Computer Science and Systems Analysis

Computer Science and Systems Analysis

Technical Reports

Miami University *Year* 1995

Application of Object-oriented Programming in Simulation: A Simulation of Case Study Using Microsoft Visual C++

Hong Chen
Miami University, commons-admin@lib.muohio.edu

MIAMI UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE & SYSTEMS ANALYSIS

TECHNICAL REPORT: MU-SEAS-CSA-1995-000

Application of Object-oriented Programming in Simulation: A Simulation of Case Study Using Microsoft Visual C++

Hong "Sherri" Chen

School of Engineering & Applied Science | Oxford, Ohio 45056 | 513-529-5928

Application of Object-Oriented Programming in Simulation

A simulation case study using Microsoft Visual C++

By

Hong 'Sherri' Chen

A THESIS
submitted to Miami University
as partial fulfillment of Master's Degree

Advisor:	Thesis Committee:
Dr. Mufit Ozden	Dr. Donald Byrkett
	Dr. James D. Kiper

August 1,1995

Application of Object-Oriented Programming in Simulation

A simulation case study using Microsoft Visual C++

A THESIS
submitted to Miami University
as partial fulfillment of Master's Degree

By

Hong 'Sherri' Chen
Miami University
Oxford, Ohio

August 1,1995

Advisor Murat Ozden
Reader Donald L Burkett
Reader James D. Kiper

ACKNOWLEDGMENTS

Many thanks to my advisor, Dr. Mufit Ozden, for his limitless guidance, support, and patience. His generosity of time and willingness to teach by example have made the completion of this project feasible. His kindness and sense of humor have made it enjoyable.

I would like to express my appreciation to my thesis committee, Dr. Donald Byrkett and Dr. Jim Kiper, for their time and helpful advice.

Special thanks to my parents, Jifeng and Yongkang Chen, for the loving support they have given me through all of my endeavors. Last, but not least, special thanks to my husband, Jie Fred Shen, who has been a constant source of encouragement and emotional support throughout my graduate career.

Abstract

In this thesis, a prototype simulation environment is introduced. Simulation has always been important for systems analysis. The original idea of this thesis stems from the fact that more flexible simulation programming tools are required by the modern analysis.

Six simulation classes are implemented in the thesis to support simple simulation cases and Microsoft Visual C++ window classes are used to build a user friendly interface. A simulation output class is also implemented to conduct simple simulation output analysis.

Based on this work, more classes and features can be added to make the simulation environment more powerful, so that the simulation environment can support different simulation situations.

Table of Content

List of Figures

List of Tables

1. Introduction

1.1 Purpose of Study

The purpose of this thesis is to develop a prototype simulation environment. By using object-oriented design and Microsoft Visual C++ visual classes, the simulation environment has a friendly user interface and a great flexibility that can be modified to support more real-world simulation cases.

1.2 Simulation Case Description

We use a simple manufacturing problem to demonstrate the main features of our simulation program. In this problem, a factory has a certain number of machines that break down and require frequent adjustment and repair; a certain number of adjusters are employed to keep the machines running. A service manager coordinates the activities of the adjusters and the broken down machines. If there are machines waiting to be repaired, the service manager maintains a queue of inoperative machines and assigns the machine in the front of its queue to the next adjuster available. Likewise, when some adjusters are not busy, the service manager maintains a queue of idle adjusters and assigns the adjuster in the front of the queue to the next machine that breaks down. The factory management wishes to get as much use as possible out of its machines and its adjusters. It is therefore interested in maximizing the machine utilization and in the adjuster utilization.

Figure 1 illustrates the system that being simulated.

This is a single-queue with multiple server system. In the system, if there exists a queue, it will be either a queue of broken machines waiting for adjusters, or a queue of idle adjusters waiting for jobs. The system clock is advanced when events are occurring.

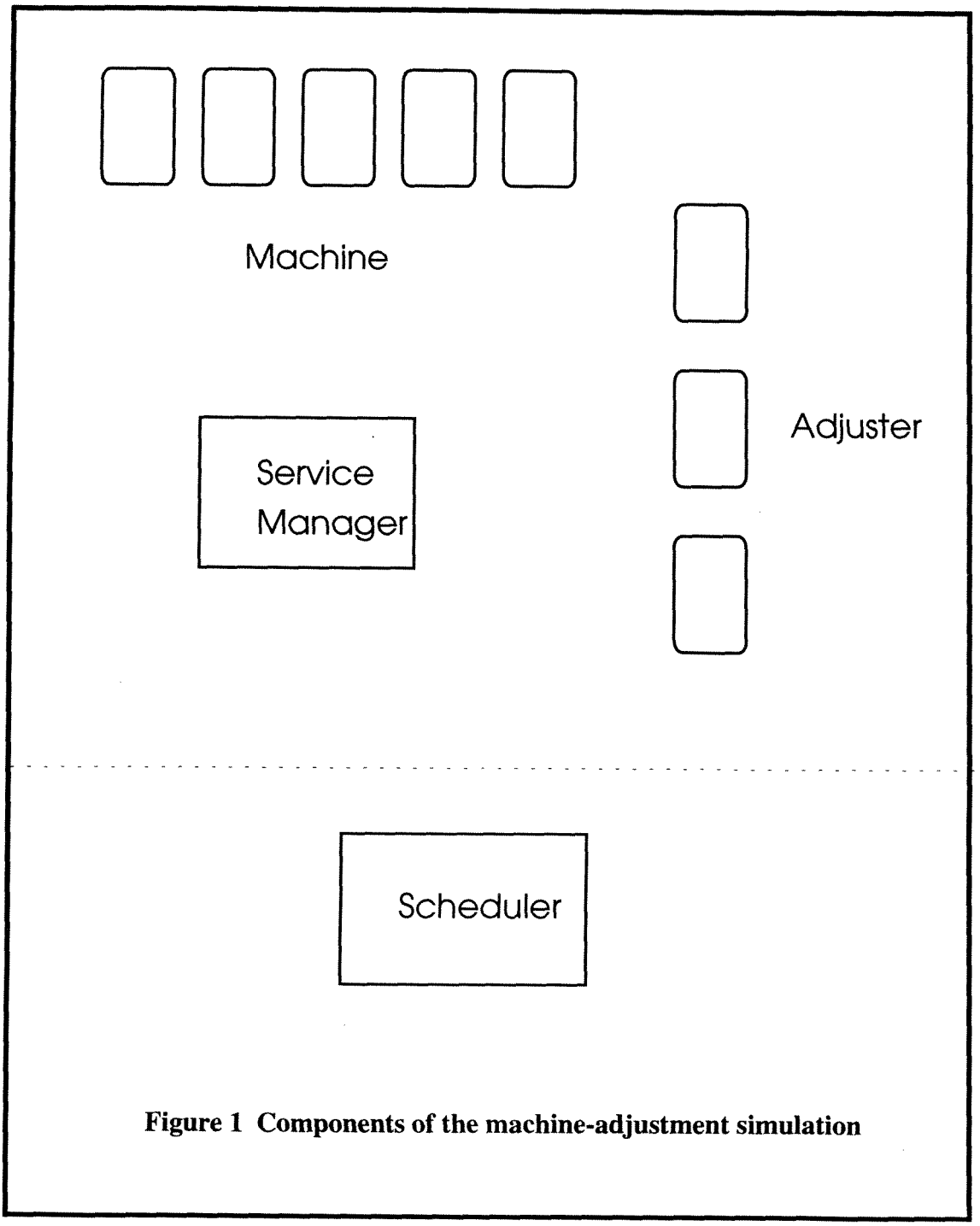


Figure 1 Components of the machine-adjustment simulation

2. Background & Literature Review

2.1 Simulation

Today, simulation analysis is a powerful problem-solving technique. Its origin lies in statistical sampling theory and analysis of complex probabilistic physical systems. Simulation analyses historically have been applied in various aspects of the production sector, such as corporate planning, inventory control systems, production lines, materials handing systems, and job shop scheduling.

2.1.1 History of Simulation

During World War II , a tremendous technological revolution took place. Companies provided a more diversified range of products and services. Manufacturing, in turn, became increasingly complex. Middle management used mathematical techniques perfected during the War to solve problems of project planning, inventory control and capital investment. Operations research techniques experienced outstanding growth and interest after the War.

With continuing opportunities for upper management to expand and modify old production processes, decisions about when and how those changes would truly be beneficial became more difficult for management to make. Operational research methods used by middle management were not powerful enough for complex processes. In 1955, the digital computer came to the aid of management. A new tool was introduced to middle management. The new tool was simulation. (Emshoff & Sisson, 1970)

2.1.2 Simulation and Modeling

One useful working definition of simulation is:

"The process of designing a mathematical or logical model of a real system and then conducting computer-based experiments with the model to describe, explain, and predict the behavior of the real system." (Naylor et al, 1966)

This definition gives us a clear concept of what simulation is and how simulation is conducted. First of all, an explicit model, mathematical or logical, is required for every simulation process. Developing a good model is fundamental to the success of any simulation analysis. Then the model should be able to be expressed in a computer language. Finally, the ultimate purpose of simulation analysis is to predict system behavior, thereby aiding decision makers in the evaluation of alternatives.

Simulation models generally are a kind of symbolic model. Symbolic models are those models that have no physical or analog relationship to the real system, but a logical one. Symbolic models can be classified as different categories, as:

- prescriptive or descriptive

  A prescriptive simulation model is used to formulate and optimize a given problem, thus providing the one best solution. A descriptive model, on the other hand, is used to describe the system behavior and leave the optimization process totally in the hands of the analyst.

- discrete or continuous

  The discrete or continuous classification refers to how the state of the system changes. Continuous status variables may take on the value of any real number,

while discrete status variables may assume only a limited, specified number of values.

- probabilistic or deterministic

  We distinguished between probabilistic and deterministic models also by focusing on the model variables. If any random variables are present, we classify the model as a probabilistic model.

- static or dynamic

  Models are classified as either static or dynamic depending on whether or not the model variables change over time.

- open loop or closed loop

  The notion of an open loop or a closed loop model is defined by the structure of the model, whether the model gets feedback from outside of the model.

Here, we are interested in descriptive, discrete, probabilistic, static and closed loop models.

Generally, in simulation models, we use terms such like entity, resource, queue, etc., to describe the simulated system. Entities represent a wide variety of animate and inanimate objects: for example, patients in a clinic, customers in a bank, cars in a toll booth queue. Resources, on the other hand, represent those objects that provide service to entities, like doctors in a clinic, bankers in a bank and tolls in a toll booth.

2.1.3 Simulation by Computer

Simulation is a computer based technique. Typically, a computer simulation language should have the following components:

- Time flow mechanisms

  There are two kinds of time flow mechanisms, fixed time increment and variable time increment. Fixed time increment advance is more efficient for systems where events occur at regular time intervals, whereas variable time increment is more efficient for systems where events occur at variable time intervals. (Blake,et al., 1964)

- An event calendar or event list

  An event list is a list of future events which is sorted from top to bottom by ascending time and ascending priority.

- Random number generators

  For any probabilistic simulation, the ability to extract random samples from specified probability distributions is an important feature.

- Statistical gathering capability

  Simulation can not be conducted without statistical gathering. Otherwise, simulation become useless and meaningless. Only by collecting data on system performance measures can people evaluate the system being simulated.

- Experiment design

Since simulation analysis is a descriptive, rather than prescriptive, technique, its successful application depends heavily on model experimentation. Effective,

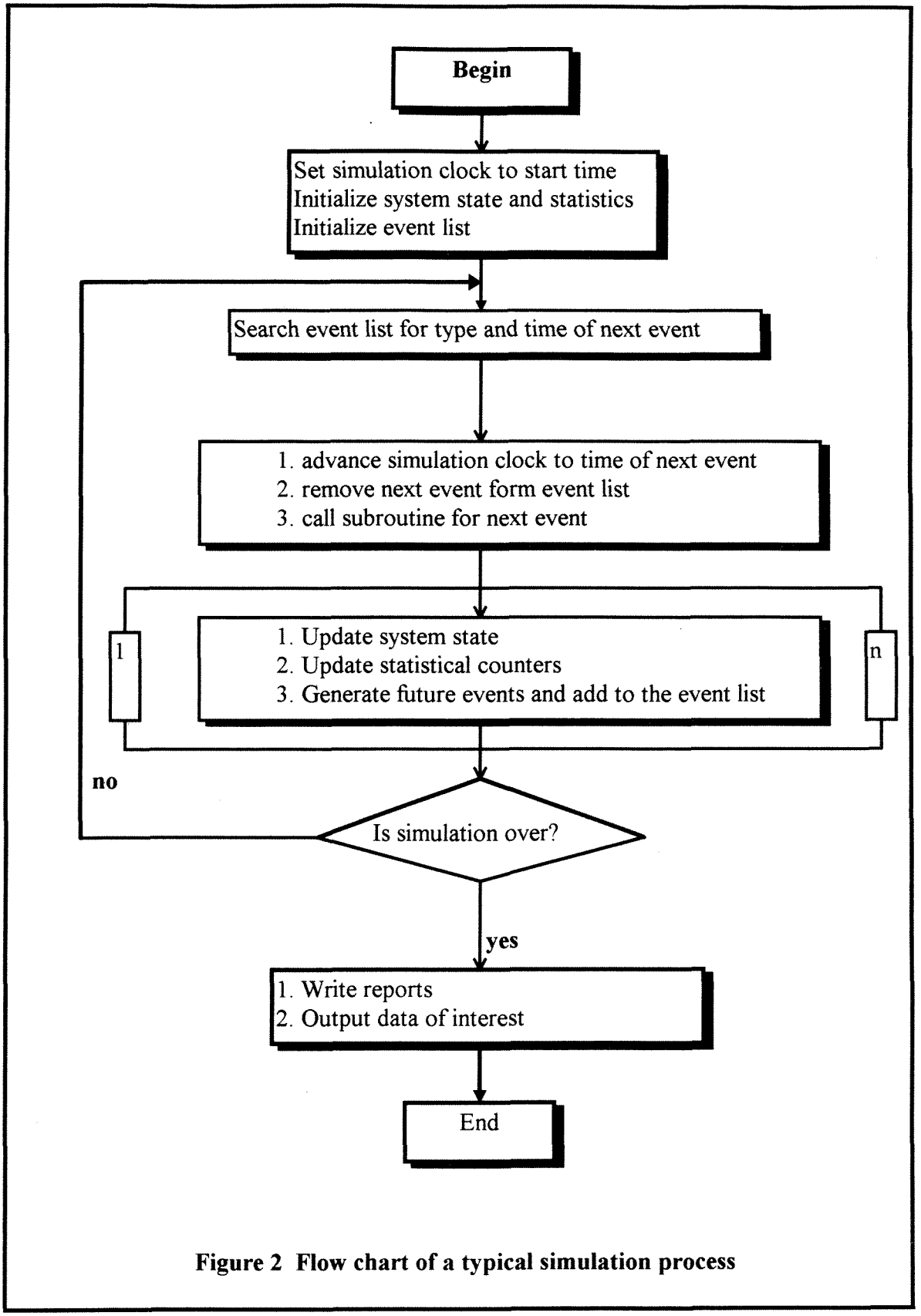


Figure 2 Flow chart of a typical simulation process

efficient experimental designs do much to enhance the ultimate quality of solutions obtained from a simulation model.

♦ Graphical animation and dynamic output

The ability to use simulation languages on microcomputers permits the graphics capability of these machines to illustrate, in some fashion, the running of the simulation model or its output.

Figure 2 presents a generic flow chart of a simulation conducted by computer. As we can see simulation begins at a starting time, usually 0, by initializing the clock, initializing statistics, and initializing the event list. The clock removes the first item on the event list, adjusts the clock time, updates the event, and then repeats. During the updating of the system, new events may be registered. The simulation will continue indefinitely until the event list becomes empty or some prespecified condition stops the simulation.

2.1.4 Computer Simulation Languages

In the beginning simulation was conducted by using common programming languages - primarily FORTRAN. However, the elements and structures of these common languages were not closely aligned with the systems to be simulated. In particular, the model development phase of simulation analysis was often protracted because of the long, complex computer code necessary to represent the model. To solve these problems, specialized computing languages were developed for simulation. As industry needs grew more complex, different simulation languages were developed to meet the needs of different types of models. Today, the most common simulation languages are GPSS, SIMSCRIPT, SLAM, and SIMAN.

Although these special purpose programming languages were an improvement for creating a simulation model, they were limited in the flexibility that they provided. Programmers had to drop down to some base language if they wanted to do something that was not provided in the simulation language. This made some simulation projects frustrating and tended to limit the variety of simulation projects that were undertaken.

Analysts are demanding better and easier simulation tools. Simulation languages must therefore continue to provide the basic simulation constructs that users have come to expect and also include new tools that provide the ability to conduct better, smarter, and more flexible simulations.

In summary, a desirable simulation language should have those features including: "automated model-code generation, effective interactive-debugging capability, facile means for changing the experimental design, ease of learning and ease of explanation to nontechnical individuals. " (Henriksen and Perry, 1983)

2.2 Object Oriented Programming And Visual C++

2.2.1 Object Oriented Programming and C++

In terms of its influence on the programming community, object-oriented problem solving is predicted to be to the nineties what structured programming was to the seventies. It has its roots in simulation theory. It is consists of identifying objects and what is to be done with those objects as specific steps in a problem solution. The programming metaphor is based on personifying the physical or conceptual objects from

some real-world domain objects in the program domain; e.g., objects can be clients in a business, foods in a produce store, or transportation ports in a factory.

Object Oriented Programming is fundamentally different from traditional procedural or algorithmic approaches. Object-oriented programming describes a system in terms of entities involved. Traditional programming approaches, on the other hand, describe systems in terms of their functionality.

Computer languages are object-oriented if they support the four specific object properties called *abstraction, encapsulation, inheritance*, and *polymorphism*. The central feature of an object-oriented approach to problem solving is the concept of an object. Objects are the unit of encapsulation and consist of the private data, shared data, and messages as given in the class description protocol for that object. Objects are instances of classes. Abstractions are represented as kinds of objects through a hierarchy of classes. Object-oriented programs consist of object-message sequences, and messages are sent to objects.

C++ is one of the several languages that support object-oriented problem solving techniques. It was developed by AT&T Bell Laboratories in the early 1980s. C++ was originally developed for simulation. It is based on C and is a superset of C. It adds polymorphism and inheritance to support object-oriented programming, while maintaining the speed and efficiency of C.

Simulation is one of the most natural applications of OOP. OOP languages offer a natural representation for program coding by allowing the creation of objects. Objects encapsulate a name, structure, and behavior into one unit. Programming flow is formed

by communication among objects. The object-oriented language C++ can be an effective simulation language environment.

2.2.2 Microsoft Visual C++

Microsoft Visual C++ is one of the most recent software environments that support C++. It contains the most powerful Windows-based application framework to date.

Each Visual C++ window application has at least the following four derived classes:

. The document class

. The view class

. The main frame window class

. The application class

The primary program tasks are divided among these four main classes.

The document class is derived from the Microsoft Foundation Class(MFC) class CDocument. It is responsible for storing the program data and for reading and writing this data to disk files.

The View class is derived from MFC class CView. It is responsible for displaying the program data (on the screen, printer, or other device) and for processing input from the user. This class manages the view window, which is used for displaying program data on the screen.

The Mainframe window class is derived from CMainframe. It manages the main program window, which serves to display a title bar, a menu bar, maximize and minimize buttons, borders, a system menu, and sometimes other user interface elements such as a

tool bar or a status bar. The view window is a child of the main program window, which means -- among other things -- that it is always displayed on top of and within the boundaries of the main program window.

Finally, the application class is derived from the MFC class CWinApp. The application class manages the program as a whole; that is, it performs general tasks that do not fall within the province of any of the other three classes, such as initializing the program and performing the final program cleanup. Every MFC Windows program must create exactly one instance of a class derived from CWinApp. (M. J. Young, 1993)

The four main classes communicate with each other and exchange data by calling each other's public member functions and by sending messages. Public functions like AfxGetApp() returns a pointer points to the CWinApp class. Table 1 summarizes the features of the four main classes for the simulation program in this thesis.

Table 1 Window application classes for the simulation environment

Class	Derived Class Name	Base Class Name	File Header	File Implementation
Document	CThesisdoc	CDocument	thesidoc.h	thesidoc.cpp
View	CTheview	CView	theview.h	theview.cpp
Mainframe	CMainframe	CFrameWnd	mainfrm.h	mainfrm.cpp
Application	CTheApp	CWinApp	theapp.h	theapp.cpp

When an application is activated, the following five events will take place:

1. the CWinApp class constructor is called
2. the program entry function, WinMain, receives control
3. WinMain calls the program's InitInstance function
4. WinMain enters a loop for processing messages
5. WinMain exits and the program terminates.

3. Simulation Study using Visual C++

The event list, random number generator, and output analysis are all important components of a simulation language.

3.1 Event List -- Calendar queue

The event list is a list of future events. A calendar queue is used in the program to implement the event list. A calendar queue is a data structure that has been proposed for representing a simulation event list. It is the most widely used in general purpose language for simulation purpose. Others are array, linked list, priority queue, etc. The calendar consists of an array with one element for each calendar page; each page corresponds to a clock tick. Figure 3 illustrate the structure of a calendar queue.

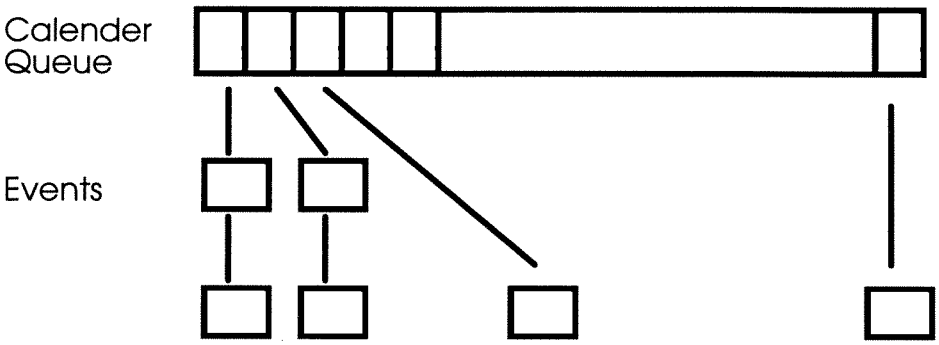


Figure 3 Structure of a calendar

There are many ways to implement a calendar queue. Each has its advantages and disadvantages.

The easiest way to implement a calendar queue is to make the number of pages in our calendar one larger than the largest time interval by which an event is scheduled ahead. This corresponds to the situation for an actual calendar in which events are always scheduled less than a calendar period ahead: each page will contain events only for the current time and not those that are to be carried out at later times in the calendar. However, this implementation wastes lots of computer memory. (McGraw,1991)

Another way to implement the calendar is to have a calendar and an event list. The event list can be a linked list, which will allocate space dynamically at run time. The length of the calendar is different for different cases. Events that are within the time period of the calendar are kept in the calendar, otherwise, they are kept in the event list. At the end of the calendar period, when the clock exceeds the length of the calendar, events stored in the event list are pulled out to fill in and reset the calendar. This method saves computer memory, but, it is not efficient for some cases, in which there are infrequent events.

The most desirable way to implement a calendar is to implement the calendar as a dynamic one. At run time, change the length of the calendar according to the density of the event on each page. But a lot of work need to be done for this implementation.

We used the second technique in our calendar queue implementation.

3.2 Random Number Generator

The definition of randomness is that a sequence of integers in a particular range is said to be random if every possible value in that range is equally likely to occur anywhere within the sequence. (S. V. Hoover, 1989)

The need to generate random numbers is important in many computing applications, especially for simulation applications. Badly generated random numbers can affect the result of the simulation. A great number of investigators have been carried out work to seek good random number generators.

In our implementation, we employed the most widely used linear congruential method, for generating random numbers, which is:

$$X_{i+1} = (a X_i + c) \% m$$

where X_i is the ith result of this formula, a, c and m are constants that needed to be predefined. So, to start the sequence of generating random number by using the above formula, we need to give X_0 a value as an input. This is called a seed. The random number generated from this formula will be:

$$Random(i) = X_i / m$$

The values for a, c, and m need to be selected carefully. During the last several decades, researchers have found some good values to fill in the formula, such as 2^{31} -1 for the modulus m, 950706376, 1226847159 and 16807 for the multiplier a, and 0 for c.

Choosing the right method and good values for the multiplier and the modulus is still not sufficient for a good random number generator. Different software has different internal length for number, Microsoft Visual C++ 2.0 uses 32-bit words and some other software uses 16-bits word. The multiplier suggested by researchers is the largest number that can be represented by a 32-bit word. Thus, it is possible that sometimes, X_i will become very large, and after multiplying by a, overflow might occur. This leads to bad random number generating. Various ways are suggested by researchers to take care of the

overflow. A good suggestion to test a random number implementation according to the above formula, where $c = 0$, $a = 16807$, is to see if X_{10001} is 1043618065 while initial seed (X_0) is set to 1. (S.K. Park, 1988)

3.3 Output Analysis

Discrete-event simulation models are different from most other types of models. Because a discrete-event simulation model brings together the confluence of many random variables, the output of the model is a random variable. Using the output of a discrete-event simulation model to answer question about the behavior and properties of the system it represents can be a difficult task. The output of a simulation model can easily be misinterpreted, resulting in false conclusions about the system it represents. Output analysis is critical to the success of simulation. (Gajanana Nadoli, 1991)

The system in our case is a non-terminating system. The factors that can affect the result of our simulation are:

1. Initial condition bias

   The data collected during the early part of the simulation may be biased by the initial state of the system. The behavior of the system during this early phase of the simulation may be misleading, or irrelevant to the questions we expect the simulation model to answer.

2. Covariance between samples

   Groups or sets of data gathered during the simulation are generally not independent of one another. If sample sets are not independent, our variance estimates will be biased.

3. Run length

Although the system itself may be nonterminating, the simulation of the system must eventually be terminated. If we terminate the system too early, we may not have a representative simulation. For very complex systems, however, it is often impractical to make extremely long runs of the simulation model.

The method of batch means is used in our program to eliminate or reduce these three problems. The method uses the mean of several independent batch runs as the result of the simulation. The size of each batch is decided after a preliminary run.

3.4 Simulation Classes

This project was based on an example that was used to introduce the concepts of object-oriented programming (Graham,1991). The original program implemented three classes, *manager*, *scheduler* and *active*, to perform a simple working shop simulation.

In our project, we implemented seven basic simulation classes. Figure 3 illustrate the relationships between these classes and Table 2 holds names and file names of these classes. As we can see, all classes have public member functions that can be called from outside of classes. These public member functions perform important roles in our implementation. We will describe each class below.

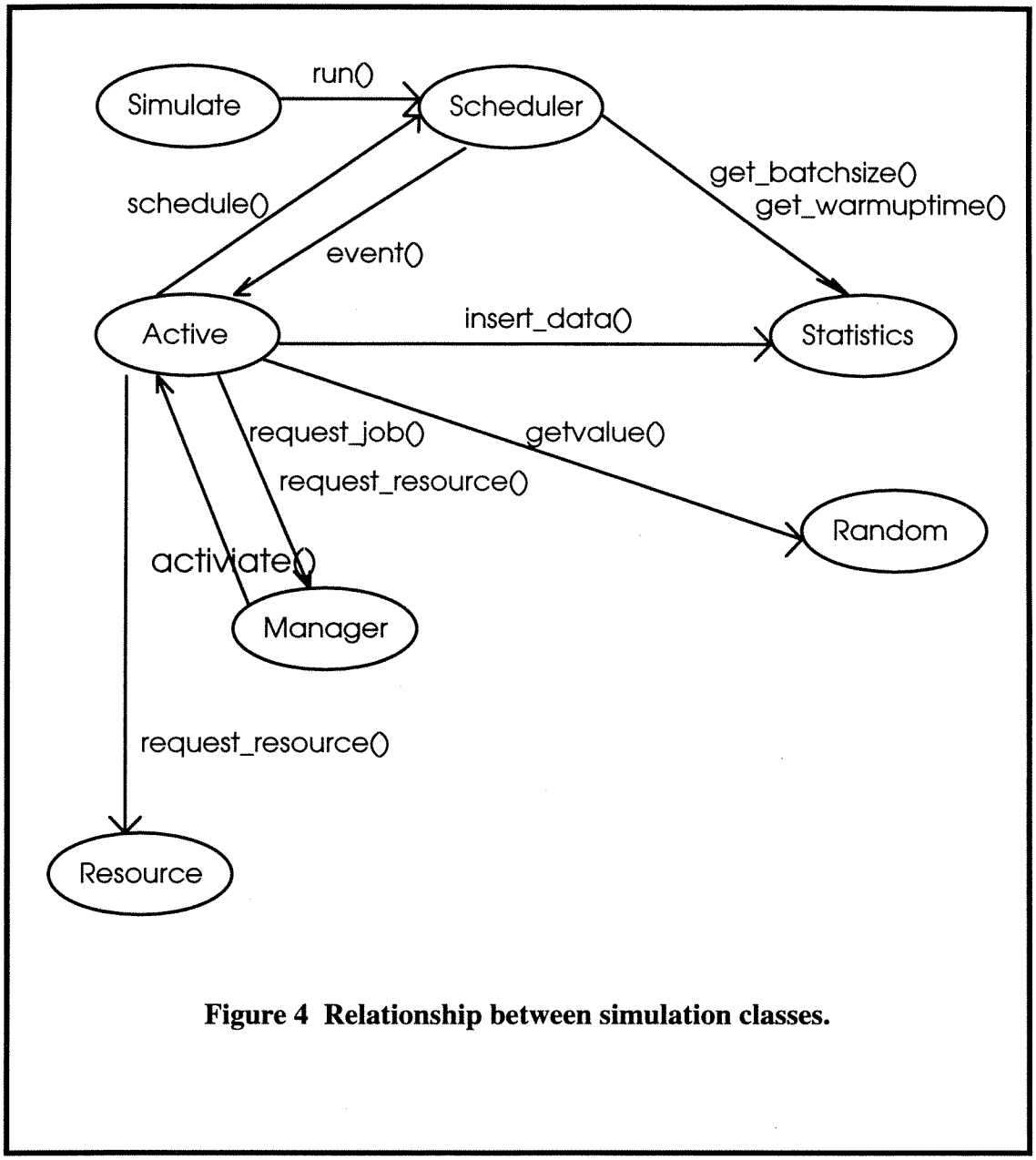


Figure 4 Relationship between simulation classes.

Table 2 Simulation Application Classes

Class Name	Header File	Implementation File
Active	Repair.h	Active.cpp
Manager	Repair.h	Manager.cpp
Scheduler	Repair.h	Scheduler.cpp
Simulate	Repair.h	Simulate.cpp
Statistic	Repair.h	Statistic.cpp
Random	Rand.h	Rand.cpp

3.4.1 Active Classes -- Active Objects

We define active objects as those that can originate spontaneous actions such as machine running or adjuster repairing. In our simulation, such "spontaneous" actions are activated or deactivated by events, such as breakdown of a machine or completion of a repair. Usually, active objects can be interpreted in simulation terms as entities or resources. Events are scheduled in advance and then triggered by an event message from the scheduler. Thus active objects are those that can receive event messages.

The defining property of active objects is that they can receive event messages via the function *event()*. Therefore, our *active* class defines *event()* as a pure virtual function that will be redefined in each derived class. Another common property of active objects is that they will be placed on linked lists by both the scheduler and the service manager.

Thus, class *active* declares a next pointer *next* and provides functions *set_next()* and *get_next()* for manipulating it:

```
class active{
    active* next;
public:
    void set_next(active* p) { next = p;}
    active* get_next() { return p;}
    virtual void event() = 0;
}
```

3.4.2 Manager Class

Usually, *manager* class represents the resource manager object of the real world. It also can be interpreted as a queue in terms of simulation. Class *manager* contains a linked list, which is a queue. For accessing the queue, class *manager* also defines private member functions as *insert_first(), insert(),* and *remove()*. Function *insert_first()* inserts an object as the first of a previously empty queue. Function *insert()* inserts an object to the end of the nonempty queue, and *remove()* removes the first object from the queue.

The manager will expect two kinds of messages, one for supplying resources to the resource manager and one for requesting resources from the resource manager. Thus, class *manager* has two member functions, *request_service()* and *request_work()*.

The class *manager* is defined as:

```
class manager{
enum who { CLIENT, SERVER,NONE};
```

```
who waiting;        // kind of object in queue
active* first;      // points to the first object in queue
active* last;       // points to the last object in queue
void insert_first( active* p);
void insert( active* p);
void remove();
public:
manager();
void request_service( active* p);        // service request
active* request_work( active* p);        // work request  }
```

3.4.3 Scheduler Class

The *scheduler* does not simulate any real-world entity but instead drives the simulation via *event()* messages. The *scheduler* also contains a calendar queue that will store event information. After the active objects have been initialized--their constructors are called--all further simulation activities are triggered, directly or indirectly, by *event()* messages. When an event is scheduled, the scheduler must store information about the event and they are sent the required *event()* message on the specified clock tick.

We implemented a certain size calendar queue to store those event that will be called in the near future and a overflow event list to store others beyond the calendar period. When a *schedule()* message come in, the scheduler will decide whether the event is in the time scope of the calendar, and insert the event into the proper place.

Besides the calendar queue and event list, *scheduler* also contains information about simulation time. It has member function as *time()* to enable other classes gain time information. Class *scheduler* is defined as:

```
class scheduler {
        int clock;              // simulation clock
        int calendar_size;   // size of calendar-queue array;
        active** calendar;   // pointer to calendar-queue array
        int index;              // for current time
public:

        scheduler ( int sz );
        int time() { return clock ; }          // return time
        void schedule ( active* p, int delay ); // schedule  event
        void run ( int ticks );                  // run simulation
}
```

3.4.4 Random Number Generator Classes

There are several classes defined in our simulation, so that different kinds of random variables can be generated, they are all derived from one virtual class *random*, The class *random* has a virtual member function *getvalue(),* that will return a random integer. The basic derived random number generator class is class *Uniform*, which will generate integer numbers that are uniformly distributed. We uses the formula which is introduced before in section 3.41 for this purpose. Other class like class *Exponential*, class *Normal*, class *Bernoulli*, etc., use the uniformly distributed number generated by class *Uniform*.

3.4.5 Statistic Class

The *statistics* class serves as an expert in the simulation. It collects data from machines during the preliminary run. Then, based on the collected data, it decides the batch size of the simulation automatically.

The *statistics* class has a linked list as a private member variable, which can dynamically change memory size at run time. The collected data is stored in this linked list, and is retrieved at the end of the preliminary run. Assessor functions as *insert()* and *get_next()* allows access to the list.

The major member function statistic class has is *get_batchsize()*, which returns the calculated batch size. Class *statistics* is defined as:

```
class statistics {
public:
        statistics();
        void insert ( stat_node* );
        void cal_var();              // calculate the variation of the results of
                                        preliminary run
        void cal_batchsize();        // calculate the batchsize to conduct complete
simulation
        void deletenode();   // free memory locations
private:
        long smallest;
        long largest;
        int batchsize;
        float variance;
```

```
    stat_node* first;
    stat_node* last;
}
```

3.4.6 Simulate Class

The *simulate* class connects all the above class together. It contains instances of all above classes as private member variables, and has member function *run()* that actually start our simulation.

3.4.7 Reusability of Simulation Class

For the classes that are previously introduced, most of them can be used in other simulation models. For example, for an inventory model, the *statistic* class, the *scheduler* class and the *random* class can be used without or with few modification.

3.5 Windows Classes for Visual Simulation

Visual classes provide a friendly user interface for our simulation, so that naive users can easily conduct their simulation.

As mentioned before, each Microsoft Visual C++ window program has four basic classes. They are: the view class, the document class, the mainframe class, and the one and only application class.

3.5.1 View Class

The THESIS view class in named CThesisView and is derived from the MFC class CView. The CThesisView header file is named thesisvw.h, and the implementation file is thesisvw.cpp. The view class is responsible for gathering simulation information from user, displaying simulation result on the screen.

3.5.2 Document Class

The THESIS document class is named CThesiDoc and is derived from the MFC class CDocument. The header file is thesidoc.h and the implementation file is thesidoc.cpp. The CThesiDoc class contains an pointer to a simulation object. It stores all the information needed for the simulation.

3.5.3 Mainframe Class and The Application Class

The Mainframe Class CMainframe is derived from the MFC class CMainframe. This class is repressible for loading the bitmap, toolbar, titlebar and the bitmapbutton at the beginning.

The Application Class CThesisApp is derived from the MFC class CApp. CThesisApp initializes the application and starts the program.

Appendix A illustrates various screens of the simulation environment.

The first screen (Appendix A-1) is the main screen of our simulation environment. It has a list of different simulation models and several buttons that lead to other screens. In the current project, only the second model on the list is implemented. Others can be added on later.

If the *Simulation Model* button is clicked, the simulation model screen will pop up (Appendix A-2). This screen provides detailed information about the model selected at the first screen. The model employed in this thesis was a descriptive, probabilistic, close loop, discrete, static and nonterminating model.

If the *Customer Setup* button is clicked, the customer information input dialog box will pop up (Appendix A-3). The data fields are the number of customer in the system, the distribution pattern of intervals between services, and the mean of this distribution.

If the *Server Setup* button is clicked, a similar dialog box will pop up asking the user to input server information (Appendix A-4).

The *Other Setup* button is an optional button. Normally, a user can use default setups to run the simulation. In case a user need to change internal simulation setups,

such as the seed for the random number generator, the size of the calendar queue, and so on, one may click the *Other Setup* button (Appendix A-5) and change the settings in the dialog box.

The *Output* button allows the user to choose the format of the simulation output. The button leads to a screen (Appendix A-6) that provides two choices to the user, one is graphic output (Appendix A-7), the other is numeric output (Appendix A - 8).

The *Run* button starts the computation process of a simulation case.

3.6 Special features of our simulation case

In our simulation, we have two classes that are derived from the visual class *active*, *machine* and *adjuster*, according to the active objects we have for our case. Figure 12 shows the class hierarchy for active objects.

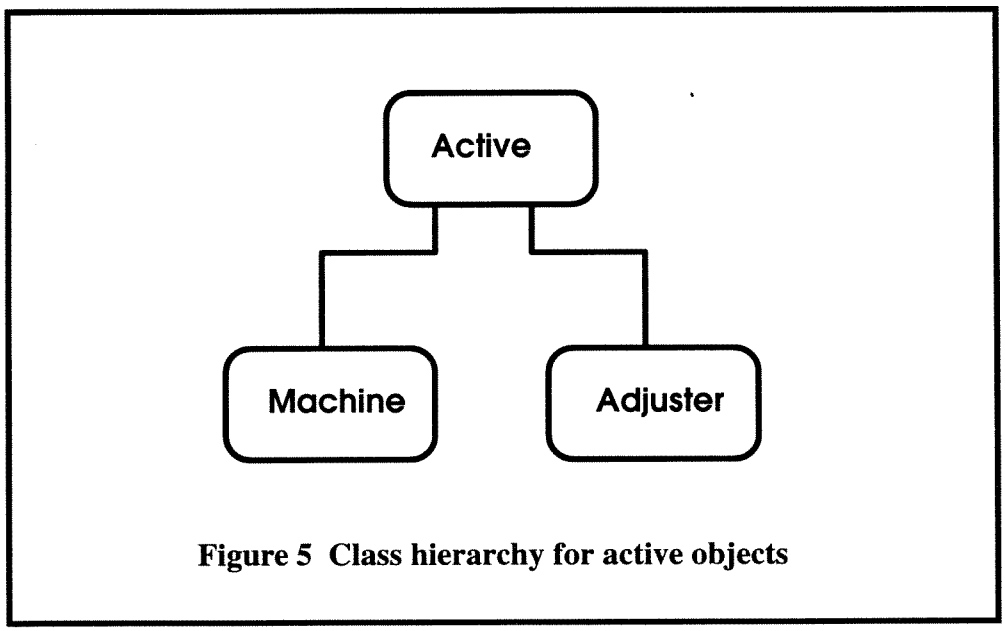


Figure 5 Class hierarchy for active objects

Both the *machine* class and the *adjuster* class inherit the member function *event(),* which will invoke an activity of machine and adjuster when the time comes. They also inherit those assessor function *set_next()* and *get_next()* that class *active* uses to provide access to the private data members.

There are also private member variables defined for class machine and adjuster for the purpose of statistic collection. Boolean variable *is_up* represents the state of a machine, that is whether a machine is running or broken down, and *is_busy* represents whether a adjuster is busy or not.

The *event()* function of machine represents the break down of a machine. In this function the machine status *is_down* is set to be TRUE, current machine's *up_time* is calculated and sent to *statistics* class, and the *require_resource()* message is sent to the *manager* class.

The *event()* function of adjuster represents repair of a machine. In this function, the machine status *is_up* is set to be TRUE, adjuster's working time is calculated and sent to the *statistics* class, and a *require_work()* message is sent to the *manager* class.

Both *machine* class and *adjuster* class have public functions that other classes can call. The function *adjusted()* of *machine* class which is usually called from *adjuster* class, sets the machine status to *is_up*, sends a *getvalue()* message to *random* class to get the next break down time, and sends a *schedule*() message to the *scheduler* class to insert the next breakdown event into the calendar. The *adjust*() function of adjuster which is usually called from the *manager* class, sets the adjuster status to *is_busy*, sends a

getvalue() to the *random* class to get the repair-time, and sends a *schedule*() message to the *scheduler* class to schedule the repair of a machine.

In our simulation, the *manager* class coordinate the repair of machines. Function *request_service(),* which is sent by a machine, looks at the queue. If the queue contains a list of waiting machine, it will put the machine in the end of the list. Otherwise, if the queue contains a list of adjuster, it will remove the first adjuster in the queue, set the adjuster's state to *is_busy*, and sends an *adjust*() message to the *adjuster* class. Function *request_work()* on the other hand, looks at the queue, if the queue contains a list of waiting machines, it will remove the first machine from the queue; otherwise, it will put the adjuster that sends the *request_work()* message into the end of the adjuster list.

4. Conclusions and Future Studies

4.1 Conclusions

Microsoft Visual C++ is a perfect software environment for simulation development. The reasons are:

1. It is very easy to build a friendly user interface for the project.

   Before they can conduct any simulation with a traditional simulation language, such as SIMAN, the users must achieve certain proficiencies in programming with that language. In contrast, the simulation environment built by using Microsoft Visual C++ is much more user-friendly. It does not require a user to have any knowledge of C++ programming language. The simulation environment provides a Windows interface that enables a user, even a naive user, to conduct simulation easily.

2. The run time is faster than a traditional simulation language, like SIMAN.

   For the same situation, this project spent only one fifth of time to run a simulation.

3. The simulation result is compatible to the result of SIMAN.

   The difference between the result of simulation from this project and SIMAN is little. In some cases, they are the same. Table 2 shows the comparison of results.

Table 3 Result comparison between simultions using SIMAN and Visual C++

		SIMAN	VISUAL C++
Number of Machine	**Number of Adjuster**	**Util of Adjuster**	**Util of Adjuster**
10	4	99.25	99.28
10	5	96.61	96.15
10	6	89.84	89.41
10	7	80.92	80.61
10	8	72.23	72.01
10	9	64.73	64.13
10	10	58.51	57.49

4. The project allows the addition of new features in the future. Based on these defined classes, other features and classes can be added to the project easily.

4.2 Future Study

The computer simulation in the analysis of complex systems will play an ever increasing role in decision making. Simulation analysis provides a means through which an analyst can use to assist in decision making of real world systems.

This thesis has given me the opportunity to use the new technology to implement new simulation tools.

Future enhancements to this simulation project can provide a more standard interface for the running of simulation experiments and defining classes that abstract all the necessary objects needed by any simulation. Hence, more simulation models can be implemented if an expert class will be added to the projects to conduct input data analysis.

5. References

1. Joseph A. Fisher, "Object Oriented Simulation Tools for Discrete-Continuous, Stochastic-Deterministic Simulation Models", University of Oregon State, 1993.

2. Stewart V.Hoover, Donald F. Perry, "Simulation, a problem-solving approach", 1989.

3. Henriksen, James,"The Integrated simulation environment", Operations Research, Vol.31, No.6,(1983).

4. Perry, R., "Simulation from the User's Point of View," unpublished paper, April 1966.

5. Michael J. Young, "Mastering Microsoft Visual C++ Programming", 1993.

6. David J., Kruglinski, "Inside Visual C++", 1993.

7. Stephen. K. Park, Keith W. Miller, " Random Number Generators: Good Ones Are Hard To Find", communications of ACM, 1988

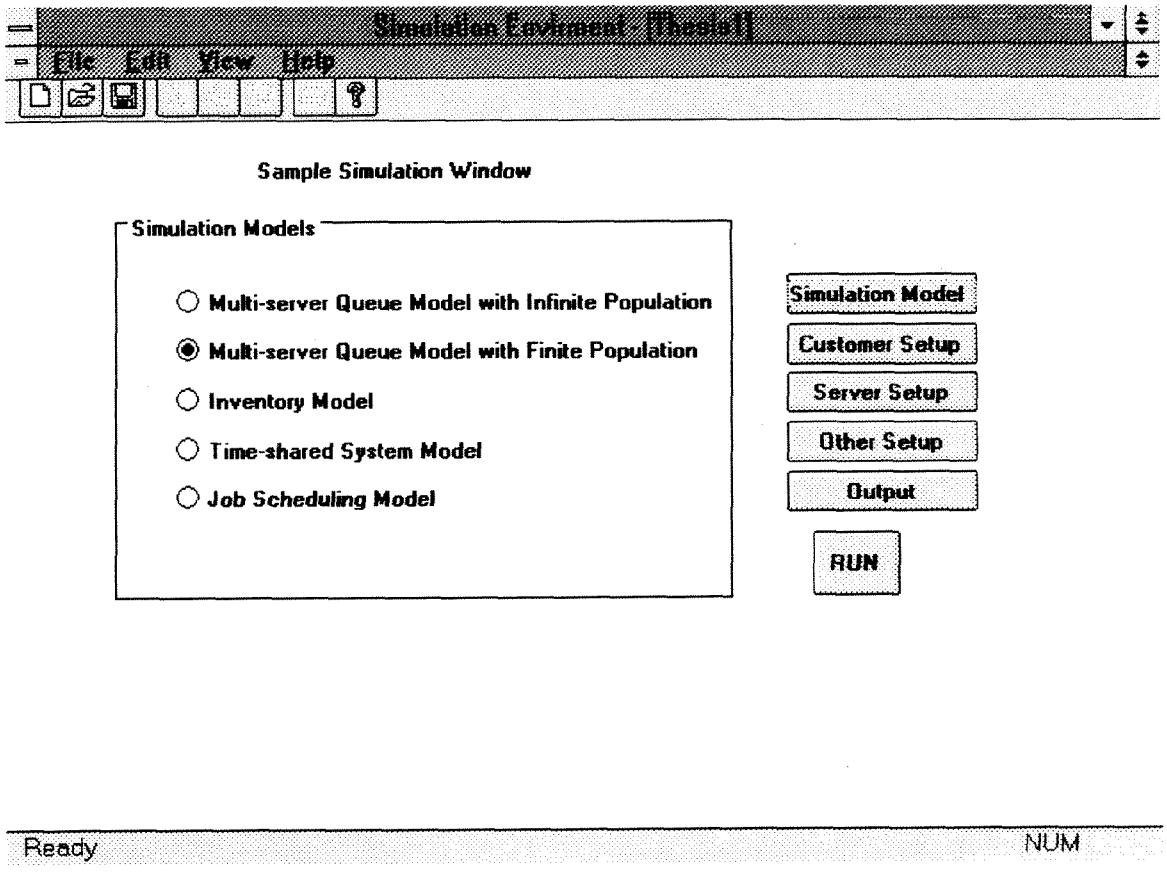
8. Gajanana Nadoli, John E. Biegil, "Use of Object-Oriented Intelligent Agents for Discrete Event Simulation Analysis", University of Centrol Florida, 1991

9. Neil Grahum, "Learning C++", 1991

Appendix A Screens of simulation environment

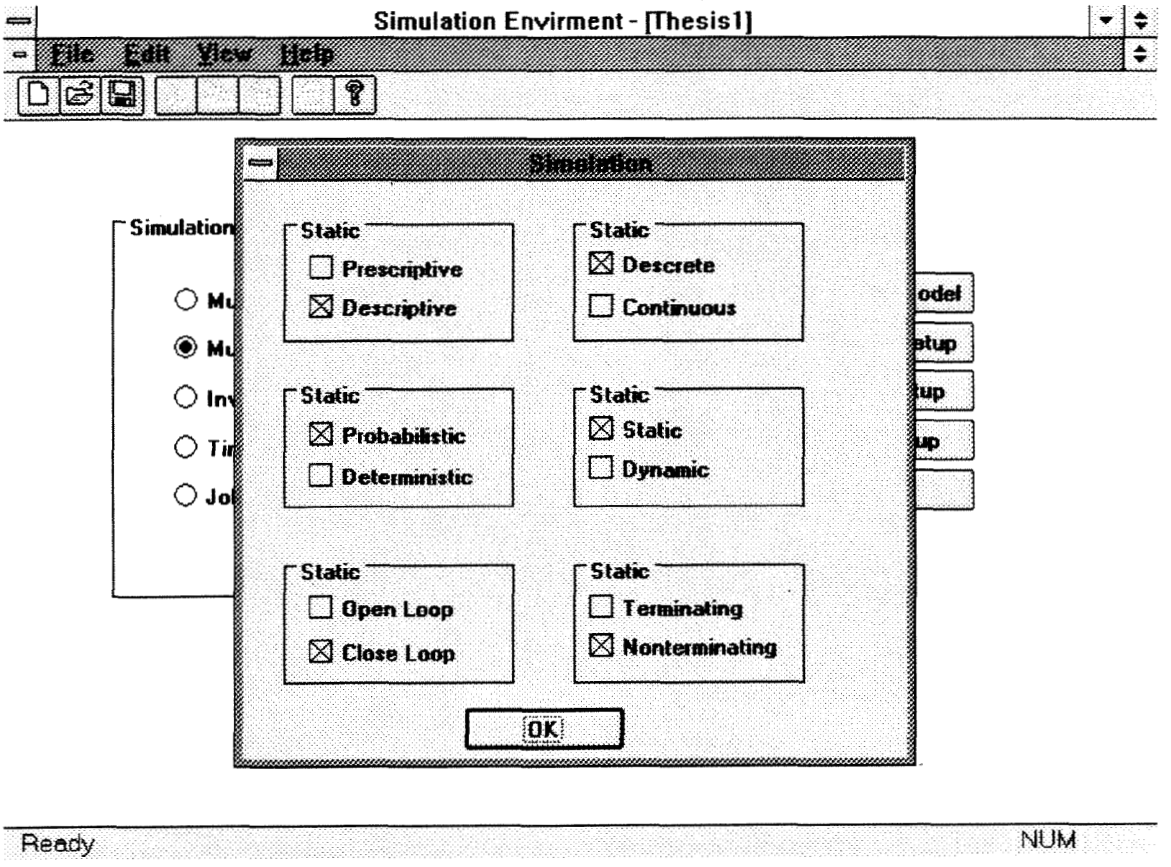
A-1. Main Screen of the simulation environment

This screen provides different simulation models for user to select. All the buttons on the right connect to dialog boxes for more detailed simulation information.



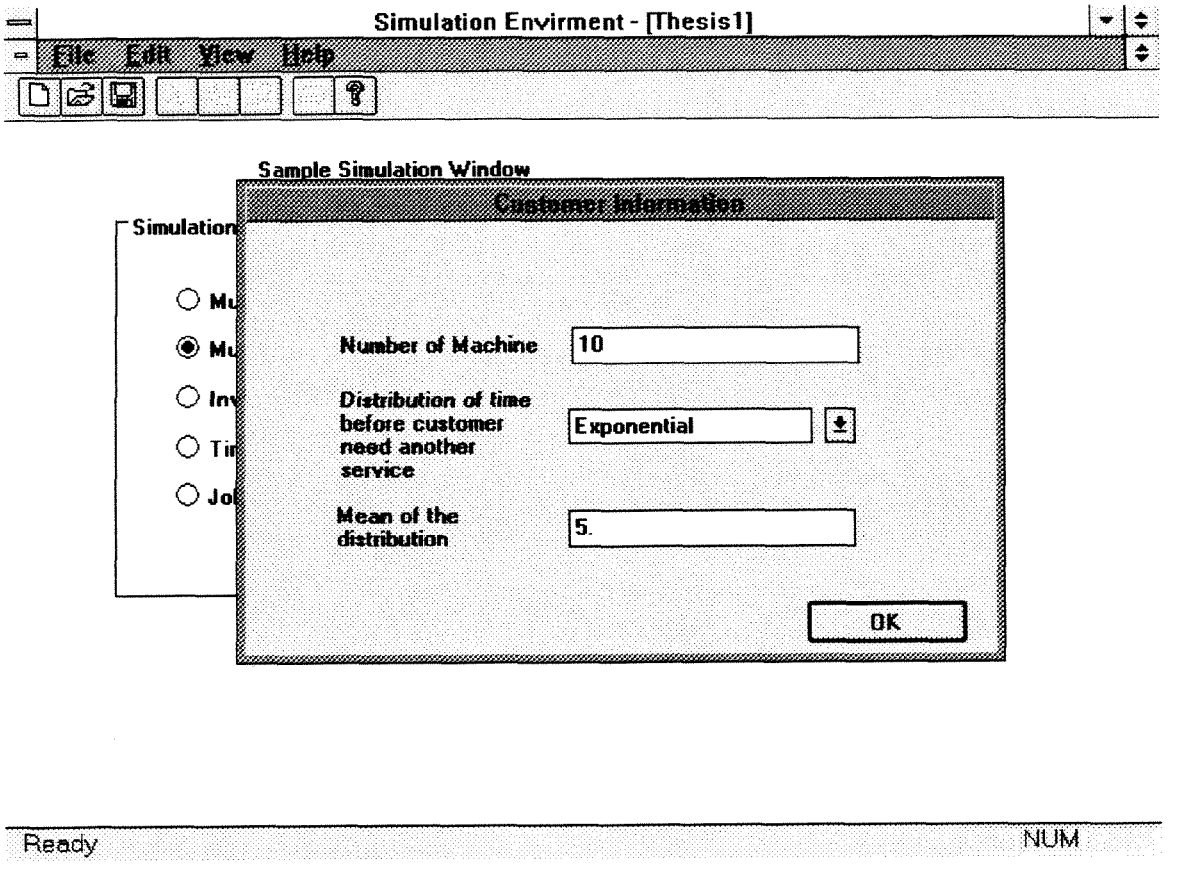
A-2. Simulation Model Screen

This screen is a read only screen. It gives user more information about the simulation model that being selected in the previous screen.



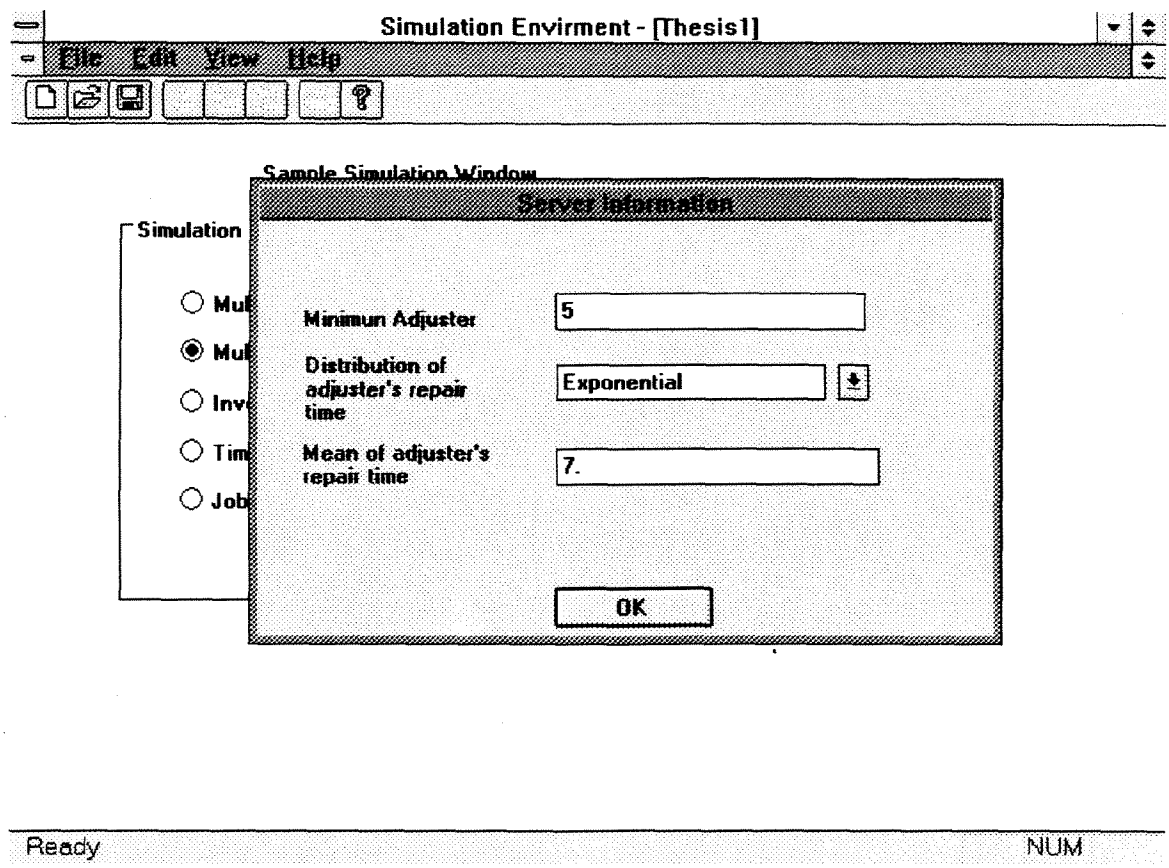
A-3. Customer Information Input Dialog Box

This screen has a widow edit box that allows a user to input customer simulation data. The combo box for input distribution has predefined different kinds of distribution that the program can provide.



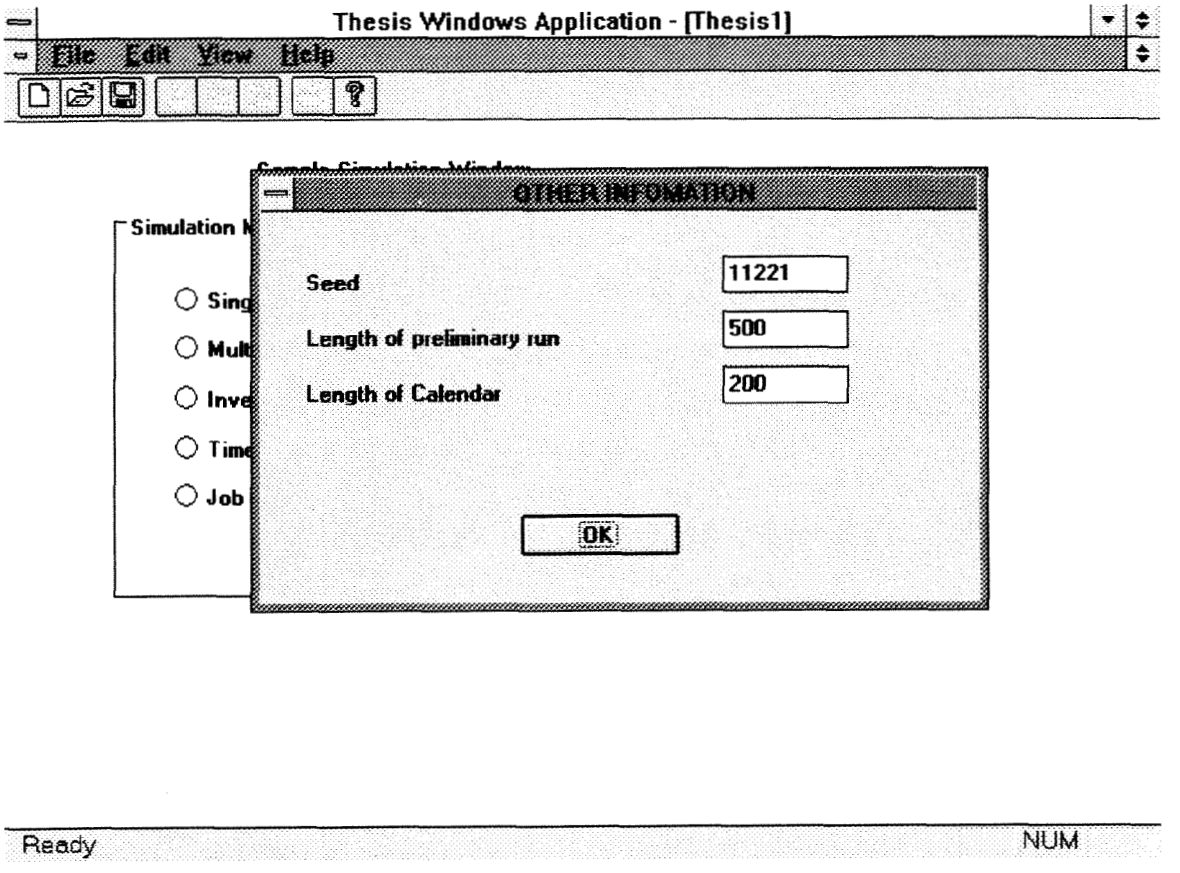
A-4. Server Information Input Dialog box

This screen has a widow edit box that allows a user to input server simulation data. The combo box for input distribution has predefined different kinds of distribution that the program can provide.



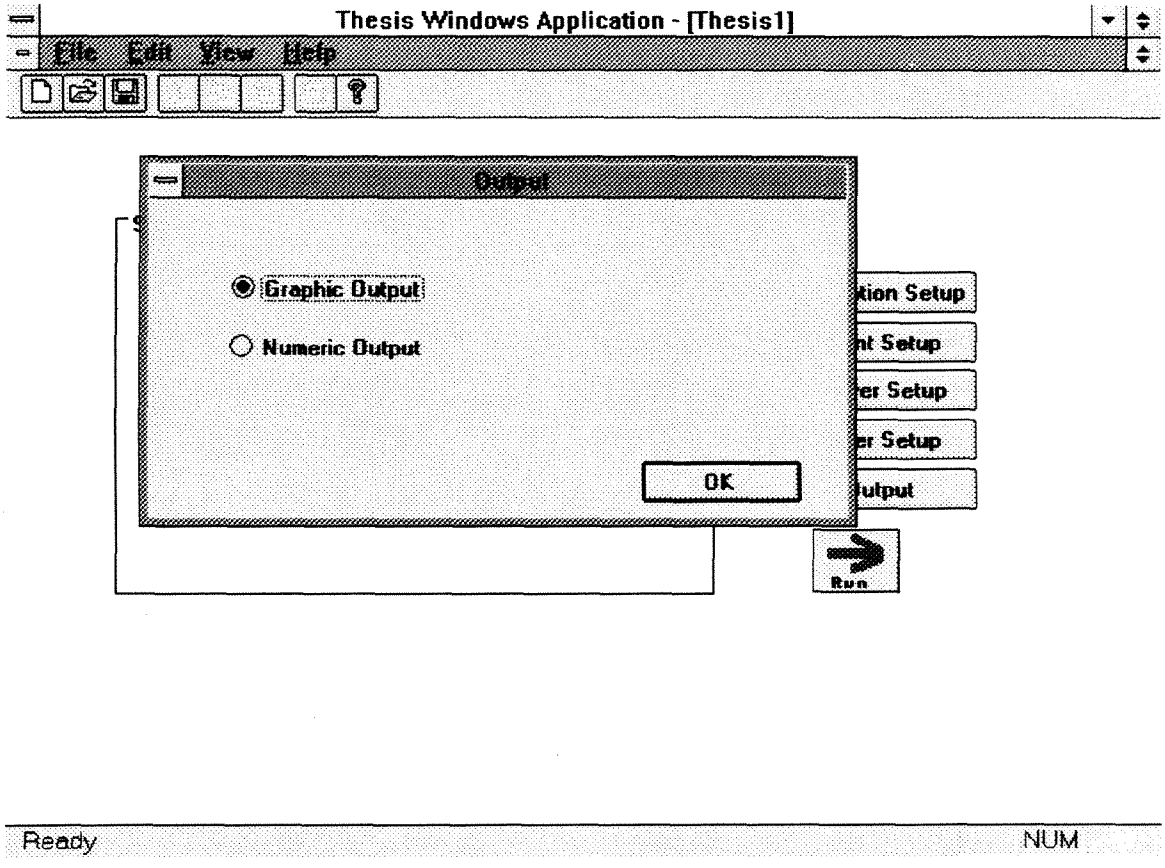
A-5. Other Information Dialog Box

This screen allows a user to change simulation data setup, such as the seed for random number generator, calendar size for event list, et. A user can either change them or use default values.



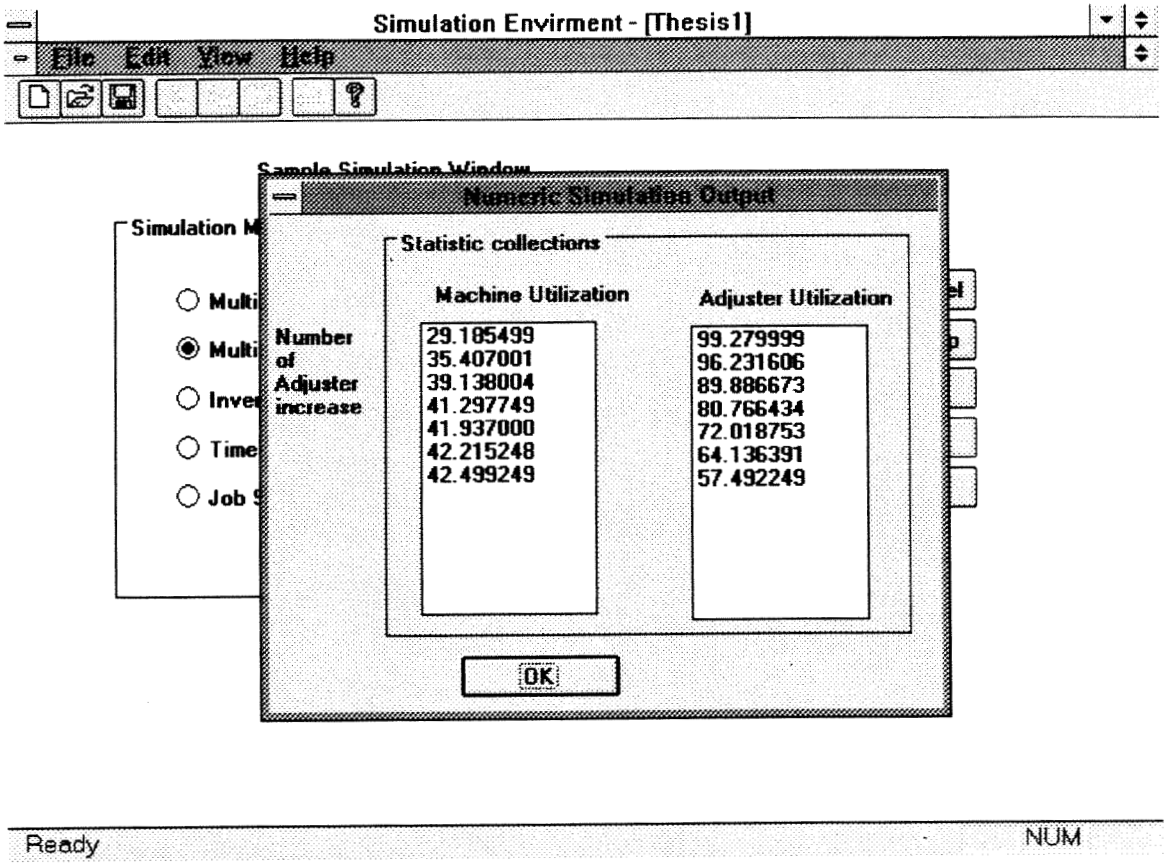
A-6. Output Select Dialog Box

This screen allows a user to change the simulation output format by clicking at one of the choices.



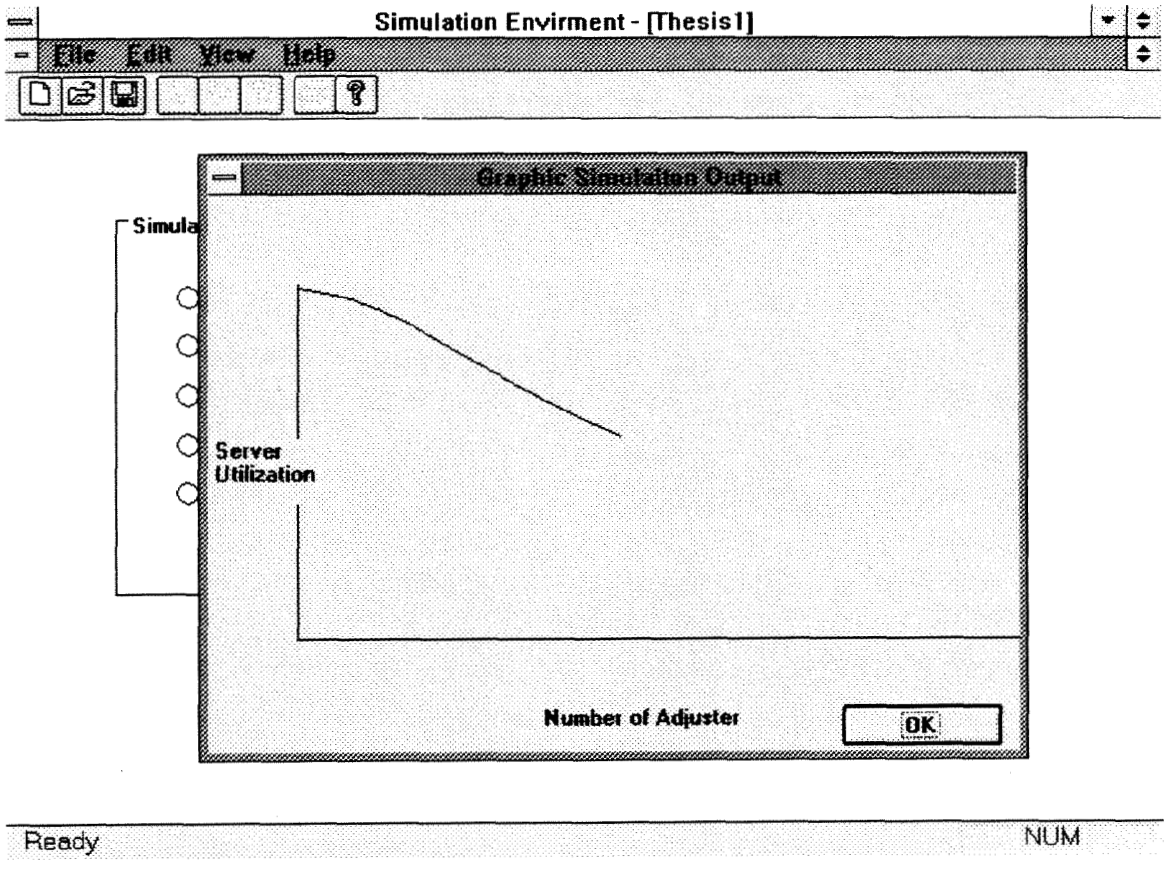
A-7. Numeric Output Screen

This screen outputs the result of simulation numerically.



A-8 Graphic Output Screen

This screen outputs the result of simulation graphically.



Appendix B Simulation Classes

B-1. Simulation Classes

```
//////////////////////////////////////////////////
// file repair.h                                //
// header file for machine-adjustment simulation  //
//////////////////////////////////////////////////


//#if !defined ( _RANDGEO_H )
//#include"randgeo.h"
//#endif

#if !defined( _RANDEXP_H )
#include "randexp.h"
#endif

//#if !defined( _RANDBERN_H )
//#include "randbern.h"
//#endif

class scheduler;
class manager;
class statistic;

class geometric
{
  int geo_max;
  int ways_to_occur;

  public:
    geometric(double mean, int max);
    int RandValue();
};


class simulation
{
  public:
    // constructor

    simulation(){}
    simulation(int &m_mNumber, double &m_mMean, int &m_aNumber, double &m_aMean,
                    unsigned &m_seed, int &m_step)
                    {
                      num_mach = m_mNumber;
                      num_adj = m_aNumber;
                      m_mean = m_mMean;
                      a_mean = m_aMean;
                      seed = m_seed;
                      calendar_size = 200;
                      duration = m_step;
                      mutil = 0;
                      autil = 0;
                      rutil = 0;
                    }
            // desturctor
            ~simulation(){}

            // member functions

            void Execsim();

  // data member
  public:
    unsigned seed;
    int num_mach;
    int num_adj;
    int num_adj1;
    int num_res;
```

```
    double m_mean;
    double a_mean;
    double mutil;
    double autil;
    double rutil;
    int calendar_size;
    long duration;
};


// abstract base class of active objects

class active
{
  public:
    void set_next( active* p) { next = p;}
    void set_time( long time) { occurtime = time;}
    long get_time() { return occurtime;}
    active* get_next() { return next;}
    virtual void event() = 0; // trigger scheduled event
    virtual void reset() = 0; // reset statistic data
  private:
    long occurtime;
    active* next; // next-object pointer for linked lists
};

//class for machine objects

class machine : public active
{
  public:
    machine( double mean, char*name, scheduler* s, manager* m,statistic* c);
    void event();              // time to break down
    void reset();              // reset statistic to 0
    void adjusted();           // repairs complete
    long up_time();       // return total up-time
  private:
    scheduler* sp;             // pointer to scheduler
    manager* mp;          // pointer to service manager
    statistic* cp;        // pointer to statistic object
    RandExponential g;
    //geometric g;    // random number generator
    int is_up;                 // state variabe
    long up_time_start;        // start of most recent up_period
    long tot_up_time;          // total up-time
    long down_time_start;  // start of break-down
    long down_time;            // break-down time include repair time
};


// class of adjuster objects

class adjuster: public active
{
  public:
    adjuster(double mean,char*name, scheduler* s, manager* m);
    void event();                          // time to finish repair
    void reset();                          // reset statistic to zero
    void adjust(machine* p);   // repairs begin
    long busy_time();         // return total busy time
  private:
    scheduler* sp;
    manager* mp;
    RandExponential g;
    //geometric g;
```

```
    machine* workp;
    int is_busy;
    long busy_time_start;
    long tot_busy_time;
};


// class for service manager

class manager
{
  enum who { MACHINES, ADJUSTERS, NONE };

  who waiting;     // kind of objects in queue
  active* first;   // points to first object in queue
  active* last;    // points to last objuct in queue

  // private functions for manipulating queue

  void insert_first(active* p)
  {
    first = last = p;
    p -> set_next(0);
  }

  void insert(active* p)
  {
    last -> set_next(p);
    p -> set_next(0);
    last = p;
  };

  void remove()
  {
    first = first -> get_next();
  }

  public:
    manager();     //{ first = last = 0; waiting = NONE; }
    void request_service(machine* p);   // service request
    void request_work(adjuster* p);     // work request
};


// class for scheduler

class scheduler
{
  public:
    scheduler(int sz);
    ~scheduler();
    long time() { return clock;}              // return time
    void schedule ( active* p, int delay); // schedule event
    void resetcal(void);
    void run ( long begin, long end);
    void prerun(long ticks);       // run simulation
    void deleteobject();

    int get_status() { return status; }
  private:
    long clock;             // simulation clock
    int calendar_size;       // size of calendar-queue array
    active** calendar;       // pointer to calendar-queue array
    long flagtime;          // record time at the beginnig of the calendar
    int status;
};
```

```
// class of statistic node
/*****************************************************************************************
*********/


class stat_link
{
  public:
    stat_link(){ step = 0; down_time = 0; }
    stat_link(long s, long d) { step = s; down_time = d; }
    void set_next( stat_link* s) { nextnode = s ;}
    long get_cur() { return down_time; }
    long get_time() { return step; }
    stat_link* get_next() { return nextnode; }


  private:
    long step;
    long down_time;
    stat_link* nextnode;

  friend statistic;
};


// class of statistic

class statistic
{
  public:
    statistic() {smallest = 100.0; largest = 0.0; warm_up = 0; listHead = 0; listTail = 0;}
    void insert( stat_link*);
    void get_warmup_time(stat_link* s);
    void cal_var();
    void cal_batchsize();
    long get_current(stat_link* s );
    long get_warm() { return warm_up; }
    float get_var() { return variance; }
    int get_batchsize() { return batchsize; }
    void deletenode();

  private:
    long smallest;
    long largest;
    long warm_up;
    int batchsize;
    float variance;

    stat_link* listHead;
    stat_link* listTail;
};
```

```
//////////////////////////////////////////////////////////////
// file active.cpp                                          //
// source file for classes machine and adjuster             //
//////////////////////////////////////////////////////////////


#include "stdafx.h"
#include "thesis.h"
#include "repair.h"
#include "thesidoc.h"
#include "thesivw.h"
//#include "repair.h"

// initialize machine object and schedule first breakdown

machine::machine( double mean, char* name, scheduler* s,manager* m,statistic* c)
        : g(mean,name)
{
  sp = s;
  mp = m;
  cp = c;
  tot_up_time = 0;
  is_up = TRUE;
  up_time_start = sp->time();
  int temp = (int)g.RandValue();
  sp->schedule(this,temp);
}

// request service for disabled machine

void machine::event()
{
  is_up = FALSE;
  down_time_start = sp->time();
  tot_up_time += sp->time() - up_time_start;
  mp->request_service(this);
}

// reset statistic data

void machine::reset()
{
  tot_up_time = 0;
}

// return repaired machine to service

void machine::adjusted()
{
  is_up = TRUE;
  up_time_start = sp->time();
  if( sp->get_status() == 1)
  {
    down_time = sp->time() - down_time_start;
    stat_link* sl;
    sl = new stat_link( up_time_start,down_time);
    cp->insert(sl);
    cp->get_warmup_time(sl);
  }
  int temp = (int)g.RandValue();
  sp->schedule( this, temp);
}
```

```
// return total up-time

long machine::up_time()
{
  /*long t = tot_up_time;
  if( is_up)
  t += sp->time() - up_time_start; */

  return tot_up_time;
}

// initialize adjuster object and, if possible
// get object's first work assignment

adjuster::adjuster( double mean, char* name, scheduler* s, manager* m)
                : g(mean, name)          //g(mean,name)
{
  sp = s;
  mp = m;
  tot_busy_time = 0;
  is_busy = FALSE;
  mp->request_work(this);

}

// complete repair on currnet machine, if possible
// get new work assignment

void adjuster:: event()
{
  tot_busy_time += sp->time() - busy_time_start;
  workp->adjusted();
  is_busy = FALSE;
  mp->request_work(this);
}

// reset

void adjuster::reset()
{
  tot_busy_time = 0;
}

// accept work assignment

void adjuster::adjust(machine* p)
{
  workp = p;
  is_busy = TRUE;
  busy_time_start = sp->time();
  int temp = (int)g.RandValue();
  sp->schedule(this, temp);
}

// return total busy time

long adjuster::busy_time()
{
  /*long t = tot_busy_time;
  if ( is_busy)
    t += sp->time() - busy_time_start; */
  return tot_busy_time;
}
```

```
////////////////////////////////////////////////////
// file manager.cpp                                //
// source file for class manager                   //
////////////////////////////////////////////////////


#include "stdafx.h"
#include "thesis.h"

#include "repair.h"
#include "thesidoc.h"
#include "thesivw.h"
//#include "repair.h"

// handle service request form disabled machine


manager::manager()
{
  first = last = 0;
  waiting = NONE;
}


void manager::request_service(machine* p)
{
  adjuster* q;
  switch(waiting)
  {
    case MACHINES:
      insert(p);
      return;
    case ADJUSTERS:
      q = (adjuster*) first;
      remove();
      if (first = = 0)
        waiting = NONE;
      q->adjust(p);
      return;
    case NONE:
      waiting = MACHINES;
      insert_first(p);
      return;
  }
}

// handle work request from idle adjuster

void manager::request_work(adjuster* p)
{
  machine* q;
  switch(waiting)
  {
    case MACHINES:
      q = (machine*) first;
      remove();
      if(first = = 0)
      {
        waiting = NONE;
      }
      p->adjust(q);
      return;

    case ADJUSTERS:
      insert(p);
      return;
```

```
      case NONE:
        waiting = ADJUSTERS;
        insert_first(p);
        return;

      default:
        return;
    }
}
```

```
//////////////////////////////////////////////////////
// file schedule.cpp                                //
// source file for class scheduler                  //
//////////////////////////////////////////////////////


#include "stdafx.h"
#include "thesis.h"

#include "repair.h"
#include "thesidoc.h"
#include "thesivw.h"

// create scheduler with calendar queue having sz elements

scheduler::scheduler(int sz)
{
  clock = 0;
  flagtime = 0;
  status = 1;
  //status = 0;
  calendar_size = sz;
  calendar = new active*[sz+1];
  for (int i = 0; i < sz+1; i++)
  {
    calendar[i] = 0;
  }

}

scheduler::~scheduler()
{
  active* p;
  for (int i = 0; i <= calendar_size; i++ )
  {
    while( (p = calendar[i]) != 0 )
    {
      calendar[i] = p->get_next();
      delete p;
    }
  }
}


void scheduler::deleteobject()
{

  active* p;
  for (int i = 0; i <= calendar_size; i++ )
  {
    while( (p = calendar[i]) != 0 )
    {
      calendar[i] = p->get_next();
      delete p;
    }
  }
}


// schedule object *p to receive enent message after
// delay ticks have elapsed

void scheduler::schedule( active* p, int delay)
{
  long t = clock + (long)delay;
  p -> set_time(t);
  t -= flagtime;
```

```
  if( t> = calendar_size)
  {
    p->set_next(calendar[calendar_size]);
    calendar[calendar_size] = p;
  }

  else
  {
    p->set_next(calendar[t]);
    calendar[t] = p;
  }
}


void scheduler::resetcal(void)
{
  active* p;
  active* templist;
  templist = 0;
  for ( int i = 0; i < calendar_size; i++)
  {
    calendar[i] = 0;
  }

  while((p = calendar[calendar_size]) != 0)
  {
    long t;
    calendar[calendar_size] = p->get_next();
    t = p->get_time();

    if (t < ( flagtime + calendar_size))
    {
      long time;
      time = p -> get_time() - flagtime;
      p->set_next( calendar[time]);
      calendar[time] = p;
    }

    else
    {
      p->set_next(templist);
      templist = p;
    }
  }  // end of while

  calendar[calendar_size] = templist;
}


/*void scheduler::saveevent(int i)
{
   do
        {
                while((p = calendar[i]) != 0)
                {
                        calendar[i] = p -> get_next();
                        p->set_next(templist);
                        templist = p;
                }

                i++;

        }while( i < calendar_size);

} */

// run simulation for given number of ticks
```

```
void scheduler::run(long begin, long end)
{
  // int ee = begin - flagtime;
   //char temp[20];
   //sprintf(temp," p = %d",ee);
   //AfxMessageBox(temp);

   active* p;
   while( clock < = end)
   {
     int i = 0;

     do
     {
       while((p = calendar[i]) ! = 0)
       {
         calendar[i] = p -> get_next();
         p->event();
       }

       i + +;
       clock + +;
     }while( i < calendar_size && clock < = end);


     if(i = = calendar_size)
     {
        flagtime + = calendar_size;
        resetcal();
     }
   }  // end of while
}

// pre-run for given number of ticks

void scheduler::prerun(long ticks)
{
   active* p;
   while ( clock < = ticks )
   {
     int i = 0;
     do
     {
       while(( p = calendar[i]) ! = 0 )
       {
         calendar[i] = p -> get_next();
         p->event();
       }

       i + +;
       clock + +;
     } while ( i < calendar_size && clock < = ticks );

     if ( i = = calendar_size)
     {
          flagtime + = calendar_size;
          resetcal();
     }
   }  // end of while

   status = 0;
}
```

```
//////////////////////////////////////////////////////
// file simulate.cpp                                //
// demonstration program for machine-adjuster simulation//
//////////////////////////////////////////////////////

#include <iostream.h>
#include <stdlib.h>
#include <stdio.h>
#include "stdafx.h"
#include "thesis.h"

#include "repair.h"
#include "thesidoc.h"
#include "thesivw.h"
//#include "repair.h"

void simulation::Execsim()
{

  //char temp[64];
 // sprintf(temp, "#mach = %d, #adj = %d",num_mach,num_adj);
 // AfxMessageBox(temp);

  srand(seed);
  int max_time = 10 * int ( m_mean > a_mean? m_mean:a_mean);
  int i;

  // create manager and scheduler
  statistic c;
  manager mngr;

  scheduler sch(calendar_size);
  char name[2] = " " ;

  // create machines

  machine** m_list = new machine*[num_mach];
  for ( i = 0; i < num_mach; i++)
  {
    m_list[i] = new machine(m_mean, name, &sch, &mngr, &c);
  }

  // create adjusters

  adjuster** a_list = new adjuster*[num_adj];
  for (i = 0; i < num_adj; i++)
  {
    a_list[i] = new adjuster(a_mean, name, &sch, &mngr);
  }

  // do successive runs of simulation; print cumulative
  // statistics after each run


    // run simulation

   //sch.prerun(duration);
   sch.prerun(500);
   c.cal_var();
   c.cal_batchsize();
   c.deletenode();

  // sch.run(0,500);

  /*for ( i = 0; i < num_mach; i++ )
  {
     m_list[i]->reset();
```

```
}

for ( i = 0; i < num_adj; i++ )
{
  a_list[i]->reset();
} */

 // clear all statistic data

 float m_util[10];
 float a_util[10];

/* AfxMessageBox("before run");
 sch.run( 500,2500);

 float m_factor = 100.0 / 2000.0 / float (num_mach);
 long tot_up_time = 0;
 for (i = 0; i < num_mach; i++)
 {
   tot_up_time += m_list[i] -> up_time();
 }


 mutil = tot_up_time * m_factor;
 char temp1[20];
 sprintf(temp1,"%f",mutil);
 AfxMessageBox(temp1);

 // compute and print average adjuster utilization

 float a_factor = 100.0 / 2000.0 / float (num_adj);

 long tot_busy_time = 0;
 for ( i = 0; i < num_adj; i++)
 {
   tot_busy_time += a_list[i] -> busy_time();
 }


 autil = tot_busy_time * a_factor;
 sprintf(temp1,"%f",autil);
 AfxMessageBox(temp1);   */
int temp;
temp = ((int)(c.get_batchsize()/1000.0)+1)*1000;

long begintime = duration;
long endtime = temp + duration;


for ( i = 0; i < num_mach; i++ )
{
  m_list[i]->reset();
}

for ( i = 0; i < num_adj; i++ )
{
  a_list[i]->reset();
}



for ( int k = 0; k < 10; k++)
{
 sch.run( begintime, endtime);

 float m_factor = 100.0 / float (temp) / float (num_mach);
 long tot_up_time = 0;
```

```
    for (i = 0; i < num_mach; i++)
    {
      tot_up_time += m_list[i] -> up_time();
    }


    m_util[k] = tot_up_time * m_factor;
    char temp1[20];
    sprintf(temp1,"%f",m_util[k]);
    //AfxMessageBox(temp1);

    // compute and print average adjuster utilization

    float a_factor = 100.0 / float (temp) / float (num_adj);

    long tot_busy_time = 0;
    for ( i = 0; i < num_adj; i++)
    {
      tot_busy_time += a_list[i] -> busy_time();
    }


    a_util[k] = tot_busy_time * a_factor;

    sprintf(temp1,"%f",a_util[k]);
    //AfxMessageBox(temp1);
    for ( int j = 0; j < num_mach; j++ )
    {
      m_list[j]->reset();
    }

    for ( j = 0; j < num_adj; j++ )
    {
      a_list[j]->reset();
    }

    begintime = endtime;
    endtime += temp;
  }


  float tot_m_util = 0;
  float tot_a_util = 0;

  for ( i = 0; i < 10; i ++ )
  {
    tot_m_util = tot_m_util + m_util[i];
  }

   mutil = tot_m_util / 10;

  for ( i = 0; i < 10; i++ )
  {
     tot_a_util = tot_a_util + a_util[i];
  }

    autil = tot_a_util / 10;

  sch.deleteobject();

}
```

```
//////////////////////////////////////////////////
// file statistic.cpp                           //
//////////////////////////////////////////////////

#include "stdafx.h"
#include "thesis.h"

#include "repair.h"
#include "thesidoc.h"
#include "thesivw.h"
//#include "repair.h"
#include <iostream.h>


void statistic::insert( stat_link* s )
{
  if (listHead = = 0)
  {
    listHead = s;
    listTail = s;
    s -> set_next(0);
  }

  else
  {
    listTail -> set_next (s);
    listTail = s;
    s -> set_next (0);
  }
}

long statistic::get_current(stat_link* s)
{
  long temp;
  temp = s->get_cur();
  return temp;
}



void statistic::get_warmup_time(stat_link* s)
{
  if ( s->get_cur() < smallest )
  smallest = s->get_cur();

  if ( s->get_cur() > largest )
  largest = s->get_cur();

  if ( s->get_cur() < largest && s->get_cur() > smallest && warm_up = = 0)
  warm_up = s->get_time();
}


void statistic::cal_var()
{
  long temp = 0;
  int i = 0;
  stat_link* current;

  current = listHead;
  do
  {
    temp + = current->get_cur();
    current = current->get_next();
    i + +;
  } while ( current ! = 0 );
```

```
  float average = temp / i;

  current = listHead;

  float t = 0;
  do
  {
    t = t + ( current->get_cur() - average )*(current->get_cur() - average);
    current = current->get_next();
  } while (current != 0 );

  variance = t / i;
}


void statistic::cal_batchsize()
{
  batchsize = (int)( 2.26 * 2.26 * variance / (0.25 * 0.25 ));
}


void statistic::deletenode()
{
   stat_link* temp;

   while ( (temp = listHead) != 0 )
   {
     listHead = temp->get_next();
     delete temp;
   }
}
```

B-2. Window classes

```
////////////////////////////////////////////////////////////////////////
// stdafx.h : include file for standard system include files,        //
//  or project specific include files that are used frequently, but  //
//      are changed infrequently                                      //
////////////////////////////////////////////////////////////////////////

#include <afxwin.h>         // MFC core and standard components
#include <afxext.h>         // MFC extensions (including VB)
```

```
////////////////////////////////////////////////////////////////
// stdafx.cpp : source file that includes just the standard includes //
//  stdafx.pch will be the pre-compiled header                  //
//  stdafx.obj will contain the pre-compiled type information   //
////////////////////////////////////////////////////////////////

#include "stdafx.h"
```

```
////////////////////////////////////////////////////////////////
// thesivw.h : interface of the CThesisView class   //
////////////////////////////////////////////////////////////////

class CThesisView : public CFormView
{
public:
   int output_flag;
   CBitmapButton m_btnRun;
   //CBitmapButton m_btnOutput;
protected: // create from serialization only
          CThesisView();
          DECLARE_DYNCREATE(CThesisView)

public:
          //{{AFX_DATA(CThesisView)
          enum { IDD = IDD_THESIS_FORM };
          int                m_iModelType;
          //}}AFX_DATA

protected:
   virtual void OnInitialUpdate();

private:
   void UpdateEntry();

// Attributes
public:
          CThesisDoc* GetDocument();

// Operations
public:

// Implementation
public:
          virtual ~CThesisView();
#ifdef _DEBUG
          virtual void AssertValid() const;
          virtual void Dump(CDumpContext& dc) const;
#endif

protected:
          virtual void DoDataExchange(CDataExchange* pDX);   // DDX/DDV support

// Generated message map functions
protected:
          //{{AFX_MSG(CThesisView)
          afx_msg void OnClickedEnter();
          afx_msg void OnButtonClient();
          afx_msg void OnButtonOther();
          afx_msg void OnButtonOutput();
          afx_msg void OnButtonServer();
          afx_msg void OnButtonSimu();
          //}}AFX_MSG
          DECLARE_MESSAGE_MAP()
};

#ifndef _DEBUG  // debug version in thesivw.cpp
inline CThesisDoc* CThesisView::GetDocument()
  { return (CThesisDoc*)m_pDocument; }
#endif

////////////////////////////////////////////////////////////////
```

```
////////////////////////////////////////////////////////////////////////////
// thesivw.cpp : implementation of the CThesisView class //
////////////////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "thesis.h"

#include "repair.h"
#include "thesidoc.h"
#include "thesivw.h"
#include "graphic.h"
#include "machine.h"
#include "adjuster.h"
#include "output.h"
#include "subsim.h"
#include "suboutpu.h"
#include "otherinf.h"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

////////////////////////////////////////////////////////////////////////////
// CThesisView

IMPLEMENT_DYNCREATE(CThesisView, CFormView)

BEGIN_MESSAGE_MAP(CThesisView, CFormView)
        //{{AFX_MSG_MAP(CThesisView)
        ON_BN_CLICKED(IDC_ENTER, OnClickedEnter)
        ON_BN_CLICKED(IDC_BUTTON_CLIENT, OnButtonClient)
        ON_BN_CLICKED(IDC_BUTTON_OTHER, OnButtonOther)
        ON_BN_CLICKED(IDC_BUTTON_OUTPUT, OnButtonOutput)
        ON_BN_CLICKED(IDC_BUTTON_SERVER, OnButtonServer)
        ON_BN_CLICKED(IDC_BUTTON_SIMU, OnButtonSimu)
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

////////////////////////////////////////////////////////////////////////////
// CThesisView construction/destruction

CThesisView::CThesisView()
        : CFormView(CThesisView::IDD)
{
        //{{AFX_DATA_INIT(CThesisView)
        m_iModelType = -1;
        //}}AFX_DATA_INIT
        // TODO: add construction code here
        output_flag = 0;
}

CThesisView::~CThesisView()
{
}

void CThesisView::OnInitialUpdate()
{
  // called on startup, on File New, and on File Open
  TRACE ( "Entering CThesisView::OnInitialUpdate\n");
  VERIFY(m_btnRun.AutoLoad(IDC_ENTER,this));
  //VERIFY(m_btnOutput.AutoLoad(IDC_BUTTON_OUTPUT,this));
  UpdateEntry();
}

void CThesisView::UpdateEntry()
{
  // called form OnInitialUpdate and OnEditClearAll
  CThesisDoc* pDoc = ( CThesisDoc*) GetDocument();
```

```
  pDoc->pSim.num_mach = 10;
  pDoc->pSim.num_adj = 5;
  pDoc->pSim.num_adj1 = 20;

  pDoc->pSim.m_mean = 5;
  pDoc->pSim.a_mean = 7;

  pDoc->pSim.duration = 500;
  pDoc->pSim.seed = 11221;
  pDoc->pSim.calendar_size = 500;


  UpdateData(FALSE);  // calls DDX
  ((CDialog*) this ) ->GotoDlgCtrl(GetDlgItem(IDC_STEP));
}



void CThesisView::DoDataExchange(CDataExchange* pDX)
{
        CFormView::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CThesisView)
        DDX_Radio(pDX, IDC_RADIO1, m_iModelType);
        //}}AFX_DATA_MAP
}

/////////////////////////////////////////////////////////////////////////////
// CThesisView diagnostics

#ifdef _DEBUG
void CThesisView::AssertValid() const
{
        CFormView::AssertValid();
}

void CThesisView::Dump(CDumpContext& dc) const
{
        CFormView::Dump(dc);
}

CThesisDoc* CThesisView::GetDocument() // non-debug version is inline
{
        ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CThesisDoc)));
        return (CThesisDoc*)m_pDocument;
}
#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CThesisView message handlers

void CThesisView::OnClickedEnter()
{
        TRACE("Entering CThesisView::OnClickedEnter\n");

        CThesisDoc* pDoc = ( CThesisDoc* ) GetDocument();
        UpdateData( TRUE );
  float machine_util[10],adjuster_util[10];

/*      pDoc->pSim.Execsim();

        char temp[20];
  sprintf(temp,"%f",pDoc->pSim.autil);
  AfxMessageBox(temp);

  sprintf(temp,"%f",pDoc->pSim.mutil);
  AfxMessageBox(temp);   */

        for (int i = 0; i < 7; i++)
        {
```

```
            pDoc->pSim.Execsim();
            char temp[20];
            sprintf(temp,"%f",pDoc->pSim.autil);
           // AfxMessageBox(temp);

            machine_util[i] = pDoc->pSim.mutil;
            adjuster_util[i]= pDoc->pSim.autil;

            pDoc->pSim.num_adj++;
          }

  if(output_flag)
  {
                  CGraphic out_dlg;
                  for( int i = 0; i< 7; i++)
                  {
                    out_dlg.x[i] = (int)(machine_util[i] * 2);
                    out_dlg.y[i] = (int)(adjuster_util[i] *2);
                  }
                  out_dlg.DoModal();
          }

          else
          {
            CSubOutput out_dlg;
            char temp[30];
            for(int j = 0; j < 7; j++)
            {
      out_dlg.machine[j] = machine_util[j];
              out_dlg.adjuster[j] = adjuster_util[j];
            }

            out_dlg.DoModal();
  }

  pDoc->Resetdata();
}

void CThesisView::OnButtonClient()
{
          // TODO: Add your control notification handler code here
          CThesisDoc* pDoc = (CThesisDoc*) GetDocument();
          CMachine machine_dlg;
          machine_dlg.m_iMachNum = pDoc->pSim.num_mach;
  machine_dlg.m_fMachMean = pDoc->pSim.m_mean;
          machine_dlg.DoModal();


  pDoc->pSim.num_mach = machine_dlg.m_iMachNum;
  pDoc->pSim.m_mean = machine_dlg.m_fMachMean;
}

void CThesisView::OnButtonOther()
{
          // TODO: Add your control notification handler code here
          CThesisDoc* pDoc = (CThesisDoc*) GetDocument();
          COtherinf other_dlg;
          other_dlg.m_lSeed = pDoc->pSim.seed;
          other_dlg.m_iStep = pDoc->pSim.duration;
          other_dlg.m_iCalendarSize = pDoc->pSim.calendar_size;
          other_dlg.DoModal();

          pDoc->pSim.seed = other_dlg.m_lSeed;
          pDoc->pSim.duration = other_dlg.m_iStep;
          pDoc->pSim.calendar_size = other_dlg.m_iCalendarSize;

}

void CThesisView::OnButtonOutput()
```

```
{
        // TODO: Add your control notification handler code here
        COutput output_dlg;
        output_dlg.DoModal();

        if(output_dlg.graphic_flag)
          output_flag = 1;
}


void CThesisView::OnButtonServer()
{
        // TODO: Add your control notification handler code here
        CThesisDoc* pDoc = (CThesisDoc*) GetDocument();
        CAdjuster adj_dlg;
        adj_dlg.m_iAdjNum = pDoc->pSim.num_adj;
        adj_dlg.m_fAdjMean = pDoc->pSim.a_mean;
        adj_dlg.m_iAdjNum1 = pDoc->pSim.num_adj1;

        adj_dlg.DoModal();

        pDoc->pSim.num_adj = adj_dlg.m_iAdjNum;
        pDoc->pSim.a_mean = adj_dlg.m_fAdjMean;
        pDoc->pSim.num_adj1 = adj_dlg.m_iAdjNum1;

}

void CThesisView::OnButtonSimu()
{
        // TODO: Add your control notification handler code here
        CSubSimulation sim_dlg;
        sim_dlg.DoModal();

}
```

```
////////////////////////////////////////////////////////////////
// thesidoc.h : interface of the CThesisDoc class   //
//          derived from MFC CDocument          //
////////////////////////////////////////////////////////////////

class CThesisDoc : public CDocument
{
protected: // create from serialization only
        CThesisDoc();
        DECLARE_DYNCREATE(CThesisDoc)

// Attributes
public:
  simulation pSim;

// Operations
public:

// Implementation
public:
        virtual ~CThesisDoc();
        virtual void Serialize(CArchive& ar);   // overridden for document i/o
#ifdef _DEBUG
        virtual void AssertValid() const;
        virtual void Dump(CDumpContext& dc) const;
#endif

public:
  void Resetdata();
protected:
        virtual BOOL OnNewDocument();

// Generated message map functions
protected:
        //{{AFX_MSG(CThesisDoc)
        afx_msg void OnEditClearAll();
        afx_msg void OnUpdateEditClearAll(CCmdUI* pCmdUI);
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};

////////////////////////////////////////////////////////////////
```

```
////////////////////////////////////////////////////
// thesidoc.cpp : implementation of the CThesisDoc class //
////////////////////////////////////////////////////

#include "stdafx.h"
#include "thesis.h"
#include "repair.h"

#include "thesidoc.h"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

////////////////////////////////////////////////////////////////
// CThesisDoc

IMPLEMENT_DYNCREATE(CThesisDoc, CDocument)

BEGIN_MESSAGE_MAP(CThesisDoc, CDocument)
        //{{AFX_MSG_MAP(CThesisDoc)
        ON_COMMAND(ID_EDIT_CLEAR_ALL, OnEditClearAll)
        ON_UPDATE_COMMAND_UI(ID_EDIT_CLEAR_ALL, OnUpdateEditClearAll)
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

////////////////////////////////////////////////////////////////
// CThesisDoc construction/destruction

CThesisDoc::CThesisDoc()
{
        TRACE( "Document object constructed\n");
}

CThesisDoc::~CThesisDoc()
{
  TRACE( "Document object destroyed\n");
  #ifdef _DEBUG
    Dump(afxDump);
  #endif
}

BOOL CThesisDoc::OnNewDocument()
{
        if (!CDocument::OnNewDocument())
                return FALSE;

        // TODO: add reinitialization code here
        // (SDI documents will reuse this document)

        return TRUE;
}

////////////////////////////////////////////////////////////////
// CThesisDoc serialization

void CThesisDoc::Serialize(CArchive& ar)
{
        if (ar.IsStoring())
        {
                // TODO: add storing code here
        }
        else
        {
                // TODO: add loading code here
        }
}
```

```
/////////////////////////////////////////////////////////////////////////////
// CThesisDoc diagnostics

#ifdef _DEBUG
void CThesisDoc::AssertValid() const
{
	CDocument::AssertValid();
}

void CThesisDoc::Dump(CDumpContext& dc) const
{
	CDocument::Dump(dc);
}
#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CThesisDoc commands

void CThesisDoc::OnEditClearAll()
{
	// TODO: Add your command handler code here

}

void CThesisDoc::OnUpdateEditClearAll(CCmdUI* pCmdUI)
{
	// TODO: Add your command update UI handler code here

}

void CThesisDoc::Resetdata()
{
  pSim.num_mach = 10;
  pSim.num_adj = 5;
  pSim.num_adj1 = 20;

  pSim.m_mean = 5;
  pSim.a_mean = 7;

  pSim.duration = 500;
  pSim.seed = 11221;
  pSim.calendar_size = 200;

}
```

```
/////////////////////////////////////////////////////////////////////////////
// mainfrm.h : interface of the CMainFrame class    //
/////////////////////////////////////////////////////////////////////////////

class CMainFrame : public CMDIFrameWnd
{
        DECLARE_DYNAMIC(CMainFrame)
public:
        CMainFrame();

public:

// Attributes
public:

// Operations
public:

// Implementation
public:
        virtual ~CMainFrame();
#ifdef _DEBUG
        virtual void AssertValid() const;
        virtual void Dump(CDumpContext& dc) const;
#endif

protected:  // control bar embedded members
        CStatusBar  m_wndStatusBar;
        CToolBar    m_wndToolBar;

// Generated message map functions
protected:
        //{{AFX_MSG(CMainFrame)
        afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
                // NOTE - the ClassWizard will add and remove member functions here.
                //    DO NOT EDIT what you see in these blocks of generated code!
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////
```

```
////////////////////////////////////////////////////////////////////////////
// mainfrm.cpp : implementation of the CMainFrame class //
////////////////////////////////////////////////////////////////////////////


#include "stdafx.h"
#include "thesis.h"

#include "mainfrm.h"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CMainFrame

IMPLEMENT_DYNAMIC(CMainFrame, CMDIFrameWnd)

BEGIN_MESSAGE_MAP(CMainFrame, CMDIFrameWnd)
        //{{AFX_MSG_MAP(CMainFrame)
                // NOTE - the ClassWizard will add and remove mapping macros here.
                //    DO NOT EDIT what you see in these blocks of generated code !
        ON_WM_CREATE()
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// arrays of IDs used to initialize control bars

// toolbar buttons - IDs are command buttons
static UINT BASED_CODE buttons[] =
{
        // same order as in the bitmap 'toolbar.bmp'
        ID_FILE_NEW,
        ID_FILE_OPEN,
        ID_FILE_SAVE,
                ID_SEPARATOR,
        ID_EDIT_CUT,
        ID_EDIT_COPY,
        ID_EDIT_PASTE,
                ID_SEPARATOR,
        ID_FILE_PRINT,
        ID_APP_ABOUT,
};

static UINT BASED_CODE indicators[] =
{
        ID_SEPARATOR,           // status line indicator
        ID_INDICATOR_CAPS,
        ID_INDICATOR_NUM,
        ID_INDICATOR_SCRL,
};

/////////////////////////////////////////////////////////////////////////////
// CMainFrame construction/destruction

CMainFrame::CMainFrame()
{
        // TODO: add member initialization code here
}

CMainFrame::~CMainFrame()
{
}

int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
```

```
	if (CMDIFrameWnd::OnCreate(lpCreateStruct) == -1)
		return -1;

  //VERIFY(m_btnRun.AutoLoad(IDC_ENTER,this));

	if (!m_wndToolBar.Create(this) ||
		!m_wndToolBar.LoadBitmap(IDR_MAINFRAME) ||
		!m_wndToolBar.SetButtons(buttons,
		  sizeof(buttons)/sizeof(UINT)))
	{
		TRACE("Failed to create toolbar\n");
		return -1;	// fail to create
	}

	if (!m_wndStatusBar.Create(this) ||
		!m_wndStatusBar.SetIndicators(indicators,
		  sizeof(indicators)/sizeof(UINT)))
	{
		TRACE("Failed to create status bar\n");
		return -1;	// fail to create
	}

	return 0;
}

/////////////////////////////////////////////////////////////////////////////
// CMainFrame diagnostics

#ifdef _DEBUG
void CMainFrame::AssertValid() const
{
	CMDIFrameWnd::AssertValid();
}

void CMainFrame::Dump(CDumpContext& dc) const
{
	CMDIFrameWnd::Dump(dc);
}

#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CMainFrame message handlers
```

```
/////////////////////////////////////////////////////////////////////////////
// thesis.h : main header file for the THESIS application //
//          the one and only application class                  //
/////////////////////////////////////////////////////////////////////////////

#ifndef __AFXWIN_H__
        #error include 'stdafx.h' before including this file for PCH
#endif

#include "resource.h"       // main symbols

/////////////////////////////////////////////////////////////////////////////
// CThesisApp:
// See thesis.cpp for the implementation of this class
//

class CThesisApp : public CWinApp
{
public:
        CThesisApp();

// Overrides
        virtual BOOL InitInstance();

// Implementation

        //{{AFX_MSG(CThesisApp)
        afx_msg void OnAppAbout();
                // NOTE - the ClassWizard will add and remove member functions here.
                //    DO NOT EDIT what you see in these blocks of generated code !
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};


/////////////////////////////////////////////////////////////////////////////
```


74

```
////////////////////////////////////////////////////////////////////////////
// thesis.cpp : Defines the class behaviors for the application.//
////////////////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "thesis.h"

#include "mainfrm.h"
#include "repair.h"
#include "thesidoc.h"
#include "thesivw.h"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CThesisApp

BEGIN_MESSAGE_MAP(CThesisApp, CWinApp)
        //{{AFX_MSG_MAP(CThesisApp)
        ON_COMMAND(ID_APP_ABOUT, OnAppAbout)
                // NOTE - the ClassWizard will add and remove mapping macros here.
                //    DO NOT EDIT what you see in these blocks of generated code!
        //}}AFX_MSG_MAP
        // Standard file based document commands
        ON_COMMAND(ID_FILE_NEW, CWinApp::OnFileNew)
        ON_COMMAND(ID_FILE_OPEN, CWinApp::OnFileOpen)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CThesisApp construction

CThesisApp::CThesisApp()
{
        // TODO: add construction code here,
        // Place all significant initialization in InitInstance
}

/////////////////////////////////////////////////////////////////////////////
// The one and only CThesisApp object

CThesisApp NEAR theApp;

/////////////////////////////////////////////////////////////////////////////
// CThesisApp initialization

BOOL CThesisApp::InitInstance()
{
        // Standard initialization
        // If you are not using these features and wish to reduce the size
        //  of your final executable, you should remove from the following
        //  the specific initialization routines you do not need.

        SetDialogBkColor();      // Set dialog background color to gray
        LoadStdProfileSettings(); // Load standard INI file options (including MRU)

        // Register the application's document templates.  Document templates
        //  serve as the connection between documents, frame windows and views.

        CMultiDocTemplate* pDocTemplate;
        pDocTemplate = new CMultiDocTemplate(
                IDR_THESISTYPE,
                RUNTIME_CLASS(CThesisDoc),
                RUNTIME_CLASS(CMDIChildWnd),      // standard MDI child frame
                RUNTIME_CLASS(CThesisView));
        AddDocTemplate(pDocTemplate);
```

```
	// create main MDI Frame window
	CMainFrame* pMainFrame = new CMainFrame;
	if (!pMainFrame->LoadFrame(IDR_MAINFRAME))
		return FALSE;
	m_pMainWnd = pMainFrame;

	// create a new (empty) document
	OnFileNew();

	if (m_lpCmdLine[0] != '\0')
	{
		// TODO: add command line processing here
	}

	// The main window has been initialized, so show and update it.
	pMainFrame->ShowWindow(m_nCmdShow);
	pMainFrame->UpdateWindow();

	return TRUE;
}

/////////////////////////////////////////////////////////////////////////////
// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
	CAboutDlg();

// Dialog Data
	//{{AFX_DATA(CAboutDlg)
	enum { IDD = IDD_ABOUTBOX };
	//}}AFX_DATA

// Implementation
protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
	//{{AFX_MSG(CAboutDlg)
		// No message handlers
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
	//{{AFX_DATA_INIT(CAboutDlg)
	//}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CAboutDlg)
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
	//{{AFX_MSG_MAP(CAboutDlg)
		// No message handlers
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

// App command to run the dialog
void CThesisApp::OnAppAbout()
{
	CAboutDlg aboutDlg;
	aboutDlg.DoModal();
}
```

```
/////////////////////////////////////////////////////////////////////////////
// CThesisApp commands
```

```
/////////////////////////////////////////////////////////////
// subsim.h : header file                    //
// header file dialog class simulation//
/////////////////////////////////////////////////////////////
// CSubSimulation dialog

class CSubSimulation : public CDialog
{
// Construction
public:
        CSubSimulation(CWnd* pParent = NULL);        // standard constructor

// Dialog Data
        //{{AFX_DATA(CSubSimulation)
        enum { IDD = IDD_DIALOG_SIMULATION };
        BOOL    m_bCloseLoop;
        BOOL    m_bContinous;
        BOOL    m_bDescrete;
        BOOL    m_bDescriptive;
        BOOL    m_bDeterminating;
        BOOL    m_bNonterminating;
        BOOL    m_bOpenloop;
        BOOL    m_bPrescriptive;
        BOOL    m_bProbabilistic;
        BOOL    m_bStatic;
        BOOL    m_bTerminating;
        BOOL    m_bDynamic;
        //}}AFX_DATA

// Implementation
protected:
        virtual void DoDataExchange(CDataExchange* pDX);      // DDX/DDV support

        // Generated message map functions
        //{{AFX_MSG(CSubSimulation)
                // NOTE: the ClassWizard will add member functions here
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};
```

```
////////////////////////////////////////////////////////
// subsim.cpp : implementation file //
////////////////////////////////////////////////////////

#include "stdafx.h"
#include "thesis.h"
#include "subsim.h"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////
// CSubSimulation dialog


CSubSimulation::CSubSimulation(CWnd* pParent /*=NULL*/)
        : CDialog(CSubSimulation::IDD, pParent)
{
        //{{AFX_DATA_INIT(CSubSimulation)
        m_bCloseLoop = TRUE;
        m_bContinous = FALSE;
        m_bDescrete = TRUE;
        m_bDescriptive = TRUE;
        m_bDeterminating = FALSE;
        m_bNonterminating = TRUE;
        m_bOpenloop = FALSE;
        m_bPrescriptive = FALSE;
        m_bProbabilistic = TRUE;
        m_bStatic = TRUE;
        m_bTerminating = FALSE;
        m_bDynamic = FALSE;
        //}}AFX_DATA_INIT
}

void CSubSimulation::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CSubSimulation)
        DDX_Check(pDX, IDC_CHECK_CLOSELOOP, m_bCloseLoop);
        DDX_Check(pDX, IDC_CHECK_CONTINUOUS, m_bContinous);
        DDX_Check(pDX, IDC_CHECK_DESCRETE, m_bDescrete);
        DDX_Check(pDX, IDC_CHECK_DESCRIPTIVE, m_bDescriptive);
        DDX_Check(pDX, IDC_CHECK_DETERMINISTIC, m_bDeterminating);
        DDX_Check(pDX, IDC_CHECK_NONTERMINATING, m_bNonterminating);
        DDX_Check(pDX, IDC_CHECK_OPENLOOP, m_bOpenloop);
        DDX_Check(pDX, IDC_CHECK_PRESCRIPTIVE, m_bPrescriptive);
        DDX_Check(pDX, IDC_CHECK_PROBABILISTIC, m_bProbabilistic);
        DDX_Check(pDX, IDC_CHECK_STATIC, m_bStatic);
        DDX_Check(pDX, IDC_CHECK_TERMINATING, m_bTerminating);
        DDX_Check(pDX, IDC_CHECK8_DYNAMIC, m_bDynamic);
        //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CSubSimulation, CDialog)
        //{{AFX_MSG_MAP(CSubSimulation)
                // NOTE: the ClassWizard will add message map macros here
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()


/////////////////////////////////////////////////////////////////////////
// CSubSimulation message handlers
```

```
////////////////////////////////////////////////
// machine.h : header file          //
////////////////////////////////////////////////

// CMachine dialog

class CMachine : public CDialog
{
// Construction
public:
        CMachine(CWnd* pParent = NULL);    // standard constructor

// Dialog Data
        //{{AFX_DATA(CMachine)
        enum { IDD = IDD_DIALOG_CLIENT };
        float       m_fMachMean;
        int                 m_iMachNum;
        //}}AFX_DATA

// Implementation
protected:
        virtual void DoDataExchange(CDataExchange* pDX);       // DDX/DDV support

        // Generated message map functions
        //{{AFX_MSG(CMachine)
                // NOTE: the ClassWizard will add member functions here
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};
```

```
////////////////////////////////////////////////////////////
// machine.cpp : implementation file //
////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "thesis.h"
#include "machine.h"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

////////////////////////////////////////////////////////////////////////
// CMachine dialog


CMachine::CMachine(CWnd* pParent /*=NULL*/)
        : CDialog(CMachine::IDD, pParent)
{
        //{{AFX_DATA_INIT(CMachine)
        m_fMachMean = 0;
        m_iMachNum = 0;
        //}}AFX_DATA_INIT
}

void CMachine::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CMachine)
        DDX_Text(pDX, IDC_EDIT_MEAN_MACH, m_fMachMean);
        DDX_Text(pDX, IDC_EDIT_NUM_MACH, m_iMachNum);
        //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CMachine, CDialog)
        //{{AFX_MSG_MAP(CMachine)
                // NOTE: the ClassWizard will add message map macros here
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()


////////////////////////////////////////////////////////////////////////
// CMachine message handlers
```

```
////////////////////////////////////////////////
// adjuster.h : header file        //
////////////////////////////////////////////////

// CAdjuster dialog

class CAdjuster : public CDialog
{
// Construction
public:
        CAdjuster(CWnd* pParent = NULL);    // standard constructor

// Dialog Data
        //{{AFX_DATA(CAdjuster)
        enum { IDD = IDD_DIALOG_SERVER };
        float       m_fAdjMean;
        int                 m_iAdjNum;
        int                 m_iAdjNum1;
        //}}AFX_DATA

// Implementation
protected:
        virtual void DoDataExchange(CDataExchange* pDX);       // DDX/DDV support

        // Generated message map functions
        //{{AFX_MSG(CAdjuster)
                // NOTE: the ClassWizard will add member functions here
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};
```

```
////////////////////////////////////////////////////////////
// adjuster.cpp : implementation file  //
////////////////////////////////////////////////////////////


#include "stdafx.h"
#include "thesis.h"
#include "adjuster.h"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

////////////////////////////////////////////////////////////////////
// CAdjuster dialog


CAdjuster::CAdjuster(CWnd* pParent /*=NULL*/)
        : CDialog(CAdjuster::IDD, pParent)
{
        //{{AFX_DATA_INIT(CAdjuster)
        m_fAdjMean = 0;
        m_iAdjNum = 0;
        m_iAdjNum1 = 0;
        //}}AFX_DATA_INIT
}

void CAdjuster::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CAdjuster)
        DDX_Text(pDX, IDC_EDIT_ADJ_MEAN, m_fAdjMean);
        DDX_Text(pDX, IDC_EDIT_ADJ_NUM, m_iAdjNum);
        DDX_Text(pDX, IDC_EDIT_ADJ_NUM1, m_iAdjNum1);
        //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAdjuster, CDialog)
        //{{AFX_MSG_MAP(CAdjuster)
                // NOTE: the ClassWizard will add message map macros here
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()


////////////////////////////////////////////////////////////////////
// CAdjuster message handlers
```

```
////////////////////////////////////////////////
// otherinf.h : header file          //
////////////////////////////////////////////////

/////////////////////////////////////////////////////////////////////////
// COtherinf dialog

class COtherinf : public CDialog
{
// Construction
public:
        COtherinf(CWnd* pParent = NULL);    // standard constructor

// Dialog Data
        //{{AFX_DATA(COtherinf)
        enum { IDD = IDD_DIALOG_OTHER };
        int                 m_iCalendarSize;
        long        m_lSeed;
        int                 m_iStep;
        //}}AFX_DATA

// Implementation
protected:
        virtual void DoDataExchange(CDataExchange* pDX);      // DDX/DDV support

        // Generated message map functions
        //{{AFX_MSG(COtherinf)
                // NOTE: the ClassWizard will add member functions here
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};
```

```
////////////////////////////////////////////////////////////
// otherinf.cpp : implementation file   //
////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "thesis.h"
#include "otherinf.h"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

////////////////////////////////////////////////////////////////////
// COtherinf dialog


COtherinf::COtherinf(CWnd* pParent /*=NULL*/)
        : CDialog(COtherinf::IDD, pParent)
{
        //{{AFX_DATA_INIT(COtherinf)
        m_iCalendarSize = 0;
        m_lSeed = 0;
        m_iStep = 0;
        //}}AFX_DATA_INIT
}

void COtherinf::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(COtherinf)
        DDX_Text(pDX, IDC_EDIT_CALENDARSIZE, m_iCalendarSize);
        DDX_Text(pDX, IDC_EDIT_SEED, m_lSeed);
        DDX_Text(pDX, IDC_EDIT_STEP, m_iStep);
        //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(COtherinf, CDialog)
        //{{AFX_MSG_MAP(COtherinf)
                // NOTE: the ClassWizard will add message map macros here
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()


////////////////////////////////////////////////////////////////////
// COtherinf message handlers
```

```
////////////////////////////////////////////////
// output.h : header file          //
////////////////////////////////////////////////

/////////////////////////////////////////////////////////////////////////
// COutput dialog

class COutput : public CDialog
{
public:
  int graphic_flag;
// Construction
public:
        COutput(CWnd* pParent = NULL);       // standard constructor

// Dialog Data
        //{{AFX_DATA(COutput)
        enum { IDD = IDD_DIALOG_OUTPUT };
                // NOTE: the ClassWizard will add data members here
        //}}AFX_DATA

// Implementation
protected:
        virtual void DoDataExchange(CDataExchange* pDX);       // DDX/DDV support

        // Generated message map functions
        //{{AFX_MSG(COutput)
        afx_msg void OnRadioGraphic();
        afx_msg void OnRadioNumeric();
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};
```

```
////////////////////////////////////////////////////////
// output.cpp : implementation file //
////////////////////////////////////////////////////////


#include "stdafx.h"
#include "thesis.h"
#include "output.h"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////
// COutput dialog


COutput::COutput(CWnd* pParent /*=NULL*/)
        : CDialog(COutput::IDD, pParent)
{
        //{{AFX_DATA_INIT(COutput)
                // NOTE: the ClassWizard will add member initialization here
        //}}AFX_DATA_INIT
        graphic_flag = 0;
}

void COutput::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(COutput)
                // NOTE: the ClassWizard will add DDX and DDV calls here
        //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(COutput, CDialog)
        //{{AFX_MSG_MAP(COutput)
        ON_BN_CLICKED(IDC_RADIO_GRAPHIC, OnRadioGraphic)
        ON_BN_CLICKED(IDC_RADIO_NUMERIC, OnRadioNumeric)
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()


/////////////////////////////////////////////////////////////////////////
// COutput message handlers

void COutput::OnRadioGraphic()
{
        // TODO: Add your control notification handler code here
        graphic_flag = 1;

}

void COutput::OnRadioNumeric()
{
        // TODO: Add your control notification handler code here
        graphic_flag = 0;
}
```

```
////////////////////////////////////////////////
// suboutpu.h : header file      //
////////////////////////////////////////////////
// CSubOutput dialog

class CSubOutput : public CDialog
{
public:
   float machine[10], adjuster[10];
// Construction
public:
        CSubOutput(CWnd* pParent = NULL); // standard constructor

// Dialog Data
        //{{AFX_DATA(CSubOutput)
        enum { IDD = IDD_DIALOG_NUMERIC };
        CListBox m_lbMachUtil;
        CListBox m_lbAdjUtility;
        //}}AFX_DATA

// Implementation
protected:
        virtual void DoDataExchange(CDataExchange* pDX);      // DDX/DDV support

        // Generated message map functions
        //{{AFX_MSG(CSubOutput)
        virtual BOOL OnInitDialog();
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};
```

```
////////////////////////////////////////////////////////
// suboutpu.cpp : implementation file//
////////////////////////////////////////////////////////

#include "stdafx.h"
#include "thesis.h"
#include "suboutpu.h"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

////////////////////////////////////////////////////////////////////
// CSubOutput dialog


CSubOutput::CSubOutput(CWnd* pParent /*=NULL*/)
        : CDialog(CSubOutput::IDD, pParent)
{
        //{{AFX_DATA_INIT(CSubOutput)
                // NOTE: the ClassWizard will add member initialization here
        //}}AFX_DATA_INIT
        for (int i = 0; i < 10; i++)
        {
          machine[i] = 0;
          adjuster[i] = 0;
        }
}

void CSubOutput::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CSubOutput)
        DDX_Control(pDX, IDC_LIST_MACH_UTIL, m_lbMachUtil);
        DDX_Control(pDX, IDC_LIST_ADJ_UTIL, m_lbAdjUtility);
        //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CSubOutput, CDialog)
        //{{AFX_MSG_MAP(CSubOutput)
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()


////////////////////////////////////////////////////////////////////
// CSubOutput message handlers

BOOL CSubOutput::OnInitDialog()
{
        CDialog::OnInitDialog();

        // TODO: Add extra initialization here
        char temp[30];
  for(int j = 0; j < 7; j++)
  {
    sprintf(temp,"%f", machine[j]);
    m_lbMachUtil.AddString(temp);
    sprintf(temp,"%f", adjuster[j]);
    m_lbAdjUtility.AddString(temp);
  }
        return TRUE;  // return TRUE  unless you set the focus to a control
}
```

```
/////////////////////////////////////////////////
// graphic.h : header file          //
/////////////////////////////////////////////////


// CGraphic dialog

class CGraphic : public CDialog
{
// Construction
public:
        CGraphic(CWnd* pParent = NULL);     // standard constructor

  int x[10],y[10];
// Dialog Data
        //{{AFX_DATA(CGraphic)
        enum { IDD = IDD_DIALOG_GRAPH };
                // NOTE: the ClassWizard will add data members here
        //}}AFX_DATA

// Implementation
protected:
        virtual void DoDataExchange(CDataExchange* pDX);       // DDX/DDV support

        // Generated message map functions
        //{{AFX_MSG(CGraphic)
        afx_msg void OnPaint();
        afx_msg void OnButton1();
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};
```

```
////////////////////////////////////////////////////////////
// graphic.cpp : implementation file //
////////////////////////////////////////////////////////////


#include "stdafx.h"
#include "thesis.h"
#include "graphic.h"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

////////////////////////////////////////////////////////////////////////
// CGraphic dialog


CGraphic::CGraphic(CWnd* pParent /*=NULL*/)
        : CDialog(CGraphic::IDD, pParent)
{
        //{{AFX_DATA_INIT(CGraphic)
                // NOTE: the ClassWizard will add member initialization here
        //}}AFX_DATA_INIT
}

void CGraphic::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CGraphic)
                // NOTE: the ClassWizard will add DDX and DDV calls here
        //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CGraphic, CDialog)
        //{{AFX_MSG_MAP(CGraphic)
        ON_WM_PAINT()
        ON_BN_CLICKED(IDC_BUTTON1, OnButton1)
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()


////////////////////////////////////////////////////////////////////////
// CGraphic message handlers

void CGraphic::OnPaint()
{
        CPaintDC dc(this); // device context for painting

        // TODO: Add your message handler code here

        // Do not call CDialog::OnPaint() for painting messages
        dc.MoveTo(50,50);
        dc.LineTo(50,250);
        dc.LineTo(500,250);
        dc.MoveTo(50,250-x[0]);
        /*for(int i = 1; i < 7; i++)
        {
          dc.LineTo(50+30*i,250-x[i]);
        } */

        dc.MoveTo(50,250-y[0]);
        for(int j = 1; j < 7; j++)
        {
         dc.LineTo(50+30*j,250-y[j]);
        }
}

void CGraphic::OnButton1()
```

```
{
        // TODO: Add your control notification handler code here
        CDialog::OnOK();

}
```

```
//{{NO_DEPENDENCIES}}
// App Studio generated include file.
// Used by THESIS.RC
//
#define IDR_MAINFRAME              2
#define IDR_THESISTYPE             3
#define IDD_ABOUTBOX               100
#define IDD_THESIS_FORM            101
#define IDD_HELP                   103
#define IDD_DIALOG_OTHER           103
#define IDR_MENU_HOWTOUSE          104
#define IDR_MENU_POP               105
#define IDD_DIALOG_NUM_MACH        106
#define IDD_DIALOG_CLIENT          106
#define IDD_DIALOG_GRAPH           107
#define IDB_BITMAP1                108
#define IDD_DIALOG_SERVER          109
#define IDD_DIALOG_OUTPUT          112
#define IDD_DIALOG_SIMULATION      113
#define IDD_DIALOG_NUMERIC         114
#define IDC_MNUMBER                1000
#define IDC_ANUMBER                1001
#define IDC_MDISTRIBUTION          1002
#define IDC_MMEAN                  1003
#define IDC_ADISTRIBUTION          1004
#define IDC_AMEAN                  1005
#define IDC_RNUMBER                1006
#define IDC_RDISTRIBUTION          1007
#define IDC_RMEAN                  1008
#define IDC_ENTER                  1009
#define IDC_MUTILITY               1010
#define IDC_AUTILITY               1011
#define IDC_RUTILITY               1012
#define IDC_SEED                   1013
#define IDC_STEP                   1014
#define IDC_CALSIZE                1015
#define IDC_STATIC_MNUMBER         1016
#define IDC_RADIO1                 1017
#define IDC_RADIO2                 1018
#define IDC_RADIO3                 1019
#define IDC_RADIO4                 1020
#define IDC_RADIO5                 1021
#define IDC_BUTTON_SERVER          1023
#define IDC_BUTTON_SIMU            1024
#define IDC_BUTTON_OTHER           1025
#define IDC_BUTTON_OUTPUT          1026
#define IDC_BUTTON_CLIENT          1027
#define IDC_COMBO_MACH             1028
#define IDC_EDIT_NUM_MACH          1029
#define IDC_EDIT_MEAN_MACH         1030
#define IDC_EDIT_ADJ_NUM           1031
#define IDC_COMBO_ADJ              1032
#define IDC_EDIT_ADJ_MEAN          1033
#define IDC_RADIO_GRAPHIC          1034
#define IDC_RADIO_NUMERIC          1035
#define IDC_EDIT_SEED              1036
#define IDC_EDIT_STEP              1037
#define IDC_EDIT_CALENDARSIZE      1038
#define IDC_CHECK_PRESCRIPTIVE     1039
#define IDC_CHECK_DESCRIPTIVE      1040
#define IDC_CHECK_DESCRETE         1041
#define IDC_CHECK_CONTINUOUS       1042
#define IDC_CHECK_PROBABILISTIC    1043
#define IDC_CHECK_DETERMINISTIC    1044
#define IDC_CHECK_STATIC           1045
#define IDC_CHECK8_DYNAMIC         1046
#define IDC_CHECK_OPENLOOP         1047
#define IDC_CHECK_CLOSELOOP        1048
#define IDC_CHECK_TERMINATING      1049
```

```
#define IDC_CHECK_NONTERMINATING      1050
#define IDC_LIST_MACH_UTIL            1051
#define IDC_LIST_ADJ_UTIL             1052
#define IDC_EDIT_ADJ_NUM1             1053
#define IDC_BUTTON1                   1054
#define ID_HELP_HOWTO                 32772
#define ID_FILE_EXIT                  32775
#define ID_SEARCH_ONSUBJUCT           32776
#define ID_SEARCH_ONLANGUAGE          32777

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS

#define _APS_NEXT_RESOURCE_VALUE      118
#define _APS_NEXT_COMMAND_VALUE       32780
#define _APS_NEXT_CONTROL_VALUE       1055
#define _APS_NEXT_SYMED_VALUE         101
#endif
#endif
```

```
//Microsoft App Studio generated resource script.
//
#include "resource.h"

#define APSTUDIO_READONLY_SYMBOLS
////////////////////////////////////////////////////////////////////////////
//
// Generated from the TEXTINCLUDE 2 resource.
//
#include "afxres.h"

////////////////////////////////////////////////////////////////////////////
#undef APSTUDIO_READONLY_SYMBOLS

#ifdef APSTUDIO_INVOKED
////////////////////////////////////////////////////////////////////////////
//
// TEXTINCLUDE
//

1 TEXTINCLUDE DISCARDABLE
BEGIN
   "resource.h\0"
END

2 TEXTINCLUDE DISCARDABLE
BEGIN
   "#include ""afxres.h""\r\n"
   "\0"
END

3 TEXTINCLUDE DISCARDABLE
BEGIN
   "#include ""res\\thesis.rc2""  // non-App Studio edited resources\r\n"
   "\r\n"
   "#include ""afxres.rc""  \011// Standard components\r\n"
   "\0"
END

////////////////////////////////////////////////////////////////////////////
#endif    // APSTUDIO_INVOKED


////////////////////////////////////////////////////////////////////////////
//
// Icon
//

IDR_MAINFRAME        ICON   DISCARDABLE    "RES\\THESIS.ICO"
IDR_THESISTYPE       ICON   DISCARDABLE    "RES\\THESIDOC.ICO"

////////////////////////////////////////////////////////////////////////////
//
// Bitmap
//

IDR_MAINFRAME        BITMAP  MOVEABLE PURE   "RES\\TOOLBAR.BMP"
RUNU                 BITMAP  DISCARDABLE     "RES\\BITMAP1.BMP"
RUND                 BITMAP  DISCARDABLE     "RES\\RUND.BMP"
OUTPUTD              BITMAP  DISCARDABLE     "RES\\BITMAP2.BMP"
OUTPUTU              BITMAP  DISCARDABLE     "RES\\BMP00001.BMP"

////////////////////////////////////////////////////////////////////////////
//
// Menu
//

IDR_MAINFRAME MENU PRELOAD DISCARDABLE
BEGIN
```

```
    POPUP "&File"
    BEGIN
        MENUITEM "&New\tCtrl+N",            ID_FILE_NEW
        MENUITEM "&Open...\tCtrl+O",        ID_FILE_OPEN
        MENUITEM SEPARATOR
        MENUITEM "Recent File",             ID_FILE_MRU_FILE1, GRAYED
        MENUITEM SEPARATOR
        MENUITEM "E&xit",                   ID_APP_EXIT
    END
    POPUP "&Edit"
    BEGIN
        MENUITEM "&Clear All",              ID_EDIT_CLEAR_ALL
    END
    POPUP "&View"
    BEGIN
        MENUITEM "&Toolbar",                ID_VIEW_TOOLBAR
        MENUITEM "&Status Bar",             ID_VIEW_STATUS_BAR
    END
    POPUP "&Help"
    BEGIN
        MENUITEM "&About Thesis...",        ID_APP_ABOUT
        MENUITEM "How to Use",              ID_HELP_HOWTO
    END
END

IDR_MENU_HOWTOUSE MENU DISCARDABLE
BEGIN
    POPUP "&File"
    BEGIN
        MENUITEM "&Print",                  ID_FILE_PRINT
        MENUITEM "&Exit",                   ID_FILE_EXIT
    END
    POPUP "&Search"
    BEGIN
        MENUITEM "On subjuct",              ID_SEARCH_ONSUBJUCT
        MENUITEM "On language",             ID_SEARCH_ONLANGUAGE
    END
    MENUITEM "&Help",                   65535
END

IDR_MENU_POP MENU DISCARDABLE
BEGIN
    POPUP "&File"
    BEGIN
        MENUITEM "&Print",                  ID_FILE_PRINT
        MENUITEM "&Exit",                   ID_FILE_EXIT
    END
END


//////////////////////////////////////////////////////////////////////
//
// Accelerator
//

IDR_MAINFRAME ACCELERATORS PRELOAD MOVEABLE PURE
BEGIN
    "N",        ID_FILE_NEW,        VIRTKEY,CONTROL
    "O",        ID_FILE_OPEN,       VIRTKEY,CONTROL
    "S",        ID_FILE_SAVE,       VIRTKEY,CONTROL
    "Z",        ID_EDIT_UNDO,       VIRTKEY,CONTROL
    "X",        ID_EDIT_CUT,        VIRTKEY,CONTROL
    "C",        ID_EDIT_COPY,       VIRTKEY,CONTROL
    "V",        ID_EDIT_PASTE,      VIRTKEY,CONTROL
    VK_BACK,    ID_EDIT_UNDO,       VIRTKEY,ALT
    VK_DELETE,  ID_EDIT_CUT,        VIRTKEY,SHIFT
    VK_INSERT,  ID_EDIT_COPY,       VIRTKEY,CONTROL
    VK_INSERT,  ID_EDIT_PASTE,      VIRTKEY,SHIFT
    VK_F6,      ID_NEXT_PANE,       VIRTKEY
```

```
  VK_F6,        ID_PREV_PANE,        VIRTKEY,SHIFT
END


//////////////////////////////////////////////////////////////////////
//
// Dialog
//

IDD_ABOUTBOX DIALOG DISCARDABLE  34, 22, 236, 103
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "About Thesis"
FONT 8, "MS Sans Serif"
BEGIN
  DEFPUSHBUTTON  "OK",IDOK,195,7,32,14,WS_GROUP
  LTEXT        "The simulation case is about a simple workshop.",
           IDC_STATIC,8,10,147,20
  LTEXT        "Objects:  machines( break down from time to time)",
           IDC_STATIC,8,33,158,8
  LTEXT        " adjusters ( repair the broken down machine )",
           IDC_STATIC,35,44,146,8
  LTEXT        "manager ( arrange the repairment of machines )",
           IDC_STATIC,38,54,146,8
  LTEXT        "scheduler ( schedule the event )",IDC_STATIC,37,65,104,
           8
  LTEXT        "Model:  m/m/a/fcfs/b/b  ( a: number of machines, b: number of adjusters)",
           IDC_STATIC,7,80,225,8
END

IDD_THESIS_FORM DIALOG DISCARDABLE  0, 0, 353, 211
STYLE WS_CHILD
FONT 8, "MS Sans Serif"
BEGIN
  PUSHBUTTON     "RUN",IDC_ENTER,256,142,43,14
  GROUPBOX       "Simulation Models",IDC_STATIC,35,31,196,134
  LTEXT        "Sample Simulation Window",IDC_STATIC,80,13,92,8
  CONTROL        "Multi-server Queue Model with Infinite Population",
           IDC_RADIO1,"Button",BS_AUTORADIOBUTTON | WS_GROUP,54,57,
           170,10
  CONTROL        "Multi-server Queue Model with Finite Population",
           IDC_RADIO2,"Button",BS_AUTORADIOBUTTON,54,74,169,10
  CONTROL        "Inventory Model",IDC_RADIO3,"Button",BS_AUTORADIOBUTTON,
           54,91,113,10
  CONTROL        "Time-shared System Model",IDC_RADIO4,"Button",
           BS_AUTORADIOBUTTON,54,108,113,10
  CONTROL        "Job Scheduling Model",IDC_RADIO5,"Button",
           BS_AUTORADIOBUTTON,54,125,113,10
  PUSHBUTTON     "Server Setup",IDC_BUTTON_SERVER,248,87,60,14
  PUSHBUTTON     "Simulation Model ",IDC_BUTTON_SIMU,248,53,60,14
  PUSHBUTTON     "Other Setup",IDC_BUTTON_OTHER,248,104,60,14
  PUSHBUTTON     "Output",IDC_BUTTON_OUTPUT,248,121,60,14
  PUSHBUTTON     "Customer Setup",IDC_BUTTON_CLIENT,248,70,60,14
END

IDD_DIALOG_OTHER DIALOG DISCARDABLE  100, 50, 228, 136
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CAPTION "OTHER INFOMATION"
FONT 8, "MS Sans Serif"
BEGIN
  DEFPUSHBUTTON  "OK",IDOK,83,105,50,14
  LTEXT        "Seed",IDC_STATIC,15,21,84,8
  LTEXT        "Length of preliminary run",IDC_STATIC,15,40,84,8
  EDITTEXT       IDC_EDIT_SEED,147,16,40,13,ES_AUTOHSCROLL
  EDITTEXT       IDC_EDIT_STEP,147,35,40,13,ES_AUTOHSCROLL
  EDITTEXT       IDC_EDIT_CALENDARSIZE,147,54,40,13,ES_AUTOHSCROLL
  LTEXT        "Length of Calendar",IDC_STATIC,15,59,84,8
END

IDD_DIALOG_CLIENT DIALOG DISCARDABLE  100, 50, 238, 150
```

```
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION
CAPTION "Customer Information"
FONT 8, "MS Sans Serif"
BEGIN
  DEFPUSHBUTTON   "OK",IDOK,178,132,50,14
  COMBOBOX        IDC_COMBO_MACH,103,65,91,66,CBS_DROPDOWN | CBS_SORT |
          WS_VSCROLL | WS_TABSTOP
  LTEXT           "Distribution of time before customer need another service",
          IDC_STATIC,30,58,67,31
  LTEXT           "Number of Machine",IDC_STATIC,30,39,65,8
  EDITTEXT        IDC_EDIT_NUM_MACH,104,37,91,13,ES_AUTOHSCROLL
  LTEXT           "Mean of the distribution",IDC_STATIC,29,98,49,23
  EDITTEXT        IDC_EDIT_MEAN_MACH,103,100,91,13,ES_AUTOHSCROLL
END

IDD_DIALOG_GRAPH DIALOG DISCARDABLE  100, 50, 257, 193
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CAPTION "Graphic Simulaiton Output"
FONT 8, "MS Sans Serif"
BEGIN
  LTEXT           "",IDC_STATIC,91,143,67,8
  LTEXT           "Number of Adjuster",IDC_STATIC,107,177,66,8
  LTEXT           "Server Utilization",IDC_STATIC,2,84,34,23
  PUSHBUTTON      "OK",IDC_BUTTON1,201,177,50,14
END

IDD_DIALOG_SERVER DIALOG DISCARDABLE  100, 50, 233, 144
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION
CAPTION "Server Information"
FONT 8, "MS Sans Serif"
BEGIN
  DEFPUSHBUTTON   "OK",IDOK,94,127,50,14
  LTEXT           "Minimun Adjuster",IDC_STATIC,14,30,72,8
  LTEXT           "Distribution of adjuster's repair time",IDC_STATIC,15,
          46,64,25
  LTEXT           "Mean of adjuster's repair time",IDC_STATIC,14,76,64,21
  EDITTEXT        IDC_EDIT_ADJ_NUM,94,25,98,13,ES_AUTOHSCROLL
  COMBOBOX        IDC_COMBO_ADJ,95,50,99,69,CBS_DROPDOWN | CBS_SORT |
          WS_VSCROLL | WS_TABSTOP
  EDITTEXT        IDC_EDIT_ADJ_MEAN,95,79,102,13,ES_AUTOHSCROLL
  EDITTEXT        IDC_EDIT_ADJ_NUM1,94,47,100,13,ES_AUTOHSCROLL |
          ES_READONLY | NOT WS_VISIBLE
END

IDD_DIALOG_OUTPUT DIALOG DISCARDABLE  0, 0, 222, 111
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CAPTION "Output "
FONT 8, "MS Sans Serif"
BEGIN
  DEFPUSHBUTTON   "OK",IDOK,157,91,50,14
  CONTROL         "Graphic Output",IDC_RADIO_GRAPHIC,"Button",
          BS_AUTORADIOBUTTON,27,26,73,10
  CONTROL         "Numeric Output",IDC_RADIO_NUMERIC,"Button",
          BS_AUTORADIOBUTTON,26,46,74,10
END

IDD_DIALOG_SIMULATION DIALOG DISCARDABLE  100, 0, 210, 200
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CAPTION "Simulation"
FONT 8, "MS Sans Serif"
BEGIN
  DEFPUSHBUTTON   "OK",IDOK,71,183,50,14
  GROUPBOX        "Static",IDC_STATIC,13,11,73,45
  CONTROL         "Prescriptive",IDC_CHECK_PRESCRIPTIVE,"Button",
          BS_AUTOCHECKBOX | WS_TABSTOP,21,25,54,10
  CONTROL         "Descriptive",IDC_CHECK_DESCRIPTIVE,"Button",
          BS_AUTOCHECKBOX | WS_TABSTOP,21,39,54,10
  GROUPBOX        "Static",IDC_STATIC,105,11,73,45
  CONTROL         "Descrete",IDC_CHECK_DESCRETE,"Button",BS_AUTOCHECKBOX |
```

```
            WS_TABSTOP,110,24,53,10
  CONTROL       "Continuous",IDC_CHECK_CONTINUOUS,"Button",
            BS_AUTOCHECKBOX | WS_TABSTOP,110,39,53,10
  GROUPBOX      "Static",IDC_STATIC,13,68,73,45
  CONTROL       "Probabilistic",IDC_CHECK_PROBABILISTIC,"Button",
            BS_AUTOCHECKBOX | WS_TABSTOP,21,83,54,10
  CONTROL       "Deterministic",IDC_CHECK_DETERMINISTIC,"Button",
            BS_AUTOCHECKBOX | WS_TABSTOP,21,97,54,10
  GROUPBOX      "Static",IDC_STATIC,105,68,73,45
  CONTROL       "Static",IDC_CHECK_STATIC,"Button",BS_AUTOCHECKBOX |
            WS_TABSTOP,110,81,53,10
  CONTROL       "Dynamic",IDC_CHECK8_DYNAMIC,"Button",BS_AUTOCHECKBOX |
            WS_TABSTOP,110,95,53,10
  GROUPBOX      "Static",IDC_STATIC,13,129,73,45
  GROUPBOX      "Static",IDC_STATIC,105,129,73,45
  CONTROL       "Open Loop",IDC_CHECK_OPENLOOP,"Button",BS_AUTOCHECKBOX |
            WS_TABSTOP,21,143,54,10
  CONTROL       "Close Loop",IDC_CHECK_CLOSELOOP,"Button",
            BS_AUTOCHECKBOX | WS_TABSTOP,21,158,54,10
  CONTROL       "Terminating",IDC_CHECK_TERMINATING,"Button",
            BS_AUTOCHECKBOX | WS_TABSTOP,110,143,53,10
  CONTROL       "Nonterminating",IDC_CHECK_NONTERMINATING,"Button",
            BS_AUTOCHECKBOX | WS_TABSTOP,110,156,62,10
END

IDD_DIALOG_NUMERIC DIALOG DISCARDABLE  100, 50, 213, 172
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CAPTION "Numeric Simulation Output"
FONT 8, "MS Sans Serif"
BEGIN
  DEFPUSHBUTTON  "OK",IDOK,61,153,50,14
  LISTBOX       IDC_LIST_MACH_UTIL,49,38,55,100,LBS_NOINTEGRALHEIGHT |
            WS_TABSTOP
  LTEXT         "Machine Utilization",IDC_STATIC,53,24,62,8
  LTEXT         "Adjuster Utilization",IDC_STATIC,137,26,64,8
  LISTBOX       IDC_LIST_ADJ_UTIL,135,40,55,101,LBS_NOINTEGRALHEIGHT |
            WS_TABSTOP
  GROUPBOX      "Statistic collections",IDC_STATIC,37,4,166,142
  LTEXT         "Number of Adjuster increase",IDC_STATIC,2,38,29,38
END


////////////////////////////////////////////////////////////////////////
//
// Dialog Info
//

IDD_DIALOG_CLIENT DLGINIT
BEGIN
   1028, 0x403, 12, 0
0x7845, 0x6f70, 0x656e, 0x746e, 0x6169, 0x006c,
   1028, 0x403, 7, 0
0x6f4e, 0x6d72, 0x6c61, "\000"
   1028, 0x403, 8, 0
0x6542, 0x6e72, 0x6c75, 0x0069,
   1028, 0x403, 11, 0
0x7254, 0x6169, 0x676e, 0x6c75, 0x7261, "\000"
   1028, 0x403, 10, 0
0x6547, 0x6d6f, 0x7465, 0x6972, 0x0063,
   0
END

IDD_DIALOG_SERVER DLGINIT
BEGIN
   1032, 0x403, 12, 0
0x7845, 0x6f70, 0x656e, 0x746e, 0x6169, 0x006c,
   1032, 0x403, 7, 0
0x6f4e, 0x6d72, 0x6c61, "\000"
   1032, 0x403, 11, 0
```

```
0x7254, 0x6169, 0x676e, 0x6c75, 0x7261, "\000"
  1032, 0x403, 10, 0
0x6547, 0x6d6f, 0x7465, 0x6972, 0x0063,
  1032, 0x403, 8, 0
0x6542, 0x6e72, 0x6c75, 0x0069,
  0
END


////////////////////////////////////////////////////////////////////////
//
// String Table
//

STRINGTABLE PRELOAD DISCARDABLE
BEGIN
  IDR_MAINFRAME        "Simulation Envirment"
  IDR_THESISTYPE       "\nThesis\nThesis Document\n\n\nThesis.Document\nThesis Document"
END

STRINGTABLE PRELOAD DISCARDABLE
BEGIN
  AFX_IDS_APP_TITLE    "Simulation Envirment"
  AFX_IDS_IDLEMESSAGE  "Ready"
END

STRINGTABLE DISCARDABLE
BEGIN
  ID_INDICATOR_EXT     "EXT"
  ID_INDICATOR_CAPS    "CAP"
  ID_INDICATOR_NUM     "NUM"
  ID_INDICATOR_SCRL    "SCRL"
  ID_INDICATOR_OVR     "OVR"
  ID_INDICATOR_REC     "REC"
END

STRINGTABLE DISCARDABLE
BEGIN
  ID_FILE_NEW          "Create a new document"
  ID_FILE_OPEN         "Open an existing document"
  ID_FILE_CLOSE        "Close the active document"
  ID_FILE_SAVE         "Save the active document"
  ID_FILE_SAVE_AS      "Save the active document with a new name"
END

STRINGTABLE DISCARDABLE
BEGIN
  ID_APP_ABOUT         "Display program information, version number and copyright"
  ID_APP_EXIT          "Quit the application; prompts to save documents"
END

STRINGTABLE DISCARDABLE
BEGIN
  ID_FILE_MRU_FILE1    "Open this document"
  ID_FILE_MRU_FILE2    "Open this document"
  ID_FILE_MRU_FILE3    "Open this document"
  ID_FILE_MRU_FILE4    "Open this document"
END

STRINGTABLE DISCARDABLE
BEGIN
  ID_NEXT_PANE         "Switch to the next window pane"
  ID_PREV_PANE         "Switch back to the previous window pane"
END

STRINGTABLE DISCARDABLE
BEGIN
  ID_WINDOW_NEW        "Open another window for the active document"
  ID_WINDOW_ARRANGE    "Arrange icons at the bottom of the window"
```

```
    ID_WINDOW_CASCADE       "Arrange windows so they overlap"
    ID_WINDOW_TILE_HORZ     "Arrange windows as non-overlapping tiles"
    ID_WINDOW_TILE_VERT     "Arrange windows as non-overlapping tiles"
    ID_WINDOW_SPLIT         "Split the active window into panes"
END

STRINGTABLE DISCARDABLE
BEGIN
    ID_EDIT_CLEAR           "Erase the selection"
    ID_EDIT_CLEAR_ALL       "Erase everything"
    ID_EDIT_COPY            "Copy the selection and put it on the Clipboard"
    ID_EDIT_CUT             "Cut the selection and put it on the Clipboard"
    ID_EDIT_FIND            "Find the specified text"
    ID_EDIT_PASTE           "Insert Clipboard contents"
    ID_EDIT_REPEAT          "Repeat the last action"
    ID_EDIT_REPLACE         "Replace specific text with different text"
    ID_EDIT_SELECT_ALL      "Select the entire document"
    ID_EDIT_UNDO            "Undo the last action"
    ID_EDIT_REDO            "Redo the previously undone action"
END

STRINGTABLE DISCARDABLE
BEGIN
    ID_VIEW_TOOLBAR         "Show or hide the toolbar"
    ID_VIEW_STATUS_BAR      "Show or hide the status bar"
END

STRINGTABLE DISCARDABLE
BEGIN
    AFX_IDS_SCSIZE          "Change the window size"
    AFX_IDS_SCMOVE          "Change the window position"
    AFX_IDS_SCMINIMIZE      "Reduce the window to an icon"
    AFX_IDS_SCMAXIMIZE      "Enlarge the window to full size"
    AFX_IDS_SCNEXTWINDOW    "Switch to the next document window"
    AFX_IDS_SCPREVWINDOW    "Switch to the previous document window"
    AFX_IDS_SCCLOSE         "Close the active window and prompts to save the documents"
END

STRINGTABLE DISCARDABLE
BEGIN
    AFX_IDS_SCRESTORE       "Restore the window to normal size"
    AFX_IDS_SCTASKLIST      "Activate Task List"
    AFX_IDS_MDICHILD        "Activate this window"
END


#ifndef APSTUDIO_INVOKED
////////////////////////////////////////////////////////////////////////////
//
// Generated from the TEXTINCLUDE 3 resource.
//
#include "res\thesis.rc2"  // non-App Studio edited resources

#include "afxres.rc"           // Standard components

////////////////////////////////////////////////////////////////////////////
#endif    // not APSTUDIO_INVOKED
```

B-3. Random number classes

```
////////////////////////////////////////////////////////////////////////////////////////////////////
// File of rand.h                                                                                 //
// Purpose : Generates uniform (0,1) random numbers                                               //
// This generator implements the recommendations of Peter W.Lewis for prime                       //
// modulus multiplicative congruential random number generators(PMMCG).                           //
// [Lewis and Orav,1989]                                                                          //
//                                                                                                //
//   The formula is : 2^31-1 = LONG_MAX = 2147483647                                              //
//              seed(i+1) = (multiplier*seed(i)) mod(LONG_MAX)                                    //
//              U(0,1) = seed(i+1) / LONG_MAX;                                                    //
////////////////////////////////////////////////////////////////////////////////////////////////////

#if !defined _RAND_H
#define    _RAND_H 1

typedef unsigned int UINT;
typedef unsigned long ULONG;
#define FALSE 0
#define TRUE 1

const int CAPTIONLENGTH = 32;

class Rand
{
  // -- Prime modulus multiplicative congruential generator -- //

  private:
    static long defaultSeed;       // increments seeds by 1,000,000
    long seed;                                                  // current RNG seed
    long initialSeed;                                           // initial RNG seed
    long multiplier;            // RNG multiplier
    long observations;          // # of times this generator was used
    char* name;                 // statistic name
    char* distName;                                             // generator name
    char* parameterA;                                           // parameter a stored as a string
    char* parameterB;                                           // parameter b stored as a string
    char* parameterC;                                           // parameter c stored as a string
    char* header;                                               // eg. "Uniform(0.0,1.0)"

    void SetDefaultSeed(void);
    long GetDefaultSeed(void);
    void SetInitialSeed(long _seed) { initialSeed = _seed ;} // can only be set once!

    void SelectMultiplierType(UINT _muliplier);
    void SelectMultiplierType1(void) { multiplier = 950706376;}
    void SelectMultiplierType2(void) { multiplier = 742938285;}
    void SelectMultiplierType3(void) { multiplier = 1226874159;}
    void SelectMultiplierType4(void) { multiplier = 62089911;}
    void SelectMultiplierType5(void) { multiplier = 1343714438;}
    void SelectMultiplierType6(void) { multiplier = 630360016;}
    void SelectMultiplierType7(void) { multiplier = 397204094;}
    void SelectMultiplierType8(void) { multiplier = 16807;}

  protected:
    // -- constructors -- //
    Rand( UINT _stream, char* _name = " ");
    Rand( UINT _stream, long _seed, char* _name = " ");

    // -- get U(0,1) random number from generator -- //
    double GetUnif01(void);        // also increments observations
    void AddOneObservation(void)  { observations++; }

    // check seed
    double GetBaseSeed(void);
  public:
    // -- destructor
    virtual ~Rand(void);

    // -- pure virtual, child must return random number -- //
```

```
    virtual double RandValue(void) = 0;

    // -- reset the statistics of the RNG -- //
    void ReInitializeStats(void) { observations = 0;}

    // -- get output information -- //
    const char*  GetName(void) { return (const char* ) name;}
    const char*  GetDistName(void) { return (const char* ) distName;}
    const char*  GetA(void) { return (const char* ) parameterA;}
    const char*  GetB(void) { return (const char* ) parameterB;}
    const char*  GetC(void) { return (const char* ) parameterC;}
    const char*  GetHeader(void);

    void SetName( char* _name);
    void SetDistName( char* _name);
    void SetParameterA_Name ( char* _name);
    void SetParameterB_Name ( char* _name);
    void SetParameterC_Name ( char* _name);

    long GetInitialSeed(void) { return initialSeed;}
    long GetMultiplier(void) { return multiplier;}
    long GetObservations(void) { return observations;}
    long GetSeed(void) { return seed;}
    void SetSeed( long _seed);
};

#endif
```

```
////////////////////////////////////////////////
// File rand.cpp                              //
////////////////////////////////////////////////

#include "stdafx.h"
#include "thesis.h"

#include "repair.h"
#include "thesidoc.h"
#include "thesivw.h"

#if !defined ( _RAND_H )
#include"rand.h"
#endif

#include<math.h>
#include<string.h>

#define tempa 147483647
long Rand::defaultSeed = 0;          // static to entire class which means every instance

// -- constructors -- //
Rand::Rand( UINT _multiplierType, char* _name)
  : seed(0),
    initialSeed(0),
    multiplier(0),
    observations(0),
    name(NULL),
    distName(NULL),
    parameterA(NULL),
    parameterB(NULL),
    parameterC(NULL),
    header(NULL)
  {
     SetName(_name);
     SetDistName("-");
     SetParameterA_Name("-");
     SetParameterB_Name("-");
     SetParameterC_Name("-");

     SelectMultiplierType( _multiplierType);
     SetDefaultSeed();
  }

Rand::Rand( UINT _multiplier, long _seed, char* _name)
  : seed(0),
    initialSeed(0),
    multiplier(0),
    observations(0),
    name(NULL),
    distName(NULL),
    parameterA(NULL),
    parameterB(NULL),
    parameterC(NULL),
    header(NULL)
  {
     SetName(_name);
     SetDistName("-");
     SetParameterA_Name("-");
     SetParameterB_Name("-");
     SetParameterC_Name("-");

     SelectMultiplierType( _multiplier);
     SetSeed(_seed);
  }

Rand::~Rand(void)
  {
    if(name)
```

```
    delete name;
  if(distName)
    delete distName;
  if(parameterA)
    delete parameterA;
  if(parameterB)
    delete parameterB;
  if(parameterC)
    delete parameterC;
  if(header)
    delete header;
}

const char* const Rand::GetHeader(void)
{
  if(header)
    return header;

  // -- make the header string from the distName and parameter? strings
  UINT strlength = strlen(distName);
  strlength += strlen(parameterA);
  strlength += strlen(parameterB);
  strlength += strlen(parameterC);

  header = new char[strlength + 1 + 4];
  strcpy(header,distName);
  strcat(header,"(");
  strcat(header, parameterA);
  strcat(header,",");
  strcat(header,parameterB);
  strcat(header,",");
  strcat(header,parameterC);
  strcat(header, "0");

  return header;
}

void Rand::SetName( char* _name)
{
  if( name )
  {
    delete name;
  }

  if( _name )
  {
    name = new char[ strlen( _name ) + 1];
    strcpy( name, _name );
  }

  else
  {
    name = NULL;
  }
}

void Rand::SetDistName( char* _distName)
{
  if( distName )
  {
    delete distName;
  }

  if( _distName )
  {
    distName = new char[ strlen( _distName ) + 1];
    strcpy( distName, _distName );
  }
```

```
    else
    {
      distName = NULL;
    }
  }

void Rand::SetParameterA_Name( char* _parameterA )
  {
    if( parameterA )
    {
      delete parameterA;
    }

    if( parameterA )
    {
      parameterA = new char[ strlen( _parameterA ) + 1];
      strcpy( parameterA, _parameterA );
    }

    else
    {
      parameterA = NULL;
    }
  }

void Rand::SetParameterB_Name( char* _parameterB )
  {
    if( parameterB )
    {
      delete parameterB;
    }

    if( parameterB )
    {
      parameterB = new char[ strlen( _parameterB ) + 1];
      strcpy( parameterB, _parameterB );
    }

    else
    {
      parameterB = NULL;
    }
  }

void Rand::SetParameterC_Name( char* _parameterC )
  {
    if( parameterC )
    {
      delete parameterC;
    }

    if( parameterC )
    {
      parameterC = new char[ strlen( _parameterC ) + 1];
      strcpy( parameterC, _parameterC );
    }

    else
    {
      parameterC = NULL;
    }
  }

  // -- get random number from generator -- //
double Rand::GetUnif01( void )
  {
    // -- return a U( 0,1 ) -- //
    seed = fmod((( double ) multiplier * ( double ) seed), ( double ) tempa );
    return ( double ) seed / ( double ) tempa;
```

```
}

void Rand::SetDefaultSeed( void )
{
  seed = GetDefaultSeed();
  initialSeed = seed;          // the user still has a chance to reset the initial
                               // seed by calling SetSeed( long _seed );
  return;
}

long Rand::GetDefaultSeed( void )
{
  // -- since defaultSeed is static, it is shared by all classes
  // -- and hence all instances
   if ( defaultSeed >= 2145000000 )
   defaultSeed = 0;

   defaultSeed += 1000000;

   return defaultSeed;
}

void Rand::SetSeed( long _seed )
{
  if( _seed <= 0 || _seed >= tempa )
  {
    SetDefaultSeed();
  }

  else
  {
    seed = _seed;
  }

  SetInitialSeed( seed );

  return;
}

void Rand::SelectMultiplierType( UINT _multiplier )
{
  switch( _multiplier )
  {
    case 2:
      SelectMultiplierType2();
      break;
    case 3:
      SelectMultiplierType3();
      break;
    case 4:
      SelectMultiplierType4();
      break;
    case 5:
      SelectMultiplierType5();
      break;
    case 6:
      SelectMultiplierType6();
      break;
    case 7:
      SelectMultiplierType7();
      break;
    case 8:
      SelectMultiplierType8();
      break;
    default:
      SelectMultiplierType1();
  }

  return;
```

```
  }

double Rand::GetBaseSeed(void)
{
  return seed;
}
```

```
////////////////////////////////////////////////////////////////////////////////////////////////////
// file randunif.h
// generates uniform( 0,1 ) random numbers or uniform( lowerbound, upperbound )
//   random numbers where lowerBound < upperBound

/**************************************************************************
 Parameters: The minimum and maximum value for the distribution specified as
         real numbers
 Range:     [ lowerBound, upperbound ]
 Mian:      (lowerBound + upperBound) / 2
 Var:       (( upperbound - lowerbound ) ^ 2 ) / 12
**************************************************************************/
////////////////////////////////////////////////////////////////////////////////////////////////////

#if !defined _RANDUNIF_H
#define     _RANDUNIF_H 1

#if !defined (_RAND_H)
#include "rand.h"
#endif

class RandUniform:public Rand
{
  private:
    // -- scale the U( 0,1 ) to a new range -- //
    double lowerBound;
    double upperBound;

  public:
    // -- constructors -- //
    RandUniform( char* _name = " " );
    RandUniform( UINT _stream, long _seed, char* _name = " " );
    RandUniform( double _lowerBound, double _upperbound, long _seed,
            char* _name = " " );
    RandUniform( double _lowerBound, double _upperBound, UINT _stream,
            long _seed, char* _name = " " );

    // -- destructor -- //
    virtual ~RandUniform( void ) {}

    // -- return a uniform number -- //
    virtual double RandValue( void );
    double GetSeedValue(void)
};

#endif
```

```
//////////////////////////////////////////////
// File of randunif.cpp                     //
//////////////////////////////////////////////

#if !defined ( _RANDUNIF_H )
#include"randunif.h"
#endif

#include<string.h>
#include<stdio.h>

// -- constructors -- //
RandUniform::RandUniform( char* _name )
  : Rand( 1, _name ),
    lowerBound( 0 ),
    upperBound( 1 )
  {
    SetDistName( "Uniform" );

    // -- set the coefficients -- //
    char temp[ 40 ];
    sprintf( temp, "%g", lowerBound );
    SetParameterA_Name( temp );
    sprintf( temp, "%g", upperBound );
    SetParameterB_Name( temp );
  }

RandUniform::RandUniform( UINT _stream, long _seed, char* _name )
  : Rand( _stream, _seed, _name ),
    lowerBound( 0 ),
    upperBound( 1 )
  {
    SetDistName( "Uniform" );

    // -- set the coefficients -- //
    char temp[ 40 ];
    sprintf( temp, "%g", lowerBound );
    SetParameterA_Name( temp );
    sprintf( temp, "%g", upperBound );
    SetParameterB_Name( temp );
  }

RandUniform::RandUniform( double _lowerBound, double _upperBound, long _seed, char* _name )
  : Rand( 1, _seed, _name ),
    lowerBound( _lowerBound ),
    upperBound( _upperBound )
  {
    SetDistName( "Uniform" );

    // -- set the coefficients -- //
    char temp[ 40 ];
    sprintf( temp, "%g", lowerBound );
    SetParameterA_Name( temp );
    sprintf( temp, "%g", upperBound );
    SetParameterB_Name( temp );
  }

RandUniform::RandUniform( double _lowerBound, double _upperBound, UINT _stream,
                          long _seed, char* _name )
  : Rand( _stream, _seed, _name ),
    lowerBound( _lowerBound ),
    upperBound( _upperBound )
  {
    SetDistName( "Uniform" );

    // -- set the coefficients -- //
    char temp[ 40 ];
    sprintf( temp, "%g", lowerBound );
    SetParameterA_Name( temp );
```

```
    sprintf( temp, "%g", upperBound );
    SetParameterB_Name( temp );
  }

double RandUniform::RandValue( void )
{
  AddOneObservation();
  return( lowerBound + ( upperBound - lowerBound ) * GetUnif01() );
}


// function to check seed

double RandUniform::GetSeedValue(void)
{
  double temp;
  temp = GetBaseSeed();
  return temp;
}
```

```
////////////////////////////////////////////////////////////////////////
// File of randexp.h                                                  //
// Generates exponential random numbers                               //
// Parameters: mean is the rate parameter and is > 0                  //
// Range:     [ 0, +infinity ]                                        //
// Mean:      Beta( mean )                                            //
// Var:       bate^2                                                  //
////////////////////////////////////////////////////////////////////////


#if !defined ( _RAND_H )
#include "rand.h"
#endif

class RandExponential:public Rand
{
  private:
    double mean;       // rate parameter

  public:
    // -- constructor -- //
    RandExponential( double _mean, char* _name = " " );
    RandExponential( double _mean, long _seed, char* _name = " " );
    RandExponential( double _mean, UINT _stream, long _seed, char* _name = " " );

    // -- destructor -- //
    virtual ~RandExponential( void ) {}

    // -- return an exponential random number -- //
    virtual double RandValue( void );
};


//#endif
```

```
//////////////////////////////////////////////////
// File of randexp.cpp                          //
//////////////////////////////////////////////////

#include "stdafx.h"
#include "thesis.h"

#include "repair.h"
#include "thesidoc.h"
#include "thesivw.h"

#include<math.h>
#include<string.h>
#include<stdio.h>

// -- constructor -- //
RandExponential::RandExponential( double _mean, char* _name )
  : Rand( 1, _name ),
    mean( fabs( _mean ))
  {
    SetDistName( "Expon" );

    // -- set the coefficients -- //
    char temp[40];
    // printf( temp, "%g", mean );
    SetParameterA_Name( temp );
  }


RandExponential::RandExponential( double _mean, UINT _stream, long _seed, char* _name )
  : Rand( _stream, _seed , _name ),
    mean( fabs( _mean ))
  {
    SetDistName( "Expon" );

    // -- set the coefficients -- //
    char temp[40];
    // printf( temp, "%g", mean );
    SetParameterA_Name( temp );
  }


// -- return an exponential random number --//
double RandExponential::RandValue( void )
  {
    AddOneObservation();
    return ( -mean * log( GetUnif01()));
  }
```