Proceedings of the 1990 Winter Simulation Conference
Osman Balci, Randall P. Sadowski, Richard E. Nance (eds.)

OBJECT ORIENTED VISUAL INTERACTIVE SIMULATION

Ranko Vujosevic

Department of Industrial Engineering
College of Engineering
The University of Iowa
Iowa City, IA 52242

ABSTRACT

The development of systems that provide visual interactive modeling and simulation capabilities represents one of the major directions in the simulation area. Characteristics of such a system, which include visual interactive model development, visualization of the simulation output, user-model interaction, and intelligent advisoring are discussed. The implementation of defined concepts in an experimental object oriented system for visual interactive simulation of flexible manufacturing systems is presented.

1. INTRODUCTION

The Visual Interactive Simulation (VIS) is a paradigm which requires the extensive participation of the user in all phases of a simulation experiment. A VIS system must provide visual capabilities for graphical display of a simulation model and animation of simulated activities. It increases the user's ability to control the simulation process. The VIS systems are becoming increasingly popular. They are developed to help a user who is not a simulation expert to conduct the simulation experiment successfully. This feature is very important since there is a rather small number of simulation experts compared to the number of problems that require the use of simulation. The assumption that the user is not a simulation expert is a potential source of problems, as stated by Law and McComas [1986]. Wrong decisions can be made in various modeling and simulation activities, such as: model development and validation, simulation input data specification, interpretation of simulation results, and so forth. To be considered a fully VIS system, it should provide advisory capabilities to help the user to overcome the lack of the simulation and/or problem domain knowledge.

In general, the basic capabilities that should be provided in a VIS system include:

1) Visual interactive modeling. The user must be able to develop a simulation model by creating the graphical description of the system to be simulated. Both the graphical representation of the system layout and the simulation model could be stored in a simulation model base and be used for developing future models.

2) Visualization of the simulation output. Changes in the state of a simulation model during the simulation experiment are animated. Corresponding changes in the simulation statistics are displayed graphically as well. The simulation results could be analyzed by the user or by an expert system.

3) User - model interaction. A VIS system must provide the user with mechanisms to stop the simulation, change some modeling and/or simulation parameters, and continue the simulation.

4) Intelligent advisoring. On-line and off-line intelligent advisory capabilities provide the user with guidelines how to accomplish modeling and simulation activities.

According to O'Keefe [1987], to be considered as a VIS system it must provide characteristics described in 2 and 3. The first three characteristics are usually described in literature on VIS [Hurrion 1986; O'Keefe 1987]. Most of the commercial VIS systems are advertised as "easy-to-use", "no-programming-required", or "you-do-not-have-to-be-a-simulation-expert" systems. But, as far as we know, only INSIGHT [Roberts 1989] has limited capabilities to help a "non-expert" coping with simulation and modeling problems. We find advisory capabilities very important and think that they must be provided as well.

Bell and O'Keefe [1987] reviewed the historical development of VIS systems since the introduction of the original idea by Hurrion [1978]. O'Keefe [1989] emphasized the suitability of VIS systems for simulation of complex systems, which require human decision making. The user is able to make certain decisions being provided with an animated display of a simulation experiment during its execution. He emphasized that "typically visual interactive simulation is successful where the interacting decision maker has to make some global decisions regarding the system, and really only needs an overall feel of performance. It is less effective when, for instance, being used to design specific decision rules due to the limited sample that is given by interaction with a few runs and the difficulty in analyzing the effect of interactions". In earlier work [O'Keefe 1987], methodologies for supporting the decision making process were described.

In this paper, the implementation of VIS concepts is illustrated with the SIMFLEX, an experimental object-oriented VIS system for simulation of Flexible Manufacturing Systems (FMSs). The blackboard problem solving model is used to support the decision making process. It allows for existence of a set of knowledge sources for solving problems which require the expert knowledge.

In the next section, the object oriented programming rationale for discrete event simulation is discussed. An object oriented framework for developing VIS systems is described in Section 3. The rest of the paper is a detailed discussion of the VIS characteristics and their implementation.

2. OBJECT ORIENTED PROGRAMMING AND DISCRETE-EVENT SIMULATION

Object oriented programming (OOP) is a way of structuring computer programs by defining a set of objects capturing information about corresponding entities that are of interest to the modeler. Information processing is performed through the communication between objects, which send messages to each other. Object oriented problem solving paradigm is based on building a solution procedure by specifying a sequence of messages to be sent to a particular object. Implementing object oriented paradigm for solving a specific problem can be summarized as a four-step procedure [Pinson and Wiener 1988]: problem definition, identification of the domain objects, identification of messages to be sent to those objects, and definition of a problem solving sequence of messages.

OOP is increasingly used for discrete event simulation (DES). Both advantages and disadvantages of OOP for developing DES systems are discussed in literature. Zeigler [1987] described compatibility between the OOP paradigm and DES. In a recent work [Zeigler 1990], the development of an approach to object oriented DES based on the concept of hierarchical, modular model construction is described. Limitations of OOP are reviewed by Rothenberg [1986] and identified as the lack of the following capabilities: modeling power, control, representation, comprehensibility, and model building.

A variety of OOP languages have been used for developing DES systems, such as: C++ [Blair and Selvaray 1989; Eldredge et al. 1990; Ali and Wyatt, 1990], MODSIM II [Bryan, 1989], Simula [Nyen, 1987], and Smalltalk [Knapp 1986, 1987; Van der Meulen 1987].

Manufacturing systems are considered as a natural application for the message - passing characteristics of the OOP paradigm. The

system abstraction is achieved by creating a set of classes representing corresponding physical components that might be included into an FMS design. An FMS model can be created by using one of two approaches for modeling of discrete systems in an OOP environment, as defined by Bezivin [1987]:

1. Network decomposition - Simulation objects communicate to each other by sending messages.

2. Client-Server decomposition - Clients are active simulation objects while servers are passive ones. A client may send messages to servers, but the reverse case is not possible.

For example, by applying network decomposition for FMS modeling, workstations can be modeled as objects and parts as messages to be sent to workstations. On the other hand, following client -server decomposition, parts can be modeled as active objects (clients), while workstations would be passive objects (servers). Parts send messages to workstations in order to be processed.

A number of Smalltalk applications for simulation of manufacturing systems have been reported. Thomasma et al. [1990a] gave an excellent overview of Smalltalk applications in manufacturing simulation. Some of the reviewed simulation systems have VIS capabilities. Thomasma et al. [1990b] described SmartSim, a VIS system for simulation of manufacturing systems. SmartSim uses Smalltalk capabilities for development of an interactive user interface for defining the physical configuration of a system, and specifying behavior of simulation objects and relationships between them. Beaumariage and Mize [1990] described an object oriented environment for modeling of manufacturing systems. In a recent paper, Guo et al. [1990] presented an object oriented system for flexible manufacturing in which the simulation is integrated with a manufacturing information system.

3. A FRAMEWORK FOR DEVELOPMENT OF AN OBJECT ORIENTED VIS SYSTEM

The very first step in development of an object-oriented VIS system is the definition of a class hierarchy which supports implementation of visual interactive modeling and simulation capabilities. The Model-View-Controller (MVC) programming paradigm implemented in Smalltalk-80 [Krasner and Pope, 1988] provides mechanisms for development of a VIS system. The MVC allows for building interactive applications by providing mechanisms for capturing information and operations in a particular problem domain (Model); graphical display of the domain information (View); and control of the user - problem domain interaction (Controller). The Smalltalk-80 class hierarchy implemented in SIMFLEX is shown bellow.

```
Object
    Model
        FMSModel
    View
        Canvas
            FMSView
    Controller
        MouseMenuController
            CanvasController
                FMSController
    Workstation
        Machine
    Part
        PartBuffer
    AutomatedGuidedVehicle
    Robot
    Distribution
    Magnitude
        Time
            SimulationTime
    Neuron
```

More modeling and simulation capabilities could be included by either defining new methods in existing classes (for example, operational and control capabilities), or creating new classes to be included in the class hierarchy (for example, a class for representing a material storage and retrieval system).

The information about a particular FMS simulation model and activities performed during a modeling and simulation process are contained in the class **FMSModel**, which is a subclass of the class **Model**. The class **FMSView** displays graphical description of the FMS simulation model and animated display during the simulation experiment. It is a subclass of the class **Canvas** created to facilitate display of a system layout which is larger then the initial screen size. The class **FMSController** controls the user-model interaction. It is a subclass of the class **CanvasController**, which allows for scrolling of a graphical window. Classes **Workstation**, **Machine**, **Part**, **PartBuffer** and **AutomatedGuidedVehicle** represent corresponding components of an FMS. These classes are instantiated as many times as needed. Different distribution functions usually used in simulation of FMSs are provided by the class **Distribution**. The MVC interaction in SIMFLEX is presented in Figure 1, adapted from Krasner and Pope [1988].

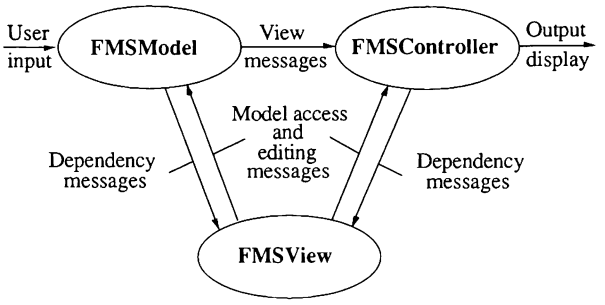


Figure 1. The Model-View-Controller Interaction in SIMFLEX

The characteristics of a VIS system, described in the introductory section, require a sophisticated user interface which supports definition of input data and simulation objectives, graphical representation of the system layout, visual representation of the simulation model, interactive model modification, display of simulation results, animation of simulated activities, and user-model interaction. The MVC supports development of such a user interface. The SIMFLEX user interface consists of a set of interactive buttons for activation of modeling and simulation activities and manipulation of the simulation model base, a window for displaying icon-based system layout representation, a window for displaying the visual simulation model representation, and a window for displaying simulation results and various messages.

A set of interactive buttons is used for guiding the modeling and simulation process. A button is an instance of the class **SwitchView** having instances of the class **OneOnSwitch** as models. An instance of the class **Object** serves as a connector to make sure that only one button is activated at a time. The window for icon - based representation of an FMS physical layout is an instance of the class **FMSView**. The model is an instance of the class **FMSModel** and the controller is an instance of the class **FMSController**. This window is defined as a **Canvas** (an instance of the class **Form**) on which an FMS layout is drawn. The visual representation of an FMS simulation model is in the second window. The model for this window is an instance variable **currentModel** of the class **FMSModel**. The window for textual messages and outputs is an instance of the class **TextCollector**.

The described framework provides mechanisms for the implementation of VIS characteristics defined in the introductory section.

4. VISUAL INTERACTIVE MODELING

Simulation modeling is a process of creating a simplified computer representation of the system to be simulated. The fundamentals of simulation modeling are described in numerous literature. An excellent overview of simulation modeling is given by Rothenberg (1989). In this paper, we are concerned with how the simulation modeling should be supported in a VIS system.

The way in which a VIS system supports simulation modeling has a great importance for the final user. Considering modeling capabilities of existing VIS systems, they could be divided into three groups:

1) General purpose VIS systems that require knowledge about

the programming and/or simulation language in which a VIS system is implemented. This requirement is a major drawback of this kind of VIS systems. However, they offer great application flexibility. Examples of such systems include CINEMA (Siman) [Poorte and Davis 1989], SIMGRAPHICS (Simscript II.5) [Bryan 1989], and SEE/WHY (Fortran) [Gilman and Billingham 1989].

2) General and specific purpose VIS systems that provide graphical capabilities for model development, but the system layout, needed for animation, is developed separately. A graphical representation scheme is used for describing simulation models. Most often the network diagrams are used, as it is implemented in TESS [Grant and Starks 1988] and INSIGHT [Roberts and Flanigan 1989]. WITNESS [Gilman and Billingham 1989] is an example of such a system for simulation of manufacturing systems.

3) Specific purpose VIS systems which provide icon-based graphical capabilities for parallel model and system layout development. These systems, most often developed for manufacturing application, are described as easy-to-use VIS systems for rapid modeling. The graphical display of the system layout is used for animation. These features can be found in XCELL+ [Conway and Maxwell 1986] and SIMFACTORY II.5 [Rohrbough 1989].

The research presented in this paper involves an object oriented implementation of modeling capabilities described for the last group of VIS systems. An attempt towards the definition of a more "intelligent" model development framework has been made. The framework is presented by describing the method of modeling knowledge representation, simulation model base, and model development capabilities.

4.1 Modeling Knowledge Representation

Various methods have been used for the computer representation of a simulation model. There is an increasing use of AI techniques for modeling knowledge representation. Elzas [1986] gave an overview of the applicability of AI techniques to knowledge representation in modeling and simulation.

In SIMFLEX, the domain modeling knowledge is encoded into an associative (semantic) network by using the NRL, a Network Representation Language [Antao 1988] developed in Smalltalk-80. Reasons for applying a semantic network for this purpose are as follows:

1) It supports the hierarchical model development.

2) It serves as a blackboard data structure for various knowledge sources implemented.

3) A knowledge network can be stored in the simulation model base and used for developing future models.

4) The hypertext user interface paradigm, provided in NRL, supports the user-model interaction, which is one of the most important features of a VIS system.

NRL provides a set of representation primitives (nodes and links) used for building a semantically consistent representation of an FMS model. Nodes are primitives that represent the knowledge encoded in the network. Nodes can represent entire concept descriptions (concept nodes), attributes belonging to a concept (attribute nodes), or instances of a generalized concept (instance nodes). For example, a node may represent concept **Machine** and capture knowledge about that particular generalized concept. A set of attributes assigned to this generalized concept is represented using attribute nodes. A machine object, an instance of the concept **Machine**, is represented by an instance node. In a semantic network, links represent associations between concepts.

When a concept node is created, an object representing the corresponding FMS component is generated. The object becomes a part of the model. Supplementary information is provided by attribute nodes that are aggregated within multiple contexts assigned to a concept node [Antao et al. 1989]. Contexts provide a mechanism for defining different sets of attributes about a concept for various user categories. The underlying representation is presented to the user as a labeled graph, as shown on a simple example in Figure 2.

The NRL interface, based on the hypertext paradigm, allows for [Antao et al. 1988]: viewing the entire network visually, displaying parts of the network, browsing through the network by tracing associations to other nodes, storing and displaying augmented descriptions at nodes, and interactively modifying the network structure by creating new links or deleting existing links.

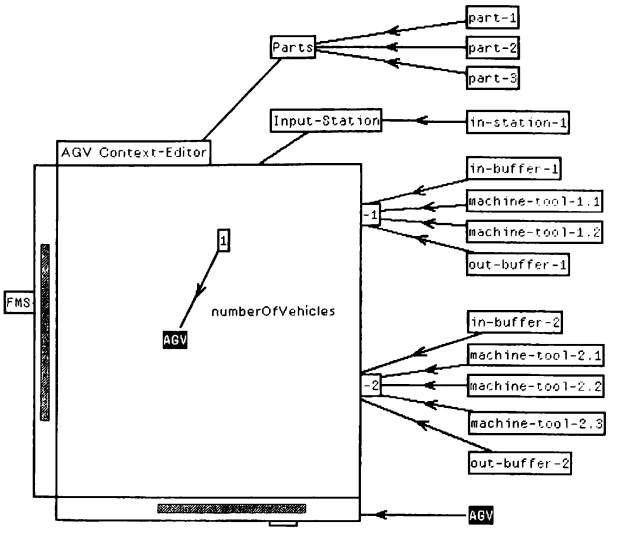


Figure 2. Visual Representation of an FMS Model

4.2 Simulation Model Base

When a simulation experiment has been performed, the user may want to save the simulation model. That model can be later used for running another simulation experiment, or for building another simulation model. This can significantly reduce the model development time. To provide this capability, a simulation model base must be maintained. It involves capabilities for storing and retrieving models from the model base and searching the model base. Kettenis [1986] discussed the importance of being able to access simulation models already developed.

Storing a simulation model in a VIS system assumes storing graphical representations of the system layout and the model. The simulation model base is implemented in Smalltalk-80 as a global variable stored in the **Smalltalk** system dictionary. For example, an FMS simulation model is stored by storing an instance of the class **FMSModel** in the global variable **ModelBase**, as shown below.

```
SaveAnFMSModel
    | modelName |
    modelName <- (FillInTheBlank request: 'Enter name of the
                    new FMS model').
    self model name: modelName.
    (Smalltalk at: #ModelBase) at: modelName put: (self
model).
```

Various approaches for retrieving and using an existing model stored in the model base are discussed in the next section.

4.3 Simulation Model Development

Simulation is usually used for either simulation of an existing system or for the design of a new system. The VIS modeling capabilities must support both cases.

4.3.1 Modeling of an Existing System

For this purpose the user is provided with icon-based graphical modeling capabilities. That allows the user to generate two representations of the system at the same time: (1) external (visual) representation, as a real-like icon-based picture on the screen, and (2) internal (logical) representation, as a set of objects representing corresponding system components, which build the simulation model.

The modeling process starts with defining general characteristics of the system. A particular component is modeled by activating the corresponding interactive button. The user is asked to define characteristics of the component and then to place an icon, represent-

ing that component, on the screen in the proper location. The icons are scaled according to the defined physical size of the system. The graphical representation of the system generated in such a way is later used for the animation purpose.

The simulation model itself is generated in parallel. The user is asked only to place nodes for the visual representation of the semantic network used for the modeling knowledge representation.

4.3.2 Modeling of a New System

The design requirements and specifications define an ideal model of the FMS, which is later compared to alternative designs developed during the conceptual design phases. Sadowski [1989] described the level of complexity of the model needed for various scenarios in design of a new manufacturing system. We discuss how such a model could be developed.

The first step in modeling a design alternative should be searching the simulation model base, which consists of overall simulation models already developed, and primitive models. The overall models are searched to find a model that can be accepted as the conceptual model. Two ways of performing this activity are described.

1. Human decision making. The user selects an existing model through a menu-driven dialogue. The user is able to list the model base and select models interactively. Characteristics of a selected model are displayed graphically and numerically. Modifications of an existing model, such as replacing or deleting icons are supported as well. Afterwards, the modified model can be stored in the model base. The advantage of this approach is flexibility obtained by using humans for the decision making process. However, the level of expertize required is significant.

2. Development of an intelligent system for model retrieval. Two ideas for development of such a system are currently being explored:

a) Using a rule-based expert system which analyzes the overall functions of the stored models. If more then one model can be selected, the ultimate selection is based on some criteria defined by the designer: total cost, physical constraints etc.

b) Using a neural network based model retrieval system. The existing simulation models are stored in an associative memory. The ideal model is then classified into one of the categories, based on which of a number of stored patterns (overall models) it most resembles.

A simulation model may also be built by performing the model synthesis using a set of primitive models. For that purpose, the approach for building hierarchical, modular discrete-event models, proposed by Zeigler [1987], can be applied. That approach is used for development of an intelligent system for model synthesis [Kusiak et al. 1990]. An outcome of the model synthesis process is shown in Figure 3.

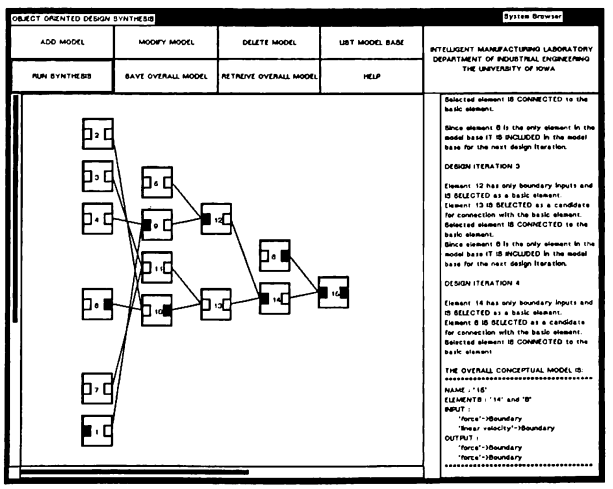


Figure 3. An Outcome of the Model Synthesis

5. SIMULATION AND ANIMATION OF SIMULATED ACTIVITIES

In this section, the development of an object oriented simulation driver is described. Then we discuss the importance of visualization of simulation statistics and animation of simulated activities for the user-model interaction. Finally, a neural network based approach for analysis of simulation results is presented.

5.1 Development of an Object Oriented Simulation Driver

An FMS model is executed by the simulation driver which consists of a number of methods defined under the **simulation** category in the class **FMSController**. The simulation driver runs the simulation model, gathers statistics, maintains a simulation clock and event list, displays simulation output, and performs concurrent animation.

One of the main issues to be addressed in building an object oriented simulation driver is the representation of the list of simulation events and part buffers within an FMS. An event list is implemented as an instance of the class **SortedCollection**, and contains events sorted according to the simulation time they occur. An event from the list is an **Association**, where the key is the simulation time the event occurs, and the value is an **Array** with the following event attributes: event type, part type, number of the workstation required to process next operation on the part, machine number within the workstation, operation type, and AGV identification number.

For example, the event (25.5 -> (1 2 1 1 transfer 2)) means that at the simulation time of 25.5 time units, part type 2 should be transfered by vehicle 1 to workstation 1 where machine 2 will perform operation 1. Some events do not require all of these attributes. The method **scheduleNextEvent** schedules events in the event list using a minimum simulation time criteria. This means that next event is always first one on the event list.

Part buffers are implemented as instances of the class **OrderedCollection**, and are maintained using the First-In-First-Out rule. The simulation clock, advanced according to the simulation time of the next event, is implemented in the class **SimulationTime**. A variety of distribution functions used in the simulation of FMSs are provided by the class **Distribution**.

5.2 Visualization of Simulation Results

The visualization of simulation statistics during a simulation experiment is a feature that significantly increases the efficiency of the user-model interaction. This capability provides the user with an insight into the current state of a simulation model. Based on that, the user is able to observe certain problems in the model and make necessary changes. For example, during the simulation of an FMS, the user can constantly check the utilization of equipment, queue lengths, part statistics etc. If some problems are encountered (for example, equipment overutilized or underutilized), the user may stop the simulation, make necessary changes, and continue the simulation. Some VIS provide such capabilities. To mention only one, Cinema [1989] dynamically displays plots and histograms representing system states variables.

Smalltalk-80 does not have built-in capabilities to support this feature. However, a package called **PluggableGages** is developed and used for monitoring values of simulation variables in an FMS simulator [Adams, 1988]. Guo et al. [1990] use the bar gages for displaying the current number of parts in part buffers. This capability is not implemented in SIMFLEX.

5.3 Animation

Animation is an essential feature of a VIS system. The graphical representation of the system layout is used to animate simulation activities in which components of the system, represented graphically, are involved. It is a valuable support for the user-model interaction since most malfunctions of the model are apparent visually. The animation helps to establish the credibility of the simulation model, especially in the eyes of people who are not simulation experts. The animation provided in existing VIS systems is either performed in parallel with the simulation execution, or, postprocessed.

The animation is a built-in capability in Smalltalk-80. Its use for simulation of FMSs is reported by Ulgen and Thomasma [1987], Adams [1988], and [Guo et al. 1990]. The animation capabilities of SIMFLEX are implemented by using the built-in Smalltalk-80 animation mechanisms. Animated objects are divided into two groups: stationary objects (workstations and part buffers) and moving objects (vehicles).

The change in the state of a stationary object is animated by changing the color of a corresponding icon. For example, different colors are provided for various states of a workstation: processing (black), idle (white), blocked (light gray), and broken (gray). Motion of vehicles is animated using the **follow:while:** message. A snapshot taken during a simulation experiment is shown in Figure 4.

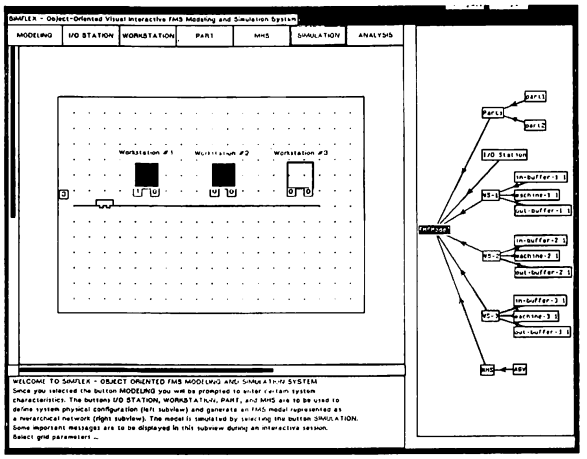


Figure 4. Animation of Simulated Activities

5.4 Analysis of Simulation Results

The simulation results must be analyzed to make necessary conclusions about the dynamic behavior of the system simulated. This is especially important when a new system is being developed. Based on the analysis of the simulation results one design alternative is accepted or modified to be used for another simulation experiment. An approach for analysis of FMS simulation results is presented [Vujosevic and Kusiak 1990b].

The system for analyzing FMS simulation results serves as a knowledge source attached to the blackboard data structure of SIMFLEX, as shown in Figure 5.

The system consists of three components:

1. A rule-based preprocessor which compares simulation results with the target values and, if necessary, generates the topology of a neural network for analysis of simulation results.
2. A neural network processor for training the neural network, reasoning about simulation results, and explanation of conclusions made.
3. A rule-based postprocessor for implementation of design modifications into the FMS simulation model.

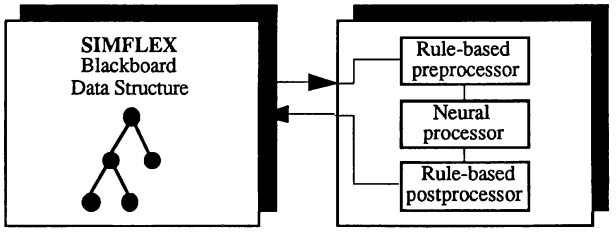


Figure 5. The Knowledge Source for Analysis of FMS Simulation Results

The rule-based preprocessor compares the production objectives and the simulation results. Based on the outcome of that comparison and the information about the FMS physical configuration, the preprocessor generates the input data to the neural processor. Information generated by this system include neurons that correspond to variables of interest, variable names, and values for input data (neurons in the input layer). The neural network used as a knowledge base for FMS diagnosis has the following characteristics:

1. It is a feedforward neural network with the input layer, one hidden layer, and the output layer. The number of neurons in each layer depends on the physical configuration of an FMS being considered. The neurons in the input layer correspond to the variables representing FMS performance measures (station utilization, material handling system utilization, and queue length). The hidden layer involves a number of neurons representing variables that describe problems that may result in an undesired FMS performance (eg., capacity problem, loading problem, etc.). The neurons in the output layer indicate the design modifications that should be implemented when a certain difficulty is encountered.

2. The neuron outputs in the input layer take three discrete values: +1, if a performance measure is above the maximum value allowed; -1, if a performance measure is below the targeted value; and 0, if a performance measure is within limits. The neuron outputs in the hidden layer and in the output layer take two discrete values: +1, if a bottleneck reason is detected (hidden layer) or if a design modification is recommended (output layer) and 0 otherwise.

Three types of rules may be identified in the rule-based preprocessor: decision rules, rules for description of the neural network topology, and rules for assigning values to neurons in the input layer.

The feedforward neural network topology, generated by the rule-based preprocessor, is the input to the neural processor developed for reasoning about FMS simulation results. The user is asked to enter a set of training examples for a particular FMS model, if the set does not exist. The perceptron learning algorithm [Wasserman 1988] is used for training the neural network.

The training process results in a set of weights assigned to the connections between neurons and biases. A knowledge base for reasoning about simulation results is obtained. The reasoning process is initiated by applying values generated by the rule-based preprocessor to neurons in the input layer. Then, the neuron outputs in the hidden layer are calculated. For each neuron the total sum of products of values of input neurons to that neuron and corresponding weights is calculated. The output of the corresponding neuron is +1 if the sum is > 0 and 0 otherwise. The results of the FMS analysis process are obtained by computing the neuron outputs in the output layer in the same manner.

The neural processor is able to explain the conclusion made by tracing down in the input layer neurons (variables) that were in favor of the conclusion obtained. The explanation of the reasoning process is given in the form of production rules.

The FMS design modification recommended is presented to the user for verification. Two cases may appear:

1. The user accepts the result of FMS analysis. The rule-based preprocessor analyzes the design modification and incorporates appropriate changes into the FMS simulation model.

2. The user does not verify the design modification. In that case, the user is able to modify an FMS simulation model using the capabilities for interactive model modification.

5.4.1 Illustrative Example

To illustrate the ideas presented, a problem of analyzing simulation results of a flexible manufacturing cell for producing a single part type is considered. The system consists of two machines, a robot for machine loading, a part buffer, and automated guided vehicle for part delivery and pick-up.

Assume that the target number of parts has not been achieved and therefore the bottleneck analysis has to be performed. Based on the FMS physical configuration, the rule-based preprocessor defines the neurons that correspond to performance measures (input layer),

possible bottleneck reasons (intermediate layer), and design modifications (output layer). The names of variables represented by neurons, and (assumed) values assigned to the input neurons based on the simulation results are as follows:

Variable name	Value
Input layer:	
1) machine 1 utilization rate	+1
2) machine 2 utilization rate	+1
3) robot utilization rate	+1
4) AGV utilization rate	0
5) queue length	+1
Intermediate layer:	
6) capacity problem	
7) loading problem	
8) transportation problem	
Output layer:	
9) increase number of machines	
10) replace robot	
11) increase number of AGVs	

The next step, performed by the neural processor, is to train the neural network using the perceptron learning algorithm. For that purpose, a small set of 8 training examples has been used. The set of training examples could be obtained from previous simulation runs or by using a heuristic knowledge.

A knowledge base for the reasoning about simulation results for this simple example of FMS analysis, with assigned weights and biases obtained from the training process, is shown in Figure 6. Although neurons are fully interconnected without feedback, for the sake of simplicity only connections with the weights not equal to zero are displayed.

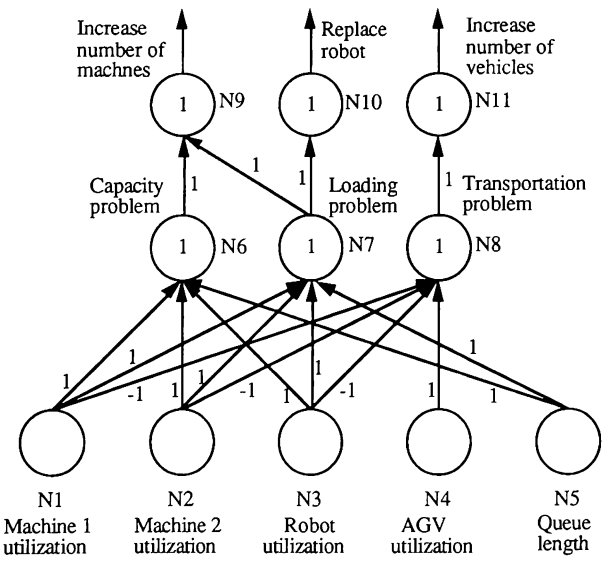


Figure 6. The Knowledge Base for Analysis of a Flexible Manufacturing Cell

Reasoning about simulation results is performed by using the input vector of values for neurons in the input layer. The output of the neurons in the hidden layer and the output layer are calculated and two conclusions made: "increase number of machines" and "replace robot".

In that case, the user can verify both conclusions, one of them, or neither one. If the conclusion "increase number of machines" is verified by the user, an instance of the class **Machine**, provided in the class hierarchy of SIMFLEX, is created and included in the simulation model. Any additional information is provided by the user.

6. USER - MODEL INTERACTION

One of the key features that must be provided in a VIS system is the user-model interaction during the execution of a simulation experiment. The user must be able to interrupt a simulation run, make changes in the model or even in the simulation driver's computer code, and continue simulation. Changes should stay recorded in the model even after simulation is completed. Schriber [1987] discussed the aspects of the user-model interaction. For simulation of FMSs, this feature is especially useful for checking different control strategies.

Smalltalk-80 is an ideal environment for implementing these features. The simulation experiment can be interrupted in some predefined points, or arbitrarily, according to the user's observation that some intervention is necessary. Smalltalk-80 provides special windows called **Debuggers** that allow inspecting objects. The debugger can be used either when the message **self halt** is encountered, when a user interrupt is signalled, or when a run-time error appears [Pinson and Wiener 1988].

The message **self halt** may be used for interrupting execution of a simulation run in predefined points. When necessary changes have been made, the simulation can be continued by selecting the menu option **proceed** available in the debugger's yellow button menu.

A user interrupt can be signalled arbitrary by using **control-c** key combination.

7. INTELLIGENT ADVISORING

The assumption made in the development of a VIS system is that the user is not a simulation expert, or even not an expert in the application area. In addition, the user might not be familiar with the programming language in which a VIS system is implemented. Such a user should be able to accomplish modeling and simulation activities efficiently. This can be achieved only if the VIS system used provides intelligent advisory capabilities. If this characteristic is not provided, a simulation non-expert is likely to make errors. On the other hand, the lack of the problem domain knowledge makes the simulation experiment practically impossible. In any case, errors made in activities, such as model development, model validation, simulation input definition, interpretation of simulation results, and so on, lead to an invalid simulation experiment.

A way of incorporating the intelligent advisory capabilities in a VIS system is presented and illustrated by describing its implementation in SIMFLEX. It assumes the existence of on-line and off-line advisory capabilities.

7.1 On-line Intelligent Advisoring

The on-line advisoring capabilities are realized by providing a set of knowledge sources attached to the blackboard data structure (further refereed as blackboard). The blackboard is the hierarchical associative network that contains FMS modeling knowledge. The knowledge sources "suggest" the user solutions of particular problems and write solutions or recommendations on the blackboard, if accepted by the user. The system for analyzing simulation results, presented in the section 4.4 is an example of such a knowledge source. To illustrate suitability of the applied blackboard model, which allows for existence of a set of knowledge sources implemented in different knowledge representation paradigms, and using different problem solving methods, we describe two knowledge sources used for on-line advisoring: an expert system for machine layout and/or material handling system (MHS) type selection, and a neural network algorithm for solving the machine layout problem.

The knowledge source for machine layout and/or MHS type selection performs following activities:

1. Suggests a machine layout type and/or a material handling system if the user does not define these parameters.
2. Checks both parameters if entered by the user. For example, if the user selects a circular layout type and an automated guided vehicle system, the expert system suggests that a robot should be selected for that machine layout type.

The expert system, implemented in the HUMBLE rule based

expert system shell [Piersol 1987], takes the data from the blackboard using an interrogator and updates the blackboard with new data - the FMS layout type and/or material handling system type.

HUMBLE, an expert system shell implemented in Smalltalk-80, allows development of rule based expert systems. The expert system for machine layout and/or material handling system type selection consists of rules about machine layout and material handling system entities, as suggested by Kusiak and Heragu [1988]. Sample rules from the HUMBLE data base are presented bellow.

if: (selectedLayoutType = 'unknown'
& (selectedMHSType = 'agv'))
then: [suggestedLayoutType is: 'linear row'].

if: (selectedLayoutType = linear single row' |
(selectedLayoutType = 'linear double row')
& (selectedMHSType = 'unknown'))
then: [suggestedMHSType is: 'agv'].

The knowledge base is attached to the blackboard in a manner similar to that suggested by Piersol [1987]. To attach a HUMBLE knowledge base to a blackboard, an interrogator which reads the blackboard must be created. Three messages **oneOf:**, **moreOf:**, and **request:** are the means by which a HUMBLE knowledge base interacts with its interrogator. For this purpose, these messages are defined differently then they are in the class **Interrogator** provided in HUMBLE. The message **oneOf:** looks for the existence of a single type of entity in the real world. It always returns true because it is assumed that there is only one machine layout type and one material handling system. The message **moreOf:** looks for more machine layouts and material handling systems. Since, they do not exist, this message always returns false. The value of a parameter is returned by the message **request:**. Whenever the inference engine asks for the value of a particular parameter, a method with a selector corresponding to the parameter name is executed.

As an example of how it works, consider the first rule presented above. When the HUMBLE inference engine encounters the parameter **selectedMHSType**, the method **selectedMHS** is executed and returns the type of the material handling system (in this example "agv") entered by the user, which is stored in the blackboard. Based on this information the expert system suggests "linear row" machine layout type.

In the same way, any knowledge source that can serve as an intelligent advisor can be attached to the blackboard. The only possible modifications are three described messages for interaction with the interrogator.

In some cases (for example, for design of a new system), the user may want to perform the optimization of the machine layout, according to the machine layout type selected. For that purpose, a neural network algorithm for solving the machine layout problem based on the Hopfield network [Hopfield and Tank 1984] has been developed. The simulation of the machine layout algorithm and the result of the optimization process are shown in Figure 7.

7.2 Off-line Intelligent Advisoring

The second approach suggests the use of an associative (semantic) network for developing a comprehensive advisory system for simulation. The NRL, originally developed for this purpose, has already been successfully used for development of an advisory system for teaching and coaching designers and students about the use of a digital logic simulation system [Antao et al. 1989]. An off-line advisory system for simulation of FMSs is currently under development, following design guidelines proposed by Antao et al. [1990]. The system is defined to provide the following capabilities: a tutorial on modeling of FMSs, a tutorial on discrete-event simulation principles, a tutorial on the VIS system used, and learning by running example simulations.

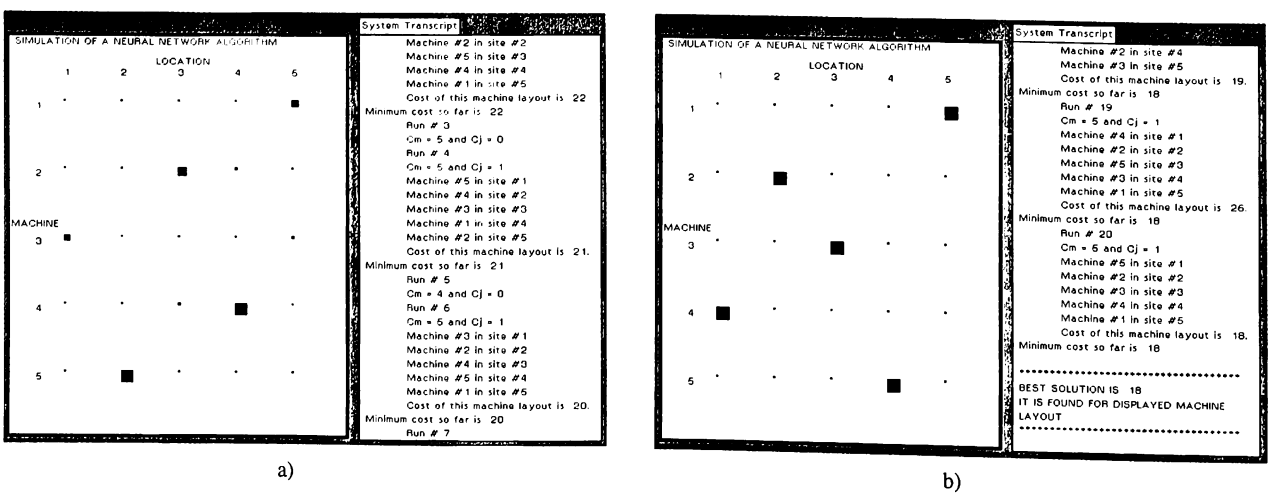
At any point, the user is able to stop the modeling and simulation process and activate the advisory system. The knowledge used for advisoring is encoded into an associative network visually displayed on the screen. The hypertext paradigm features, implemented in NRL, allows the user to explore the knowledge network through a mixed initiative dialogue. The user access to the tutorial knowledge is guided visually by coloring nodes. The successfully completed nodes are grayed lightly, the nodes that are supposed to be explored next are fully displayed, and the nodes not accessible at that point are blacked out. The user is able to explore (learn) concepts (stored in nodes) that are of his/her interest in a particular point during the modeling and simulation process.

A set of simulation examples is stored in nodes. The purpose is to allow the user to exercise the concepts learned, before performing an actual simulation experiment.

8. CONCLUSIONS AND FUTURE WORK

The primary goal of the research presented was to explore applicability of the object-oriented programming paradigm in development of VIS systems. The result is an experimental system for VIS of FMSs. VIS has been found to be a natural domain for application of the paradigm and graphical capabilities provided in Smalltalk-80. The research has been focused on the technical aspects of this application. Commercial aspects have not been considered.

The other activity involved the application of AI techniques. The blackboard model based problem solving method implemented allow for application of different AI techniques, such as neural networks and expert systems, for solving various modeling and simulation tasks. An approach for providing the intelligent advisory capabilities in a VIS system is presented.



a) b)

Figure 7a-b. Simulation of a Neural Network Algorithm for Solving 5-Machine Layout Problem. **a** Intemediate State, **b** Best Solution Found

The ongoing research activities are directed towards the development of a blackboard model based intelligent FMS design system. In addition to the features currently under development, as described in the paper, it requires development of a set of knowledge sources that perform specific FMS design activities, such as: machine selection, machine cell formation, cell layout formation etc.

ACKNOWLEDGEMENTS

The author would like to thank Brian Antao for valuable advices on the implementation of NRL and Professors Steven Peterson and John Bourne for support and encouragement for a large part of this work. Special thanks to Vanderbilt University for providing an opportunity to live and work in a friendly academic environment.

REFERENCES

Adams S.S. (1988), "MetaMethods," *HOOPLA!*, January, 9-23.

Ali, M.S. and D.L. Wyatt (1990), "An Object-Oriented Approach to Knowledge-Based Digital Circuit Simulation," In *Proceedings of the SCS Multiconference on Object Oriented Simulation*, A. Guasch, Ed. Society for Computer Simulation, San Diego, CA, 38-42.

Antao, A.A.B. (1988), "NRL: A Network Representation Language for Representing Knowledge in Advisory Systems," M.S. Thesis, Department of Electrical Engineering, Vanderbilt University, Nashville, TN.

Antao, A.A.B., J.R. Cantwell, A.J. Brodersen, and J.R. Bourne (1989), "An Advisory System for Digital Logic Simulation," In *Proceedings of the Second Int. Conf. on Ind. and Engr. Applications of AI & Expert Systems*, 775-778.

Antao, A.A.B., J.A. Brodersen, J.R. Bourne, and J.R. Cantwell (1990), "Building Intelligent Tutoring Systems for Teaching Simulation in Engineering Education," *IEEE Transactions on Education* (in press).

Beaumariage, T. and J.H. Mize (1990), "Object Oriented Modeling: Concepts and Ongoing Research," In *Proceedings of the SCS Multiconference on Object Oriented Simulation*, A. Guasch, Ed. Society for Computer Simulation, San Diego, CA, 7-12.

Bell, P.C. and R.M. O'Keefe (1987), "Visual Interactive Simulation - History, recent developments, and major issues," *Simulation* 49, 3, 109-116.

Bezivin, J. (1987), "Some Experiments in Object-Oriented Simulation," In *Proceedings of the OOPSLA'87 Conference*, ACM Press, New York, NJ, 394-405.

Birch, M.J., T.J. Terrel, R.J. Simpson, and H.P. Feszczur (1986), "FORSSIGHT and its application to an FMS simulation study," In *Simulation: Applications in Manufacturing*, R.D. Hurrion, Ed. IFS (Publications), UK, 163-172.

Blair, E.L. and S. Selvaraj (1989), "DISC++: A C++ Based Library for Object Oriented Simulation," In *Proceedings of the 1989 Winter Simulation Conference*, E.A. MacNair, K.J. Musselman, and P. Heidelberger, Eds. IEEE, Piscataway, NJ, 301-306.

Bryan, O.F. (1989a), "MODSIM II - An Object Oriented Simulation Language for Sequential and Parallel Processors," In *Proceedings of the 1989 Winter Simulation Conference*, E.A. MacNair, K.J. Musselman, and P. Heidelberger, Eds. IEEE, Piscataway, NJ, 172-176.

Bryan, O.F. (1989b), "Productivity Tools in Simulation: SIMSCRIPT II.5 and SIMGRAPHICS," In *Proceedings of the 1989 Winter Simulation Conference*, E.A. MacNair, K.J. Musselman, and P. Heidelberger, Eds. IEEE, Piscataway, NJ, 164-170.

Conway, R. and W. Maxwell (1986), "XCELL: A Cellular, Graphical Factory Modelling System," In *Proceedings of the 1986 Winter Simulation Conference*, J.R. Wilson, J.O. Henriksen, and S.D. Roberts, Eds. IEEE, Piscataway, NJ, 160-163.

Eldredge, D., J.D. McGregor, and M.K. Summers (1990), "Applying the Object-Oriented Paradigm to discrete Event Simulations Using C++ Language," *Simulation* 54, 2, 83-91

Elzas, M.S. (1986), "The Applicability of Artificial Intelligence Techniques to Knowledge Representation in Modelling and Simulation," In *Modelling and Simulation Methodology in the Artificial Intelligence Era*, M.S. Elzas, T.I. Oren, and B.P. Zeigler, Eds. Elsevier Science Publishers, Amsterdam, The Netherlands, 19-40.

Garrison, W.J. (1989), "NETWORK II.5 Tutorial: Network Modeling Without Programming," In *Proceedings of the 1989 Winter Simulation Conference*, E.A. MacNair, K.J. Musselman, and P. Heidelberger, Eds. IEEE, Piscataway, 205-214.

Gilman, R. A. and C. Billingham (1989), "A Tutorial on SEE WHY and WITNESS," In *Proceedings of the 1989 Winter Simulation Conference*, E.A. MacNair, K.J. Musselman, and P. Heidelberger, Eds. IEEE, Piscataway, NJ, 192-200.

Goldberg, A. and D. Robson (1983), *Smalltalk - 80: The Language and Its Implementation*, Addison-Wesley Publishing Company, Reading, MA.

Grant, E.M. and D.W. Starks (1988), "A Tutorial on TESS: The Extended Simulation Support system," In *Proceedings of the 1988 Winter Simulation Conference*, M.A. Abrams, P.L. Haigh, and J.C. Comfort, Eds. IEEE, Piscataway, NJ, 136-140.

Guo, D., D.H. Norrie, and O.R. Fauvel (1990), "Object-Oriented Flexible Manufacturing Simulation," In *Proceedings of the 1990 Summer Computer Simulation Conference*, B. Svrcek and J. McRae, Eds. The Society for Computer Simulation, San Diego, CA.

Heragu, S. and A. Kusiak (1988), "A Knowledge-Based System for Machine Layout," In *Expert Systems: Structures and Solutions in Manufacturing Design and Planning*, A. Kusiak, Ed. SME, Dearborn, MI, 303 - 321.

Hopfield, J.J. and D.W.Tank (1985), ""Neural" Computation of Decisions in Optimization Problems," *Biological Cybernetics* 52, 141-152.

Hurrion, R.D. and R.J.R. Secker (1978), "Visual Interactive Simulation: An Aid to Decision Making," *Omega* 6, 5, 419-426.

Hurrion, R.D. (1986), "Visual Interactive Modeling," In *Simulation: Applications in Manufacturing*, R.D. Hurrion, Ed. IFS (Publications), UK, 3-13.

Kettenis, D.L. (1986), "Knowledge-Based Model Storage and Retrieval: Problems and Possibilities," In *Modelling and Simulation Methodology in the Artificial Intelligence Era*, M.S. Elzas, T.I. Oren, and B.P. Zeigler, Eds. Elsevier Science Publishers, Amsterdam, The Netherlands, 101-112.

Knapp, V.E. (1986), "The Smalltalk Simulation Environment - Part 1," In *Proceedings of the 1986 Winter Simulation Conference*, J.R. Wilson, J.O. Henriksen, and S.D. Roberts, Eds. IEEE, Piscataway, NJ, 125 -128.

Knapp, V.E. (1987), "The Smalltalk Simulation Environment - Part 2," In *Proceedings of the 1987 Winter Simulation Conference*, A. Thesen, H. Grant, and W.D. Kelton, Eds. IEEE, Piscataway, NJ, 146 -151.

Krasner, G.E. and S.T Pope (1988), "A Cookbook for Using the Model-View-Controller User Interface Paradigm in Smalltalk-80," ParcPlace Systems, Palo Alto, CA.

Kusiak, A., E. Szczerbicki, and R. Vujosevic (1990), "Intelligent Design Synthesis: An Object Oriented Approach," Working Paper 90-15, Department of Industrial Engineering, The University of Iowa, Iowa City, IA.

Law, A.M. and M. McComas (1986), "Pitfalls in the Simulation of Manufacturing Systems," In *Proceedings of the 1986 Winter Simulation Conference*, J.R. Wilson, J.O. Henriksen, and S.D. Roberts, Eds. IEEE, Piscataway, NJ, 539-542.

Nii, P. H. (1986), "Blackboard Systems: The Blackboard Model of Problem Solving and the Evolution of Blackboard Architectures," *AI Magazine* 7, 2, 39-53.

Nyen P.A. (1987), "A Comprehensive Environment to Object Oriented Simulation of Manufacturing Systems," In *Simulation in Computer Integrated Manufacturing*, K.E. Wichmann, Ed. Society for Computer Simulation, San Diego, CA, 21-25.

O'Keefe, R.M. (1987), "What is Visual Interactive Simulation? (And is There a Methodology for Doing it Right?)," In *Proceedings of the 1987 Winter Simulation Conference*, A. Thesen, H. Grant, and W.D. Kelton, Eds. IEEE, Piscataway, NJ, 461-464

O'Keefe, R.M. (1989), "The Role of Artificial Intelligence in Discrete-Event Simulation," In *Artificial Intelligence, Simulat-*

ion, and Modeling, L.E. Widman, K.A. Loparo, and N.R. Nielsen, Eds. John Wiley & Sons, New York, NJ, 359-380.

Piersol, K. W. (1987), *HUMBLE V2.0 Reference Manual*, Xerox Special Information Systems Vista Laboratory, Pasadena, CA .

Pinson, L.J. and R.S. Wiener (1988), *An Introduction to Object Oriented Programming and Smalltalk*, Addison-Wesley Publishing Company, Reading, MA.

Poorte, J.P. and D.A. Davis (1989), "Computer Animation with CINEMA," In *Proceedings of the 1989 Winter Simulation Conference*, E.A. MacNair, K.J. Musselman, and P. Heidelberger, Eds. IEEE, Piscataway, NJ, 147-154.

Roberts, D.S. and M.A. Flanigan (1989), "Simulation Modeling and Analysis with INSIGHT: A Tutorial," In *Proceedings of the 1989 Winter Simulation Conference*, E.A. MacNair, K.J. Musselman, and P. Heidelberger, Eds. IEEE, Piscataway, NJ, 291-300.

Rohrbough, M.C. (1989), "Introduction to SIMFACTORY II.5," In *Proceedings of the 1989 Winter Simulation Conference*, E.A. MacNair, K.J. Musselman, and P. Heidelberger, Eds. IEEE, Piscataway, NJ, 201-204.

Rothenberg, J. (1986), "Object-Oriented Simulation: When do we go from here?," In *Proceedings of the 1986 Winter Simulation Conference*, J.R. Wilson, J.O. Henriksen, and S.D. Roberts, Eds. IEEE, Piscataway, NJ, 464-469.

Rothenberg, J. (1989), "The Nature of Modeling," In *Artificial Intelligence, Simulation, and Modeling*, L.E. Widman, K.A. Loparo, and N.R. Nielsen, Eds. John Wiley & Sons, New York, NJ, 75-91.

Sadowski, R.P. (1989), "The Simulation Process: Avoiding the Problems and Pitfalls," In *Proceedings of the 1989 Winter Simulation Conference*, E.A. MacNair, K.J. Musselman, and P. Heidelberger, Eds. IEEE, Piscataway, NJ, 72-78.

Schriber, T.J. (1987), "The Nature and Role of Simulation in the Design of Manufacturing Systems," In *Simulation in Computer Integrated Manufacturing*, K.E. Wichmann, Ed. Society for Computer Simulation, San Diego, CA, 5-18.

Thomasma, T., Y. Mao, and O. Ulgen (1990a), "Manufacturing Simulation in Smalltalk," In *Proceedings of the SCS Multiconference on Object Oriented Simulation*, A. Guasch, Ed. Society for Computer Simulation, San Diego, CA, 93-97.

Thomasma, T., O. Ulgen, and Y. Mao (1990b), "Tools for Designing Material Handling Control Logic," In *Proceedings of the SCS Multiconference on Object Oriented Simulation*, A. Guasch, Ed. Society for Computer Simulation, San Diego, CA, 19-22.

Thompson, M.B. (1989), "AutoMode II: The System Builder," In *Proceedings of the 1989 Winter Simulation Conference*, E.A. MacNair, K.J. Musselman, and P. Heidelberger, Eds. IEEE, Piscataway, NJ, 235-242.

Ulgen, O.M. and T. Thomasma (1986), "Simulation Modeling in an Object-Oriented Environment Using Smalltalk-80," In *Proceedings of the 1986 Winter Simulation Conference*, J.R. Wilson, J.O. Henriksen, and S.D. Roberts, Eds. IEEE, Piscataway, NJ, 474-484.

Van der Meulen P.S. (1987), "INSIST: Interactive Simulation in Smalltalk," In *Proceedings of the OOPSLA'87 Conference*, ACM Press, New York, NJ, 366-376.

Vujosevic, R. (1990a), "Object Oriented Visual Interactive Modeling and Simulation of Flexible Manufacturing Systems," M.S. Thesis, Department of Mechanical Engineering, Vanderbilt University, Nashville, TN.

Vujosevic, R. and A. Kusiak (1990b), "Neural Network Based Analysis of Flexible Manufacturing Systems," Working Paper 90-16, Department of Industrial Engineering, The University of Iowa, Iowa City, IA.

Vujosevic, R. (1990c), "Using a Neural Network Algorithm for Solving Machine Layout Problem," Unpublished Report, Department of Industrial Engineering, The University of Iowa, Iowa City, IA.

Wasserman, P.D. (1989), *Neural Computing: Theory and Practice*, Van Nostrand Reinhold, New York, NY.

Zeigler, B.P. (1987), "Hierarchical, Modular Discrete Event Modelling in an Object Oriented Environment," *Simulation* 49, 5, 219-230.

Zeigler, B.P. (1990), *Object-Oriented Simulation with Hierarchical, Modular Models*, Academic Press, San Diego, CA.