1 Introduction to Computer Simulation

Computer simulation is used to reduce the risk associated with creating new systems or with making changes to existing ones. More than ever, modern organizations want assurance that investments will produce the expected results. For instance, an assembly line may be required to produce a particular number of autos during an eight hour shift. Complex, interacting factors influence operation and so powerful tools are needed to develop an accurate analysis. Over the past few decades, computer simulation software, together with statistical analysis techniques have evolved to give decision makers tools equal to the task. As the world grows more technical and the need for precision becomes more important, the margin for error will continue to shrink. Business, industry, and governments cannot afford to make educated guesses during systems development. For that reason, computer simulation is more important than ever.

1.1 Simulation Defined

Simulation uses a model to develop conclusions providing insight on the behavior of real-world elements being studied. Computer simulation uses the same concept but requires the model be created through computer programming. While this field has grown and flourished with availability of powerful software and hardware, its origins in the desire to forecast future behaviors, run quite deep.

Men and women have attempted to foretell the future since ancient times. Kings employed wizards and soothsayers. Various religions used prophets. Seers such as French apothecary Michel de Nostredame (better known as Nostradamus) became famous with their visions of the future. Others attempted to make predictions based on birth dates and the stars. Crystal balls, bones, and tarot cards were all used as tools to probe the future.

Although this book does not advocate those methods, in the same way modern chemists bear a relationship to the ancient alchemist, the modern simulation practitioner has a relationship with the ancient prophet. Of course, methodologies used by modern simulation analysts bear virtually no similarity with the prediction methods used in ancient times. However, there are common elements. For instance, each sought to remove the risk of a future event or behavior and reduce uncertainty. The prophet tried to accomplish this with the magic available at the time. Today, the simulation analyst uses the modern magic of mathematical principles, experimentation, computer science and statistics.

1.1.1 Computer Simulation's Basic Nature

Computer simulation can be classified as a branch applied mathematics. The use of computer simulation increased due to availability of computing power and improvements in programming languages. Added to this are inherent difficulties or even impossibilities to accurately describe complex real world systems using analytical or purely mathematical models. For these reasons, a tool that can represent these complexities accurately is required. Computer simulation can be broadly defined as:

"Using a computer to imitate the operations of a real world process or facility according to appropriately developed assumptions taking the form of logical, statistical, or mathematical relationships which are developed and shaped into a model."

The result can be manipulated by varying a set of input parameters to help an analyst understand the underlying system's dynamics. The model typically is evaluated numerically over a simulated period of time and data is gathered to estimate real world system characteristics. Generally, the collected data is interpreted with statistics like any experiment.

1.1.2 Computer Simulation Uses

Computer simulation can be an expensive, time consuming, and complicated problem solving technique. Therefore, certain circumstances warrant its use. Situations well suited to its application include the following (Table 1.1):

General Situation	Examples
Real system does not yet exist and building a prototype is cost prohibitive, time-consuming or hazardous.	Aircraft, Production System, Nuclear Reactor
System is impossible to build.	National Economy, Biological System
Real system exists but experimentation is too expensive, hazardous or disruptive to conduct.	Proposed Changes to a Materials Handling System, Military Unit, Transportation System, Airport Baggage Handling System
Forecasting is required to analyze long time periods in a compressed format.	Population Growth, Forest Fire Spread, Urbanization Studies, Pandemic Flu Spread
Mathematical modeling has no practical analytical or numeric solution.	Stochastic Problems, Nonlinear Differential Equations

Table 1.1 Situations Warranting Computer Simulations

1.1.3 General Benefits of Computer Simulation

Using computer simulation for analysis has many advantages over other decision making techniques. Among these advantages are:

1. Allows Experimentation without Disruptions to Existing Systems – In systems that already exist, testing new ideas may be difficult, costly, or impossible. Simulation allows a model to be developed and compared to the system to ensure it accurately reflects current operation. Any desired modifications can be made to the model first, the impact on the system examined, and then a decision to implement the changes in the real world system can be made.

Example: Changing the Line

An automotive assembly line may run 24 hours a day, seven days a week. Shutting the line down, putting in a temporary modification (which may or may not speed up a process), and resuming production would be very expensive.

Example: Adding Equipment

Unless the machinery was already purchased, the process of adding it to the line and trying it firsthand would be impossible.

2. Concept can be Tested Prior to Installation – A computer simulation will allow concepts to be tested prior to the installation of new systems. This testing may reveal unforeseen design flaws and give designers a tool for improvement. If the same flaws were discovered after installation, changes to the system might end up being very costly or even impossible to implement.

Example: Purchase of an Automatic Storage and Retrieval system (ASRS)

Engineers in a medium sized company decide to replace an existing warehouse with an ASRS. After stretching their tight budget to its outer limits, the equipment was procured and installed. Several months of usage revealed the new system was unable to keep up with the demands for pallets being entering and removed from the racks. Their assembly line process began to bog down because material wasn't arriving from storage in a timely fashion. The budget was gone and upper management told manufacturing to live with their decision. The entire situation could have been avoided by simulating the ASRS system prior to purchase.

3. Detection of Unforeseen Problems or Bugs – When a system is simulated prior to installation and found to work in concept, the model is often refined to include finer details. The detailed simulation may reveal unforeseen problems or bugs that may exist in the system's design. By discovering these problems prior to installation, debug time and rework costs can be avoided. In addition, improvements to system operation may be discovered.

Example: Traffic light simulation

Analysts were running data through the model of a new traffic light to be installed near a busy industrial park. Traffic seemed to flow smoothly most of the time but one exception existed. On average, daily traffic flows were heaviest from east to west. However, the largest factory had a shift change at 3:30 PM and that caused traffic to run heavier in the north and south directions. By detecting this fact, analysts were able to produce a recommendation that would lengthen the north and south green light times between 3:00 and 4:00 PM Traffic would run smoother and congestion would be eased.

4. Gain in Knowledge on System – A primary benefit of the simulation process is an increase in overall system knowledge. At the start of a simulation project, especially in modeling of complex systems, knowledge is often dispersed among many different people. Each individual is an expert in his or her particular area. In order to develop a working simulation, all this information needs to be gathered together then woven into a complete picture. This process of bringing all the pieces together ends up being of great value by providing those involved with an education on the system. The simulation analyst ends up being a sleuth seeking out information from various sources to produce a finished picture. In the situation where simulations are conducted on a regular basis, channels for the information gathering process need to be established. This will speed up the process considerably and allow data to flow from individual experts to simulation.
5. Speed in Analysis – After a model has been developed, it is possible to run the simulated system at speeds much greater than would be attainable in the real world. With many simulation languages, multiple experiments can be set up and run, adding to the time savings. A finished model can take anywhere from fractions of seconds to hours of run time to produce results. But these results can represent minutes, hours, days or even years of system time. Many manufacturing plants keep a simulation of their assembly processes on hand. Each morning before starting production, engineers enter anticipated system inputs to their model and predict how long the daily operations will take.
6. Forces System Definition – In order to produce a valid, working model of a system, it is important all aspects of that system be known. If incorrect or incomplete definitions exist, the model will be inaccurate and should not be used as an analysis tool. Therefore, development of a simulation ends up forcing analysts to fully define all parameters pertinent to its operation. If certain facts cannot be determined with complete certainty, a safe guess should be calculated. When the model is run, if this guess is found to be a trouble spot causing a system bottleneck, then a more careful investigation and possibly additional research time is indicated as being necessary.
7. Enhances Creativity – Having a simulation on hand can enhance creativity in the design of a system. For instance, an engineer may conceive two possible solutions for a particular problem on the factory floor. One solution is guaranteed to work but is more expensive. The second solution involves a new technology which is less expensive, but somewhat risky. Without any means of analyzing the two possible courses of action, the more conservative one will be chosen. If a model of the system is available, both potential solutions could be tried and compared. If the less expensive one was found to perform as desired, then the creativity of the engineer could be exercised without the risk of failure. Additionally, other more radical ideas could be tested at the model level with little more ventured than the time of the analyst.

All these simulation advantages have a common thread: the reduction of risk. Simulation is a risk reduction method. Uncertainty is reduced and replaced with certainty about the expected operation of a new system or about the effects of changes to an existing system.

1.1.4 General Limitations of Computer Simulation

Simulation is not a perfect cure-all that works in every case to remove every risk from uncertain decisions. It has limitations and disadvantages. Some of these follow:

Expensive – The creation of a computer model often can be an expensive method of analysis. Although lower priced simulation packages are available, most large scale modeling efforts represent a major investment in training, software, hardware, analysis and development time.

Time Consuming – Modeling does not always produce quick answers to questions. In most cases, data collection, model development, analysis, and report generation will require considerable amounts of time. The simulation process can be sped up through two methods: reduction of detail (scaling) and by using generic code libraries. By reducing the level of detail, general concept questions can be answered much faster. However caution should be exercised when using this approach. Model accuracy can also be affected by eliminating key details. In situations where many similar simulations will be run, a generic simulation, or code library can be created. This reusable resource will prevent reinventing the wheel for each simulation project. This is the main idea behind the use of simulators.

Yields Only Approximate Answers – Discrete event simulation relies on the use of random number generators to provide model input. Since the input has a random element, some uncertainty is also associated with the output. In order to produce meaningful results, statistics must be used as a tool for interpreting output. All outputs are estimates of the true behavior of the system. It is important to recognize this fact and treat simulation results as close approximations and use statistical testing to draw conclusions.

Difficult to Validate – Validation is the process of making sure the computer model accurately represents the system being studied. When the system does not yet exist, this can become a formidable task. The opinions of experts and intuition must be relied on as a means of insuring the model runs in the same way the system will. Whenever judgment and opinion are being used, the possibly of error exists. It often is better to take a conservative point of view and make sure the system's predicted operation is no better than expectations for actual operation. Additionally, the human element in computer simulation can be difficult to accurately capture and model.

Accepted as Gospel – Another problem that can occur over the course of a simulation study is the tendency of users to accept output as gospel. Simulation is a tool used by humans and is subject to any error that a person can make. Output reports should always be subject to rigorous review by the data user. Not only should statistical testing be employed, but common sense should be used as a mechanism for acceptance. Many times problems can be detected by thinking through the process being simulated and the formulation of opinions on what should happen. If output data doesn't appear to be in line with expectations it should be investigated more closely. In this way, a problem may come to light.

1.2 Different Types of Simulation

Computer simulation takes a variety of forms and the term has acquired various meanings in different fields. A related term, often used interchangeably with computer simulation is computer modeling. Generally speaking, computer simulation is a broad term that includes the practice of generating inputs from simulated users and then passing these values into actual computer software (this practice is sometimes called emulation). For example, flight simulators can both simulate flight operations as well as control actual flight software. Computer modeling narrows the scope of application and implies all system aspects are represented using a computer. While this book uses the broader term computer simulation, most of its context relates to computer modeling.

Computer simulations have been used successfully in many fields including engineering, production management, business, the sciences, technology, architecture, entertainment, government, military and logistics/transportation. Several types of computer simulation are commonly studied for use in engineering and business environments. These include continuous, Monte Carlo, discrete event, and agent-based modeling. Although many authorities consider discrete event to be a form of Monte Carlo simulation, they will be evaluated separately since, in practice, each is used uniquely and for different applications.

1.2.1 Continuous Simulation

Continuous simulation is concerned with modeling a set of equations, representing a system, over time. This system may consist of algebraic systems, game theoretic models, statistical models or differential equations set up in such a way as to change continuously to represent the ebb and flow of parameters associated with the system state. An example of a continuous simulation is the model of a four wheel drive suspension system in which the dynamics of running over different terrains could be examined. Continuous simulations are often used in conjunction with CAD (Computer Aided Drafting) systems or within mathematical modeling software packages.

Another example of a continuous simulation is a model of competition between two populations. Biological models of this type are known as predator-prey models. The environment consists of two populations which interact with each other. The predators depend on the prey as a source of food. If the number of predators grows too fast, the prey available will decrease and predators will starve. If the number of predators drops, the number of prey will increase. This relationship can be analyzed with a continuous simulation using partial derivatives. The mathematics of this two species system was worked out by the noted theorist, Volterra. He demonstrated that with no outside interference, a fluctuating relationship similar to the graph shown in Figure 1.2 would result.

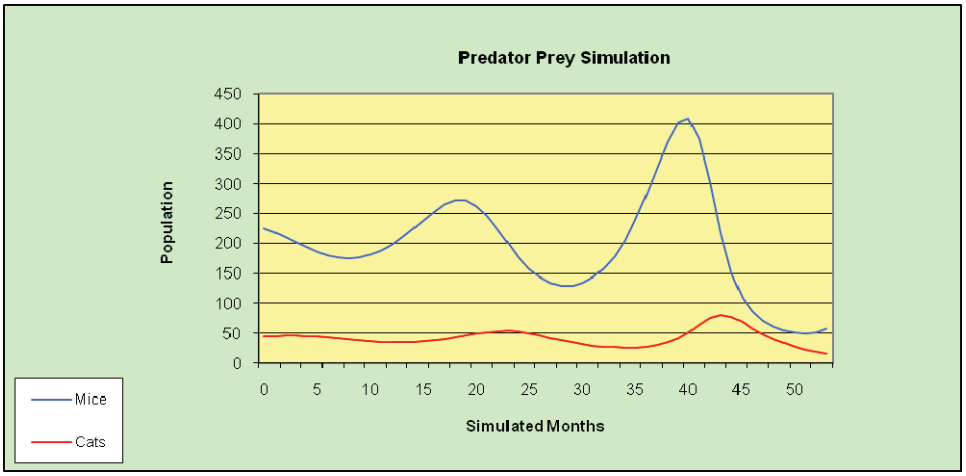

Figure 1.2 Predator Prey Simulation Model

Population growth, urban growth, hurricane predictions, weather forecasting, disease spread and fermentation models are all examples of systems that are suitable candidates for continuous simulation models. The term System Dynamics, first used by Jay W. Forrester in the 1950s, is also used to describe continuous simulation modeling. In general, system dynamics describes system behavior as interrelated, interacting feedback loops, each of which can either directly or indirectly impact another.

Continuous Simulations are commonly developed using spreadsheets, specialized mathematics software such as MATLAB or Mathematica, specialized modeling software like MATLAB's Simulink, or are developed using traditional programming languages like Visual Basic or C++.

1.2.2 Monte Carlo Simulation

The name Monte Carlo invokes thoughts of gambling, gaming and chance. John Von Neumann used the code name Monte Carlo for his experiments, based on the use of random numbers, conducted at Los Alamos during the initial development of the atomic bomb. The name became popular and is now used to represent simulations that are "a scheme employing random numbers, which is used for solving certain stochastic or deterministic problems where the passage of time plays no role" (Law and Kelton, 2000). The last part of this definition (e.g. the passage of time) distinguishes Monte Carlo from discrete event simulation. Monte Carlo generally removes time from the model, whereas discrete event simulation is based on the passage of time. The use of random number generators gives Monte Carlo simulation characteristics not common to continuous simulation.

In the Visual Basic.Net programming language, the RND() function returns a random number which can be used to simulate events which are not completely predictable but occur according to particular probabilities. For instance, consider the following paintball competition simulation written in the form of a Monte Carlo simulation.

This model predicts the numbers of Red and Blue team members to survive a match subject to the assumptions listed in Table 1.2.

1) The match is conducted by long range. Any paintballer can hit a randomly selected opponent with equal ease.
2) No two paintballers ever select the same target.
3) The match is continued until one side is completely wiped out or until 100 paintballs are fired.
4) Every paintball that hits an opposing team member removes them from the competition.
5) In the context of a Monte Carlo simulation, the entire match takes place in zero simulated time.
6) The probability of "which side shoots a paintball next" is based on the percentages of Red and Blue team members who are currently active and part of the match.
7) Initial team sizes are varied and recorded.
8) Average ending team sizes are based on 1000 matches.

Table 1.2 Monte Carlo Simulation Assumptions

Figure 1.3 provides a screen capture of the user interface for the Monte Carlo simulation of the paintball match. The source code, written in Visual Basic.Net, is shown in Figure 1.4.

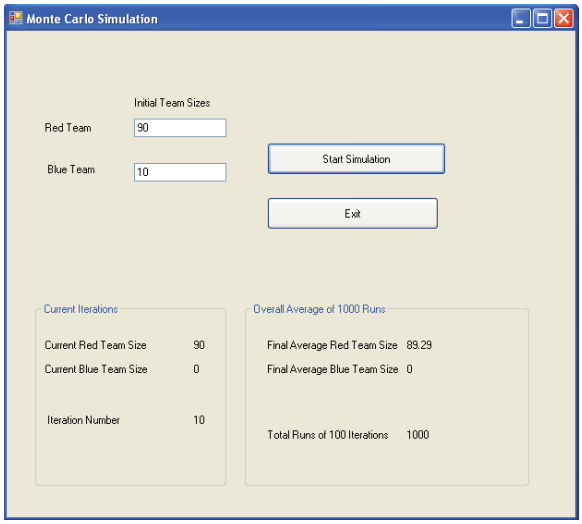

Figure 1.3 User Interface for Paintball Match Monte Carlo Simulation

```
Button1                                                    Click
Public Class Form1

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
        Dim BlueTeamSize As Decimal
        Dim RedTeamSize As Decimal
        Dim PercentPlayersLeft As Decimal
        Dim RandomDraw As Decimal
        Dim OverallAverageReds As Decimal = 0
        Dim OverallAverageBlues As Decimal = 0
        Dim Runs As Integer
        Dim NumberIterations As Integer
        Dim NumberRunsDesired As Integer = 1000

        For Runs = 1 To NumberRunsDesired
            NumberIterations = 0
            RedTeamSize = Convert.ToInt32(TextBox1.Text)
            BlueTeamSize = Convert.ToInt32(TextBox2.Text)
            Do While NumberIterations < 100 And RedTeamSize > 0 And BlueTeamSize > 0
                PercentPlayersLeft = RedTeamSize / (RedTeamSize + BlueTeamSize)
                RandomDraw = Rnd()
                If RandomDraw > PercentPlayersLeft Then
                    RedTeamSize = RedTeamSize - 1
                Else
                    BlueTeamSize = BlueTeamSize - 1
                End If
                NumberIterations = NumberIterations + 1
                Label9.Text = NumberIterations.ToString
                Label5.Text = RedTeamSize.ToString
                Label7.Text = BlueTeamSize.ToString
            Loop
            OverallAverageReds = (RedTeamSize + OverallAverageReds)
            OverallAverageBlues = (BlueTeamSize + OverallAverageBlues)
        Next
        Label12.Text = Math.Round((OverallAverageReds / Runs), 2).ToString
        Label13.Text = Math.Round((OverallAverageBlues / Runs), 2).ToString
        Label15.Text = NumberRunsDesired.ToString

    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click
        Me.Close()
    End Sub
End Class
```

Figure 1.4 Source Code for Paintball Match Monte Carlo Simulation

The model was run for 1,000 matches for each of the scenarios depicted in Table 1.3. The greater the number of runs, the closer the number of remaining players would converge to an exact solution. The value of this type of simulation is that a few minutes work on a computer gives an answer that would be difficult to evaluate in a real life situation. Another advantage inherent in this simulation is the ease with which parameters can be altered and different experiments run.

Initial Red Team Members	Initial Blue Team Members	Red Team Members Remaining	Blue Team Members Remaining
60	10	89.29	0.00
80	20	77.28	0.00
70	30	62.86	0.00
60	40	43.95	0.00
50	50	9.74	9.65
40	60	0.00	43.77
30	70	0.00	62.83
20	80	0.00	77.32
10	90	0.00	89.30

Table 1.3 Paintball Match Simulation Remaining Team Members

1.2.3 Discrete Event Computer Simulation

Discrete Event Simulation is characterized by the passage of blocks of time during which nothing happens, punctuated by events which change the state of the system. An example to illustrate this is a simple queuing system consisting of bank customers arriving at an automatic teller machine (ATM). Customers arrive, wait for service if the machine is in use, receive service and then depart. The following assumptions can be made:

1. Arriving customers wait in front of the ATM (if it is in use) in a single queue.

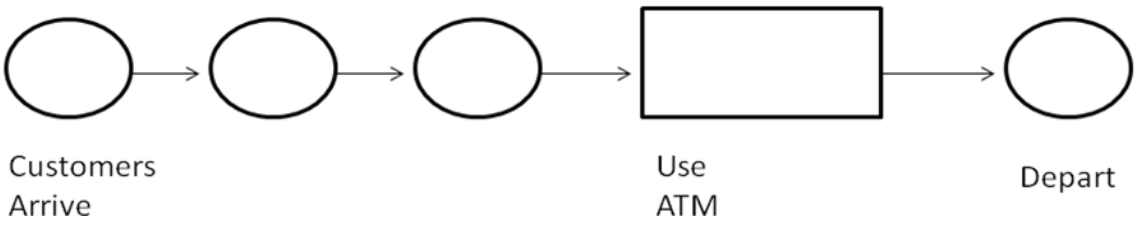

Figure 1.5 ATM Queuing System

2. The time between customer arrivals is as follows:

1 min	5% of arrivals	6 min	20% of arrivals
2 min	7% of arrivals	7 min	10% of arrivals
3 min	8% of arrivals	8 min	8% of arrivals
4 min	10% of arrivals	9 min	7% of arrivals
5 min	20% of arrivals	10 min	5% of arrivals

Figure 1.6 Arrival Times

1. The time ATM service takes is as follows:

1 min	10% of arrivals	4 min 25% of arrivals
2 min	25% of arrivals	5 min 10% of arrivals
3 min	30% of arrivals	

Figure 1.7 Service Times

2. The simulation is written in GPSS World which is a discrete event simulation language from Minuteman Software (http://www.minutemansoftware.com/).
3. The purpose of this simulation is to determine maximum customer queue length, average service wait time, and percent time the ATM machine is in use.
4. The simulation will be run for 1000 hours of simulated time.

The GPSS source code is shown in Figure 1.8. Many of the model statements may seem cryptic but in a later chapter, model construction will be discussed in detail.

```
* Sample ATM Simulation Roger McHaney
************************************************************************
* This function represents customer arrival times
************************************************************************
100    ARRIVE FUNCTION   RN2,D10
0.05,1/0.12,2/0.20,3/0.30,4/0.50,5/0.70,6/0.80,7/0.88,8/0.95,9/1.0,10
************************************************************************
* This function represents ATM use times
************************************************************************
200    SERVE  FUNCTION   RN3,D5
0.10,2/0.35,3/0.55,4/0.80,5/1.0,6
************************************************************************
* First customer is created then used to 'create' additional customers*
* according to timing specified in 'Arrive' Function                   *
************************************************************************
300        GENERATE   1,0,1,1
400 CREATE ADVANCE    FN$ARRIVE
500        SPLIT      1,ATM
600        TRANSFER   ,CREATE
************************************************************************
* Queuing and Use of ATM Machine is modeled in this segment           *
************************************************************************
700 ATM    QUEUE      WAIT
800        SEIZE      MACHINE
900        DEPART     WAIT
1000       ADVANCE    FN$SERVE
1100       RELEASE    MACHINE
************************************************************************
* Customer leaves system                                              *
************************************************************************
1120       TERMINATE
************************************************************************
* Model runs and timing are regulated                                 *
************************************************************************
1130       GENERATE   60
1140       TERMINATE  1
1150       START      1000
```

Figure 1.8 GPSS Simulation Source Code

Figure 1.9 provides a listing of the simulation's output. Even if you have never seen a simulation output before, many of the statistics are easily interpreted. For instance, the queue statistics reveal a maximum waiting line size of 5 customers. The average wait for the ATM machine was 1.381 minutes. The ATM machine was in use 76.3 percent of the time. The average length of the customer waiting line was .251 people. The machine was used 10898 times and 10899 customers entered the system. Of course, a simulation such as this can be based on actual customer arrival rates and service times observed at an ATM machine.

```
GPSS World Simulation Report - Untitled Model 1.8.1
                  Tuesday, June 09, 2009 11:12:31

           START TIME           END TIME  BLOCKS  FACILITIES  STORAGES
                0.000          60000.000    12        1          0


              NAME                       VALUE
          ARRIVE                      10000.000
          ATM                             5.000
          CREATE                          2.000
          MACHINE                     10003.000
          SERVE                       10001.000
          WAIT                        10002.000

 LABEL              LOC  BLOCK TYPE     ENTRY COUNT CURRENT COUNT RETRY
                    1    GENERATE            1             0       0
CREATE              2    ADVANCE         10899             1       0
                    3    SPLIT           10898             0       0
                    4    TRANSFER        10898             0       0
ATM                 5    QUEUE           10898             0       0
                    6    SEIZE           10898             0       0
                    7    DEPART          10898             0       0
                    8    ADVANCE         10898             0       0
                    9    RELEASE         10898             0       0
                   10    TERMINATE       10898             0       0
                   11    GENERATE         1000             0       0
                   12    TERMINATE        1000             0       0

FACILITY         ENTRIES  UTIL.   AVE. TIME AVAIL. OWNER PEND INTER RETRY DELAY
 MACHINE          10898    0.763       4.203  1        0    0    0     0       0


QUEUE              MAX CONT. ENTRY ENTRY(0) AVE.CONT. AVE.TIME   AVE.(-0) RETRY
 WAIT                5    0  10898   6415     0.251      1.381      3.358   0
```

Figure 1.9 GPSS World Output Statistics

If the simulated ATM utilization percentage was close to 100% a decision could be made to determine if an additional machine is needed. This simulation could also be modified to represent what would happen during peak usage times. For instance, if Friday was payday at a large nearby plant, a surge of users could be expected to use the ATM after receiving their checks. Data representing this surge could be entered into the simulation model and analyzed to assess impact on machine utilization and waiting line times.

1.2.4 Agent-Based Modeling

Agent-based modeling addresses the simultaneous interactions of multiple agents to simulate, recreate, study, and predict complex phenomenon. The concept of agent-based modeling is that an overall behavior emerges through the micro-level interactions of individual agents. The primary assumption is that simple local behaviors generate complex height level behavior. Individual agents are modeled according to individual characteristics and are generally assumed to be rational, acting in their own interests which may be economic or socially derived. The model will use local heuristics and simple decision-making rules that create the larger environment.

Most agent-based models have the following elements:

1. multiple agents modeled and scaled with various levels of detail (granularity)
2. decision-making heuristics and rules
3. adaptive behaviors or learning
4. interaction rules or topology
5. environment for interaction often consisting of constrained resources

1.3 Brief History of Simulation

Generally speaking, simulation evolved from the natural human desire to remove risk from the decision making process. In ancient times, rulers often relied on prophets to foretell the outcome of a military action. In modern times, the same desire is manifested in sophisticated military models used to closely examine and statistically anticipate the outcome of particular actions and maneuvers. Methodologies have changed but the overarching goals remain constant: risk reduction and better decision making.

Early simulation efforts can be traced back to World War II when Jon Von Neumann and Stanislaw Ulam developed Monte Carlo simulation techniques to aid in their understanding and development of the atomic bomb. The modern era of simulation began in earnest during the 1950s when new concepts and methods for creating simulations were implemented with programs in available computer languages such as machine code, assembly language, or FORTRAN. Computer hardware was expensive, scarce, slow, and not always reliable. Also computer languages were not designed for simulation applications. In spite of these environmental shortcomings, the value of computer simulation became apparent. Early modeling techniques were developed and implemented on available hardware and software platforms.

Many efforts proceeded in parallel during the early days of computer simulation. Tocher was among the first to describe the application of computers to sampling experiments. These computer-based sampling experiments are now considered to be among the first true computer simulations. In the early 1960s, Geoffrey Gordon of IBM introduced a computer language called GPSS (General Purpose Simulation System) intended to manage the general overhead associated with running simulations on a computer (e.g. timing mechanisms, resource representation, entities, et cetera). This new language first was used at IBM to analyze complex systems but quickly gained widespread acceptance among various organizations and the military. In 1962, the Rand Corporation announced Harry Markowitz, Bernard Hausner, and Herbert Karr had developed the SIMSCRIPT simulation language. This software was developed as an inventory modeling tool for the United States Air Force. During this same time period, Norwegian scientists Dahl and Nygaard released the SIMULA language which, in addition to being a simulation language, was the first object oriented programming language.

The development of a simulation language industry and the realization that many parallel and similar efforts were taking place across the United States and Europe led to the establishment of workshops, support organization, and conferences intended to communicate simulation progress, reduce redundancy efforts and provide faster advances. In March, 1964 the Workshop on Simulation Languages was held at Stanford University and provided the first formal venue for developers and simulation users to exchange ideas. The need for a regular annual conference was apparent and in 1967, the first Winter Simulation Conference was held. By 1968, the Society for Computer Simulation (SCS) had became an official sponsor and gained widespread popularity as a leading organization for simulation practitioners.

Another major simulation language emerged late in the 1960s with the release of the general purpose simulation language GASP. GASP eventually evolved into the SLAM family of languages. In the 1970s, as computing power increased and hardware costs decrease, a wide variety of simulation software products entered the market. In 1977, after IBM corporation discontinued support of GPSS/V, James O. Henriksen announced a new and improved version of GPSS called GPSS/H making GPSS the first multivendor simulation language.

In the early 1980s, the advent of the personal computer led to further developments in the simulation marketplace. Two new major simulation languages were released during this time SLAM by Pritsker Corporation in 1980 and SIMAN by the Systems Modeling Corporation in 1983. Other developers targeted the largely untapped market of easy-to-use industrial simulators. During the 1980s, numerous simulation products were developed and marketed. At the same time, established simulation software companies continued to expand their product lines with animation packages, simulation development toolkits, and enhancements to existing languages. By the 1990s, the simulation market was more commercialized and segmented. More uses emerged. Simulation software fell into eight major categories with numerous offerings in each area (Table 1.4).

Simulation Software Category	Description	Example Products
General Purpose Software	Simulation languages and general software used to write simulation models.	GPSS/H, GPSS/PC, SIMAN, Simula, SLAM, SLX
Manufacturing Oriented Software	Products specifically designed for use in the analysis of manufacturing and production systems.	ProModel, AutoMod, WITNESS, ShowFlow 2.5
Planning & Scheduling Software	General purpose and specific manufacturing software tools are often used for planning and scheduling but separate software products supporting this area has emerged.	Simul8 Planner, AutoSched
Special Purpose Modeling Software / Simulators	Other specialty area simulation packages concentrate on specific areas like communications, health care, manufacturing, service industries, education, and so forth.	MedModel, a medical service environment simulation system; ServiceModel, a service industry simulation package for use with banks, schools, offices and other applications; ns-3, a network simulator
Simulation Environments	Simulation environments contain many utilities to conduct a simulation study. These capabilities include input data analysis, model entry support, scenario management, animation, and output data analysis.	Arena, GPSS/World
Animators	Animation software allows the simulation to be dynamically displayed on the screen of a computer using a graphic format.	Wolverine Software Corporation's PROOF; Arena integrates animation with underlying simulation software
Rapid Modeling Tools	Rapid modeling or rough cut modeling tools are used to develop quick models or perform feasibility studies prior to embarking on a full blown modeling effort.	Spreadsheet software is most commonly used or broadly scaled models are developed with simulation languages or simulators Manuplan and SimStarter are examples of software packages that were developed for this function but are no longer sold
Simulation Support Software	Support software includes tools to aid in the simulation process. Among these are tools used for data analysis, distribution determination, and reporting.	ExpertFit SIMSTAT 2.0

Table 1.4 Simulation Software Categories

During the 1990s, simulation product vendors focused on putting tools in the hands of end-users. Software such as AutoMod and Micro Saint gained in popularity with automatic input data collection features, programming free deployment and graphical interfaces. Additionally, web-based simulation emerged. This approach to simulation means programs are developed and executed over the Internet specifically using a web browser. The web increasingly became viewed as an environment for simulation applications.

Another growing area of simulation, agent-based modeling, began gaining popularity during the 1990s and found application in a variety of business, social, and technical areas. Agents-based models were applied to supply chain problems, consumer behavior, social interaction, workforce management, stock market analysis, pandemic group models, traffic patterns, and other areas. Agent-based models tested how changes in local behaviors impact large scale emergent behaviors. Languages such as Swarm and Repast are used in agent-based modeling.

As simulation development moved into the 2000s, the industry continued to grow both in sales and products available. Today, hundreds of simulation products are available with specialty products in numerous areas. A list of these can be found at:

http://www.lionhrtpub.com/orms/surveys/Simulation/Simulation1main.html

1.4 Bibliography

D.T. Brunner, and J.O. Henriksen, "A General Purpose Animator," In Proceedings of the 1989 Winter Simulation Conference, Society for Computer Simulation, San Diego, California, pp. 155–163 (1989).

S. Cox, "GPSS/PC Graphics and Animation," In Proceedings of the 1988 Winter Simulation Conference, Society for Computer Simulation, San Diego, California, 129–135 (1988).

O. Dahl and K. Nygaard, "Simula An ALGOL Based Simulation Language," Communications of the ACM, 9: 9, pp. 671–678 (1963).

G. Gordon, "A General Purpose Systems Simulator," In Proceedings of EJCC, Macmillian, New York, pp. 87–104 (1961).

A.M. Law and W.D. Kelton, Simulation Modeling and Analysis, 3rd Edition, McGraw-Hill Book Company (2000).

Lion Heart Publications. http://www.lionhrtpub.com/orms/orms-8-06/fragent.html

J. McLeod (Editor), Proceedings of the 1988 Conference: Pioneers & Peers, Society for Computer Simulation, San Diego, California (1988).

A.A.B. Pritsker, The GASP IV Simulation Language, John Wiley and Sons, New York (1974).

A.A.B. Pritsker, Introduction to Simulation and SLAM II, Systems Publishing Corporation, West Lafayette, Indiana (1986).

J.O. Henriksen, "An Improved Events List Algorithm," In Proceedings of the 1977 Winter Simulation Conference, Society for Computer Simulation, San Diego, California, pp. 546–557 (1977).

T.J. Schriber, "Perspectives on Simulation Using GPSS," In Proceedings of the 1988 Winter Simulation Conference, Society for Computer Simulation, San Diego, California, pp. 71–84 (1988).

H.M. Markowitz, B. Hausner, and H. Karr, SIMSCRIPT A Simulation Programming Language, Prentice Hall, Englewood Cliffs, New Jersey (1963).

R.W. McHaney, Computer Simulation: A Practical Perspective, Academic Press, San Diego (1991).

R.W. McHaney, "Use cases and personas: uses in service sector simulation development," International Journal of Simulation and Process Modelling (IJSPM), Vol. 4, No. 3/4, pp 264–279 (2008).

R.M. O'Keefe, "What is Visual Interactive Simulation? (And is There a Methodology for Doing it Right?)," In Proceedings of the 1987 Winter Simulation Conference, Society for Computer Simulation, San Diego, California, pp. 461–464 (1987).

J.J. O'Reilly and W.R. Lilegdon, "SLAM II Tutorial," In Proceedings of the 1988 Winter Simulation Conference, Society for Computer Simulation (San Diego, California), 85–89 (1988).

C.D. Pegden, Introduction to SIMAN, Systems Modeling Corporation, State College, Pennsylvania (1987).

C.D. Pegden, R.E. Shannon, and R.P. Sadowski, Introduction to Simulation Using SIMAN, McGraw Hill, New York (1990).

S. Robinson, "Discrete-Event Simulation: From the Pioneers to the Present, What Next?" Journal of the Operational Research Society, 56 619–629 (2005).

S. Robinson, Simulation: The Practice of Model Development and Use, Wiley, Chichester (2004).

J. Rottenbach, "Simulation Software Acts in Other Areas," Managing Automation (January), pp. 44–45 (1991).

D.A. Samuelson, and C.M. Macal, "Agent-Based Simulation Comes of Age: Software opens up many new areas of application," OR/MS Today, August (2006).

D.A. Taylor, Object Oriented information System Planning and Implementation, John Wiley and Sons, New York (1992).

M.B. Thompson, "AutoMod II: The System Builder," In Proceedings of the 1989 Winter Simulation Conference, Society for Computer Simulation, San Diego, California, pp. 235–242 (1989).

K.D. Tocher, "The Application of Automatic Computers to Sampling Experiments," Journal of the Royal Statistical Society (B:16), 39 (1954).

K.D. Tocher, The Art of Simulation, The English Universities Press Limited, London, England (1963).