CHAPTER 94
Simulation Packages

ABE NISANCI
Bradley University

ROBERT E. SCHWAB
Caterpillar Inc.

1. INTRODUCTION

The increased complexity and cost of modern service and manufacturing systems, coupled with the advances in computer and simulation technologies, have led to an extensive use of simulation. Parallel to this development, simulation languages, and products have also proliferated. In the early 1970s, there were only a few special-purpose languages such as SIMSCRIPT, Q-GERT, GASP, GPSS, and DYNAMO, and general-purpose languages such as ALGOL, PL/1, and FORTRAN (Nance 1996; Schwab and Nisanci 1992). Initially, the most significant use of simulation was by the military. Currently, simulation plays a significant role in almost every major service and manufacturing industry throughout the world. The *Directory of Simulation Software* published by the Society for Computer Simulation (Rodrigues 1994), lists 174 simulation products that are available in the marketplace. The annual Buyer's Guide of simulation software (Institute of Industrial Engineers 1999) lists 27 companies providing 40 simulation software products. Twenty-eight simulation products are listed under the categories of general purpose simulation software, manufacturing-oriented software, business process engineering, simulation-based scheduling, animators, simulation support software, and optimization (Banks 1998a). The simulation software market is very dynamic, resulting in many mergers and new companies replacing the existing ones. There are many more products that are in various states of development and use at companies and universities. Most major companies employ at least one simulation specialist on staff or have developed a close relationship with a consulting firm that offers simulation service. Major manufacturers started requiring simulation analysis prior to approving significant capital expenditures as early as the 1980s (Krepchin 1988; Schwab 1987). This soaring interest in simulation and the simultaneous availability of PCs have produced a corresponding growth in simulation companies and products. Simulation products can be divided into two distinct categories: simulation tools and associated/complementary products. Traditionally, simulation tools include languages and simulation packages (sometimes called simulators). However, the difference between languages and packages seems to disappear over time. A simulation package may have all the power of a simulation language. Complementary products are used to support simulation serving as front and back ends.

1.1. Simulation Languages

Simulation languages allow the user to mathematically describe a process using constructs and syntax specifically designed for the analysis of dynamic and stochastic systems (Smith 1987). Early simulation models were coded in general-purpose procedural languages such as FORTRAN. This approach resulted in spending a large percentage of time developing, debugging, and verifying the model. The new languages alleviated most of these problems by significantly eliminating the need for programming. Most modern languages have an internal simulation clock, entity tracking, some form of statistical output, and animation capability. Typically, modern simulation languages have the ability to handle a broad range of problems and are flexible enough to handle complex decision rules.

1.2. Simulation Packages

Simulation packages have been in development for the last 15 years. They are a product of simulation companies' and simulation users' desire for the benefits of simulation and a belief that the simulation process should be accomplished more easily and results produced more quickly. All packages use some programming language as the base language, but some packages use a specific simulation language (e.g., Arena uses SIMAN, and AweSim uses Visual SLAM). All packages are designed to make the model-development phase much easier by limiting the number of constructs, providing a menu-driven format, presenting fill-in menus, and tying simulation and animation together. Today, all simulation packages are designed for specific industries such as manufacturing, communication, supply chain management, and business process improvement.

1.3. Complementary Products

Complementary products include all simulation support software, which are usually developed independently and later incorporated into a package as a front or end use. Some examples consist of curve-fitting programs, input data generators, scenario builders, programs for file management and data handling, graphics and animation builders, and optimization programs. The vendor of the package or the users of the package develop some upon a perceived need, such as specific material-handling extensions or applications. Some of the popular complementary simulation products are (Banks 1998a; Institute of Industrial Engineers 1999):

- ExpertFit (Averill M. Law & Associates): Automatically determines which of 39 probability distributions best represents a data set. It puts the selected distribution into the proper format for 30 simulation products. ExpertFit provides goodness-of-fit tests, more than 30 graphical plots, a distribution viewer, and batch mode operation.
- StatFit (Geer Mountain Software): Fits input data to one of 21 theoretical continuous and discrete distributions and provides relative comparisons between distribution types.
- Proof Animation (Wolverine Software Corporation): Provides animation for discrete event simulation software.

- OptQuest (OptTek Systems): An optimization tool that guides the process of selection of system inputs and then executes the model by running alternative scenarios for each set of inputs in order to enhance system performance (Laguna 1997; Glover and Kelly 1999). OptQuest is available with ARENA, Micro Saint, and Taylor ED.
- SimRunner: Given the input factor levels and system performance measures, SimRunner is employed in two stages. The first stage involves determining the important input factors, and the second stage involves determining the best factor levels to improve the system performance. It is included in ProModel's optimization module (Heflin and Harrel 1998).

2. SELECTION STRATEGY

Even with the tremendous increase in the number of simulation products and simulation companies, the selection of simulation tools need not be difficult if an organized approach is developed. This includes two key elements: the development of a strategy for the use of the products and the knowledge of the characteristics required in the products (Schwab 1987). The strategy has an impact on the features of the tool that is chosen. The features of the tool should be tailored for the people and uses of that tool. Strategy development for the use of the product is essential for the product's benefits to be fully exploited. A good strategy addresses how the tool will be used, who will use the tool, how the users will be trained, and how the users will be supported. In developing such a strategy, it would be useful to establish a team consisting of those who will use, maintain, and provide support for the product.

2.1. Establish the Uses of the Tool

First, the type of application for the tool should be specified. Types of application include:

- One product for one application
- One product for multiple applications
- Multiple products for multiple applications

Typically, individuals charged with investigating simulation tools start by selecting one product for one application. This allows for a fair evaluation of the specific simulation product in an application for which it was designed. This approach also minimizes start-up and training costs.

The one-product-for-multiple-applications approach is particularly helpful in integrating operations separated by distance or function. Selecting one product allows for a common communication tool that can be understood by everyone trained in the tool. In selecting only one tool for all applications, some sacrifice may be made in terms of modeling efficiency. Each product has features that make specific model development easy. As an example, CACI offers a product (COMNET) to design communication systems, and AutoSimulations (AutoMod) makes automatic guided vehicle systems design easy. Selecting only one product allows for standardization, model sharing, and communication but risks some inefficiency in model development.

The multiple-products-for-multiple-projects approach has the opposite advantages and disadvantages of the previous approach. It always allows for the selection of the most appropriate tool for the application. This approach is essential for consulting firms offering simulation services to diverse companies. A broad range of products and abilities is necessary to meet the diverse needs of clients. The disadvantage to this approach is a dilution of simulation personnel. If people must be trained on a number of products, then specialization in any one tool is sacrificed. As more people are cross-trained on various products, training costs increase. However, more flexibility and diversification is achieved.

Second, it is important to have an understanding of the specific use to which the tool will be put. The tool may be used for evaluation purposes such as the analysis of different layout alternatives for a manufacturing system. It may then be used for detailed design to define and fully describe the operational characteristics of the system. The tool may also be used in narrow application such as the design of communication systems or automatic guided vehicle systems. The product itself, however, may be used in broad applications from preliminary design to detailed design.

2.2. Select the Users of the Tool

Users of the simulation tool play an important part in its success. Matching the wrong people to the right tool is just as dangerous as matching the wrong tool to the project. Users of simulation may have a broad educational background and work experience. A flexible approach to the education and experience of the simulation user must be considered in the selection of a product. This also influences training requirements, which are discussed later. If personnel options are limited, then the training program for these individuals must be flexible enough to compensate for their varied backgrounds. On the other hand, if no time or money is available for training, then individuals must be chosen with enough previous training and experience to begin work on simulation projects immediately. Personnel selected for simulation projects should meet several criteria:

- Familiarity with the general process
- Analytical and computer skills
- Knowledge of probability, statistics, and design of experiments

Simulation is not just computer programming. A simulation analyst must be proficient in all of the above areas. Simulation tools are statistical tools; therefore, an understanding of statistics is essential to make effective use of any simulation product.

2.3. Design the Training Program

Even if the software supplier is to be the primary source of training, it is important to establish a comprehensive training plan as an integral part of an overall simulation strategy. The training plan should include the following major areas:

- *Fundamentals:* The fundamental issues related to the use of simulation tools might include training in the simulation process, data collection and preparation, model building, experimentation, and input and output data analysis. All of these areas can be taught by linking them to a specific simulation tool.
- *Tool training:* Training in the use of the tool is critical. Ideally, the training is accomplished over a number of days and includes an exposure to all the features of the tools. Instructors need to provide a wide range of examples and have extensive simulation experience in order to answer the questions of trainees effectively. Trainees should attend simulation classes with specific work-related projects and should begin working on them immediately after training. This helps cement the concepts of the class firmly in the minds of the trainees. Professional training and frequent use represent the only way to become proficient with any simulation tool.
- *Application training:* Many users of simulation products find it difficult to begin to use the product for a specific task even after they have received training in how to manipulate it. It may be necessary to show users how to use the product as it specifically applies to their work. An example of this would be a warehouse engineer who takes simulation training where most examples in the class are from manufacturing. The engineer may find it difficult to return to the job and immediately make application to the warehouse. This difficulty may be overcome by allowing the new simulation person to work with and be quided by an experienced simulation person. Another alternative might be to retain a consulting firm to develop the first model and allow the new simulation person to follow and assist in the entire model-development process. All new users should be taught the use of gradual complexity in the model-building process. They need to start with a simple representation of the process and get the model running. Then they can gradually add more and more complexity until the process has been modeled completely.
- *Programming:* Within the framework of modern simulation tools, what the users need to program today is more like programming an Excel spreadsheet than programming per se. However, some simulation tools provide use of an underlying language such as FORTRAN, Pascal, or C in addition to the syntax for the simulation language. This feature enables the user to program complex decision rules and other logic not capable of being handled by the simulation language itself.
- *Hardware and operating system:* New users may be unfamiliar with the hardware on which the simulation software runs, as well as the operating system utilized on the computer. A short course covering these topics enhances productivity.

2.4. Assess Support Needs

Care should be taken to determine in what manner the users will be supported. Obviously, if there is only one user, then the only reasonable approach is to depend on customer support from the software supplier. However, a site with multiple users presents some opportunities for self-support. As an example, one person at the site could be designated as the simulation support for everyone and could spend a certain number of hours each week learning more details about the simulation tool. This may result in faster problem resolution. Responsiveness to support needs, means of providing support, and geographical proximity of the support services need to be considered in selecting vendors

3. EVALUATING SIMULATION TOOL CHARACTERISTICS

Number of studies exist related to simulation software evaluation criteria and methodologies (Tumay and Harrington 2000; Banks 1998a; Nikoukaran et al. 1998; Hlupic 1997; Hlupic and Paul 1995; Pritsker 1995; Davis and Wiliams 1994; Law and Kelton 1991). The characteristics listed below should not be considered as all-inclusive but should be used as an aid in assembling a list for the software evaluation process. As an example, if the strategy calls for purchasing one package for only one computer, then portability is not a consideration. For a given set of criteria, there may be additional items that should appear on the list. The characteristics of any simulation tool can be divided into six major areas: language, operational, input, output, hardware, and vendor.

3.1. Language Characteristics

Language characteristics include the features used in writing code and providing useful external support tools. Some simulation tools allow the user to write complex rules. All the features in this section deal with writing simulation instructions.

- *Portability:* If the computing environment is made up of different hardware platforms, then the software should be usable on a wide variety of machines, from mainframes to PCs, and operating systems such as OS/2, Windows, and UNIX. Also important are factors such as special graphics card or driver requirements, ability to run on a network, special compiler needs, and memory (RAM) requirements.
- *Readability:* If the system generates or uses some type of computer code, it should be understandable and easily read by people as well as computers. Commands should be "English-like" statements to assist the writer and reviewer in debugging the model. The language should permit the use of comment statements within the code.
- *Documentation:* The documentation, consisting of installation instructions, test models, and user manual, must provide the user with all the information required to run and debug the models. It must also provide a thorough understanding of all aspects of the software and a large number of example problems and their detailed solutions. An extensive and well-organized index to all documents is a necessity.
- *Data requirements:* The software must allow for both deterministic and stochastic input. Sampling from system-supplied distribution functions should include theoretical statistical distributions such as exponential, normal, triangular, uniform, Poisson, beta, gama, erlang, and log-normal. In addition to these, user-defined distribution functions should be permitted.
- *Capability to support different worldviews:* Depending on the characteristics studied, a discrete or continuous change model may be suitable. In discrete change models, the variables of interest may change in a discrete fashion at any point in time or at certain points in time. In continuous change models, the variable of interest changes continuously at any point in time or at certain points in time (Pritsker and O'Reilly 1999). A project may need both discrete and continuous modeling. If so, they must be supported by the same software, or two different pieces of software must be purchased. Continuous modeling can be used with chemical-related processes such as tank filling and emptying and in discrete processes that can be modeled as continuous processes, such as high-speed canning.
- *Algorithms and modeling features:* A library of available algorithms can potentially save a significant amount of time in the model-building process. For example, automatic guided vehicles and automatic storage and retrieval systems modules are extremely useful. Additional features such as cost modeling, scheduling and schedule generation, breakdown generation, and maintenance planning also enhance modeling productivity.
- *Capability to have user-written interface:* The ability to interface with user-written routines permits the development of industry specific routines or decision logic, which can be filed in a library and reused. This allows flexibility and avoids duplication.
- *Compatibility with other software packages:* Integration is becoming more important with software packages. The ability to access other software packages allows the simulation person to work more efficiently and effectively. Interface to spreadsheet products is helpful in the collection and organization of data. An interface with computer-aided design software can be helpful to reduce the layout development time of animation. If a database package has already been chosen, the ability to integrate the simulation software with that package is extremely useful. An ability to interface as well with statistical analysis software adds additional powers and enhances user productivity.
- *Ability to communicate with other applications on a dynamic basis:* The concept of distributed interactive simulation that includes the ability to exchange data with other applications, including

other simulations, is becoming widely popular. For example, Micro Saint and Arena are both using Microsoft COM features that allow their tools to work with other applications that may even reside on different computers.

- *Ease of error recovery:* The software must be able to handle user-generated errors, especially in the interactive mode, and continue to function after receiving such an error.

3.2. Operational Characteristics

Operational characteristics involve the features in which the software interfaces with the user, except for code writing, which was considered above. These characteristics can make a significant contribution to the productivity of the modeler.

- *Applicability:* Ideally, a single piece of software should allow the novice to use the tool with a minimum amount of training and an absence of frustration. It should also allow the expert to use the tool to perform difficult, complicated simulations of a wide spectrum of systems.
- *Availability and use of debugging features:* The software must contain error diagnostics and debugging features, which help the modeler rapidly check out the model code. An interactive debugger with the ability to set stopping points and step through the model is useful in debugging complicated simulations. An important feature in model debugging is the ability to generate a detailed standard or customized trace to determine whether the model operates properly.
- *Ease of use and intuitiveness:* The software should have features to help automate the model-building process. Desirable features might include menu-driven operation, fill-in menus, and prompting for omitted values. The casual user requires online, context-sensitive help in the model-building process. The software should be accessible in an interactive mode.

  The software needs intuitive icons, menu arrangements, and keystroke sequences. Software that requires the user to enter obscure keystrokes to bring up the help screen prevents the user from making intuitive guesses about forgotten operations. Significant amounts of time are then wasted looking through a user's manual. Since most software programs use the F1 key to bring up the help screen, an intuitive system would have some form of help available by pressing the F1 key.
- *Ease of parameter variation:* The software should allow automatic parameter variation over a specified range of values and hence allow the running of multiple replications without requiring user interface. This feature can also aid in performing sensitivity analyses.
- *Ease of model revisions:* The software must allow changes to be made to the existing model quickly and easily. The best solution is rarely chosen first. The ability to change the model, the routing of the entities, and the parameters, and change them quickly, is important.
- *Interactive:* The software should allow changes in the model during the simulation run. Such changes are vital for doing "what-if" analysis with the model, although they invalidate the statistics collected during the run.
- *Availability of checkpoints:* The software should be able to print out data about the model, in tabular or graphical form, at specified intervals before the end of the simulation. This allows a look at intermediate results at critical times such as shift change.
- *Optimization:* The software should also have an optimization feature to guide the user in selecting system inputs and executing the model to enhance system performance.

3.3. Input Characteristics

The simulation model may require input from different sources. The following features may save significant time and effort in preparing the necessary input:

- Ability to read from external files, databases
- Ability to interact with popular spreadsheet, computer-aided design, and process-planning software
- Ability to trigger input functions throughout the simulation process
- Features to define, analyze, and display input in terms of statistical distributions
- Automated definition of model entities, their attributes, and rules for their interaction
- Ability to interact dynamically with other software applications

3.4. Output Characteristics

The results of simulation form the foundation for many conclusions regarding a process under consideration. Therefore, the availability and presentation of output results is important.

- *Reports:* The reports generated by the software must be clear and easy to interpret. The report format should provide graphical output such as simple bar charts and histograms as well as the standard statistical listing. Because all applications of simulation have some differences, the software must enable the user to generate special reports and control the amount, format, and type of output. A list of the standard statistics that may be included in an output report may consist of:
  1. Activity statistics: Average, minimum and maximum duration, standard deviation, entity count, current status
  2. Waiting statistics: Average, minimum and maximum length and wait time, standard deviation, queue current status
  3. Resource statistics: Average and maximum utilization, standard deviation, current status
- *Graphics:* The graphics should display a wide variety of graphs and charts (bar, pie, histogram) in a user-designated format. The ability to relate data across different scenarios helps during the analysis phase.
- *Animation:* Animation allows the user to represent the physical process graphically and describe and visualize the system being modeled. The animation portion should have a standard library of symbols and icons to assist the user in developing a picture of the process. Furthermore, an icon builder helps design shapes that are not included in the standard library. Animation can be helpful in the verification and validation of the model, as a selling tool to management or others not closely involved with the project as an operator-training tool, and as a technique to solicit ideas from experienced workers. Although experienced workers might never have used a computer product and might even feel intimidated if asked to do so, they could be drawn quickly into a discussion about the viability of a process change if they were shown an animated picture.
- *Output analysis assistance:* Automation of output analysis is certainly a characteristic of the new generation of simulation systems. Features that compare or relate data from replications or probable scenarios provide significant support to the user in reaching conclusions. Assistance in steady state analysis, confidence interval estimation, automatic stopping rules, sample size determination, and so on is very desirable.

3.5. Hardware Characteristics

Most organizations have hardware requirements that must be followed when selecting software. Therefore, the purchase of any software must be influenced by the hardware requirements of the organization and the flexibility and limitations of the simulation software. The compatibility of the software with the hardware already in place, file sharing capability between machines and facilities, and the capability to support the output devices already in place need to be considered.

3.6. Vendor Characteristics

The websites of the vendors are important sources of information. Many useful information about the companies and their products can be obtained from these sites. A critical examination of the company that develops the product requires significant attention. The process of acquiring a simulation tool, however, should not be viewed as a simple exchange of product for money. The company developing the product plays an important part in the successful use of the simulation product. Therefore, attention must be paid to certain features of the supplier. These features include:

- *Presence of ongoing software-development efforts:* The company should have an ongoing program of software development for both existing products and new applications. Otherwise, the user risks product obsolescence.
- *Training:* The company supplying the software must provide high-quality training programs that include both onsite and offsite classes. For large installations, training for systems personnel should also be available.
- *User support:* The company must have qualified technical personnel available during working hours to answer software questions. A functional user group with regular meetings is a good source of contacts and information. The user group can provide an avenue to influence the software-development priorities of the simulation company.
- *Company performance:* It is important that the company be established in the modeling and simulation business with an extensively field-tested software package that is in use by a significant number of corporate and academic clients. Nothing can be more frustrating than to have a supplier of simulation software go out of business and leave no one to call for support.
- *Cost/value relationship:* The price must be consistent with the job expected of the software, the features offered, and the training and support supplied with the software. In addition to

licensing, costs related to maintenance and upgrades, training, support, consultancy, and run times also need to be considered.

- *Business history:* Information such as the number of employees, the business focus (products vs. consultancy), experience, the customer base, the financial status, and references provide insight about the success and continuity of the vendor.

3.7. Evaluation Methodology

The abundance of simulation tools and the need to evaluate their numerous characteristics have turned tool selection into a major issue. Based on the May 1998 *IIE Solutions* "Last Word" reader survey on simulation (Garnett 1999), the capabilities of a simulation tool have been ranked in order of importance by 41 participants as follows:

1. Supply chain modeling
2. Ability to import CAD files
3. Customizable programming language
4. Cost modeling and analysis
5. Input distribution fitting and design of experiments

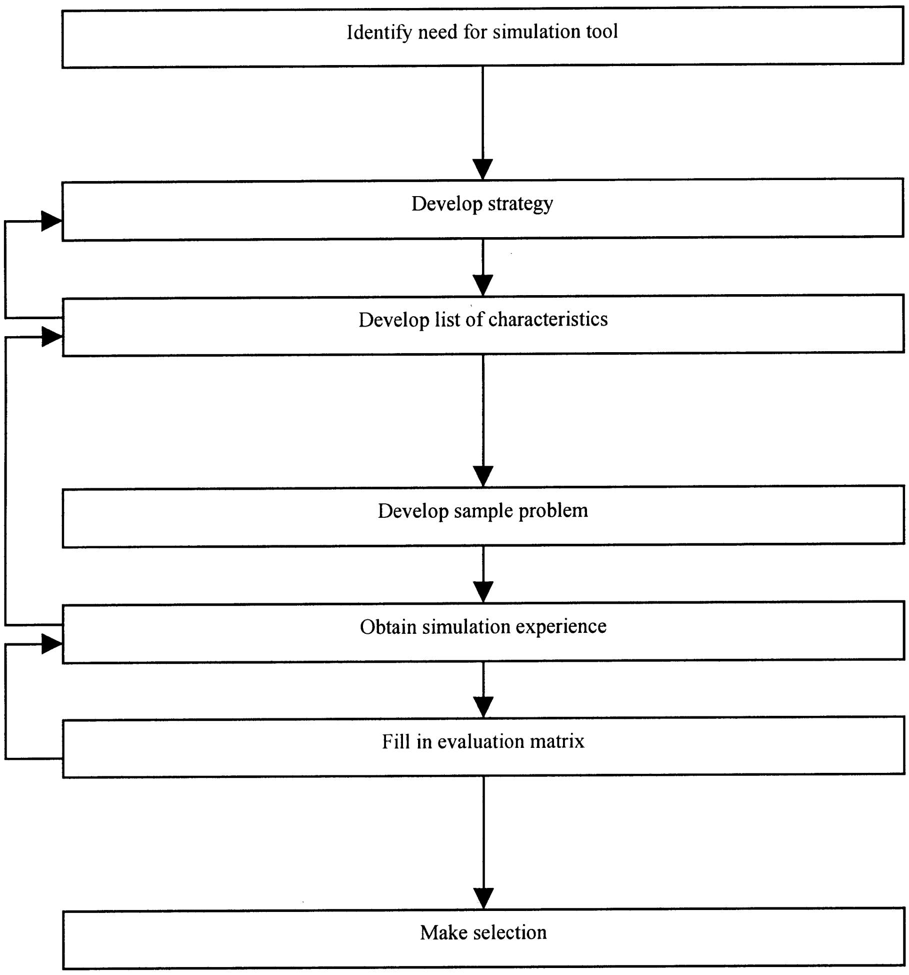
Identify need for simulation tool
Develop strategy
Develop list of characteristics
Develop sample problem
Obtain simulation experience
Fill in evaluation matrix
Make selection

Figure 1 Simulation Software-Selection Flowchart.

6. Process plan modeling
7. Output analysis systems
8. Graphical and programmable model-building capabilities

Based on the same survey, attributes of a simulation tool have been ranked in order of importance as follows:

1. Quality of graphics and animation
2. Availability of training and consulting
3. Price

TABLE 1 Software Evaluation Matrix

		Software 1		Software 2	
Importance (1)	Desired Features	Rank (2)	Score (1 × 2)	Rank (3)	Score (1 × 3)
	Language				
	Portability				
	Readability				
	Documentation				
	Data requirements				
	World views				
	Library of algorithms				
	User code interface				
	Interface with:				
	Databases				
	Spreadsheets				
	CAD software				
	Etc.				
	Compatibility				
	Operation				
	Applicability				
	Debugging features				
	Ease of use				
	Intuitive				
	Parameter revision				
	Model revision				
	Interactive				
	Check points				
	Reusability				
	Output				
	Reports				
	Graphics				
	Animation				
	Exportability				
	Hardware				
	Compatibility				
	File sharing				
	Output devices				
	Vendor				
	Ongoing development				
	Training				
	User support				
	Customer base				
	Cost/value relationship				
Total Score			Score 1		Score 2

4. Speed of execution
5. Flexibility
6. Ease of use

References cited in Section 3 also propose methodologies for evaluating simulation tools. The proposed methodology to select the appropriate software is outlined in Figure 1. Once the need for simulation software is identified, a strategy needs to be developed (Section 2).

Given the company objectives and constraints, a list of desired characteristics needs to be formulated. These characteristics are listed in Table 1. A simple yet realistic sample problem must be constructed to evaluate the software. Simulation experience must be gained either by using the software over a period of time or by participating in simulation activities with trained individuals. The software can be compared using an evaluation matrix like the one presented in Table 1. The decision maker must decide on the desired characteristics to be included in the study and the importance of these characteristics. During the experimentation with the sample problem, ranks can be assigned to the characteristics of the software subjected to evaluation. Once all the software is evaluated, the total score for each software can be calculated. The selection can be made comparing the total scores and the costs of each alternative. Since the assignment of importance factors and ranks is subjective, including more people in the evaluation process will strengthen the decision.

4. SIMULATION TOOLS

This section is prepared using the user's manuals/guides, demo software, articles, books, and promotional material obtained from simulation software vendors. The websites of the vendors were also very useful.

4.1. Languages

Once the simulation model is formulated, it needs to be translated into a computer language for execution. In the early days of simulation, general-purpose computer languages were the only alternatives. Regardless of the language used, simulation modelers have been burdened with the task of writing procedures and functions for creating and deleting entities (transactions), creating random numbers and variates, updating simulation time and system status, and recording and analyzing data for output (Schwab and Nisanci 1992). This inefficiency in the model translation stage has led to the development of powerful special purpose simulation languages. These languages are capable of supporting discrete, continuous, and combined modeling world views. Furthermore, discrete change simulation languages offer event scheduling, activity scanning, or process orientations in modeling systems. Uses of software engineering has led to reduced software development costs and improved flexibility. Significant advances have been made in the field of model translation.

The following section reviews some of the most popular simulation languages and their features.

AweSim is a descendant of SLAM II (O'Reilly and Lilegdon 1999a; Pritsker and O'Reilly 1998, 1999; Symix Systems/Pritsker Division 1999) that supports a wide range of tasks necessary to execute a simulation project. It is a general-purpose simulation system that is distributed by Symix Systems, Inc., formerly Pritsker Corporation. It takes advantage of the latest Windows technology and includes the Visual SLAM simulation language. AweSim provides the user with three frameworks (network, discrete, and continuous) that can be used independently or combined depending on the problem. This feature of AweSim enables virtually modeling any system or process. Programming is not needed for network models, yet flexibility is provided through allowing user coded inserts in C or Visual Basic. Network models can be combined with continuous and discrete-event models that can be developed using the object-oriented technology of Visual Basic, C, or Visual C++. AweSim's architecture has the openness and interconnectivity to store, retrieve, browse, and communicate with externally written software applications such as databases, spreadsheets, and word-processing programs.

A simulation project with AweSim consists of scenarios that represent alternative system configurations. Scenarios consist of component parts that are created by software programs, referred to as builders, that are provided by AweSim. Through an executive window, project, scenarios, components (network, subnetwork, control and user data, etc.), simulate, and report options are defined. Component builders are accessed through the Components menu. The project maintainer, which examines changes made to the current scenario, determines a model translation or compilation requirement. Based on the prompt from the project maintainer, the user initiates a translation task. AweSim allows multiple tasks to be performed in parallel while a simulation is executed as well as switching between tasks. Some of the features of AweSim are:

- *Building models:* A network is a basic component that graphically displays the flow of entities through the system being modeled. A network consists of nodes (processing locations) that are connected by activities that describe routing and operation time requirements. Nodes are used for functions such as removing or entering entities, seizing or freeing resources, changing variable values, etc. Entities may also be described by assigning attribute values such as weight, arrival time, etc. A network is built interactively in AweSim by selecting symbols from a graphical palette and dragging them to the desired location on a screen. Required informations about the symbols are specified by filling out a form that is supported by online error checking. Context-sensitive help and search capabilities are also provided.

- *Subnetworks:* These are reusable network objects that are provided for hierarchical models and are invoked from a calling network. This feature enables creating some form of a model depository that can be used by different modelers in different projects. Hence, it enables saving modeling time and sharing modeling expertise. Subnetwork builder works like the network builder but contains slightly different nodes.
- *Model output:* Output from various scenarios can be compared both graphically (in the form of bar charts, histograms, plots) and textually through a report browser. Multiple windows of graphical output, and side-by-side comparison of textual output are also provided. Information in the AweSim database can be exported to other Windows packages, such as Excel and Word.
- *Animation:* Multiple scenarios can be developed for a single scenario using "point and click." The animator manipulates graphical items one wants to move and the background on which they appear. Storing the symbols in standard Windows bitmap format allows them to be exchanged between programs using the Windows clipboard.
- *Interactive execution environment* (*IEE*)*:* Enables the modeler to examine, modify, save, or load the current system status. This feature facilitates debugging and verifying the model.
- *Scenario selector:* Allows screening a set of scenarios or selecting the best scenario.
- *Integration with other software:* AweSim is built on a relational database that can be accessed with tools such as Dbase, Access, FoxPro, and Excel. AweSim provides the capability to exchange data between its input and output files and these tools. Graphical elements produced by CAD, drawing, or paint programs can also be loaded into AweSim.

GPSS (General Purpose Simulation System) is a process-oriented special purpose simulation language (Crain 1998; Schriber and Brunner 1998; Banks et al. 1995, 1996; Pritsker 1995; Schriber 1991). Geoffrey Gordon developed GPSS about 1960. It is a language to model discrete systems. GPSS consists of a well-defined vocabulary (set of blocks) and a grammar. Each block carries information related to its location, operation, and operands. Operation descriptions define the tasks the blocks perform. Operands specify how these tasks need to be performed. A block may not have a location name and operands. Each block is represented by a differently shaped figure. A GPSS model consists of a selection of blocks connected with lines that describe the operation of the system model. GPSS defines dynamic and static model entities. Dynamic entities represent the units of traffic and are referred to as transactions. Static entities represent system resources. As the simulation runs, transactions enter the model and start moving over as many blocks as possible until they encounter a user-defined delay, are removed from the system, or are denied entry by the next block. The procedure summarized above describes the flow of entities in a GPSS model and the interactions between different types of model entities. Ease of learning the language and reduction of the modeling time have made GPSS one of the popular discrete-event simulation languages. Various versions of GPSS exist, such as GPSS/H (Wolverine Software Corporation) and GPSS/PC and GPSS World for Windows (both by Minuteman Software). GPSS World includes drag-and-drop modeling, a high-performance model translator, point-and-shoot debugging, an embedded programming language, built-in probability distributions, programmable experiments, interactive operation of all commands, and interactive graphical text views of running simulations.

SIMAN is the simulation language within Arena. First introduced in 1983 by Systems Modeling Corporation (Pegden et al. 1995; Kelton et al. 1998; Schriber and Brunner 1998), SIMAN provides the use with three distinct but fully integrated modeling frameworks. Each framework is divided into two sections: a system model framework and an experimental framework. The system model permits the user to describe the process, while the experimental frame permits the user to supply the parameters and characteristics describing the system operation. SIMAN provides modeling blocks for describing the system model and experimental elements for use in the experimental frame, including several blocks and elements that provide SIMAN with materials-handling modeling capabilities as part of the basic package. In the discrete-event framework, SIMAN provides a complement of user-definable and user-callable functions and subroutines to be used alone or in conjunction with the block framework. These subprograms allow the user to manipulate entity attributes and files easily and provide for complex decision rules. In the continuous framework, system state equations can be integrated over time while simultaneously integrating with the blocks and discrete-event frameworks.

CACI Products Company provides SIMSCRIPT II.5. Developed by Kiviat, Villanueva, and Markowitz, it is an event-oriented discrete-event simulation language (Kiviat et al. 1969; Russell 1983). SIMSCRIPT II.5 consists of features for developing time-driven simulation models. It includes both event- and process-oriented simulation constructs, multiple random-number generators, range of random distribution functions, automatic statistics collection, an integrated continuous simulation capability, and built-in graphics support for model input and output. Its world view facilitates the mapping of real-world modeling requirements directly onto its language features. Its general-purpose programming constructs, such as arithmetic operators, text manipulation, arrays, and subroutines, and the other features are accessed using English-like programming syntax that supports the self-documenting nature of the code. It provides extensive error checking through entity referencing, array bounds, and memory use. It contains a rich library of simulation objects that is designed to fulfill common simulation requirements, activity scheduling, and constrained resource modeling. Compilation and linking of all parts of a program are done automatically. Its graphic editor helps in creating and editing graphic images and charts and user interface objects such as menus and dialog boxes. Images and graphics are importable, and graphics library elements are fully portable. The debugger feature provides detection of programming and simulation errors, and a detailed trace reporting. SIMSCRIPT II.5 also provides integrated graphics and animation with scaling, rotating, and positioning capabilities. Using its graphics editor, presentation graphics such as pie charts, level meters, and bar graphs can be produced. The supported platforms consist of Windows 95/NT, SPARC, HP9000/700, SGI, DECAlpha, and IBM AIX.

4.2. Packages

Early offerings were no-programming products. Many were not versatile enough to handle complex decision-making rules. The biggest growth of simulation packages has been in the manufacturing area with products for PCs. Figure 2 lists the major components and functions of simulation packages. Some of the packages and their features are described below.

Arena (Sadowski et al. 1998; Kelton et al. 1998; Snowdon et al. 1998) is a graphical modeling and animation system. It is a product of Systems Modeling Corporation. Arena consists of a complementary family of products (Arena Product Suite) to meet the various needs for simulation in an enterprise through a common software interface and compatible features. These products share a common software foundation. The Business Edition (Arena BE) targets business and other systems. Arena Standard Edition (SE) targets detailed modeling of discrete and continuous systems and provides more flexibility. Arena Professional Edition enhances Arena SE with the capability to build custom simulation objects. Call$im targets call-center analysis, and HiSpeed$im targets high-speed production line modeling. Arena provides a hierarchical structure consisting of alternative and interchangeable templates of graphical simulation modeling and analysis modules. The user combines these modules in building a simulation model. Modules are grouped into a panel to compose a template. Different modeling constructs and capabilities are obtained by switching templates and bringing modules from different panels and templates. For example, modules from the SIMAN template can be pulled and SIMAN constructs can be mixed with the higher-level modules from another template. Complex decision rules can be coded using Visual Basic, FORTRAN, or C/C++. Each level in the hierarchical structure is managed with a single graphical user interface. Arena's application solution templates (collection of modules for an area, e.g., Call$im) provide applications for modeling specific areas. Arena is compatible with Microsoft Windows and Microsoft Office. It also enables integration with other technologies and applications such as databases, drawing/modeling (AutoCAD) products, and spread sheets such as Visio, Oracle, and Microsoft Office. Arena provides user interface features such as drag-and-drop, context-sensitive click menus, and customizable tool bar. It also provides a project bar for accessing modeling constructs and navigating through model hierarchy. Using Arena (Business Edition), process dynamics is represented in a hierarchical flowchart with system information stored in data spreadsheets. It also has a built-in activity-based costing and system performance data that enables analyzing business, manufacturing, service, and other systems. Arena also provides input and output analyzer for entering, analyzing and reviewing data. OptQuest for Arena is also provided for simulation optimization.

AutoMod version 8.7 (AutoSimulations 1999; Banks 1998a; Phillips 1998a, b) is an industrial-oriented simulation system provided by AutoSimulations Inc. It uses CAD-like graphics to define the physical components of a system. A procedural language is used to define the logical elements. This language is based on action statements that combine the power of a structured language with the ease of English-like syntax. If needed, the user may define C functions. The communications module allows communications between two or more models and third-party applications such as spreadsheets or Visual Basic programs. All graphics are represented in 3D space, and complex motion (kinematics and velocity) of equipment can be simulated. The kinematics module allows simulating the robots and other equipment containing moving parts and integrating them into a simulation model. AutoMod's CAD features are used to define layout and material-handling and storage systems by maintaining distance, space, and size characteristics in 3D. All graphics created can be viewed with unlimited control with respect to translation, rotation, scale, perspective, and so on. The animation runs in real time with the simulation. The model execution is interactive, which enables the user to

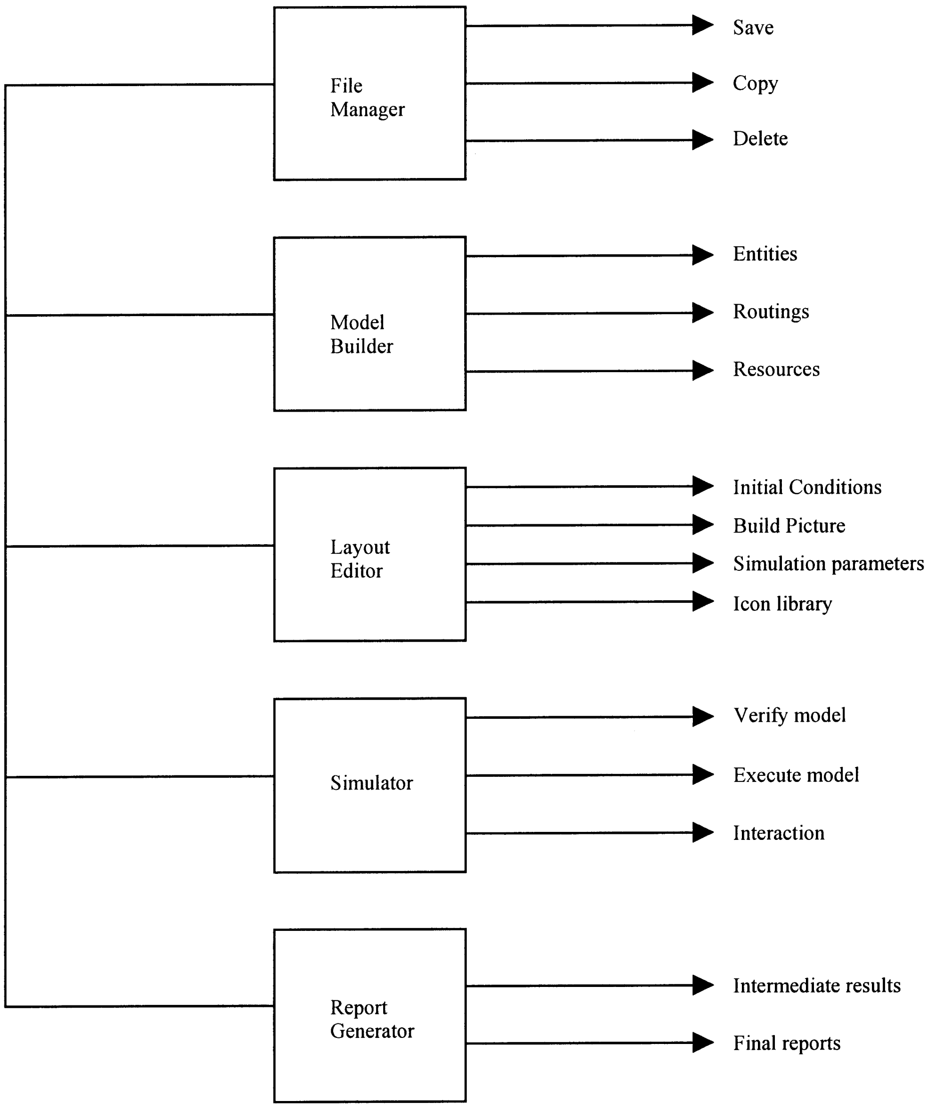

Figure 2 Major Components and Functions of Simulation Packages.

stop and resume the simulation, cancel animation, and select objects for detailed statistics. AutoMod has a material-handling orientation. Entities moving around the system are called loads, and loads are worked on by processes. The software features a well-developed array of predefined material-handling systems, including conveyors, automatic guided vehicle systems (AGVs), automated storage and retrieval systems (ASRS), bridge cranes, and power and free conveyors.

AutoMod also has support tools: AutoStat, Simulator, and AutoView. AutoStat provides enhanced analysis of output such as calculating confidence intervals, design of experiments capability, optimization, and steady-state analysis. The user defines the desired analysis through a point-and-click Windows interface. AutoView, a postprocess package, allows the creation of a directed walk-through of the output. It allows panning, zooming, and moving back and forth in time and space. The animation view can be attached to a moving model entity to get an inside view of the system. Simulator, which is shared with AutoSched AP (ASAP), provides a template for capacity planning. ASAP is a simulation-based, finite capacity planning and scheduling tool. It is used to increase throughput, reduce in-process inventory, and increase resource utilization. ASAP takes into consideration constraints such as shift schedules, work setup rules, batching, preventive maintenance, machine efficiency, and operator skill classes. Its built-in task-selection rules enable the modeling of selection of machines and personnel to enhance performance. Its output consists of reports, Gantt charts, and utilization graphs. In addition to scheduling, ASAP is also used as a planing tool for capacity analysis.

FACTOR/AIM (AIM) (Pritsker 1995; Symix Systems/Pritsker Division 1997; O'Reilly and Lilegdon 1999b) is a special-purpose simulation system for use in manufacturing decision support in areas such as engineering design, scheduling, and planning. A product of Symix Systems, Pritsker Division, it is used to build, simulate, and animate models and analyze graphically and statistically the simulation results obtained from various scenarios. AIM uses the language of manufacturing, which eliminates the need to learn a language-based syntax and abstract a system to fit syntax. AIM components, which have both operational and graphical characteristics, consist of machines, operators, materials, parts (loads), batches (group of loads), process plans or routings, conveyors, and so on. It also has nongraphical components such as work calendar, orders, shifts, and breakdowns. These components, coupled with a comprehensive set of predefined manufacturing rules, provide a powerful framework for modeling and simulation analysis of manufacturing systems. The modeler has the option of coding customized rules in C language if the built-in rules do not represent the desired logic.

The current version of AIM (8.1) supports user code with the latest version of Visual C++. AIM uses the Windows platform, where the model and the data are stored in a Microsoft Access database. This feature enables importing and exporting data and preparing customized reports. Access application wizards can be used in developing decision support applications with dialog boxes and menus. AIM consists of five modules: Executive, which triggers other modules, controls data collection, and supports database and scenario management; Simulator, which builds and runs simulation with animation; Batch Simulator, which supports experimentation through replications and multiple runs; Reporter, which provides textual reports; and Gantts, which provides detailed schedule information. A model is created through interactive point-and-click operations to select and locate graphical components that can be further refined using dialog boxes. Next, parts and their routings are defined. A model can also be prepared using automatic loading if the existing information is in electronic form. Animation of an AIM model is created automatically. An interactive run process provides support for verification and validation through automatic animation, reports and graphs summarizing system status, detailed information on each model component, and the ability to view all scheduled events. Multiple replicate and scenario runs can be made through the Batch Simulator module.

AIM provides an extensive set of standard output reports and graphs in addition to customized reports and graphs that can be created using Access report wizards. Furthermore, output can be moved to Excel Spreadsheet for additional processing. Schedule reports can be created and communicated using the GANTT chart module. AIM also has cost-modeling capability. Product costs in four groups, operating, overhead, inventory, and lateness, can be defined using fill-in-the-blank forms. For example, operating costs resulting from part production and flow, such as machining, fixture, operator, and consumables, can be tracked. Using the cost-modeling feature of AIM, cost-simulation and output reports can be generated. Cost reports such as average operating expense per part, value of inventory, and total expense incurred can be prepared. Cost reporting can be customized to generate cash flow reports and pro forma income statements. The user also has the option of selecting cost reports for the entire system, by part or resource type.

Mast Simulation Environment is a product of CMS Research Inc. (1999). It is an integrated software tool for addressing manufacturing system design and evaluation problems. It provides a single environment to aid decisions on design, acquisition, and operation of manufacturing systems. MAST offers a pull-down menu toolbar to access commonly used features, can launch multiple results, and has a cost modeling option. It utilizes CAD graphics for layout and animation and allows importing CAD layout as background. It has features such as exporting and importing data to and from spreadsheets and databases. It provides station worktable and animation libraries with zoom, scroll, scale, and speed adjustment options. Launching multiple results at one time is also an option. Mast is available for Windows 95/98/NT.

Micro Saint version 3.0 (Laughery 1999; Plott et al. 1999; Ernst et al. 1999; and Barnes and Laughery 1998) is a PC-based discrete-event simulation software program available from Micro Analysis & Design. With Micro Saint, any process can be modeled as a network of tasks that can be represented in a flow diagram. Sets of decision rules and algorithms consisting of algebraic and logical expressions specific to a process can be developed using Micro Saint's own C-like language. The user defines the terms that are used. A graphical user interface is used for model development and testing. The icons and background for the Action View animation and the flowcharting symbols are customizable. Drawings from other graphics programs can be imported through the clipboard. Micro Saint symbolically animates the network diagram as it executes the model, using the routing choices and random number seed to generate task times that are supplied by the user. The user also selects statistics charts or graphical representations to analyze the data collected during the simulation run. Queue, resource, and task statistics are collected automatically as well as the snapshots of user-defined variables at user-defined triggers. Micro Saint also has a resource wizard that can be used to create new resources quickly and allocate them to tasks in the process. The syntax checker searches the model for automatic detection and display of errors. OptQuest, included with Micro Saint, automatically searches for and finds optimal or near-optimal solutions to simulation problems and fine-tunes simulation parameters. Micro Saint has built-in features using Microsoft COM services that allow it to communicate dynamically (i.e., as the model runs) with other applications such as Visual Basic, C++ applications, and even other simulations. Micro Saint is available for Windows 95, 98, and NT, and a Unix/Linux version variant is available under the title of Integrated Performance Modeling environment.

MODSIM III is a product of CACI Products Company (Tumay and Wood 1999; Goble and Wood 1998; and Lewellen and Tumay 1998). MODSIM III is an object-oriented discrete-event simulation language that provides a complete development environment (built on top of Microsoft's Visual Studio environment) for developing models that are visual, interactive, and hierarchical. Its features include run-time libraries, graphical user interface and results presentation tools, database access, and hooks to HLA. Its components, such as Simulation Layer, Graphics Editor, Compilation Manager, and Interactive Debugger, support the development of advanced simulation models. Seamless interoperability with other tools is achieved through its ability to generate C++, which compiles using the platform's native compiler. Objects are defined in two separate blocks of code. The definition block defines the object type through its variables (everything the object knows) and methods (everything the object can do) which is used for interface specification. The implementation block defines the logic of what the objects do (their behaviors) and their affect on their state variables. MODSIM III sequences the execution of the methods of object instances to ensure correct sequences for the events. Its features to manage the scheduling, interaction, and synchronization of behaviors provide increased readability and consistency. Its features for modeling concurrent and interacting behaviors qualify it as a development tool.

MODSIM III provides a library of tested objects that can be customized. For example, objects may compete and queue for resources based on a priority while accumulating statistics on resource utilization, waiting time, and so on. Inheritance, which is a benefit of object-oriented software construction, provides object libraries that can be customized. New objects evolve through inheritance with additional capabilities resulting in enhanced software-development productivity. Inheritance also enriches the graphical properties of objects and facilitates the creation of interactive and graphically managed models. The graphical editor supports creation and configuration of icons, menus, dialog boxes, and presentation charts. Through interactive graphical editing, scenarios are defined on screen and animation and plots can be viewed while the simulation is running. The graphics environment allows easy access to animation, dynamic presentation graphics, and user interface toolkits using SIMDRAW graphics editor. Its DatabaseMod module enables access to ODBC-compliant databases and retrieving and committing data to databases. HLA is supported through run-time interface (RTI), which can be called from languages such as C++, Java, and Ada. RTI supports both local intranets and the global Internet. MODSIM III with SimGraphics is available on Windows 95/98/NT and Unix platforms. This enables load-balancing a simulation model through a network or displaying the user interface and animated graphical output on different and remote computers.

ProModel is a product of ProModel Corporation (Heflin and Harrell 1998; ProModel Corporation 1999). It is a menu-driven, mouse-activated discrete-event simulation and animation tool. The model development is graphical and object oriented. Modeling is done by selecting from a set of modeling elements such as resources, downtimes, and shifts and modifying the appropriate elements. All input is provided graphically, with information being grouped by object type and presented in a format similar to a spreadsheet. User coding in C, Pascal, or Basic can be linked to the models and called during the simulation. Its top-level menu allows the user to select model editor, run simulation, view output, or quit. The model editor permits full model definition, routing, scheduling, resource description, and transportation. All standard statistical distributions are available, and StatFit is provided to aid the user in selecting the proper distribution. The model editor also allows full control over the graphical presentation. A standard set of icons is provided and presented for selection by number. An icon editor is also provided to create special symbols. ProModel provides an automated model builder that prompts the user for all information required for modeling the process. It is organized logically and guides the user from creation of the model name to part and routing definition and finally to layout definition for the animation.

A pop-up menu is provided to prompt the user in defining a statement or expression. Its graphics editor allows scaling, rotating, copying, and so on, and drawings from other packages can be imported. Animation development is integrated with the model definition. Run simulation permits viewing of the animation while the simulation is running. The user retains full control over the animation. The simulation may be paused at any time to allow intermediate statistics to be viewed. ProModel is interactive during the run time, enabling model parameters to be changed and thus facilitating the "what-if" analysis. Basic statistics such as utilization and throughput are available, as well as cost statistics. It provides features to select reports and provides tabular and graphical reports on all system performance measures. Output reports from different simulation runs can be compared on the same graph. ProModel also provides model-merging capabilities. It can exchange data with external files and application programs such as Excel. SimRunner optimization capability is also provided.

QUEST is a product of Deneb Robotics, Inc. (Donald 1998). It is a discrete-event simulation package used to simulate discrete flow processes. QUEST uses 3D geometry and combines a graphical user interface with material-flow logic grouped in modules consisting of labor, conveyors, AGVs, kinematics, power and free conveyors, and ASRSs. For unique flow logic problems, QUEST's Simulation Control Language (SCL) can be used. It also provides a value-added costing module for activity-based costing analysis. Results can be viewed with graphical and numerical analysis capabilities. It consists of features to create a 3D virtual factory environment. Its other features consist of the ability to import and export 2D or 3D CAD geometry and data (e.g., spreadsheets, MRP systems), a graphical user interface and a programming language, and the ability to provide integrated solutions through interaction with other Deneb software. Quest also provides a virtual collaborative engineering (VCE) environment that enables modelers in remote locations to interact with the same model. QUEST integrates virtual reality (VR) devices such as head-mounted displays, stereo glasses, and cyber gloves to immerse the modelers within the factory simulation.

SDI Industry is a product of Simulation Dynamics (Siprelle et al. 1998). It is a discrete-event simulation package designed specifically for the process-oriented industries, such as chemicals, foods, consumer goods, and pharmaceuticals. SDI industry uses discrete rate technology to simulate processes in these industries. It adds an embedded database and specially designed functions to Extend for simulating high-speed and high-volume processes. Extend is the underlying simulation engine, developed by Imagine That Inc. SDI industry is a graphical interactive program that provides a hierarchical framework of model templates, flow blocks to model material flows, control blocks to manage flow, and an integrated database to store management information. A model is developed through four hierarchical levels, consisting of equipment (process), operations, systems, and plant. Each level is provided with a template and with a process management block to control the process at that level and interface with the other levels. A relational database is used to store and manage data centrally. Data requirements for the control blocks at each level are provided by standardized tables such as "materials and brands," and "shift schedule." A database viewer is provided to view and edit table data in Extend. Simulation Dynamics provides two more products: SDI Industry Pro provides scheduling and control tools, and SDI Supply Chain is used to model the dynamics of a supply chain from source to user.

SimEngine is a product of Industrial Modeling Corporation (1999). It is a 3D simulation tool that incorporates object-oriented programming. SimEngine builds on Delphi and C++ object-oriented languages, which feature inheritance, polymorphism, and encapsulation. Every SimEngine entity is an object with exposed properties, methods and events. Models are built by manipulating the objects in visual environment and in code. SimEngine also works with C++ Builder, which incorporates an ANSI standard version of C++. Delphi and C++ Builder share a development environment that is similar to Visual Basic. The environment features visual design tools, an object browser, code window, and debugging tools. SimEngine builds on these development tools by adding simulation objects such as orders, routings, resources, and vehicles. The basic model building is done via visually manipulating these objects in 3D simulation explorer. Objects are created and moved using drag-and-drop techniques, and their properties are set in the object inspector. Code writing becomes necessary only when implementing complex routing logic. Database components enable reading information directly from company databases. It has native links available to SQL databases. Its ActiveX feature enables posting of the models to a website.

SIMUL8 is a product of Visual Thinking International Inc. (1999). It is a discrete-event simulation package to simulate a business or a manufacturing process. SIMUL8 provides objects such as work centers, storage bins, conveyors, entry/exit points, resources, variables (including spreadsheets), labels (attributes), and work items. Objects are linked automatically. Models are built on the screen via point-and-click operations and by drawing them with the mouse. The drawings are detailed by using intuitively organized options. After dragging and dropping the parts of the process onto the screen, clicking on "run" shows the full animation of the products flowing through the system. All animation is automatic with full zoom and pan options and distance on the screen relates to distance and time in the model. By clicking on resources, entities, and so on, their parameters (speeds, setups, constraints, etc.) can be modified and status updates through counts and reports can be obtained. These capabilities are extended using Excel or VBA. SIMUL8 can also import Visio flowcharts to begin the simulation process. SIMUL8 also conforms to the SDX (simulation definition exchange) file standard, which allows interchange of structure and detailed data between simulation and other packages (e.g., CAD) that contain information related to simulations.

Taylor ED (Enterprise Dynamics) is the product of F&H Simulations (Hullinger 1999). It replaces the Taylor II (Nordgren 1998) software, which is being phased out. Taylor ED is an object-oriented software system for modeling, simulating, visualizing, and monitoring dynamic (discrete) flow process activities and systems. It uses an atom concept. An atom is defined as a four-dimensional object that has a location and speed (x, y, z) and a dynamic behavior (time). Atoms can inherit behavior, contain other atoms, and be created, destroyed, and moved onto one another. Atoms also communicate with each other. Within Taylor ED, everything, such as resources, products, a model, and a record, is referred to as an atom. Through its open architecture, it allows users to access standard libraries of atoms (logistics suite) to build models or create their own atoms. 4DScript language (atom editor) is used to create the atoms and to control atom's behavior, the user interface, and model visualization. A model is built by selecting, dragging, and dropping atoms from the library onto the screen. This is followed by connecting the atoms and defining routings of entities/atoms. Routing is defined by linking the output channels of an atom to the input channels of another one. Complex routing rules can be assigned using a pull-down menu. Finally, each atom is assigned a logic through defining atom's parameters, that is, its behavior and functionality. Clicking on the atom opens editing windows. Taylor ED includes experiment and optimizer (OptQuest) modules through which experimental and run conditions and performance measures can be defined. Results consisting of animation, predefined, and user-defined reports, and graphs can be viewed in 2D or 3D concurrently with simulation. Results can also be exported to external software programs. Taylor ED also monitors real-time systems by linking external programs to read and write information. Run time and real time can be synchronized, enabling the model to monitor the system in real time.

Call Center MAESTRO (Model Builders) is a decision support tool that is composed of a discrete-event simulation engine and is used to effectively staff and configure a call center. COMNET III (CACI) is a network simulation tool to predict LAN, WAN, and enterprise network performance. Its add-on modules consist of application profiler to predict the impact of a new client application, satellite and mobile module, and circuit-switched traffic module. SIMPROCESS (CACI) is a hierarchical and integrated simulation tool to enhance productivity for business process modeling and analysis. Packages such as ASSEMBLY, ERGO, and TELEGRIP are also offered by Deneb. These products are directed towards more specific applications and can be integrated with QUEST. Visual Simulation Environment (VSE) is a product of ORCA Computer, Inc. (Balci 1996). VSE is a discrete-event, general purpose, object-oriented, picture- and component-based visual simulation model development and execution product that can be used in solving complex problems such as air traffic control and communications networks.

4.3. Applications

Over 170 simulation products listed in the Directory of Simulation Software (Rodrigues 1994) published by the Society for Computer Simulation cover a wide range of application areas, such as aerospace, AI/Expert systems, business, communications, education, manufacturing, military, and networks. The annual Buyer's Guide of simulation software (Institute of Industrial Engineers 1999) lists 27 companies providing 40 simulation software products. In this list, the application areas are classified as general purpose, manufacturing, construction, service, health care, material handling, warehousing and distribution, and enterprise and business planning. Banks (1998a) classifies and gives examples of 27 simulation software products as general-purpose, manufacturing-oriented, and business process reengineering products. Simulation has also been used to study areas such as urban systems, economic systems, biological systems, environmental and ecological studies, computer and communication systems, project planning and control, and financial planning (Pritsker and O'Reilly 1999). Within each of these areas, a multitude of topics has been subjected to simulation analysis. For example, under transportation systems, topics include railroad system performance, truck scheduling and routing, air traffic control, terminal and depot operations, and issues related to supply chain management. The accelerated rate of change in simulation technology and the increasing complexity of problems associated with modern systems will undoubtedly widen the areas and the topics necessitating simulation studies (Nisanci 1997, 1998, 1999).

5. PITFALLS INHERENT IN THE USE OF SIMULATION TOOLS

Although there are many dangers inherent in the use of simulation products, only the major ones are addressed here (see Musselman 1998; Norman and Banks 1998; Rohrer and Banks 1998; Banks and Gibson 1996, 1998; Harrel and Tumay 1994; Pritsker Corporation 1993). Careful use of simulation will always be rewarded, while a disregard for the dangers may have disastrous consequences.

5.1. General Pitfalls

The general pitfalls consist of:

5.1.1. Having the Tool and Not Using It

Strange as this may seem, this is a real danger. Sooner or later, most project engineers will feel some pressure to reduce the time or expenditures required to complete a project. Simulation analysis is an easy target for elimination because the impact of its elimination is not noticed immediately. Cutting manpower or reducing the project scope has immediate and noticeable impact on the project. The project can move ahead, detailed design can be done, and equipment can be purchased and installed, all without any felt loss of simulation analysis. Usually the first time the lack of simulation analysis is experienced is during start-up when design flaws and oversights typically caught by simulation are revealed. When an inadequate design is revealed, the solution is usually expensive in redesign time and equipment cost. These costs, however significant, may still be dwarfed by the loss in revenue due to the failure to deliver the goods or services on time.

Complex systems often defy the intuition and the traditional simplified analysis (including rules of thumb) that have been faithfully passed down through the engineering generations. Complex systems with complex interactions demand detailed analysis using simulation tools. The project engineer assumes a large and unnecessary risk in not using available simulation tools when the project calls for them.

5.1.2. Failure to Establish and Adhere to Project Objectives

A clear and specific set of objectives must be established at the beginning of every simulation project. These objectives should address the questions to be answered by the project engineer using the simulation tool. Issues to consider include:

1. Configuration, quantity, utilization of nonconsumable resources
2. Quantity of consumable resources required
3. Total time to provide goods or services to the customer
4. Critical processes and bottlenecks
5. Process capability (throughput)

Once management personnel have agreed upon the objectives, they need to direct the project to its conclusion. Every new idea or suggested change to the project needs to be tested against the approved objectives. Value-adding changes should be incorporated into the objectives and reapproved. Suggested changes that add nothing to the project or do not support the objectives should be rejected. Periodically during the course of the project, the objectives should be reviewed to make certain that the project would answer all questions raised at the beginning. Failure to monitor the project in relation to the objectives may allow project personnel to pursue information and solutions that do not contribute to the objectives. This results in delays of the project's completion.

5.1.3. No Review or Inadequate Review

One of the greatest mistakes made by project engineers who have committed time and money to perform a simulation is neglecting to take the additional time to properly review the model. A review should take place after the completion of data collection and before the beginning of model building. A review of the anticipated modeling assumptions should be done in addition to a discussion of any problems with data collection. A process flowchart, providing a thorough description of the process being modeled and the method of modeling, should be developed and presented to all involved. As the project develops, additional interim progress reviews should be held as often as needed.

The final review should take place after completion of all the experimentation but before the submission of the project report. The final review process should include at least the four following key elements:

1. Thorough understanding of the assumptions. Particularly important would be a review of the elements of the process which were not modeled and why they were excluded from the analysis.
2. Review of all parameter values. Care should be taken to ensure that data used in the analysis is correct and up to date.
3. Line-by-line review of all code, including model input values. This is essential, especially if others do the modeling, e.g., simulation department or consultant.
4. Review of all conclusions and the supporting data. Careful review of the data that lead to the conclusions ensures that they are properly supported. The results and conclusions should also make reasonable sense.

At this point, simulation plays a key role in the development of the design parameters. Therefore, a thorough review is necessary to make certain that the model and the values used as a basis for the design are indeed correct. The engineer who depends on others to do simulation takes an enormous risk if the final product is not thoroughly reviewed. To assume that correct work is done by an internal simulation expert or a consultant is risky. Conceivably, one piece of information improperly translated, one parameter off by a factor of two, or one part of the process not modeled as it will operate can invalidate the results of the simulation. A thorough review of the model and the results is critical to the overall success of the project.

5.1.4. Not Having the Proper View of the Tool

The proper view of the simulation tool leads to proper expectations regarding what the tool can do and what results should be obtained.

5.1.5. Not Viewing Simulation as a Statistical Tool

Simulation is a statistical tool and must be used as such if reliable results are to be obtained. Its basic task is to collect data about how a system changes with time.

5.1.6. Thinking of Simulation as an Optimization Tool

Because simulation is a data-collection tool only, it should not be thought of as an optimization tool. Parameters are put into the tool and data are generated about the system (e.g., throughput, resource utilizations, congestion) and the components in their relationship as it was entered. Enhancement of any design only takes place as the engineer reviews the output from a simulation run and adjustments are made to the model and the model parameters. Improvement is therefore an external, manual process. However, incorporation of automated scenario-generation capabilities and optimization packages into simulation tools significantly enhances this process.

5.2. Pitfalls Related to Methodology

5.2.1. Failure to Establish Appropriate Model Boundaries and Parameters

Model boundaries become important in modeling because equipment and items not modeled are considered to be unlimited. Unlimited resources are not a problem if the cost of the resource is relatively low. However, resources having a significant impact on the process (e.g., fixtures that cost $50,000 and computer hardware) should all be modeled. Modeling can also be used to describe restrictions on the flow through the model. Not describing a restriction would mean that there is no restriction and therefore that flow is unlimited.

5.2.2. Improper Data Collection

In the simulation process, data collection may be the biggest problem faced by the project team. Critical data may not be available. If data are available, they may not be in a useful form.

5.2.3. Not Verifying Data

It is important to know the source of the data and how good they are. Downtime data carefully gathered over several years of maintenance are certainly more significant than a wild guess by a maintenance foreman.

5.2.4. Not Collecting the Proper Data

Because data collection is often a difficult task, data gatherers are often tempted to shortcut the process, especially if they do not understand the statistical implications. It is possible to believe that an average time for a transaction might be just as good (and of course easier to obtain) than the raw data reduced to a distribution. The use of average time leads to a deterministic solution and can invalidate the simulation study.

5.2.5. Avoidance of the Proper Level of Analysis

Because simulation is a statistical tool that collects statistical data about a system or process, statistical techniques must be used to examine and analyze the data. As an example, since each run of the simulation produces one sample point and since this process involves sampling for a distribution, it is impossible to determine the relationship of a single point to the true mean. Multiple runs are required to generate enough points to determine a calculated mean, standard deviation, and confidence intervals. Then and only then should the design proceed. Knowledge and use of statistical techniques with simulation is essential to produce reliable results.

5.2.6. Validation Problems

A successful simulation study must produce a solution credible enough to be used by the decision makers (Balci 1998). Validation may take more time than building the model.

6. CURRENT STATUS AND TRENDS

There have been continuous efforts to use computers in all stages of simulation besides model execution. These efforts have been significantly enhanced by the development in various technologies such as computer technology, animation, expert systems and artificial intelligence, optimization, and experimentation. Considering the stages of simulation, significant advances have been made in designing the simulation tools that have the following features:

- Assisting the modeler in problem and system definition. Intelligent user interfaces guide the user in producing a quicker and more complete definition of the problem. Complex manufacturing systems are defined quickly with the selection and placement of icons and computer-guided interrogation.
- Model building features in the form of interactive flow charting (block or network diagrams) with a one-to-one correspondence to the simulation program. These features also include a knowledge base consisting of a generic model that can be tailored to a specific model or submodels that can be combined to form the required model using object-oriented programming. Model building features tailored to specific industries allow access to simulation by nonsimulation personnel.
- Evaluation of data requirements based on problem and system definition and objectives. Automatic data-acquisition capability to prepare the data directly from an external company database.
- Model translation feature to provide automatic computer code generation using either program generators or data-driven simulators and thus eliminating the need to teach users a simulation language.
- Model verification features consisting of expert diagnosis programs for debugging models. Output and system animation features to evaluate the logical consistency of a model.
- Features to assist strategic and tactical planning. Automatic detection of steady-state and rundown periods, testing for autocorrelation, and checking for sample size sufficiency.
- Experimentation and automatic search features to find the best combination of resource configurations and levels and operating strategies.
- Automatic documentation of experimental conditions in graphical or tabular form in addition to standard simulation output.
- Highly developed graphical user interfaces that integrate the entire modeling process.

Successful new-generation simulation systems will be expected not only to automate the stages of simulation but to integrate them as well. The following developments are expected to impact future simulation software:

- The significant increase in research, development and application of object-oriented simulation. This technology is becoming the choice for modeling large, complex, and distributed systems. Modeling and simulation using an object-oriented language will increase due to its advantages in compilation, execution, and extensibility (Roberts and Dessouky 1998; Keller and Tchernev 1997).
- With high level architecture (HLA), an individual simulation or set of simulations that was developed for one purpose can be applied to another application. The HLA will provide a structure that will support reuse capabilities available in different simulations that will reduce the cost and time required. Through HLA, simulations can exchange data dynamically and interact with each other as they run. HLA will facilitate interoperability, software interface, and data exchanges. Simulation software will incorporate these developments, as is the case with MODSIM III and Micro Saint (Yu et al. 1998).
- Artificial intelligence and simulation have mutually beneficial connections. Model-based expert systems, simulations built out of neural networks, and smart interfaces to simulation tools are commonplace. It is expected that AI and simulation will join forces to model and simulate intelligent behavior (Wildberger 1999).
- Advances in parallel and distributed simulation are increasingly being applied to complex systems that require significant run times (George et al. 1999; Low et al. 1999; Fujimoto 1998).
- Web-based simulation to exploit Web technology is expected to grow. The number of languages such as Simjava (a discrete-event simulation library for Java) is expected to increase (Triadi and Barta 1999; Narayanan et al. 1997).
- Online simulation efforts for real-time management and control of complex systems will also grow (Gonzales and Davis 1997).
- More specialized products for specific environments will be developed, e.g., TRANSIM by Los Alamos Research Laboratories to simulate transportation and urban traffic.

- Links between virtual reality and simulation will grow stronger.
- Developments in Internet applications, enterprise resource planning applications, and embedded simulations will also impact the development of simulation tools (Banks 1998b).

REFERENCES

Autosimulations Inc. (1999), AutoMod Version 9.0 Student Manual, Bountiful, UT.

Balci, O. (1996), *Visual Simulation Environment User's Guide,* Orca Computer, Inc., Blacksburg, VA.

Balci, O. (1998), "Verification, Validation, and Testing," in *Handbook of Simulation,* J. Banks, Ed., John Wiley & Sons, New York, pp. 335–393.

Banks, J. (1998a), "Software for Simulation," in *Handbook of Simulation,* J. Banks, Ed., John Wiley & Sons, New York, pp. 813–835.

Banks, J. (1998b), "The Future of Simulation Software: A Panel Discussion," in *Proceedings of the Winter Simulation Conference,* (Washington, DC), D. J. Medeiros, E. F. Watson, J. S. Carson, and M. S. Manivannan, Eds., pp. 1681–1687.

Banks, J., and Gibson, R. R. (1996), "Getting Started in Simulation Modeling," *IIE Solutions,* Vol. 28, No. 11, pp. 34–39.

Banks, J., and Gibson, R. R. (1998), "Simulation Evolution," *IIE Solutions,* Vol. 30, No. 11, pp. 26–29.

Banks, J., Carson, J. S., and Sy, J. N. (1995), *Getting Started with GPSS/H,* 2nd Ed., Wolverine Software Corporation, Annandale, VA.

Banks, J., Carson, J. S., and Nelson, B. L. (1996), *Discrete Event Simulation,* 2nd Ed., Prentice-Hall, Inc., Englewood Cliffs, NJ.

Barnes, C. D., and Laughery, K. R. (1998), "Advanced Uses for Micro Saint Simulation Software," in *Proceedings of the Winter Simulation Conference,* (Washington, DC), D. J. Medeiros, E. F. Watson, J. S. Carson, and M. S. Manivannan, Eds., pp. 271–274.

CMS Research Inc. (1999), *MAST Simulation Environment Demonstration Manual,* Oshkosh, WI.

Crain, R. C. (1998), "Simulation with GPSS/H," in *Handbook of Simulation,* J. Banks, Ed., John Wiley & Sons, New York, pp. 235–240.

Davis, L., and Williams, G. (1994), "Selecting a Manufacturing Simulation System," *Integrated Manufacturing Systems,* Vol. 5, No. 1, pp. 23–32.

Donald, D. L. (1998), "A Tutorial on Ergonomic and Process Modeling Using QUEST and IGRIP," in *Proceedings of the Winter Simulation Conference,* (Washington, DC), D. J. Medeiros, E. F. Watson, J. S. Carson, and M. S. Manivannan, Eds., pp. 297–302.

Ernst, A. T., Krishnamoorthy, M., Nott, H., and Sier, D. (1999), "Micro Saint," *OR/MS Today,* April, pp. 58–62.

Fujimoto, R. M. (1998), "Parallel and Distributed Simulation," in *Handbook of Simulation,* J. Banks, Ed., John Wiley & Sons, New York, pp. 429–464.

Garnett, J. (1999), "The Last Word on Simulation," *IIE Solutions,* Vol. 31, No. 1, pp. 45–47.

George, A. D., Fogarty, R. B., Markwell J. S., and Miars M. D. (1999), "An Integrated Simulation Environment for Parallel and Distributed System Prototyping," *Simulation,* Vol. 72, No. 5, pp. 283–294.

Glover, F., and Kelly, J. P. (1999), "Combining Simulation and Optimization for Improved Business Decisions," Systems Modeling Corporation, Sewickley, PA.

Goble, J., and Wood, B. (1998), "MODSIM III: A Tutorial with Advances in Database and HLA Support," in *Proceedings of the Winter Simulation Conference,* (Washington, DC), D. J. Medeiros, E. F. Watson, J. S. Carson, and M. S. Manivannan, Eds., pp. 199–204.

Gonzales, F. G., and Davis, W. J. (1997), "A Simulation Based Controller for Distributed Discrete Event Systems with Application to Flexible Manufacturing," in *Proceedings of the Winter Simulation Conference,* (San Diego), S. Andradottir, K. J. Healy, D. H. Withers and B. L. Nelson, Eds., pp. 845–852.

Harrell, C. R., and Tumay, K. (1994), *Simulation Made Easy,* IIE Press, Norcross, GA.

Heflin, D. L., and Harrell, C. R. (1998), "Simulation Modeling and Optimization Using ProModel," in *Proceedings of the Winter Simulation Conference,* (Washington, DC), D. J. Medeiros, E. F. Watson, J. S. Carson, and M. S. Manivannan, Eds., pp. 191–197.

Hlupic, V. (1997), "Simulation Software Selection Using SimSelect," *Simulation,* Vol. 69, No. 4, pp. 231–239.

Hlupic, V., and Paul, R. J. (1995), "Manufacturing Simulators and Possible Ways to Improve Them," *International Journal of Manufacturing Systems Design,* Vol. 2, No. 1, pp. 1–10.

Hullinger, D. G. (1999), "Taylor Enterprise Dynamics," Internal Report, F&H Simulations Inc., Orem, UT.

Industrial Modeling Corporation (1999), *SimEngine Trial Edition Guide,* Seattle.

Institute of Industrial Engineers (1999), "Buyer's Guide: Simulation Software," *Solutions,* Vol. 31, No. 5, pp. 52–59.

Keller, P., and Tchernev, N. (1997), "Object Oriented Methodology for FMS Modeling and Simulation," *International Journal of Computer Integrated Manufacturing,* Vol. 10, No. 6, pp. 405–434.

Kelton, W. D., Sadowski, R. P., and Sadowski, D. A. (1998), *Simulation with Arena,* WCB/McGraw-Hill, New York.

Kiviat, P. J., Markowitz, H., and Villanueva, R. (1969), *SIMSCRIPT II Programming Language,* Prentice-Hall, Englewood Cliffs, NJ.

Krepchin, I. P. (1988), "We Simulate All Major Projects," *Modern Materials Handling,* August, pp. 83–86.

Laguna, M. (1997), "Optimization of Complex Systems with OptQuest," Graduate School of Business Management, University of Colorado, Boulder.

Laughery, R. (1999), "Using Discrete-Event Simulation to Model Human Performance in Complex Systems," in *Proceedings of the Winter Simulation Conference,* (Phoenix), P. A. Farrington and H. B. Nembhard, Eds.

Law, A. M., and Kelton, W. D. (1991), *Simulation Modeling and Analysis,* 2nd Ed., McGraw-Hill, New York.

Lewellen, M., and Tumay, K. (1998), "Network Simulation of a Major Railroad," in *Proceedings of the Winter Simulation Conference,* (Washington, DC), D. J. Medeiros, E. F. Watson, J. S. Carson, and M. S. Manivannan, Eds., pp. 1135–1138.

Low, Y., Lim, C., Cai, W., Huang, S., Hsu, W., Jain, S., and Turner, S. J. (1999), "Survey of Languages and Runtime Libraries for Parallel Discrete-Event Simulation," *Simulation,* Vol. 72, No. 3, pp. 170–186.

Musselman, K. J. (1998), "Guidelines for Success," in *Handbook of Simulation,* J. Banks, Ed., John Wiley & Sons, New York, pp. 721–743.

Nance, R. E. (1996), "A History of Discrete Event Programming Languages," in *History of Programming Languages,* T. J. Bergin, and R. Gibson, Eds., ACM Press, New York, and Addison-Wesley, Reading, MA, pp. 369–427.

Narayanan, S., Schneider, N. L., Patel, C., Carrico, T. M., and DiPasquale, J. (1997), "An Object Based Architecture for Developing Interactive Simulations Using Java," *Simulation,* Vol. 69, No. 3, pp. 153–171.

Nikoukaran, J., Hlupic, V., and Paul, R. J. (1998), "Criteria for Simulation Software Evaluation," in *Proceedings of the Winter Simulation Conference,* (Washington, DC), D. J. Medeiros, E. F. Watson, J. S. Carson, and M. S. Manivannan, Eds., pp. 399–406.

Nisanci, A. (1997), "Modeling of FMS Pallet/Fixture Contention Rules," in *Proceedings of Summer Computer Simulation Conference,* (Arlington, VA), M. S. Obaidat and J. Illgen, Eds., Society for Computer Simulation International, San Diego, pp. 323–326.

Nisanci, A. (1998), "Comparative Analysis of AGV and Multi-Layer Conveyor Systems Using Simulation," in *Proceedings of Summer Computer Simulation Conference,* (Reno, NV), M. S. Obaidat, F. Davoli, and D. DeMarinis, Eds., Society for Computer Simulation International, San Diego, pp. 15–20.

Nisanci, A. (1999), "Multi-Project and Multi-Resource Lot Sizing and Scheduling," in *Proceedings of Summer Computer Simulation Conference,* (Chicago), M. S. Obaidat, A. Nisanci, and B. Sadoun, Eds., Society for Computer Simulation International, San Diego, pp. 131–136.

Nordgren, W. B. (1998), "Taylor II Manufacturing Simulation Software," in *Proceedings of the Winter Simulation Conference,* (Washington, DC), D. J. Medeiros, E. F. Watson, J. S. Carson, and M. S. Manivannan, Eds., pp. 263–267.

Norman, V., and Banks, J. (1998), "Managing the Simulation Project," in *Handbook of Simulation,* J. Banks, Ed., John Wiley & Sons, New York, pp. 745–764.

O'Reilly, J. J., and Lilegdon W. R. (1999a), "Introduction to AweSim," in *Proceedings of the Winter Simulation Conference,* (Phoenix), P. A. Farrington and H. B. Nembhard, Eds.

O'Reilly, J. J., and Lilegdon, W. R. (1999b), "Introduction to FACTOR/AIM," in *Proceedings of the Winter Simulation Conference,* (Phoenix), P. A. Farrington and H. B. Nembhard, Eds.

Pegden, C. D., Shannon, R. E., and Sadowski, R. P. (1995), *Introduction to Simulation Using SIMAN,* 2nd Ed., McGraw-Hill, New York.

Phillips, T. (1998a), "AutoMod by Autosimulations," in *Proceedings of the Winter Simulation Conference,* (Washington, DC), D. J. Medeiros, E. F. Watson, J. S. Carson, and M. S. Manivannan, Eds., pp. 213–218.

Phillips, T. (1998b), "AutoSched AP by Autosimulations," in *Proceedings of the Winter Simulation Conference,* (Washington, DC), D. J. Medeiros, E. F. Watson, J. S. Carson, and M. S. Manivannan, Eds., pp. 219–222.

Plott, B., Wojciechowski, J. Q., and Kilduff, P. A. (1999), "Command and Control Human Performance Modeling," *CSERIAC GATEWAY,* Vol. 10, No. 1, pp. 10–11.

Pritsker, A. A. B. (1995), *Introduction to Simulation and SLAM II,* 4th Ed., John Wiley & Sons, New York.

Pritsker, A. A. B., and O'Reilly, J. J. (1998), "Introduction to AWESIM," in *Proceedings of the Winter Simulation Conference,* (Washington, DC), D. J. Medeiros, E. F. Watson, J. S. Carson, and M. S. Manivannan, Eds., pp. 249–256.

Pritsker, A. A. B., and O'Reilly, J. J. (1999), *Simulation with Visual SLAM and AweSim,* John Wiley & Sons, New York.

Pritsker Corporation (1993), *Simulation: A Decision Support Tool,* Pritsker Corporation, West Lafayette, IN.

ProModel Corporation (1999), *User's Guide,* Orem, UT.

Roberts, C. A., and Dessouky, Y. M. (1998), "An Overview of Object Oriented Simulation," *Simulation,* Vol. 70, No. 6, pp. 359–368.

Rodrigues, J. M. (1994), *Directory of Simulation Software,* Society for Computer Simulation, San Diego.

Rohrer, M., and Banks, J. (1998), "Required Skills of a Simulation Analyst," *IIE Solutions,* Vol. 30, No. 5, pp. 20–23.

Russell, E. C. (1983), *Building Simulation Models with SIMSCRIPT II.5,* CACI Inc., Los Angeles.

Sadowski, D., Bapat, V., and Drake, G. (1998), "The Arena Product Family: Enterprise Modeling Solutions," in *Proceedings of the Winter Simulation Conference,* (Washington, DC), D. J. Medeiros, E. F. Watson, J. S. Carson, and M. S. Manivannan, Eds., pp. 205–212.

Schriber, T. J. (1991), *An Introduction to Simulation Using GPSS/H,* John Wiley & Sons, New York.

Schriber, T. J. and Brunner, D. T. (1998), "How Discrete Event Simulation Works," in *Handbook of Simulation,* J. Banks, Ed., John Wiley & Sons, New York., pp. 765–811.

Schwab, R. E. (1987), "Development of Simulation at Caterpillar Inc.," Internal Report, Peoria, IL.

Schwab, R. E., and Nisanci, H. I. (1992), "Simulation Languages," in *Handbook of Industrial Engineering,* 2nd Ed., G. Salvendy, Ed., John Wiley & Sons, New York.

Siprelle, A. J., Phelps, R. A., and Barnes, M. M. (1998), "SDI Industry: An Extend Based Tool for Continuous and High Speed Manufacturing," in *Proceedings of the Winter Simulation Conference,* (Washington, DC), D. J. Medeiros, E. F. Watson, J. S. Carson, and M. S. Manivannan, Eds., pp. 349–358.

Smith, J. M. (1987), *Mathematical Modeling and Digital Simulation for Engineers and Scientists,* John Wiley & Sons, New York.

Snowdon, J. L., El-Taji, S., Montevecchi, M., MacNair, E., Callery, C. A., and Miller, S. (1998), "Avoiding the Blues for Airline Travelers," in *Proceedings of the Winter Simulation Conference,* (Washington, DC) D. J. Medeiros, E. F. Watson, J. S. Carson, and M. S. Manivannan, Eds., pp. 1105–1112.

Symix Systems/Pritsker Division (1999), *AweSim Demonstration Software Guide, Version 3.0,* West Lafayette, IN.

Symix Systems/Pritsker Division (1997), *FACTOR AIM: Working Model Primer, Version 8.0,* West Lafayette, IN.

Triadi, M. N., and Barta, T. M. (1999), "Experience with a Web-Based Discrete-Event Simulator," in *Proceedings of Summer Computer Simulation Conference,* (Chicago), M. S. Obaidat, A. Nisanci, and B. Sadoun, Eds., *Society for Computer Simulation International,* San Diego, pp. 93–96.

Tumay, K., and Harrington, H. (2000), Simulation Modeling Methods, McGraw-Hill, New York, NY.

Tumay, K., and Wood, B. (1999), ‘‘MODSIM III and Its Applications,’’ in *Proceedings of the Winter Simulation Conference,* (Phoenix), P. A. Farrington and H. B. Nembhard, Eds. Visual Thinking International Inc. (1999), *SIMUL8 Manual and Simulation Guide,* Reston, VA.

Visual Thinking International Inc. (1999), *SIMUL8 Manual and Simulation Guide,* Restin, VA.

Wildberger, A. M. (1999), ‘‘AI and Simulation,’’ *Simulation,* Vol. 72, No. 2, pp. 115–116.

Yu, L. C., Steinman, J. S., and Blank, G. E. (1998), ‘‘Adapting Your Simulation for HLA,’’ *Simulation,* Vol. 71, No. 6, pp. 410–420.