Proceedings of the 1992 Winter Simulation Conference
ed. J. J. Swain, D. Goldsman, R. C. Crain, and J. R. Wilson

GRAPHICAL OBJECT-ORIENTED DISCRETE-EVENT SIMULATION SYSTEM

Liang Y. Liu

Department of Civil Engineering
University of Illinois-Urbana/Champaign
Urbana, IL 61801-2397, U.S.A.

Photios G. Ioannou

Department of Civil & Envir. Engineering
University of Michigan
Ann Arbor, MI 48109-2125, U.S.A.

ABSTRACT

Building graphical models and running simulations directly on the models provides an attractive way of performing simulation. Most existing network-based simulation systems use graphical models to represent real world systems; however, users have to create separate simulation input files (usually text) to run the simulation. Inconsistency and mistakes between the two representations of a model are common and difficult to detect. By using object-oriented programming, modeling elements of a simulation system can be designed as objects that combine both interactive graphics and discrete-event simulation functions. This integration provides not only the friendly user interface to build simulation models but also the desired direct simulation on graphical networks. COOPS is a new discrete-event simulation system that utilizes this integration concept. It is a PC-based general purpose discrete-event simulation system capable of modeling systems such as service and manufacturing systems, or construction operations. COOPS runs on computers with 386, 486 or better CPU on top of Microsoft Windows 3.0 or above. This paper presents the modeling concept of COOPS and the design that makes the integration possible.

1 INTRODUCTION

Most existing network-based simulation systems use graphical models to represent a real world system; however, they require the creation of separate simulation input files (usually text) to run the simulation. Although the graphical models represent more vividly the characteristics of a system, a more abstract format, the text file, is generally required to perform the simulation. Mistakes and inconsistencies between the two representations of a model are common and difficult to detect. By using object-oriented programming, simulation modeling elements can be designed to contain the functionality of both interactive computer graphics and discrete-event simulation. This combination allows building simulation directly and interactively, and simulation can be performed directly on the graphical network.

This paper presents the modeling concepts and the design of a new discrete-event simulation system, COOPS, which utilizes object-oriented programming to combine the functionality required for interactive computer graphics and discrete-event simulation. All modeling elements, including activities, resources, queues, routers, consolidations, and links are implemented as objects that function appropriately when drawn on the screen and when simulation is executed. Being object-oriented, each modeling element can be modified and enhanced independently without affecting other parts of the overall simulation system.

2 COOPS MODELING CONCEPTS

COOPS was originally developed by the authors to demonstrate the benefits of object-oriented simulation design and, at the same time, overcome some of the limitations of existing simulation systems. COOPS adopts the scheme of a network-based simulation model that traces its roots to Q-GERT (Prikster 1978) and CYCLONE (Halpin 1976). While COOPS uses a simple set of primitive modeling elements, it provides a powerful mechanism for modeling discrete-event systems, such as service systems, manufacturing systems, and construction operations.

A COOPS simulation model is a precedence network that consists of three kinds of objects: nodes, links, and attachments (Figure 1). Four types of nodes have been defined: activities, queues, consolidations, and routers. Each type has its own distinctive graphical representation and function within the network. Links are graphically represented as arrows showing the precedence relationship between two nodes and symbolizing the direction of resource flows. There are three kinds of attachments: specific resources, resource units, and flags. These attachments, when attached to a

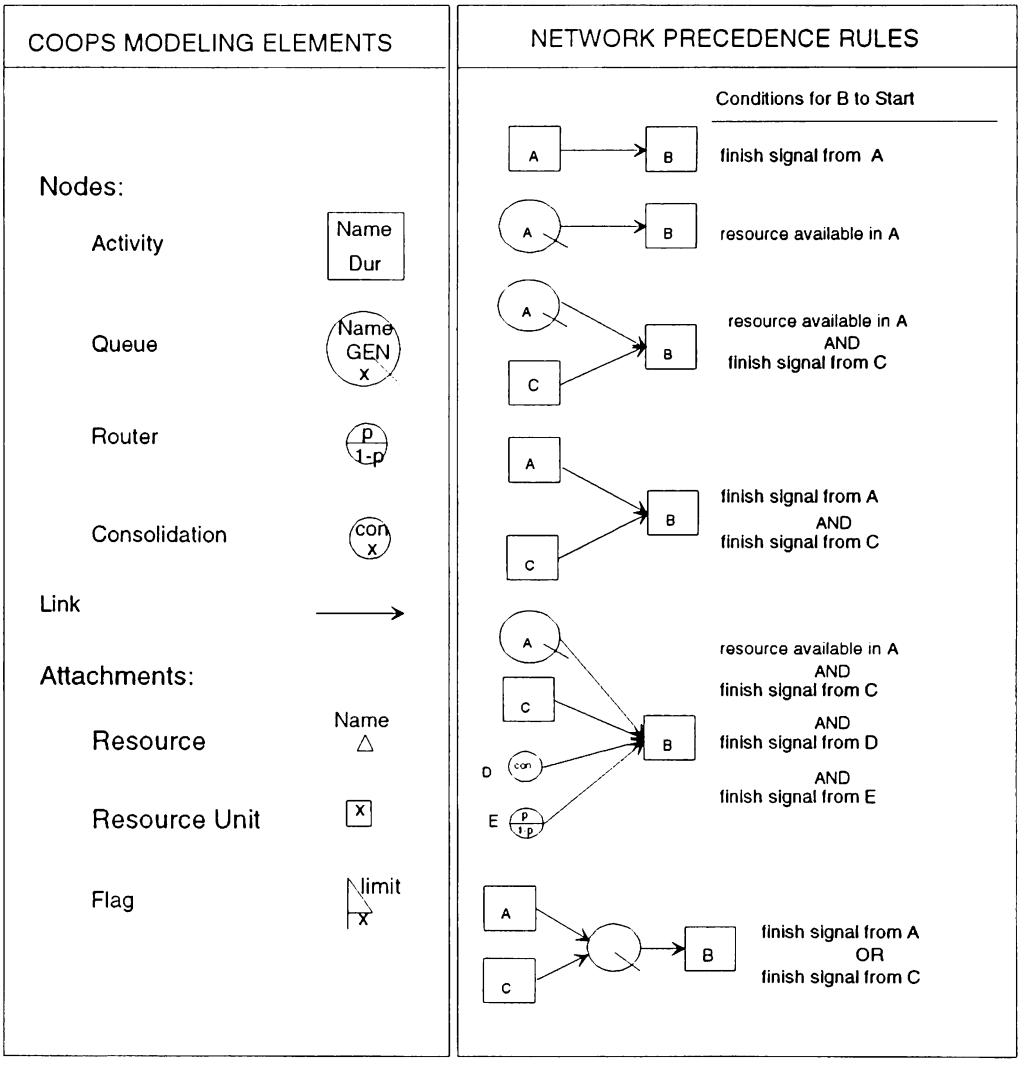

Figure 1 COOPS Modeling Elements & Network Precedence Rules

queue or a link, will affect the characteristics of the attached modeling element. In short, modeling in COOPS involves assembling the basic modeling elements to construct a network that represents a system under study, and then performing the simulation.

2.1 Resources

A resource depicted in the model represents the attributes of a real world resource. Typical resources are labor, equipment, materials, time, and space. The following resource attributes are tracked during simulation: units, state, flows, interactions, transformations, and queuing behavior.

Simulation models can have two kinds of resources: "generic" and "specific." Generic resources are assumed to be identical and interchangeable, whereas specific resources are individually identified. For example, ten workers in an operation can be modeled as either a pool of ten units of generic resources or ten individual specific resources. Specific resources are tracked individually during simulation; therefore, more detailed statistics are collected. Only group statistics are kept for generic resources.

2.2 Queues

A queue can be viewed as a storage area for resources. Resources in queues are in an idle state. The contents of a queue change whenever resources are captured by a successor activity or released to the queue by a preceding node during simulation. The rules for changing the resource contents of a queue can be summarized as follows:

- When a directly-preceding node finishes, the contents of the queue are increased by the units

of resource entering the queue.

- When an activity starts, the contents of each directly preceding queue are decreased by the units of resources captured by the activity.

Generic resources are stored in queues as integer values without reference to a particular unit of measure. The system does not know how these numbers are interpreted by the user, for example, whether they represent the number of workers or gallons of paint. It is the user's responsibility to select the appropriate units of measure and to ensure that the contents of queues are increased or decreased by the correct amounts. Furthermore, the units of measure for one queue should be compatible with those of other queues and with those required by the following activities.

Compatibility of resource units is necessary in many modeling situations. For this reason queues are associated with a GEN function. The GEN function in a queue "splits" each incoming resource unit into GEN number of units. This provides queues with the capability of resource "generation."

2.3 Activities

Activities represent tasks that need to be performed in an operation. Generally, resources are needed for an activity to start. When an activity starts, it captures the required resources until its completion. Activities are the only nodes that can follow a queue. Each activity has a duration which can have one of six different types of probability distributions: deterministic, uniform, normal, beta, gamma, and triangular. Activities are also assigned priority values to determine which will use a resource when two or more activities are competing for the same resource. The higher the priority value, the higher the activity's priority to use a resource.

2.4 Routers

Routers are nodes used to represent a binary random selection in a production process. They are graphically represented as a circle with a line in the middle that separates two real numbers representing the upper and the lower link probabilities. Normally, a node will activate all links leading to its successor nodes. Routers are used in situations where it is desirable to activate only one of these links (and the associated succeeding node) based on a probabilistic selection process. Although routers provide only binary random selections, they can be combined in series to model multiple random selections.

2.5 Consolidations

A consolidation represents the transformation, combination, or consumption of resources. it merges a pre-defined number of incoming generic resources to produce one unit of a finished resource. It is the logical opposite of a queue with a GEN function. Note that consolidations affect only generic resources. Specific resources are assumed to maintain their characteristics and integrity throughout a simulation and simply pass through a consolidation node unaffected.

2.6 Links

Links specify the direction of resource flows and the precedence relationships between nodes. Specific resource routing is modeled by attaching specific resources to links. In this case, the link allows only the attached specific resources to flow through.

2.7 Link Resource Requirement

An operation typically requires different combinations of resources, whose units must be balanced according to the operational logic. Link resource requirement units (LRR) are attachments that specify the number or units of generic resources that will be added to a succeeding queue or removed from a preceding queue. LRRs complement the use of GEN functions in queues and the use of consolidations when balancing resource units in a network.

2.8 Flags

Flags are special attachments used in conjunction with queues to set stopping conditions. When the contents of a queue reach the limit set by its attached flag, the simulation stops. More than one flag may be used in a model to set up multiple stopping conditions.

2.9 Calendars

The availability of resources in a COOPS model is controlled by their associated calendars. A calendar defines the work-break schedule of a resource. For example, a calendar named "9-to-5", can have the work-break schedule: 4,1,4,15. Assuming that simulation time units represent hours, this calendar represents a typical working day of two four-hour shifts. The cycle time is the sum of all work-break times (4+1+4+15=24). A calendar can be defined by assigning a combination of work-break pairs to represent any working schedule of a resource.

For efficiency, COOPS uses a system calendar to define the default working schedule. The system

calendar defines the working schedule for all resources in a model. However, a resource can have its own calendar, which, when defined, will override the system calendar. For example, some resources may work overtime while the system (default) calendar works 9 to 5.

Obviously, since calendars control the availability of resources, they also affect the calculation of the finish time of an activity. When an activity starts, it captures the required resources from its preceding queues. During the duration of an activity, if any of the captured resources would need to take a break based on their calendar, the break time must be considered and the activity finish time adjusted.

2.10 Precedence Rules and Network Logic

The logic of a network dictates the processing sequence of a simulation. An activity cannot start until all the required resources are available. When an activity precedes another activity, the successor can start only after receiving a finish signal from the predecessor. The conditions under which an activity can start are summarized in Figure 1.

3 USER INTERFACE & EXAMPLE

A basic objective of COOPS design was to provide a friendly graphical user interface with which to construct models and perform the simulation. This interface allows building simulation models interactively and performing simulation directly on the graphical network. Data for modeling elements are entered/modified by using interactive dialog boxes. Figure 2 shows the COOPS main display window with a machine service station model example and a dialog box for activity data entry and modification.

The main display window consists of several functional areas. The up-arrow on the upper right corner of the window allows maximizing the display to full screen while the down-arrow will minimize COOPS into an icon similar to most Windows applications. The window's caption bar displays the path and file name of the current model. Below the caption bar is the menu which provides functions such as file access, editing, viewing, and simulation control. The two numbered shaded bars - one horizontal, the other vertical - are rulers that show the current drawing area relative to the selected page size. To the left of the window is the

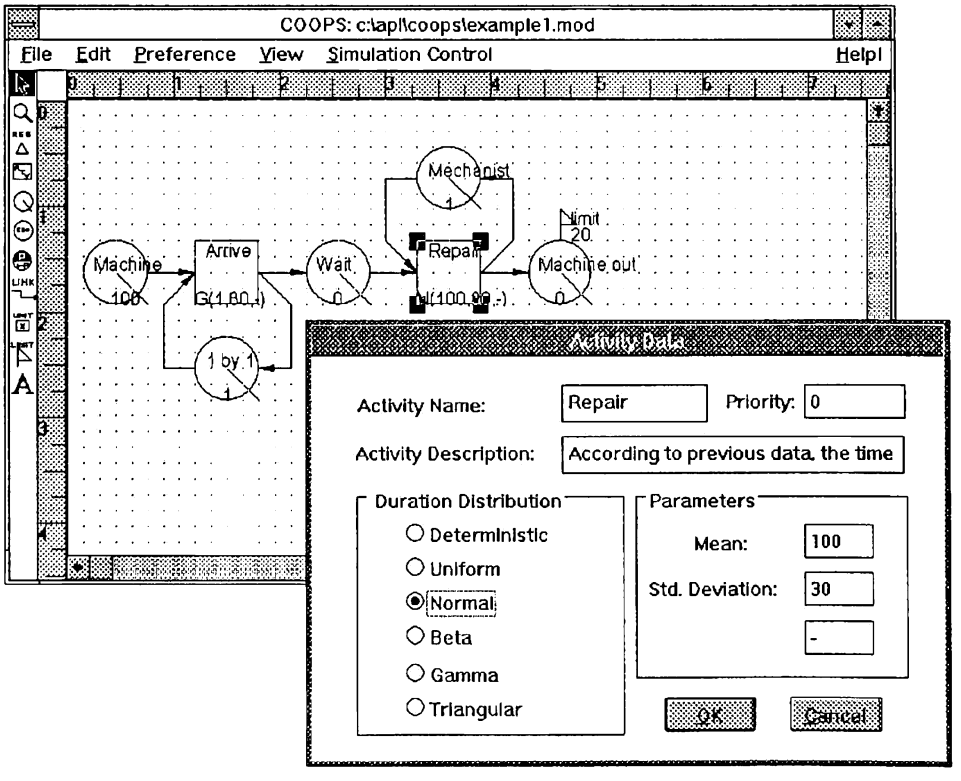

Figure 2 COOPS User Interface & Example

selection toolbox that contains all COOPS modeling elements and modeling building tools. The area to the right of the vertical ruler is the model drawing area, which contains grids to help line up and connect modeling elements. This area functions as a canvas for creating (drawing) simulation models. Editing features are provided to allow copy, cut, and paste operations.

COOPS simulation models are created by first selecting the desired modeling element from the selection toolbox using the mouse. The toolbox will highlight the current selection in reverse video. A subsequent click on the model drawing area will create and draw the corresponding modeling element. A dialog box will pop up to solicit data for the newly created element. Once proper data are entered, the modeling element will be graphically displayed in the model drawing area and can be dragged and moved to any position. A model is created by repeating this process of creating and linking elements into a network. Other input, such as the simulation time limit and random number seed are defined via the menu. Simulation is accomplished by having these modeling elements send messages to each other. The result of a simulation consists of the network model, the simulation event list, a simulation trace report, and the time-related statistics for each node.

The example in Figure 2 shows a model of a machine maintenance station. The units of resources defined in queues represent the starting conditions of the system. For example, there is only one mechanic working in the shop, and there is no machine in the shop at the beginning of the simulation. Machine breakdown and repair times are random variables. In this example, machines arrive at the shop following an exponential distribution (Gamma with K=1) with an average inter-arrival time of 80 minutes. Repair time follows a normal distribution with a mean of 100 minutes and a standard deviation of 30 minutes. Figure 2 shows an interactive dialog box for defining the duration and other attributes of an activity. In the dialog box, the parameters change dynamically according to the type of distribution selected. Once defined, the distribution type and its parameters are displayed inside the activity graphic display. For example G(1,80) represents Gamma distribution with K=1 (i.e. exponential) and an average inter-arrival time of 80. This simulation is set to stop when 20 machines are repaired because a flag (limit=20) is attached to the queue that counts the number of serviced machines. Notice that the left three nodes and links constitute a mechanism of generating random arrivals of machine breakdowns that follow an exponential distribution. This construct is similar to the GENERATE function in GPSS.

4 COOPS DESIGN

Two important features of the COOPS system design make possible the concept of integrating interactive computer graphics and direct simulation: (1) the connection between nodes and links to form a network and (2) the message-based simulation control. COOPS is implemented by using an object-oriented programming paradigm. All program code has been defined in classes with methods and instance variables determined according to object behavior analysis (Gibson 1990). After the classes were defined, instances of these classes (i.e. objects) were used to create COOPS. Simulation is performed by objects sending messages to each other and receiving messages from the user.

4.1 CONNECTION BETWEEN NODES

Links define the precedence relationships between nodes (activities, queues, consolidations, and routers) within a network. Links also show the direction of resource flows. It is therefore necessary to implement a mechanism to establish the precedence relationships both during model building and simulation.

When building models, links are drawn as polylines, which contain many line segments giving the user the flexibility to connect nodes without crossing over other nodes. A question arises as to the degree of accuracy necessary to make a connection, considering the round-off error of floating point calculation in a digital computer. To solve this problem, COOPS implements a grid and follows a proximity rule. The grid confines the borders of nodes and the points of links to lie on grid

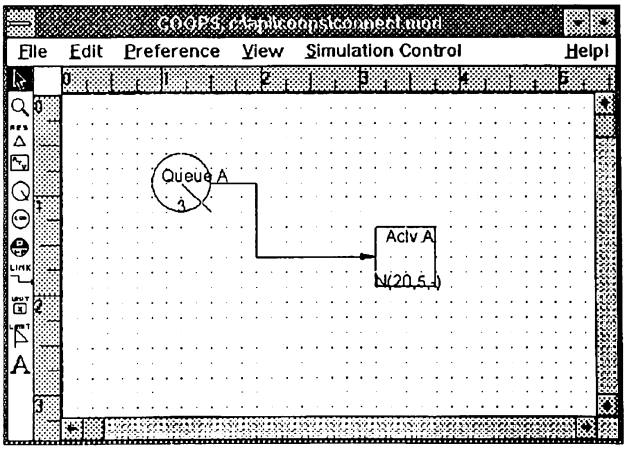

Figure 3 Link and Node Connection

points. This confinement helps line up modeling elements and also allows the proximity rule to take effect. A link that appears visually connected may in fact be digitally apart by a very small number in terms of the X or Y coordinates. To allow some tolerance, the proximity rule accepts the establishment of a connection if the distance between the end point of a link and a node is within half of the grid point distance.

During the execution of a simulation, a node must recognize which are its predecessors and successors in order to capture or release resources to the correct nodes. This mechanism is accomplished by associating the nodes at both ends of a link and setting internal pointers. This association allows the nodes to communicate with each other appropriately during simulation. For example, activity A in Figure 3 will capture resources from Queue A because of the link between them.

4.2 MESSAGE-BASED SIMULATION CONTROL

COOPS simulation is accomplished by message-passing among three groups of entities: (1) the modeling elements, (2) the simulation control, and (3) the user interface. In fact, entities in these three groups are implemented as objects with extensive functionality. Simulation is performed by a variety of active objects (modeling objects, user interface objects, and the simulation control object) communicating with each other. The active objects that cooperate during a COOPS simulation are shown in Figure 4. The SimuWind, SimPalette, SimDraw, and Network objects constitute the user interface and interact directly with the user. The SimulationControl object is the control center during simulation. These objects, along with others, work harmoniously to accomplish the task of building graphical models and performing simulation through message passing.

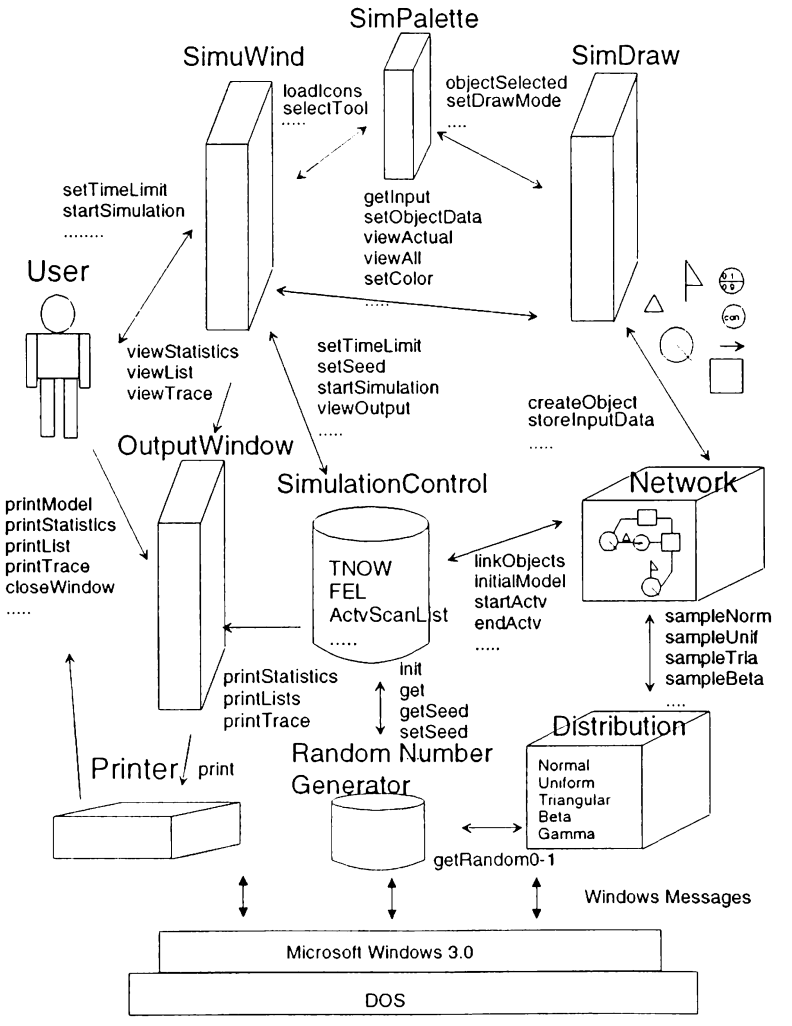

Figure 4 COOPS System Design

4.2.1 MESSAGE-PASSING— MODEL BUILDING

At the model building stage, the selection of a modeling element through the mouse sends a message to the SimuWind object. This message is then passed on to the SimPalette object to determine the current selection. A subsequent positioning within the model drawing area will trigger a message to be sent to the SimPalette object from the SimDraw object querying the current selection. A return message will make the SimDraw object create a new modeling element. The creation message triggers the modeling element to prompt the user for appropriate data entry. Once proper data is entered, the modeling element sends a message to the Network object and stores itself in the Network object.

4.2.2 MESSAGE-PASSING— SIMULATION

At the start of a simulation, a user sends a startSimulation message, and the SimuWind object receives and relays the message to the SimulationControl object, which is the control center during a COOPS simulation. The SimulationControl object contains several instance variables, such as TNOW, FEL, and ActvScanList. TNOW stores the simulation clock time. FEL (Future Event List) is a list of future event times during simulation. ActvScanList is a list of all activities in a COOPS model. During simulation, the SimulationControl object sends messages to the Network object to retrieve the data stored in the modeling elements. If the activity duration is a random variable, messages are sent to the Distribution object to sample the duration from a particular distribution, such as uniform, normal, triangular, etc.

The SimulationControl object performs two tasks alternatively: (1) scanning and scheduling activity end event times, and (2) advancing the simulation clock time to the next earliest future event. The messages sent by the SimulationControl object include startSimulation, getNextEvent, scheduleEvent, stopActivity, startSuccessors, sampleDuration, etc. The scanning/scheduling and advancing the clock tasks proceed alternately until the simulation time limit or the limit in a flag has been met. During simulation, time-related information is posted to one of the nodes' instance variables, timeStatistics. They can later respond to the printReport message to print out the statistic of a queue, a resource, or an activity.

5 CONCLUSION

COOPS utilizes object-oriented programming to integrate interactive computer graphics and discrete-event simulation. This integration allows the user to build simulation models graphically similar to a drawing system. Simulation is performed directly on the graphical network without having to build a separate text file for model definition. Being object-oriented and modular in design, the system's functionality can be extended easily. The base system can be linked with knowledge-based systems to provide an integrated simulation environment for planning, modeling, and decision making.

The current version of COOPS was developed using object-oriented programming language - Actor 3.0 from the Whitewater Group - and runs under Microsoft Windows 3.0 or above. Free copies of COOPS can be obtained from the authors.

REFERENCES

Balci, O. 1988. "The Implementation of Four Conceptual Frameworks for Simulation Modeling in High-level Languages", *Proceedings of the 1988 Winter Simulation Conference*, San Diego: Society for Computer Simulation.

Bank, J. and Carson, J.S. 1984. *Discrete-Event System Simulation*, New Jersey: Prentice-Hall.

Harris, R.B. 1978. *Precedence and Arrow Network Techniques for Construction*, New York: John Wiley & Sons.

Ioannou, P.G. 1990. *UM-CYCLONE User & Reference Manuals*, Ann Arbor, Michigan: Dept. of Civil & Envir. Eng., Univ. of Michigan.

Law, A.M. and Kelton. D.K 1991, *Simulation Modeling and Analysis*, 2nd. edition, New York: McGraw Hill.

Liu, L.Y. and P.G. Ioannou. 1992. "Graphical Object-Oriented Simulation System for Construction Process Modeling," *Proceedings of the Eighth National Conference on Computing in Civil Engineering,* New York: American Society of Civil Engineers.

Pritsker, A.A. 1978. *Modeling and Analysis Using Q-GERT Networks*, New York: John Wiley & Sons.

Pritsker, A.A. 1986. *Introduction to Simulation and Slam II,* New York: John Wiley & Sons.

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