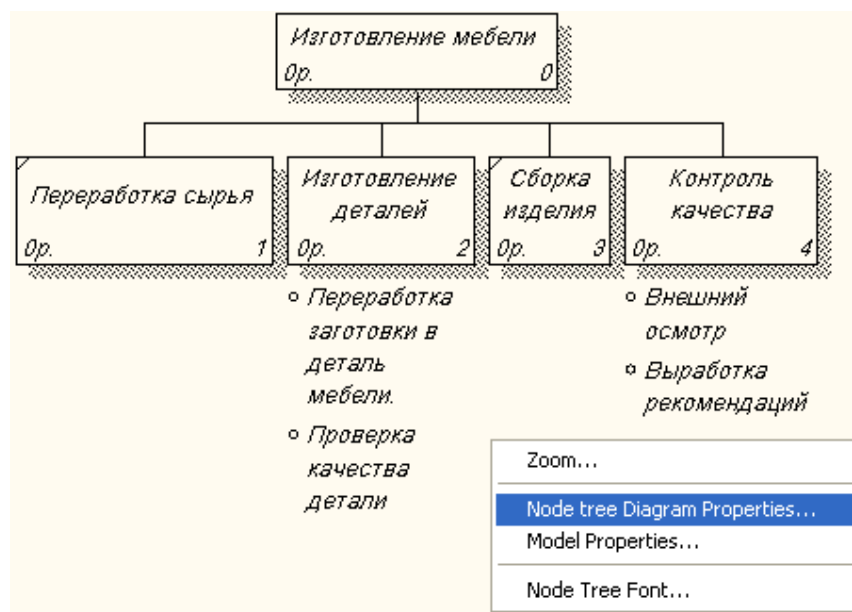


Рис. 4.6. Выбор меню *Node tree Diagram Properties*

5. Во вкладке **Style** диалога **Node Tree Properties** отключите опцию **Bullet Last Level** (рис. 4.7.).

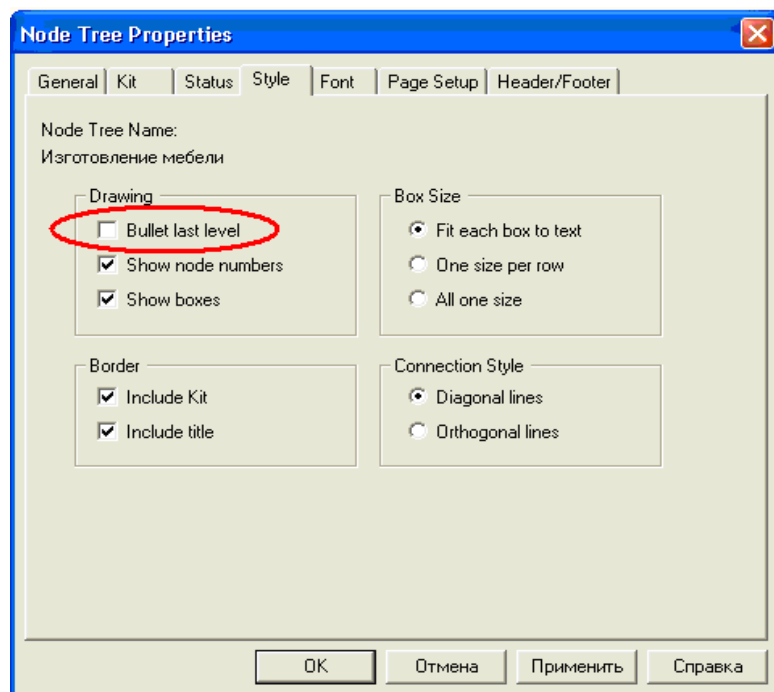


Рис. 4.7. Диалоговое окно *Node Tree Properties*

6. Щелкните по кнопке **ОК**.
7. Проверьте полученный результат (рис. 4.8.).

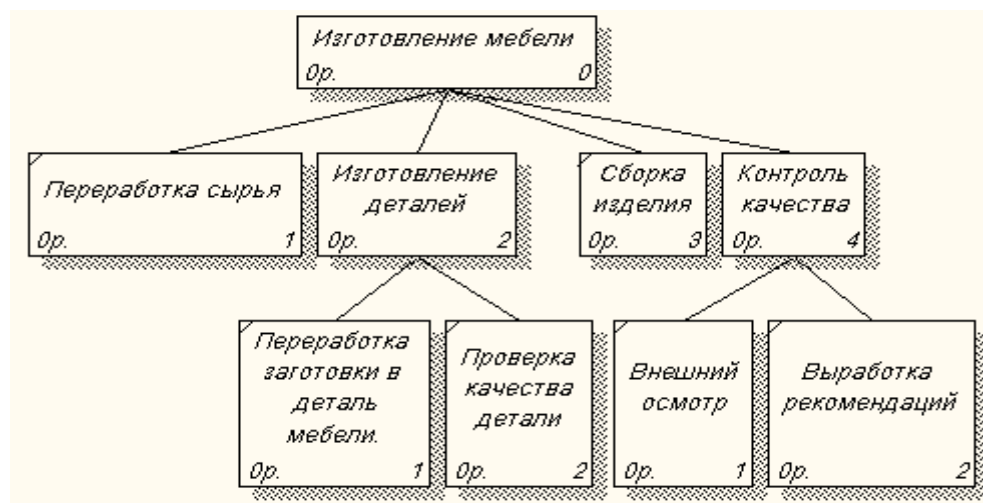


Рис. 4.8. Дерево узлов

8. Самостоятельно создайте диаграмму Дерево узлов с ортогональными линиями к работе «Изготовление деталей».
9. Проверьте себя (рис. 4.9.).

ПРАКТИЧЕСКАЯ РАБОТА №7

Тема: «Декомпозиция действий. Требования IDEF3 к описанию модели бизнес-процесса»

Цель работы: Получить практические навыки в построении IDEF3-модели бизнес-процесса средствами пакета BPWin.

Оборудование: IBM PC

3. Создание контекстной IDEF3-диаграммы.

Для создания диаграммы в нотации IDEF3 необходимо выбрать в системном меню пункт File/New. В диалоге, который изображен на рисунке 2.1, необходимо определить имя модели и используемый метод (IDEF3).

Рис. 2.1. Определение метода при создании диаграммы

После нажатия на кнопку Ok появляется диалоговое окно Properties, в котором необходимо определить автора модели. После нажатия на кнопку Ok появляется окно диаграммы с контекстной диаграммой, содержащей единственный блок (работу верх-него уровня). Введите имя блока, выбрав в контекстном меню пункт Name

В IDEF3 вместо понятия «функциональный блок» используется понятие «еди-ница работы (Unit of Work, UOW)» или «работа (activity)». UOW изображаются пря-моугольниками с прямыми углами и имеют имя, выраженное отглагольным существи-тельным, обозначающим процесс действия. Обычно номер работы состоит из номе-ра родительской работы и порядкового номера на текущей диаграмме. Контекстную диаграмму можно декомпозировать аналогично декомпозиции в нотации IDEF0, т.е. с помощью инструмента ., но в диалоге указывается нотация IDEF3. На декомпозиционной диаграмме необходимо разместить и именовать все блоки, например, как на рисунке 2.4.

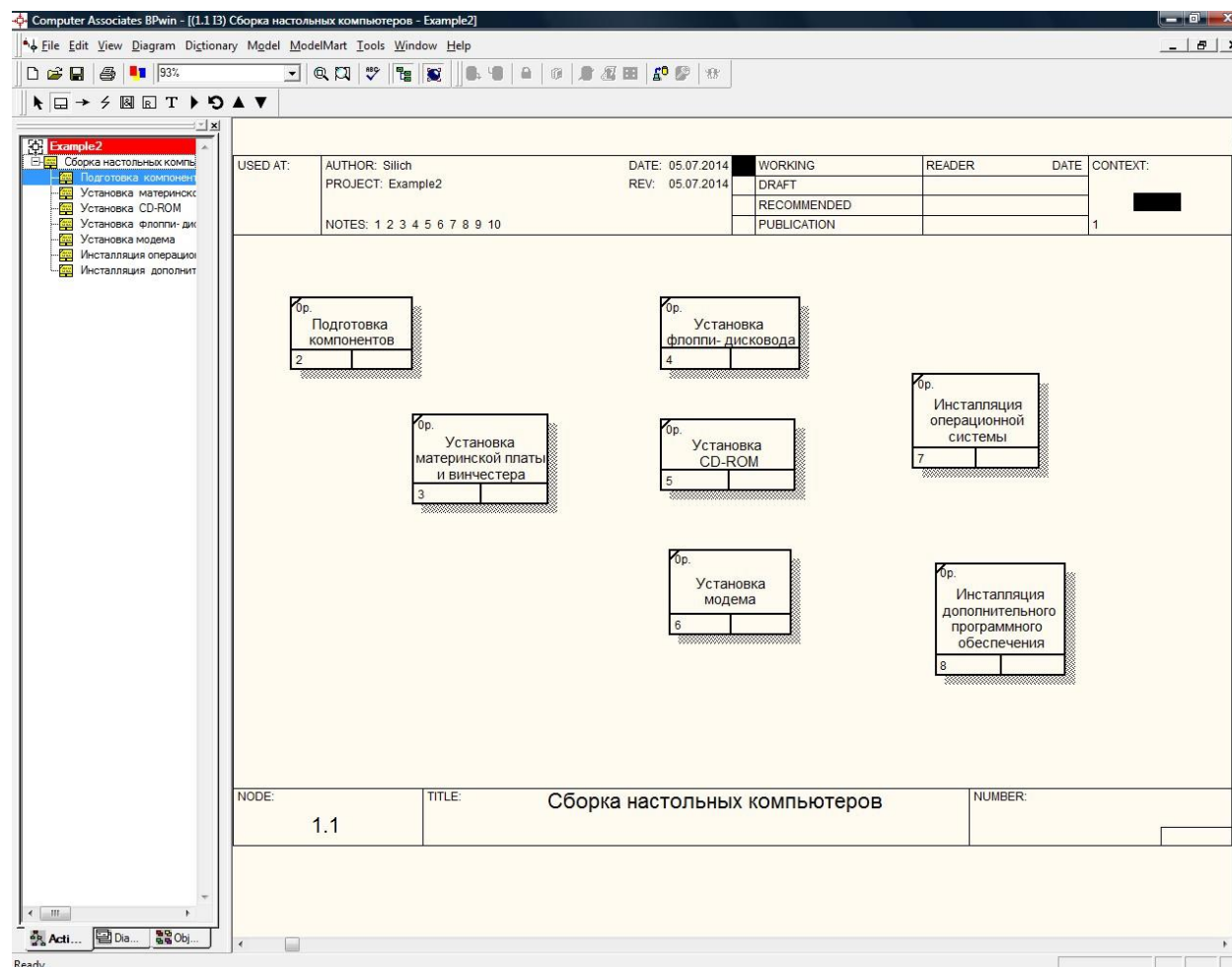


Рис. 2.4. Декомпозиционная диаграмма с функциональными блоками

В отличие от IDEF0-диаграмм блоки обычно располагаются не по диагональной схеме, а по линейной – справа налево, так, чтобы работа, которая выполняется первой, была на левом краю диаграммы, а выполняемая последней – на правом. По-токи работ, выполняемые параллельно, а также альтернативные потоки работ, располагают друг над другом. Иногда работы располагают в порядке следования сверху вниз. 23

Рис. 2.4. Декомпозиционная диаграмма с функциональными блоками

Для связывания блоков используются стрелки. Все связи в IDEF3 однонаправленные и могут быть направлены куда угодно, но обычно диаграммы IDEF3 стараются построить так, чтобы связи были направлены слева направо. В IDEF3 различают три типа стрелок, изображающих связи, стиль которых устанавливается во вкладке Style диалога Arrow Properties (пункт Style контекстного меню).

Для отображения логики взаимодействия стрелок при слиянии и разветвлении или для отображения множества событий, которые могут или должны быть завершены перед началом следующей работы используются **перекрестки (Junction)**. Окончание одной работы может служить сигналом к началу нескольких работ, или же одна работа для своего запуска может ожидать окончания нескольких работ.

Различают перекрестки слияния (Fan-in Junction) и разветвления (Fan-out Junction) стрелок. Перекресток не может использоваться одновременно для слияния и для ветвления. Все перекрестки на диаграмме нумеруются, каждый номер имеет префикс J. Можно редактировать свойства перекрестка при помощи диалога Junction Properties (вызывается из контекстного меню). В отличие от IDEF0 в IDEF3 стрелки могут сливаться и разветвляться только через перекрестки.

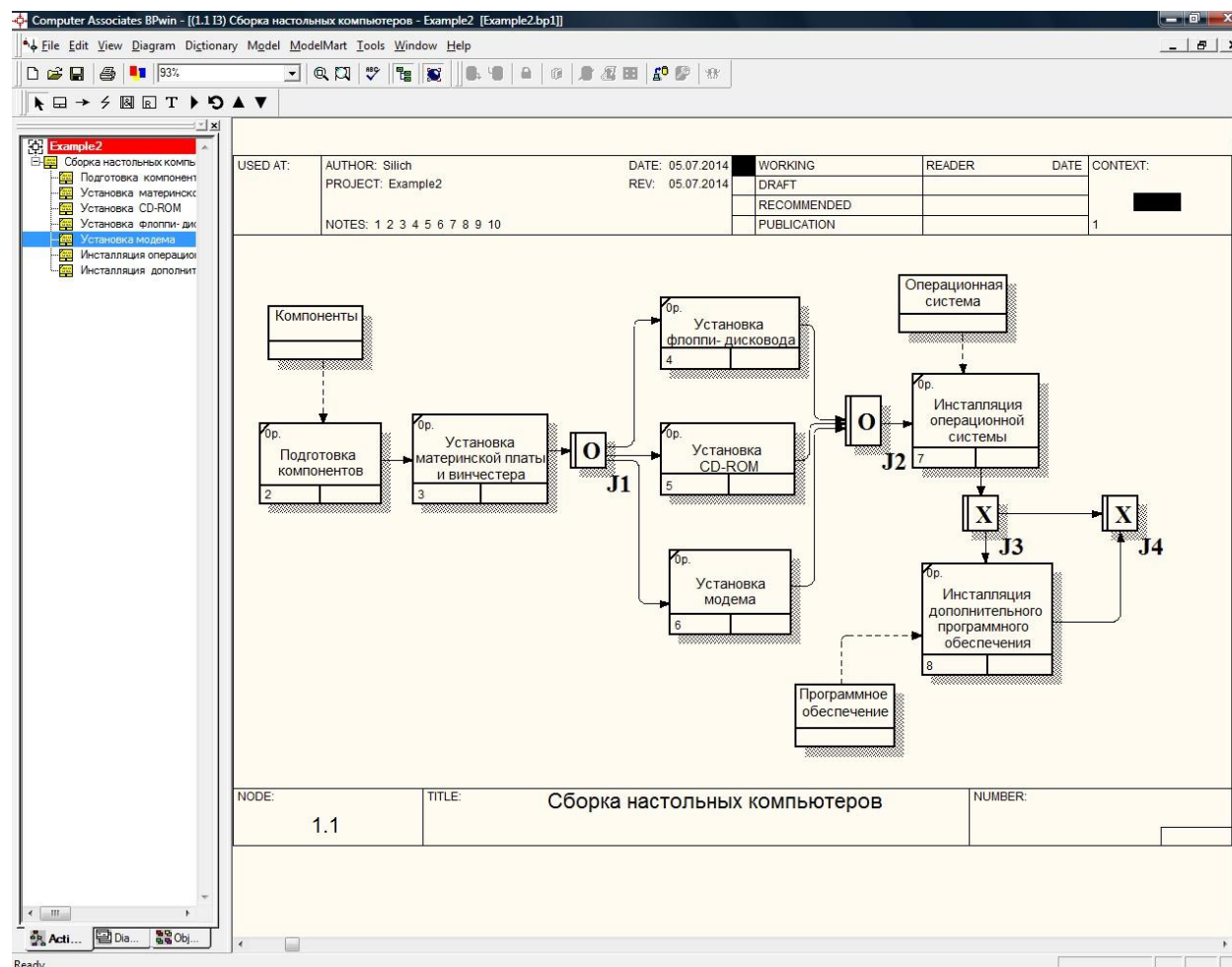


Рис. Рис. 2.6. Декомпозиционная диаграмма с перекрестками

Практическая работа № 8.

Тема: Определение сценария, границ моделирования, точки зрения. Определения действий и объектов.

Цель: Изучить определение сценария, границ моделирования, точки зрения. Определения действий и объектов.

Оборудование: IBM PC.

Теоретическая часть

Метод описания процессов IDEF3

IDEF3 - это метод, имеющий основной целью дать возможность аналитикам описать ситуацию, когда процессы выполняются в определенной последовательности, а также описать объекты, участвующие совместно в одном процессе.

IDEF3 может быть также использован как метод создания процессов. IDEF3 дополняет IDEF0 и содержит все необходимое для построения моделей, которые в дальнейшем могут быть использованы для имитационного анализа.

Диаграммы. Диаграмма является основной единицей описания в IDEF3. Важно правильно построить диаграммы, поскольку они предназначены для чтения другими людьми (а не только автором).

Объект ссылки. Объект ссылки в IDEF3 выражает некую идею, концепцию или данные, которые нельзя связать со стрелкой, перекрестком или работой (рис. 7.11). Объект ссылки изображается в виде прямоугольника, похожего на прямоугольник работы. Имя объекта ссылки задается в диалоге Referent Properties (пункт контекстного меню Name), в качестве имени можно использовать имя какой-либо стрелки с других диаграмм или имя сущности из модели данных. Объекты ссылки должны быть связаны с единицами работ или перекрестками пунктирными линиями. Официальная спецификация IDEF3 различает три стиля объектов ссылок - безусловные (unconditional), синхронные (synchronous) и асинхронные (asynchronous). BPwin поддерживает только безусловные объекты ссылок. Синхронные и асинхронные объекты ссылок, используемые в диаграммах переходов состояний объектов, не поддерживаются.

Методика выполнения упражнения

- 1 Откройте файл Комп.br1
- 2 Перейдите на диаграмму A2 и декомпозируйте работу "Сборка настольных компьютеров" (рисунок 7.1).



Рисунок 6.12 – Диаграмма A2 с объектом декомпозиции

- 3 В диалоге **Activity Box Count** (рисунок 7.2) установите число работ 4 и нотацию **IDEF3**.

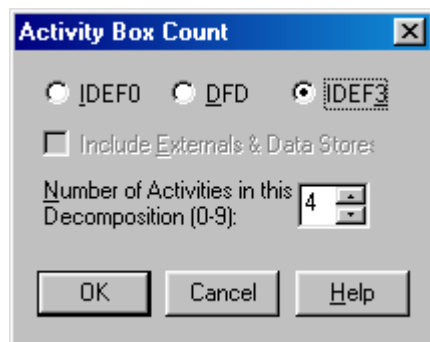


Рисунок 6.13 - Выбор нотации IDEF3 в диалоге **Activity Box Count**

Возникает диаграмма **IDEF3** (рисунок 7.3), содержащая работы **Unit of Work (UOW)**, также называемыми единицами работы или работами (**activity**). Правой кнопкой мыши щелкните по работе с номером 1, выберите в контекстном меню **Name** и внесите имя работы "**Подготовка компонентов**" (рисунок 6.14).

Затем во вкладке **Definition** внесите определение работы с номером 1 "**Подготавливаются все компоненты компьютера согласно спецификации заказа**" (рисунок 6.15).

4 Во вкладке **UOW** диалогового окна **Activity Properties** внесите свойства работы 1 в соответствии с данными таблицы 6.1.

Таблица 6.3 - Свойства UOW диалогового окна **Activity Properties**

Objects	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
Constraints	Установка модема требует установки дополнительного программного обеспечения

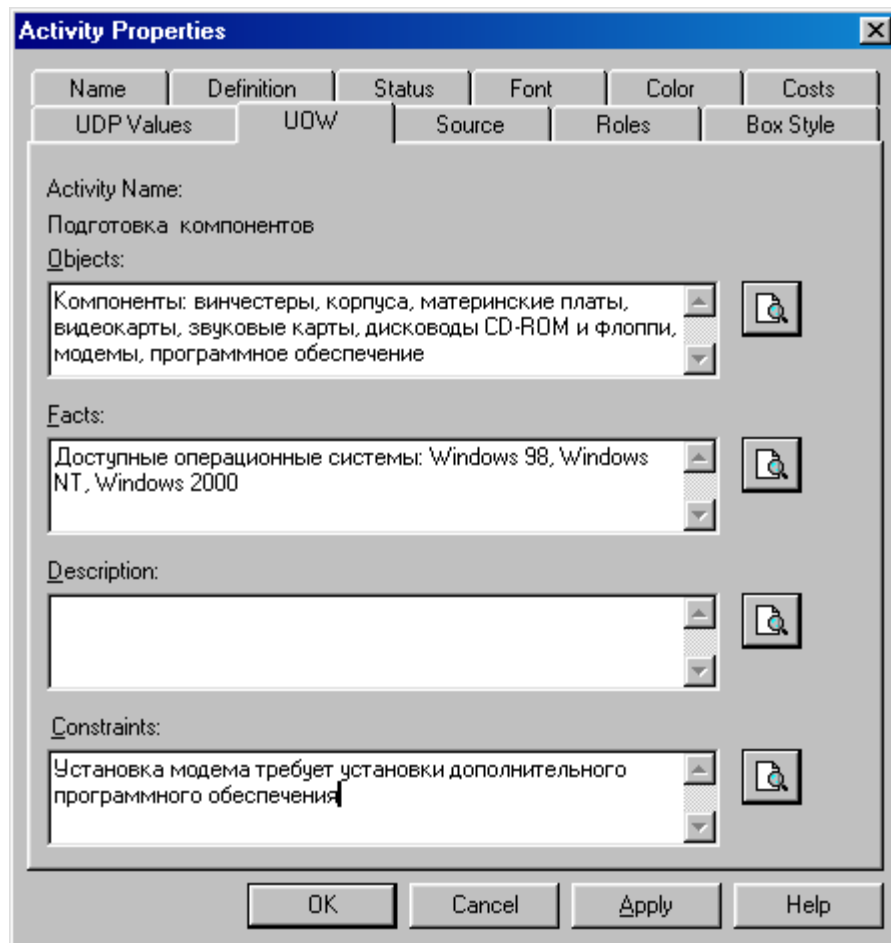


Рисунок 6.16 – Диалоговое окно **Activity Properties** вкладка **UOW**


5 Внесите в диаграмму еще 3 работы (кнопка ) и присвойте имена работам с номерами 2...7 в соответствии с данными таблицы 7.2:

Таблица 6.4 – Названия работ

Номер работы	Название работы
2	Установка материнской платы и винчестера
3	Установка модема
4	Установка дисководов CD-ROM
5	Установка флоппи- дисководов
6	Инсталляция операционной системы
7	Инсталляция дополнительного программного обеспечения

Диаграмма **IDEF3** должна выглядеть так, как показано на рисунке 6.17.

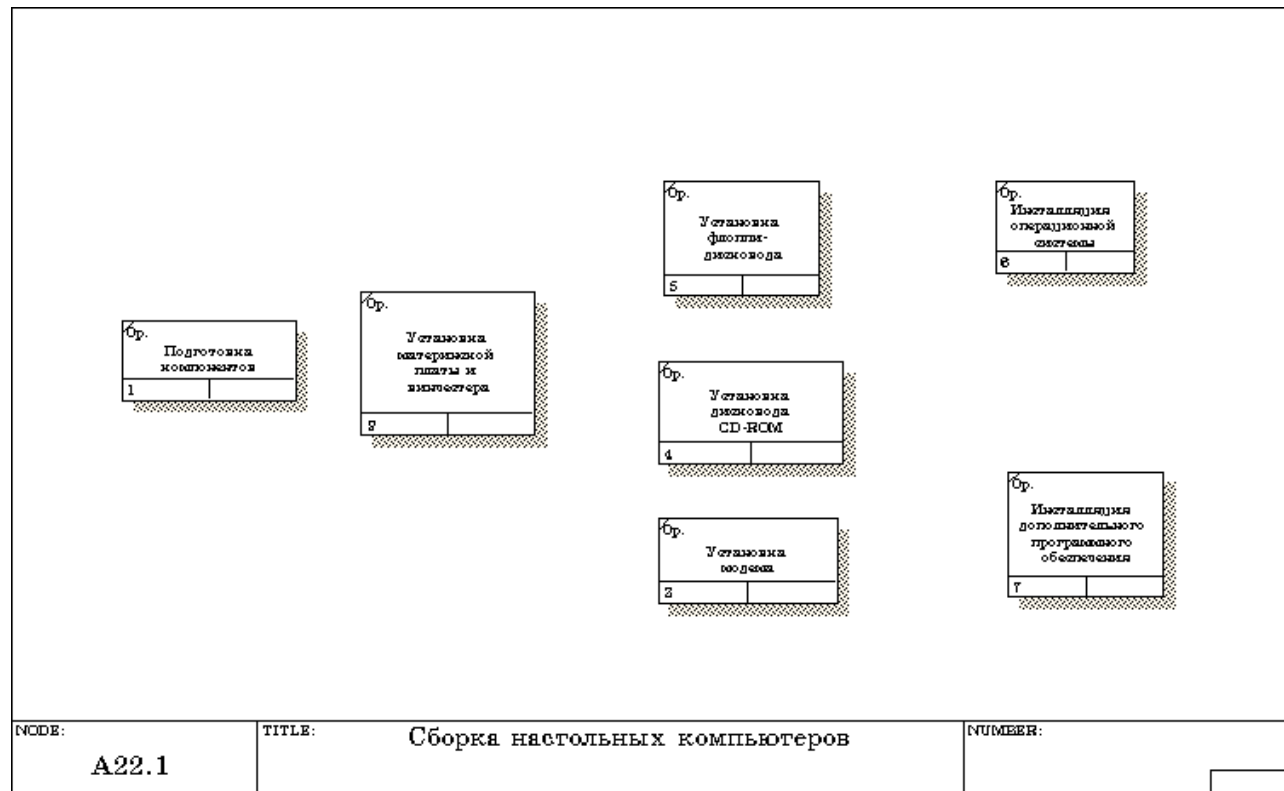



Рисунок 6.17 – Диаграмма **IDEF3** после присвоения работам названий

6 С помощью кнопки  палитры инструментов создайте объект ссылки. Внесите имя объекта внешней ссылки " **Компоненты**" (рисунок 6.18).

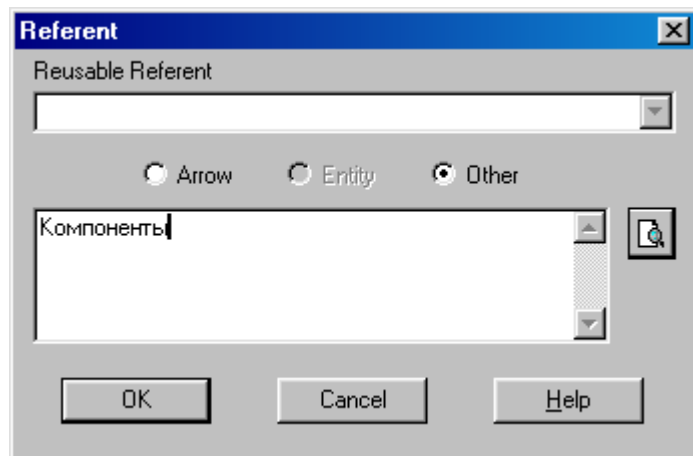


Рисунок 6.18 – Создание объекта ссылки

Свяжите стрелкой объект ссылки и работу "**Подготовка компонентов**" (рисунок 6.19).

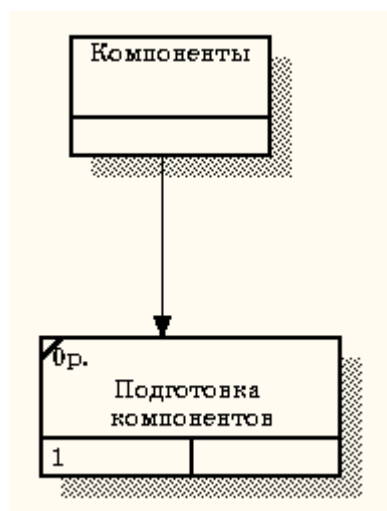


Рисунок 6.19 - Объект ссылки и работа **"Подготовка компонентов"**
связаны стрелкой

Измените стиль стрелки, связывающей объект ссылки и работу **"Подготовка компонентов"**, воспользовавшись диалоговым окном **Arrow Properties** как показано на рисунке 6.20.

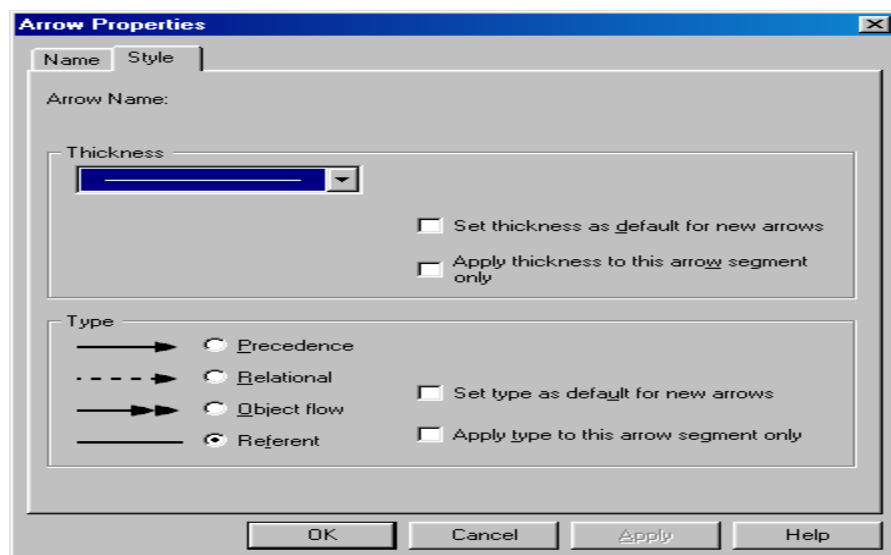


Рисунок 6.20 – Изменение стиля стрелки

7 Свяжите стрелкой работы **"Подготовка компонентов"** (выход) и **"Установка материнской платы и винчестера"** (вход). Измените стиль стрелки на **Object Flow**.

На диаграммах IDEF3 имя стрелки может отсутствовать, хотя **BPwin** показывает отсутствие имени как ошибку.

Результат выполнения пункта 6 показан на рисунке 6.21.

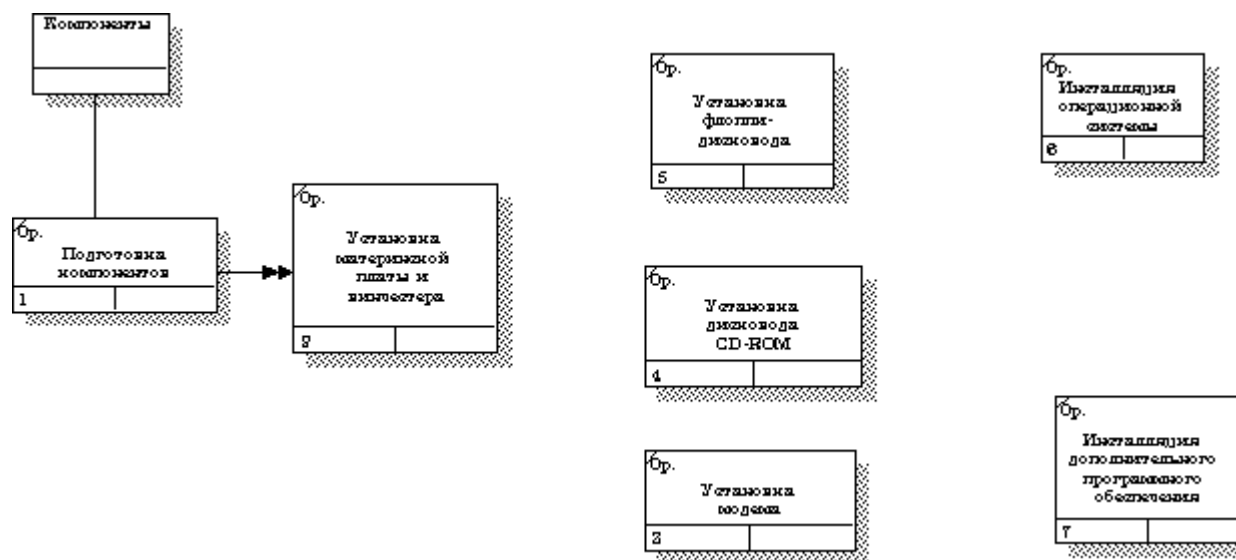



Рисунок 6.21 - Результат создания UOW и объекта ссылки

8 С помощью кнопки  на палитре инструментов внесите два перекрестка типа "**асинхронное ИЛИ**" (рисунок 6.22)

Свяжите работы с перекрестками, как показано на рисунке 6.23.

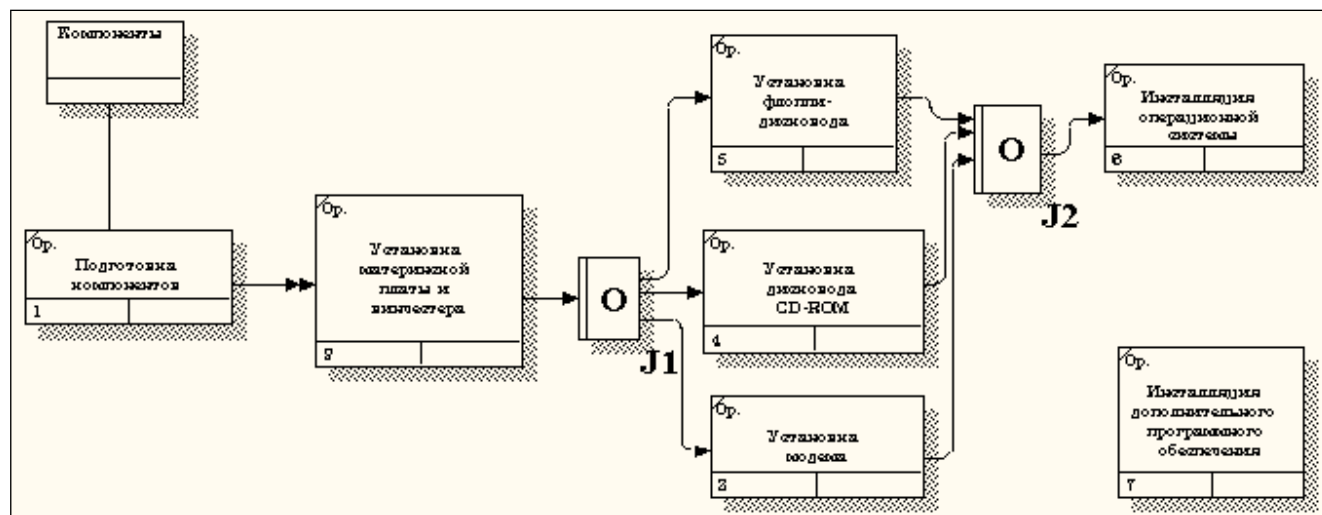


Рисунок 6.23 - Диаграмма IDEF3 после создания перекрестков

9 Правой кнопкой щелкните по перекрестку для разветвления **J1 (fan-out)**, выберите **Name** и внесите имя "Компоненты, требуемые в спецификации заказа" (рисунок 6.24).

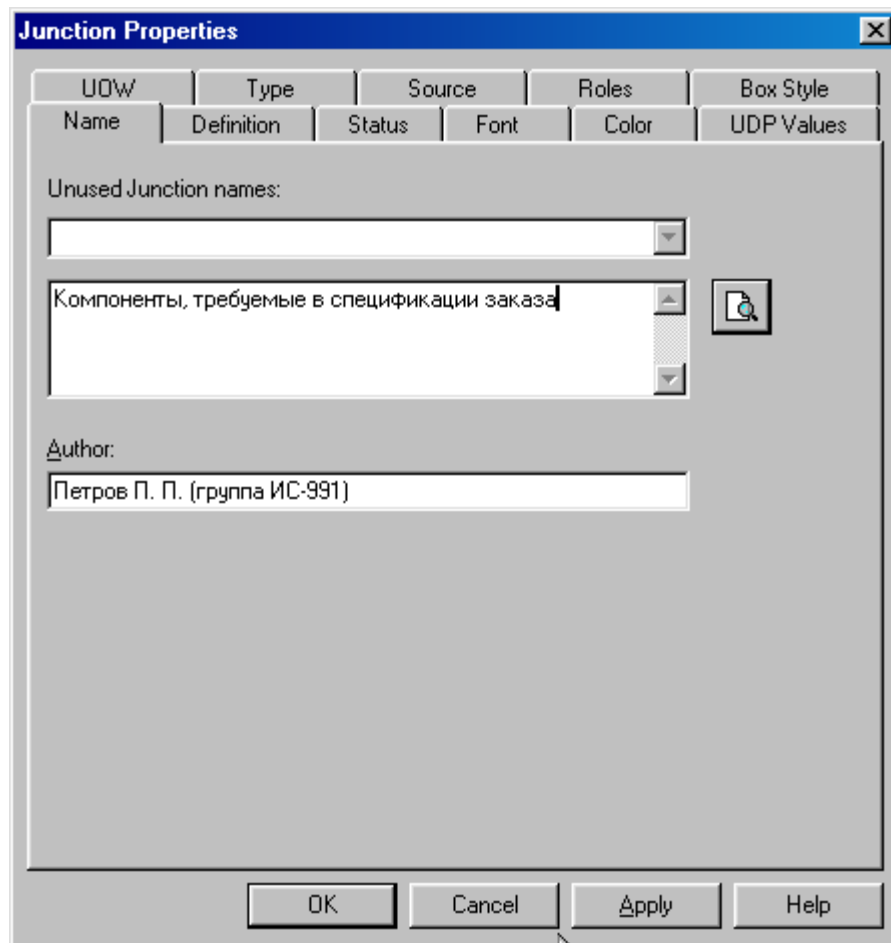



Рисунок 6.24 – Присвоение имени перекрестку **J1**

10 С помощью кнопки  палитры инструментов введите в диаграмму еще один объект ссылки и присвойте ему имя **"Программное обеспечение"**.

11 Создайте два перекрестка типа "исключающее ИЛИ". Свяжите работы и соответствующие ссылки, как это показано на рисунке 6.25.

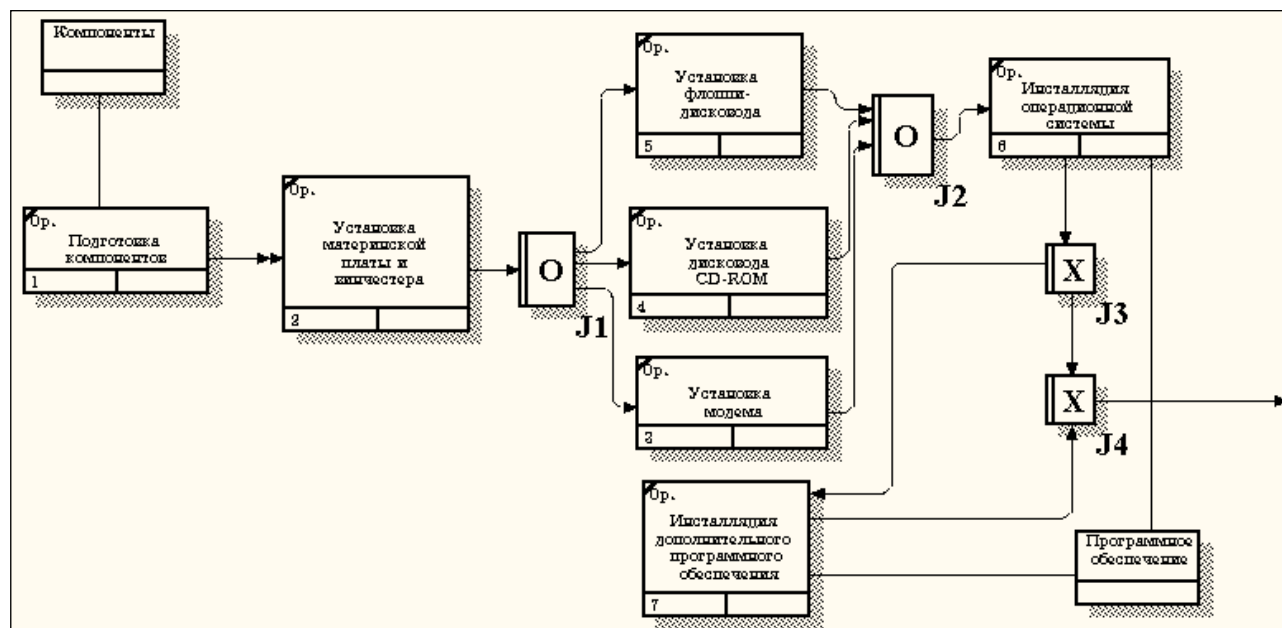


Рисунок 6.25- Результат выполнения работы 6

ПРАКТИЧЕСКАЯ РАБОТА №9

Тема: "Последовательность и параллельность взаимосвязь моделей IDEF0 и IDEF3"

Цель: Изучить последовательность и параллельность взаимосвязь моделей IDEF0 и IDEF3

Оборудование: IBM PC

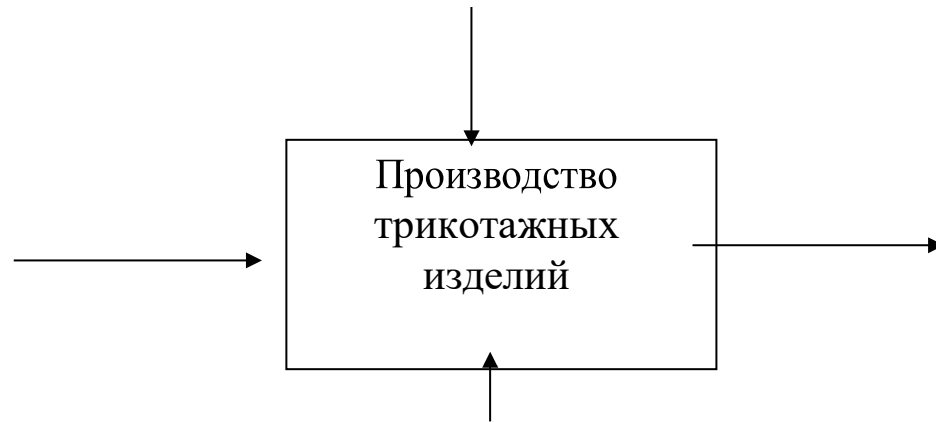


Рисунок 1 Диаграмма нулевого уровня

Выполнить декомпозицию IDEF3

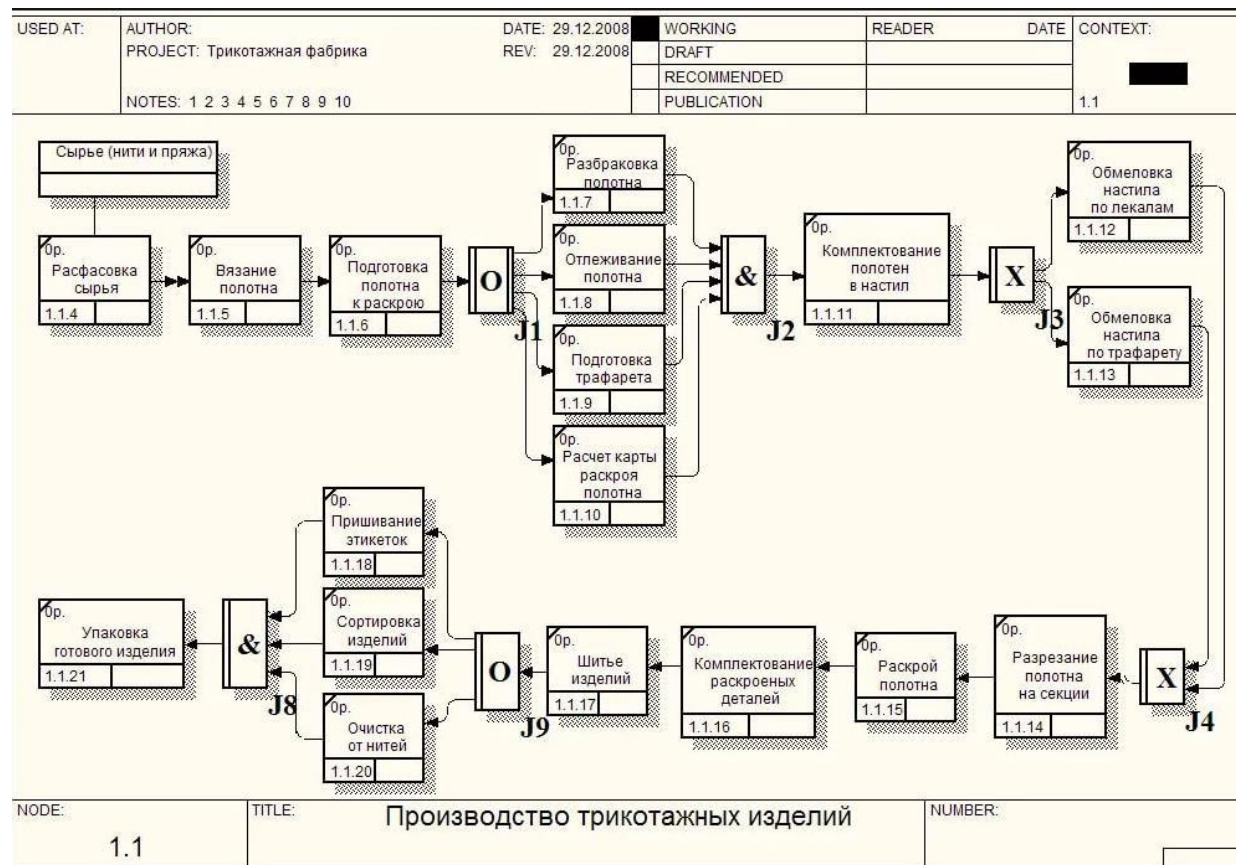


Рисунок 2 Диаграмма первого уровня

Практическая работа №10

Тема работы: Моделирование потоков данных.

Цель работы: Изучить основные объекты диаграммы потоков данных

Вопросы:

Назовите основные объекты диаграмм потоков данных	
Назовите названия кнопок панели инструментов	
На данной диаграмме назовите	
процессы;	
накопитель данных;	
потоки данных	
автора	
Название проекта	
Как изменить вид стрелок?	
Как изменить цвет объекта:?	

ПРАКТИЧЕСКАЯ РАБОТА №11

Тема: “Два подхода к построению DFD-моделей. Нумерация объектов”

Цель: Изучить два подхода к построению DFD-моделей. Нумерация объектов

Оборудование: IBM PC

Создание контекстной DFD -диаграммы.

Контекстная DFD-диаграмма создается так же, как и аналогичная диаграмма в нотации IDEF0 или IDEF3 (пункт меню File/New), но в диалоге

нужно указать тип модели – Data Flow (DFD). Определите автора модели в диалоговом окне Properties.

Появится окно с контекстной диаграммой, содержащее работу (процесс обработки информации) верхнего уровня.

Работы в DFD представляют собой функции системы, преобразующие входы в выходы. Хотя работы изображаются прямоугольниками со скругленными углами, смысл их совпадает со смыслом работ IDEF0 и IDEF3. Так же как работы IDEF3, они имеют входы и выходы, но не поддерживают управления и механизмы, как IDEF0. Работа на контекстной диаграмме обычно именуется по названию системы, например *"Система обработки информации"*.

Внешние сущности изображают входы в систему и/или выходы из системы. Как правило, они представляют собой материальный предмет или физическое лицо, например: *Заказчик, Пользователь, Персонал, Поставщик, Клиент, Банк*. Внешние сущности изображаются в виде прямоугольника с тенью и обычно располагаются по краям диаграммы. Одна внешняя сущность может быть использована многократно на одной или нескольких диаграммах. Обычно такой прием используют, чтобы не рисовать слишком длинных и запутанных стрелок.

Пример контекстной DFD-диаграммы приведен на рисунке 3.1.

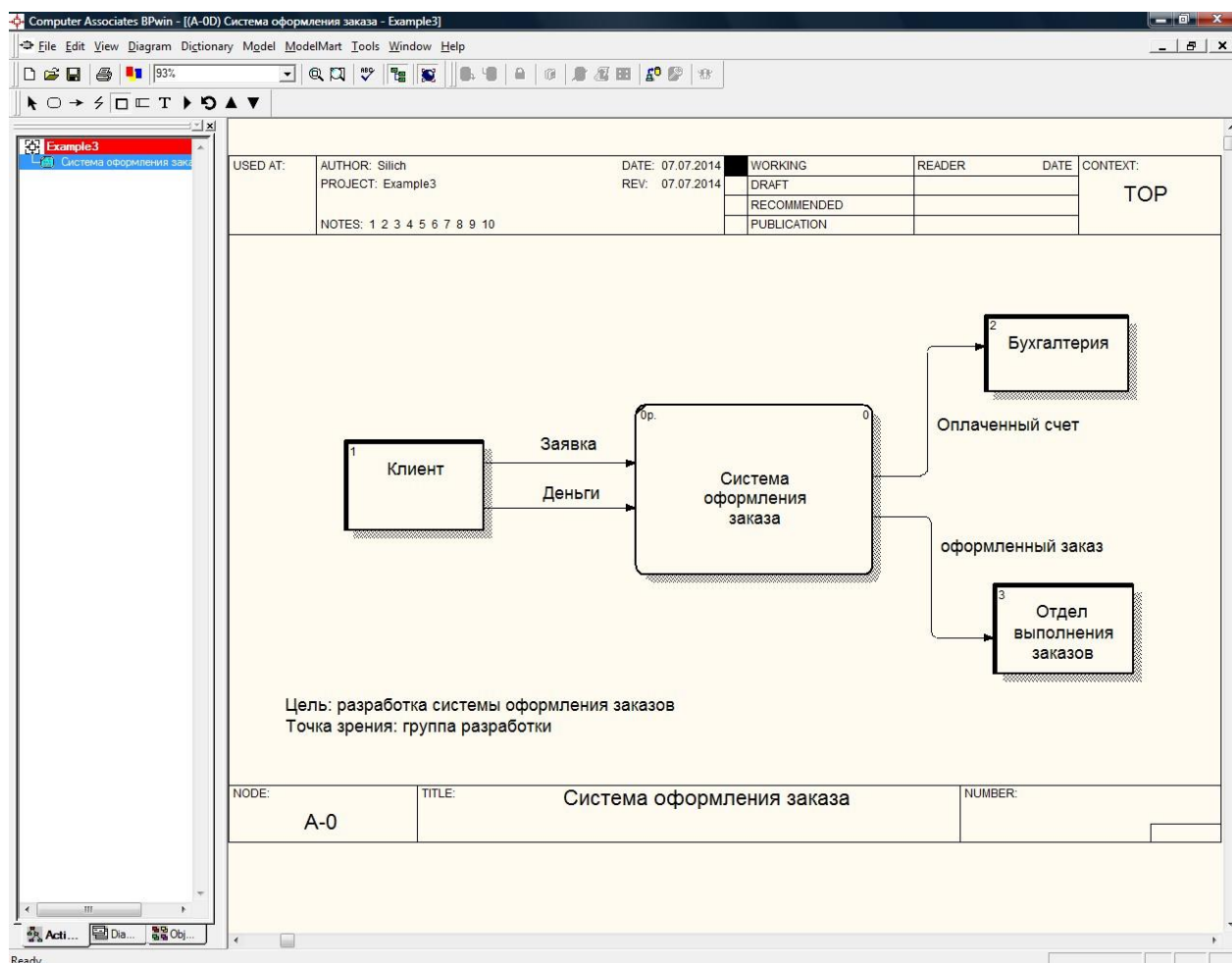


Рис. 3.1. Контекстная диаграмма в нотации DFD

На диаграмму декомпозиции с контекстной диаграммы будут перенесены стрелки входа и выхода родительской работы. Они будут представлены в виде граничных стрелок (см. рис. 3.2). Согласно нотации DFD диаграмма не должна иметь граничных стрелок – все стрелки должны начинаться и заканчиваться на работах, хранилищах данных или внешних сущностях. Поэтому следует удалить все граничные стрелки, создать соответствующие внешние сущности и вместо граничных провести внутренние стрелки, связывающие внешние сущности и работы. При этом на контекстной диаграмме стрелки входа и выхода родительской работы, соответствующие удаленным граничным стрелкам, будут иметь знак туннелирования в виде квадратных скобок.

Пример декомпозиционной DFD-диаграммы приведен на рисунке 3.3.

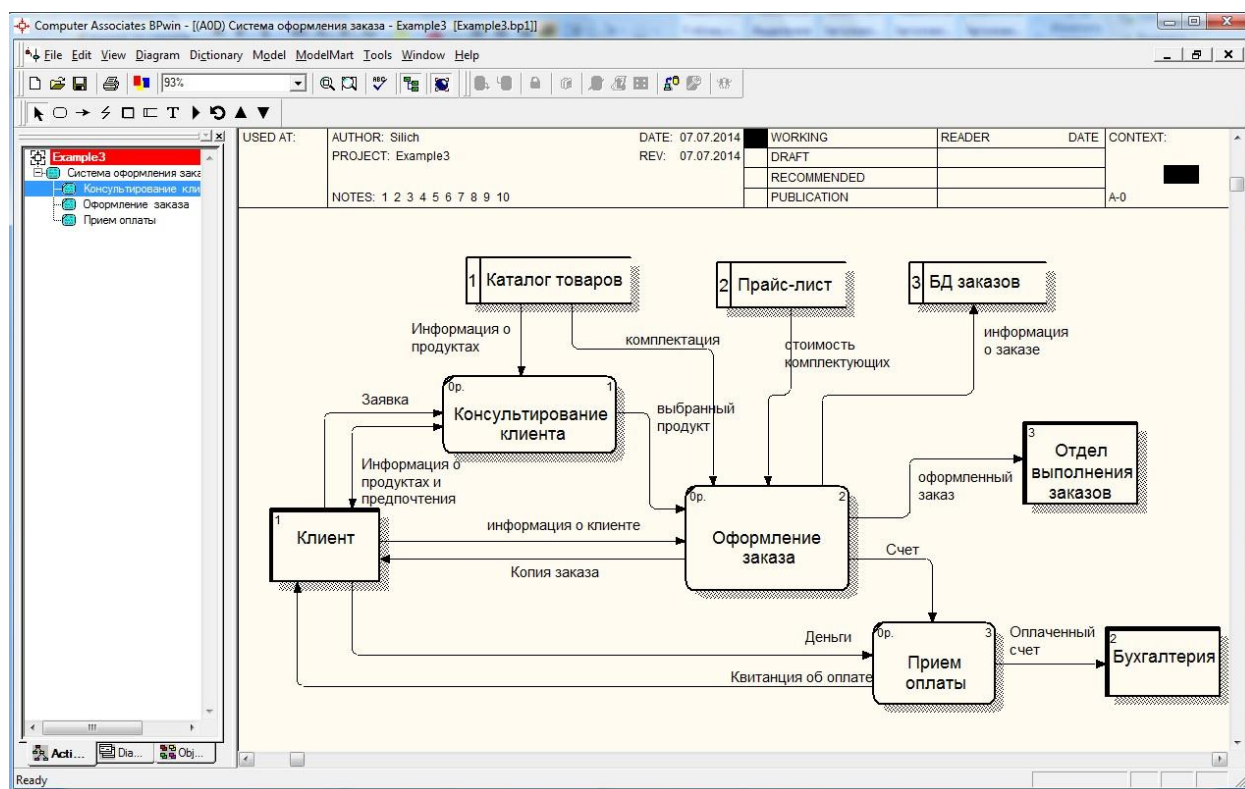


Рис 3.2. Декомпозиционная DFD-диаграмма

ПРАКТИЧЕСКАЯ РАБОТА №11

Тема: “Два подхода к построению DFD-моделей. Нумерация объектов”

Цель: Изучить два подхода к построению DFD-моделей. Нумерация объектов

Оборудование: IBM PC

Создание контекстной DFD -диаграммы.

Контекстная DFD-диаграмма создается так же, как и аналогичная диаграмма в нотации IDEF0 или IDEF3 (пункт меню File/New), но в диалоге нужно указать тип модели – Data Flow (DFD). Определите автора модели в диалоговом окне Properties.

Появится окно с контекстной диаграммой, содержащее работу (процесс обработки информации) верхнего уровня.

Работы в DFD представляют собой функции системы, преобразующие входы в выходы. Хотя работы изображаются прямоугольниками со скругленными углами, смысл их совпадает со смыслом работ IDEF0 и IDEF3. Так же как работы IDEF3, они имеют входы и выходы, но не поддерживают управления и механизмы, как IDEF0. Работа на контекстной диаграмме обычно именуется по названию системы, например *"Система обработки информации"*.

Внешние сущности изображают входы в систему и/или выходы из системы. Как правило, они представляют собой материальный предмет или физическое лицо, например: *Заказчик, Пользователь, Персонал, Поставщик, Клиент, Банк*. Внешние сущности изображаются в виде прямоугольника с тенью и обычно располагаются по краям диаграммы. Одна внешняя сущность может быть использована многократно на одной или нескольких диаграммах. Обычно такой прием используют, чтобы не рисовать слишком длинных и запутанных стрелок.

Пример контекстной DFD-диаграммы приведен на рисунке 3.1.

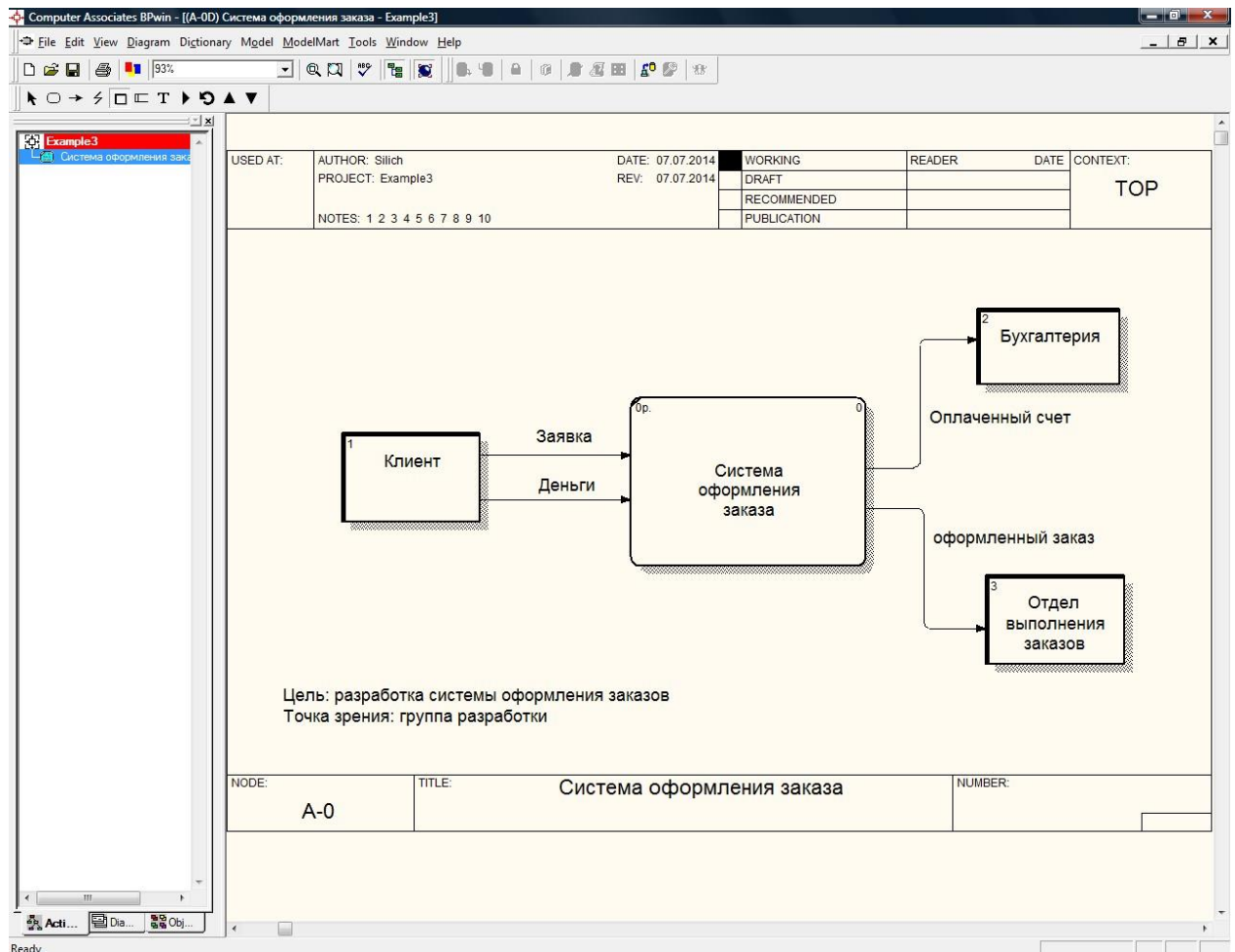


Рис. 3.1. Контекстная диаграмма в нотации DFD

На диаграмму декомпозиции с контекстной диаграммы будут перенесены стрелки входа и выхода родительской работы. Они будут представлены в виде граничных стрелок (см. рис. 3.2). Согласно нотации DFD диаграмма не должна иметь граничных стрелок – все стрелки должны начинаться и заканчиваться на работах, хранилищах данных или внешних сущностях. Поэтому следует удалить все граничные стрелки, создать соответствующие внешние сущности и вместо

граничных про-вести внутренние стрелки, связывающие внешние сущности и работы. При этом на контекстной диаграмме стрелки входа и выхода родительской работы, соответствующие удаленным граничным стрелкам, будут иметь знак туннелирования в виде квадратных скобок.

Пример декомпозиционной DFD-диаграммы приведен на рисунке 3.3.

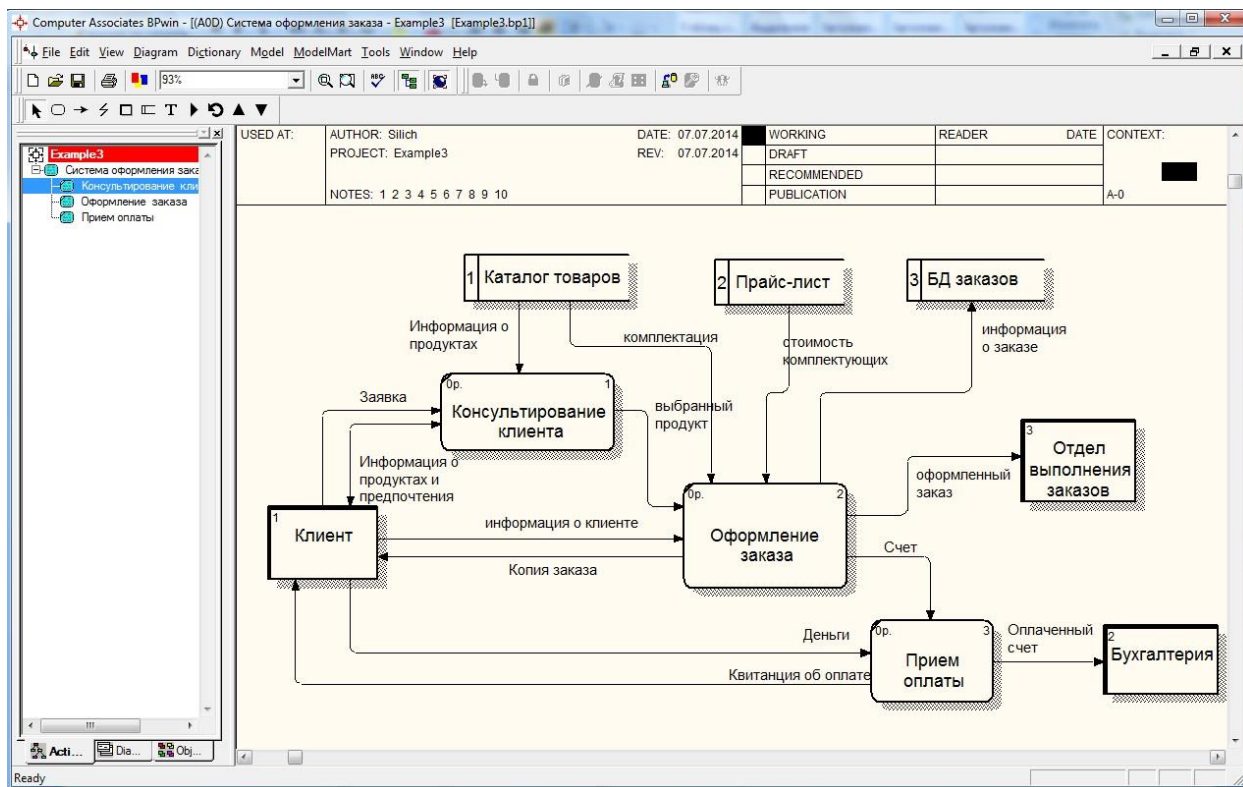


Рис 3.2. Декомпозиционная DFD-диаграмма

Практическая работа № 12

Тема: «Оформление моделей, другие виды моделей IDEF0»

Цель: Научиться оформлять модели и другие виды моделей IDEF0.

Оборудование: IBM PC.

Настройка параметров стоимостного анализа.

1. Откройте файл **Lab6.bp1**, сохраненный на предыдущем уроке.
2. В диалоговом окне **Model Properties (Model - Model Properties)** во вкладке **ABC** установите единицы измерения денег и времени – рубли и часы (Рисунок 1.).

Model Properties

General | Purpose | Definition | Source | Status | Numbering | Display | Layout | **ABC Units** | Page Setup | Header/Footer | Shapes | Draw Style

Model Name:
Изготовление мебели

Cost

Currency description: Рубли | Symbol placement: 1p.

Symbol: p. | Number of decimals in diagrams: 0 | Number of decimals in reports: 2

Time

Time Unit: Days | Decimals in frequency values: 2 | Decimals in duration values: 2

OK | Отмена | Применить | Справка

Рисунок 1. Вкладка ABC Unit диалога Model Properties

3. Перейдите в **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 комплектов мебели, контролем над сборкой и проверкой качества изделий
Рабочая сила	Затраты на оплату рабочих, занятых изготовлением изделий
Компоненты	Затраты на закупку компонентов

4. Для отображения стоимости каждой работы в нижнем левом углу прямоугольника перейдите в меню **Model - Model Properties** и во вкладку **Display** диалога **Model Properties** включите опцию **ABC Data** (Рисунок 2.).

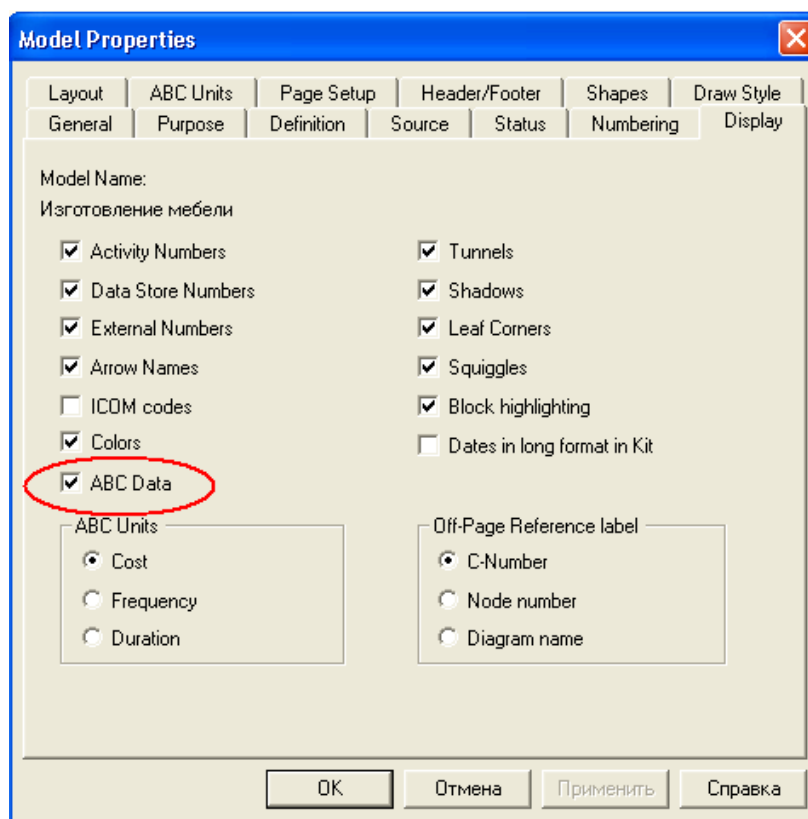


Рисунок 2. Вкладка *Display* диалога *Model Properties*

5. Для изображения стоимости, частоты или продолжительности работы переключите радиокнопки в группе **ABC Units** (Рисунок 3.).

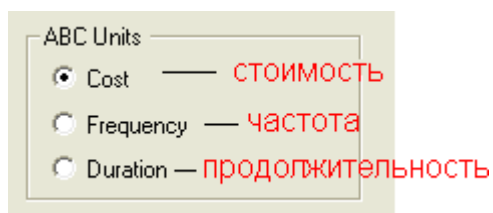


Рисунок 3. Назначение радиокнопок в группе *ABC Units*

6. Для назначения стоимости работе «Переработка сырья» щелкните по ней правой кнопкой мыши и выберите в контекстном меню **Cost**.

Откроется диалоговое окно для внесения стоимости затрат (Рисунок 4.):

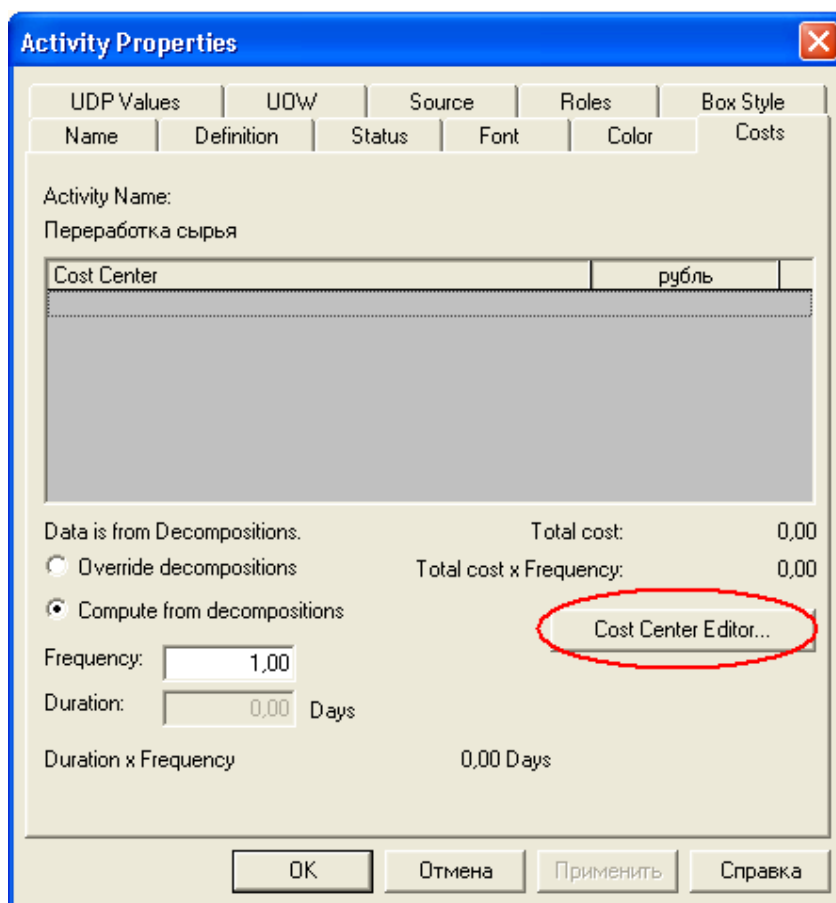


Рисунок 4. Диалог Activity Properties

7. В диалоговом окне **Cost Center Editor** добавьте центры затрат «Компоненты», «Рабочая сила», «Управление» (Рисунок 5.).

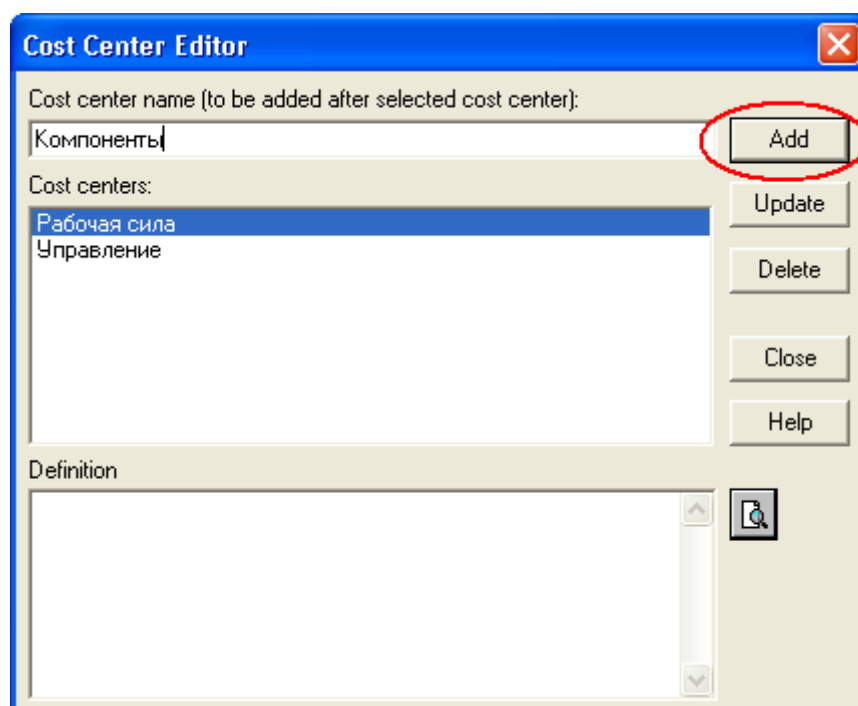


Рисунок 5. Диалог Cost Center Editor

8. Во вкладке **Costs** диалога **Activity Properties** укажите частоту проведения данной работы в рамках общего процесса (**Frequency**) и продолжительность (**Duration**).

9. Выберите в списке один из центров затрат и в окне **Cost** задайте его стоимость (Рисунок 6.).

Activity Properties

UDP Values | UOW | Source | Roles | Box Style

Name | Definition | Status | Font | Color | Costs

Activity Name:
Переработка сырья

Cost Center	
Компоненты	20 000,00
Рабочая сила	1 500,00
Управление	100,00

Data is from this level.

☒ Override decompositions ☐ Compute from decompositions

Frequency: 5,00

Duration: 1,00 Days

Duration x Frequency: 0,00 Days

Total cost: 21 600,00

Total cost x Frequency: 108 000,00

Cost Center Editor...

OK Отмена Применить Справка

Рисунок 6. Внесение стоимости в центры затрат

ПРАКТИЧЕСКАЯ РАБОТА №13

Тема: "Разбиение и объединение моделей"

Цель: Выполнить разбиение и объединение моделей.

Оборудование: IBM PC

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.

ВРwin использует для слияния и разветвления моделей стрелки вызова.

Для слияния необходимо выполнить следующие условия:

- обе сливаемые модели должны быть открыты в ВРwin;
- 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 диагональную 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.

Появляется диалог, в котором следует указать опции слияния модели. При слиянии моделей объединяются и словари стрелок и работ. В случае одинаковых определений возможна перезапись определений или принятие определений из модели источника. То же относится к именам стрелок, хранилищам данных и внешним ссылкам.

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 туннелировать вручную.

В процессе слияния модель-источник остается неизменной и к модели-цели подключается фактически ее копия. Не нужно путать слияние моделей с синхронизацией. Если в дальнейшем модель-источник будет редактироваться, эти изменения автоматически не попадут в соответствующую ветвь модели-цели.

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 во всплывающем меню пункт Split Model. В появившемся диалоге Split Options 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 ее имя будет совпадать с именем новой модели, и, наконец, 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.

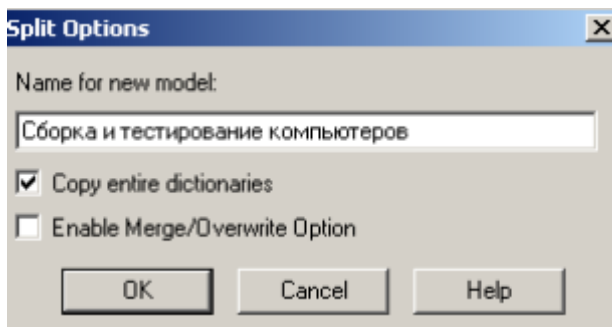


Рисунок 1. Диалог Split Option

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.

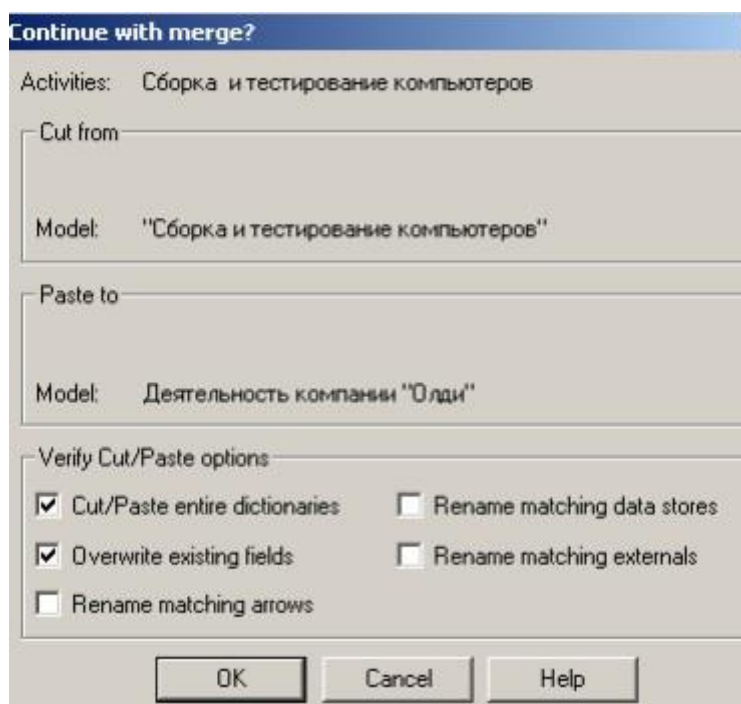


Рисунок 2. Диалог Continue with merge?

В диалоге Merge Model включите опцию Cut/Paste entire dictionaries щелкните по ОК.

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?

Практическая работа №14

Тема: “Получение отчетов по модели.”

Цель: Получить навыки построения отчетов по модели.

Оборудование: IBM PC.

1. Для работ «Изготовление деталей», «Сборка изделия», «Контроль качества» на диаграмме A1 самостоятельно внесите параметры ABC из табл. 1

Таблица 1 Стоимости работ на диаграмме A1

Имя работы (Activity Name)	Центр затрат (Cost Center)	Сумма центра затрат (Cost Center Cost), руб.	Продолжител ь-ность (Duration), день	Частота (Frequen cy)
Отслеживание расписания и управление сборкой мебели и её проверкой	Управление	1500,00		
Переработка сырья	Управление	100,00		
	Рабочая сила	1500,00	1,00	5,00
	Компонент ы	20000,00		
Изготовление деталей	Управление	200,00		
	Рабочая сила	2500,00	2,00	5,00
	Компонент ы	200,00		
Сборка мебели	Управление	600,00		
	Рабочая сила	1500,00	2,00	5,00
	Компонент ы	200,00		
Контроль качества	Управление	600,00		
	Рабочая сила	1000,00	1,00	5,00
	Компонент ы	0,00		

2. Проверьте себя (Рисунок 1.)

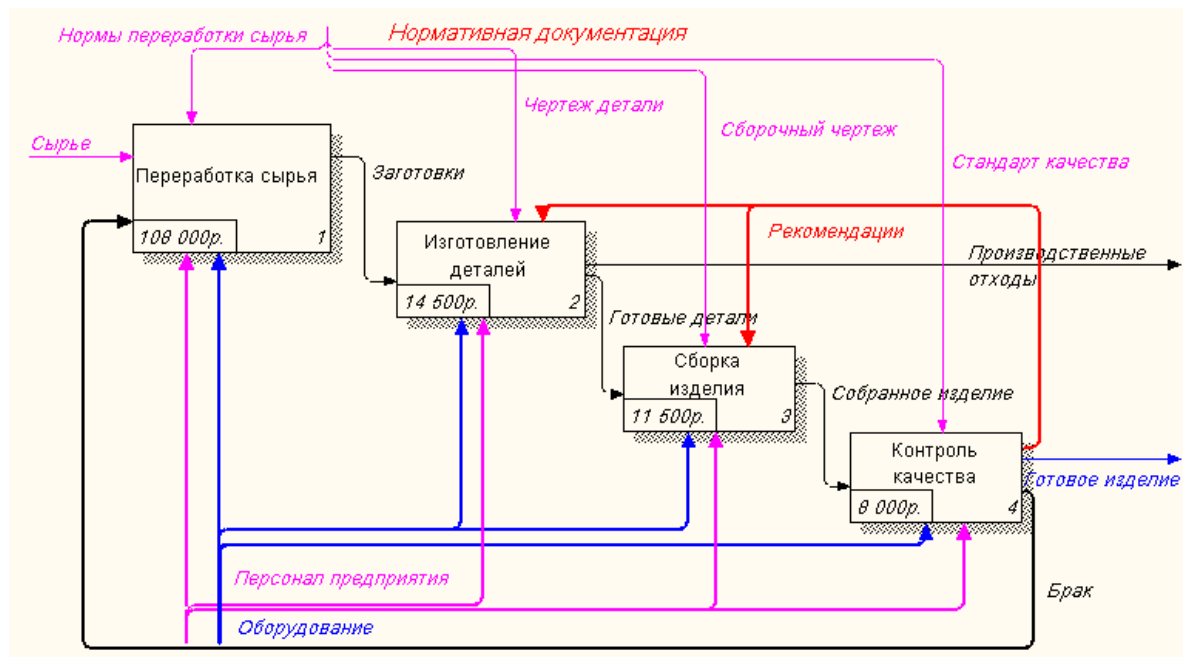


Рисунок 1. Результат стоимости работ на диаграмме А1

3. Посмотрите результат – стоимость работы верхнего уровня (Рисунок 1.).

Составление отчета.

Для того чтобы сгенерировать отчет, выполните следующие действия:

1. Выберите пункт меню **Activity Cost Report (Tools – Reports - Activity Cost Report)**.
2. Укажите пункты в диалоге **Activity Based Costing Report (Рисунок 2.)**, по которым хотите получить сведения.

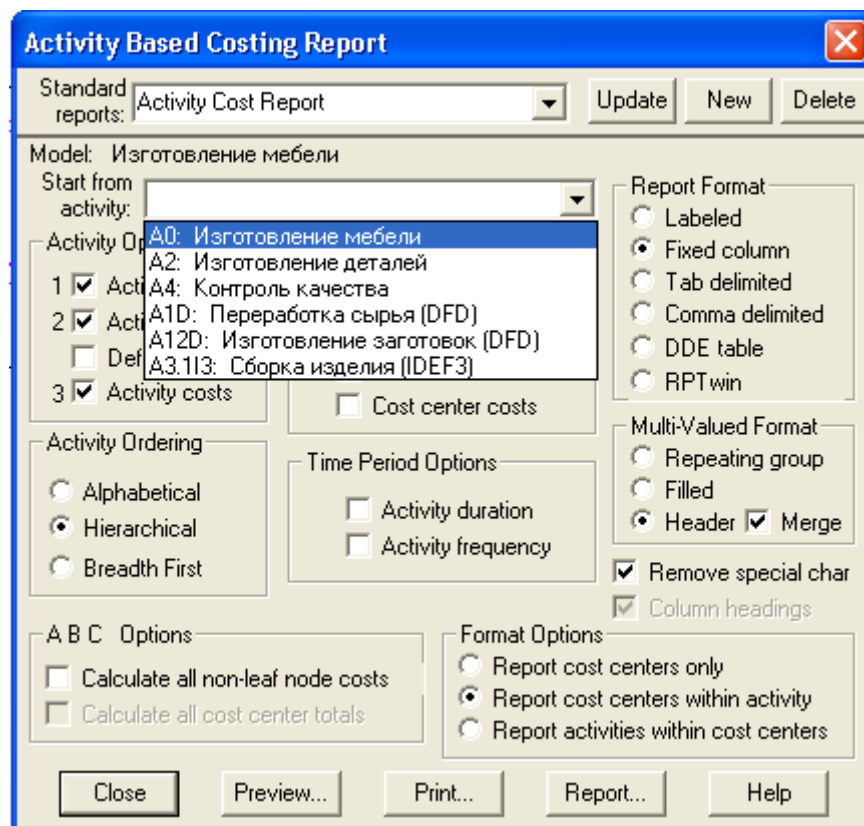


Рисунок 2. Диалог Activity Based Costing Report

ПРАКТИЧЕСКАЯ РАБОТА №15

Тема: “Получение отчётов по модели”

Цель: Получить отчеты по модели

Оборудование: IBM PC

Практическое задание «Разработка диаграммы функциональной декомпозиции»


На предыдущих лабораторных работах вы построили контекстную диаграмму процесса "Изготовление мебели" и провели его детализацию с помощью диаграммы верхнего уровня. Последним шагом построения модели является **функциональная декомпозиция**, т.е. разбиение сложных процессов на более простые. Этот процесс декомпозиции продолжается до достижения нужного уровня подробности.

Детализация процесса «Изготовление деталей».

1. Откройте файл **Lab2.bp1**, сохраненный на предыдущем уроке.
2. Проведите детализацию процесса **1.2. ИЗГОТОВЛЕНИЕ ДЕТАЛЕЙ** с помощью диаграммы нижнего уровня. Данные представлены в таблице 3.1:

Таблица 3.1. Детализирование процесса «Изготовление деталей»

Процесс	Вход	Выход
1.2.1 – Переработка заготовки в деталь	Заготовки	Готовые детали
1.2.2 – Проверка качества деталей	Готовые детали	Готовые детали, брак
Управляющие стрелки и стрелки механизмов, указанные на диаграмме верхнего уровня должны быть и в диаграмме детализации.		

3. Выберите инструмент  и щелкните по блоку ИЗГОТОВЛЕНИЕ ДЕТАЛЕЙ;
4. В диалоговом окне введите число, на которое будет произведена декомпозиция - 2;
5. Укажите тип диаграммы **IDEF0** (Рисунок 1.) и нажмите ОК.

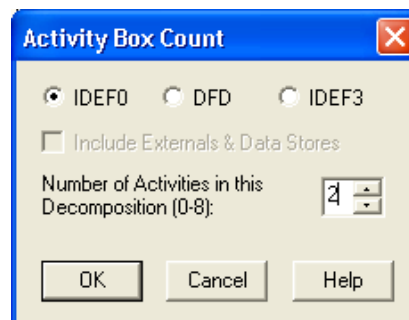


Рисунок 1. Диалоговое окно декомпозиции блока

Вы получите диаграмму декомпозиции уровня **A2** (Рисунок 2.).

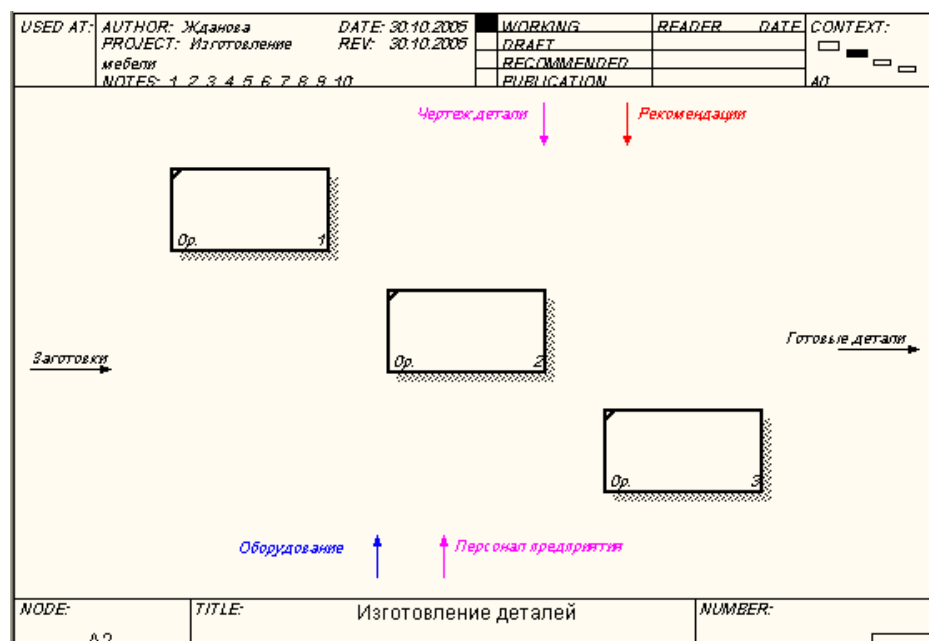


Рисунок 2. Декомпозиция уровня A2

6. Укажите названия процессов;

7. Соедините дугами обозначенные процессы, используя данные из таблицы 3.1;
8. Проверьте себя (**Рисунок 3.**).

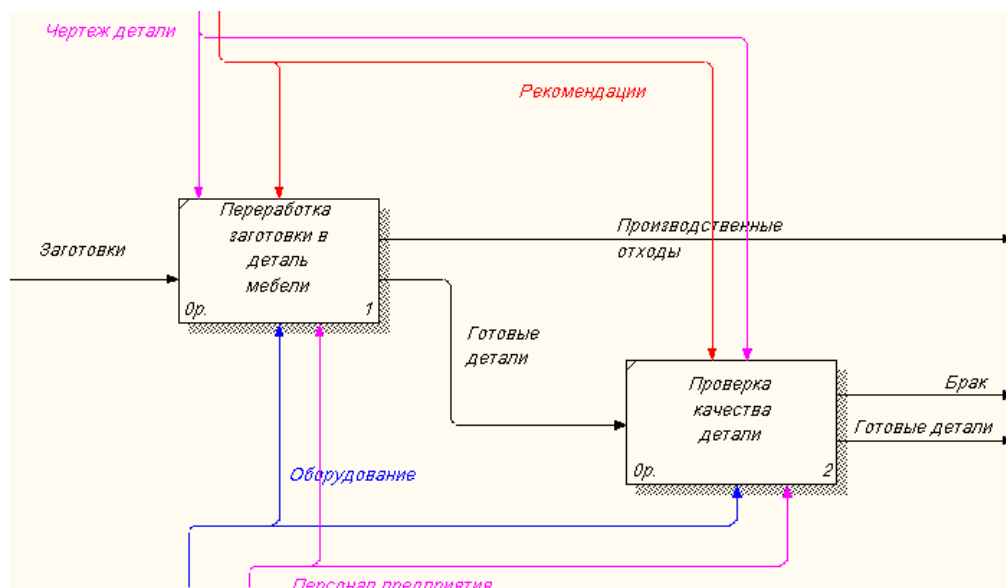


Рисунок 3. Детализация процесса ИЗГОТОВЛЕНИЕ ДЕТАЛЕЙ

Детализация процесса «Контроль качества».

1. Самостоятельно выполните детализацию процесса КОНТРОЛЬ КАЧЕСТВА.

После выполнения работы у вас должна получиться следующая диаграмма

(**Рисунок 4.**):

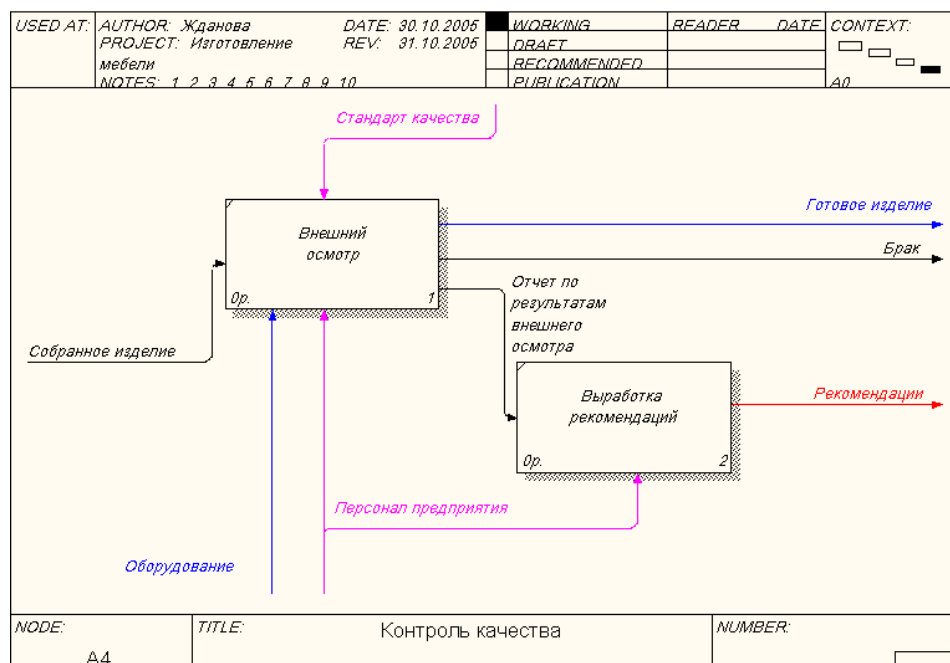


Рисунок 4. Детализация процесса КОНТРОЛЬ КАЧЕСТВА

Описание свойств модели.

IDEF0-модель предполагает наличие четко сформулированной цели, единственного субъекта моделирования и одной точки зрения. Для внесения области, цели и точки зрения в модели IDEF0 в BPwin следует:

1. Выбрать пункт меню **Model - Model Properties**, вызывающий диалог **Model Properties** (Рисунок 5.);

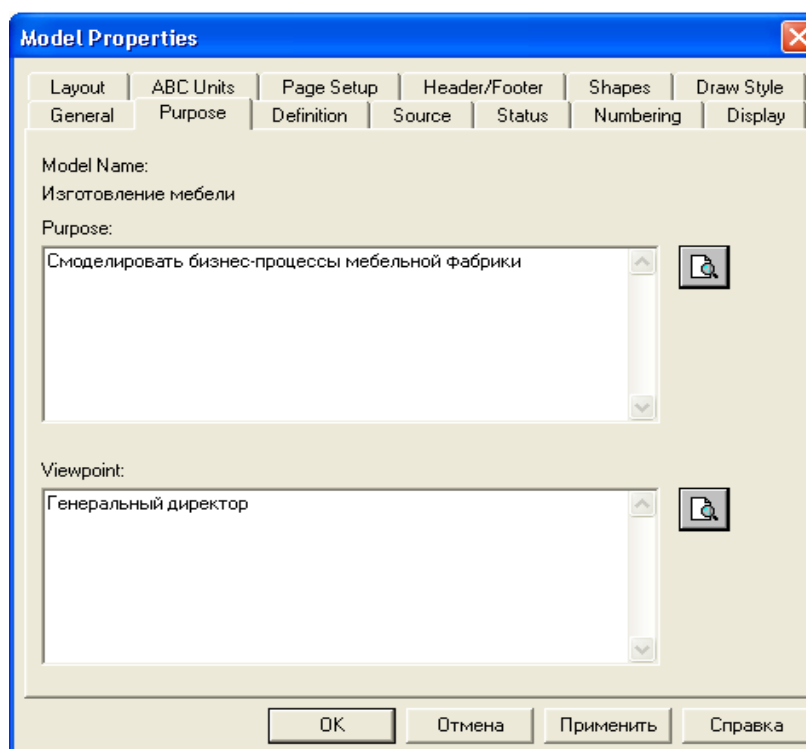


Рисунок 5. *Диалог задания свойств модели*

2. Во вкладку **Purpose** внести цель и точку зрения, а во вкладку **Definition** – определение модели;

Цель и точку зрения принято выносить на контекстную диаграмму **A-0** в виде текстового блока. После описания они появятся на контекстной диаграмме в виде текстового блока. Описание производится на уровне контекстной диаграммы.

Для описания цели и точки зрения следует:

3. Перейти на уровень диаграммы **A-0**;
4. Выбрать кнопку текста **T** на палитре инструментов;
5. Щелкнуть мышью в позиции предполагаемого ввода текста;
6. В диалоговом окне набрать нужный текст и установить опцию значимости (обычный текст, цель или точка зрения) (**Рисунок 6.**).

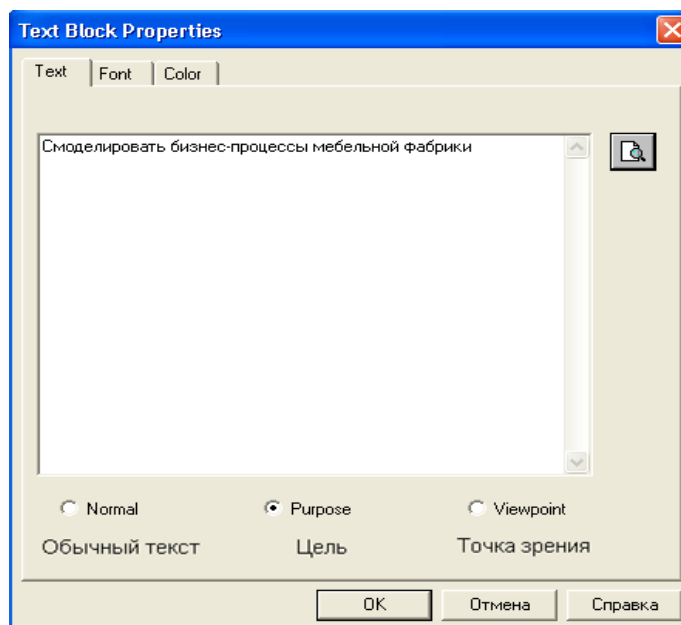


Рисунок 6. Установка опции *Text*

7. Во вкладке **Status** того же диалога опишите статус модели (черновой вариант, рабочий, окончательный и т.д.), время создания и последнего редактирования (отслеживается в дальнейшем автоматически по системной дате);
8. Во вкладке **Source** опишите источники информации для построения модели (например, «Опрос экспертов предметной области и анализ документации»);
9. Вкладка **General** служит для внесения имени проекта и модели, имени и инициалов автора и временных рамок модели.

Составление отчета.

Результат описания модели можно получить в отчете **Model Report**.

1. Диалоговое окно настройки отчета по модели вызовите из пункта меню **Tools – Reports - Model Report**.
2. Выберите необходимые поля, при этом автоматически отображается очередность вывода информации в отчете (рис. 3.9.);

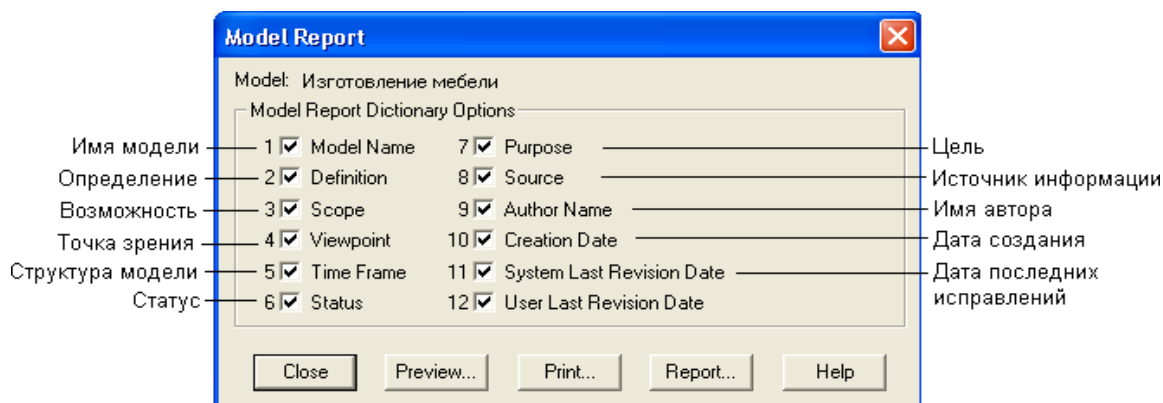


Рисунок 7. Диалоговое окно выбора информации для отчета

3. Нажмите на кнопку **Preview**, чтобы просмотреть отчет (**Рисунок 8.**).

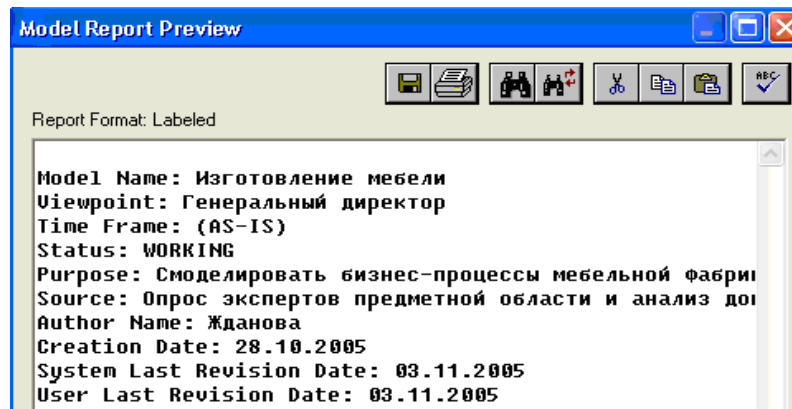


Рисунок 8. Отчет о модели

Сохранение полученной диаграммы.

1. В меню **File** выберите **Save As**.
2. Укажите путь к своей папке и имя файла **Lab3.bp1**.
3. Нажмите **OK**.

Контрольные вопросы

1. Как нумеруются модели в иерархии **IDEF0**?
2. Дайте понятие определению **Дерево узлов**.
3. Какой процесс в разработке модели называют функциональной декомпозицией?
4. Как можно вынести цель и точку зрения проекта на диаграмму?
5. Для чего необходимо составление отчета?

Информационные источники

Основные источники:

1. Долженко А.И. Технологии командной разработки программного обеспечения информационных систем [Электронный ресурс] / А.И. Долженко. — Электрон. текстовые данные. — М. : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. — 300 с. — 2227-8397. — Режим доступа: <http://www.iprbookshop.ru/39569.html>
2. Куликов И.М. Технологии разработки программного обеспечения для математического моделирования физических процессов. Часть 1. Использование суперкомпьютеров, оснащенных графическими ускорителями [Электронный ресурс] : учебное пособие / И.М. Куликов. — Электрон. текстовые данные. — Новосибирск: Новосибирский государственный технический университет, 2017. — 40 с. — 978-5-7782-2195-6. — Режим доступа: <http://www.iprbookshop.ru/45044.html>
3. Вичугова А.А. Инструментальные средства разработки компьютерных систем и комплексов [Электронный ресурс] : учебное пособие для СПО / А.А. Вичугова. — Электрон. текстовые данные. — Саратов: Профобразование, 2017. — 135 с. — 978-5-4488-0015-3. — Режим доступа: <http://www.iprbookshop.ru/66387.html>

Дополнительные источники:

1. Методические указания и задание на контрольную работу по дисциплине Технологии разработки программных комплексов и CASE-средства [Электронный ресурс] / . — Электрон. текстовые данные. — М. : Московский технический университет связи и информатики, 2016. — 37 с. — 2227-8397. — Режим доступа: <http://www.iprbookshop.ru/63365.html>
2. Шандриков А.С. Стандартизация и сертификация программного обеспечения [Электронный ресурс] : учебное пособие / А.С. Шандриков. — Электрон. текстовые данные. — Минск: Республиканский институт профессионального образования (РИПО), 2018. — 304 с. — 978-985-503-401-9. — Режим доступа: <http://www.iprbookshop.ru/67740.html>