Automation and Robotics in Construction X
G.H. Watson, R.L. Tucker and J.K. Walters (Editors)

Modeling of building construction activities using forms

Amr A. Oloufa[a]

[a]Department of Architectural Engineering, Penn State University, 104 Engineering Building "A", University Park, PA 16802, U.S.A.

Abstract

In the course of improving the efficiency of building construction activities, the application of discrete event simulation tools may be extremely beneficial. Simulation is used to evaluate construction scenarios and allows the construction manager to evaluate between alternatives. Selection of an optimum mix of equipment and crews may also be studied.

However, the practice of simulation in building construction has not been popular for a variety of reasons. Perhaps a major reason is the investment in time and cost needed to develop the simulation model and program the corresponding code in a simulation language. The situation is further aggravated by the lack of training in simulation in most undergraduate and graduate engineering curricula.

This paper reports on ongoing research aimed at simplifying the development of simulation models. The construction manager will select equipment and crews from a library of resources and then specify the interaction of these resources in an intuitive mode with no programming.

With the advent of object-oriented modeling and programming, simulation entities may be described in a format that encapsulates their characteristics and capabilities. The user will be capable of changing the number and/or mix of equipment with little or no change to the underlying simulation code.

In this paper we will develop a non-network based methodology for specifying the interaction between construction resources and explain an approach for mapping to an object-oriented simulation language. The utility of this approach will be demonstrated in a building construction application.

1. INTRODUCTION TO MODELING AND SIMULATION

Discrete event simulation is a tool for the study of operations. Shanon (1975) defines simulation as *"The process of designing a model of a real system and conducting experiments with this model for the purpose either of understanding the behavior of the system or of evaluating various strategies (within the limits imposed by a criterion or set of criteria) for the operation of the system".*

Simulation can be used to evaluate BETWEEN a set of competing alternatives. The simulation approach may be used to study almost any problem. Each simulation run becomes an experiment on the system that can be observed and controlled.

The simulation study encompasses a multitude of phases starting from the development of the system's model which is intended to serve as a surrogate of the system under consideration. The model should encompass the major system components that are likely to affect its behavior. The design and composition of the simulation model is always a tradeoff between its simplicity and how well it approximates the system considered. The incorporation of every possible feature of the system is time consuming and impractical. On the other hand, the omission of one or more items will render the model useless. For this reason, the success of modeling is largely a function of the correct choice of entities and parameters to be used in the simulation.

After the simulation model is developed, the model is coded in a simulation language. Two important concepts are involved namely validation and verification. Validation is concerned with how well the model truly represents the simulated system and is independent of simulation language issues. Verification on the other hand is concerned with computer language implementation issues, i.e. similar to debugging of programming languages. Validation and verification are iterative in nature and the typical simulation study should involve a complete and reliable investigation of both.

When the developers of the simulation model are confident of the accuracy of its logical (i.e. *validation*) and physical (i.e. *verification*) representations, the model is subjected to simulation "runs" under various operating conditions. Statistical data is collected during these runs and analyzed later for its significance. Conclusions about the model and the study can then be drawn.

It is obvious that a major ingredient for the success of a simulation study is related to the ease of development of the simulation model. Perhaps a major reason against widespread application of simulation in construction is related to the time and cost needed for developing and coding the simulation model. The author's ongoing research reported here strives to simplify the development and coding of the simulation model. The ultimate goal is the development of an environment where the simulation model can be created through an interactive dialogue with the construction project team. Since comprehensibility of the simulation model by its end users is a major step towards the credibility and success of the simulation study, this system will allow the individuals with the most intimate knowledge of the project to develop the simulation model .

2. OBJECT-ORIENTED MODELING & PROGRAMMING

Object-oriented approaches bridge the gap between the physical system and its computer representation. The object-oriented view is that the system is composed of interacting physical objects that are the central focus of the simulation. The first object-oriented programming language developed in the

1960's, named SIMULA, was a simulation language. This language laid the foundations of object-oriented programming despite of its commercial failure. Recently, the emergence of object-oriented simulation languages for manufacturing applications has received a lot of attention (Cammarata et al (1987), Knapp (1986, 1987), Roberts et al (1988), Rothenberg (1986) and Ulgen (1986)).

In describing a system, we define its components and how these components interact together (Booch 1991). We also declare the valid operations these components engage in, and how this engagement affects their states before, during and after these operations. It is important to distinguish between object-oriented modeling and object-oriented programming. Object-oriented modeling deals with the description of the system in terms of its resources. Each resource embodies in its definition its respective capabilities. Resources may inherit one or more capabilities from other resources and could therefore be defined as a subset of other resources. A conventional (*non object-oriented*) programming language may be used to translate the object-oriented model to its computer counterpart. However, what used to be understandable physical components will now be represented in terms of queues, methods, etc. in a simulation language. This is the strongest reason for the communications gap between users and programmers.

A better approach for the mapping of an object-oriented model to its computer counterpart is through the utilization of an object-oriented computer language. This approach establishes a direct mapping between the model and its coded description. However, it is worthwhile to emphasize that the utilization of object-oriented programming without a corresponding object-oriented modeling approach yields its benefits exclusively to the computer programmer, not the software user. It is estimated that companies spend 70-80% of their software budget on software update not writing software from scratch. Therefore it is extremely advantageous to develop a methodology that simplifies code development, modification and extension. The software user however, is not able to determine, and indeed has no interest in whether the application program was coded in any specific programming language.

3. A FORM-BASED APPROACH

Shu (1988) defines visual programming as "*The use of meaningful graphic representations in the process of programming*", and "*the application of graphical techniques and pointing devices to provide visual environments for program construction and execution; for information retrieval and presentation and for software design and understanding*". Several researchers (Gordon and McNair (1987), Davis *et al* (1988), Glicksman (1986), Thomasma and Ulgen (1988), Tseng *et al* (1988) and Ozden (1991)) and a few software packages such as XCELL+ (Conway and Maxwel 1991) and SIMFACTORY (Tumay 1987) have attempted to automate the model building process through visual programming. All these implementations however were designed specifically for manufacturing

environments and are incapable of representing some common situations in construction simulation. In construction applications, Riggs (1986), Odeh *et al* (1991), and Ioannu (1992) used icons to create the simulation model. However, the user has to be familiar with the simulation language CYCLONE which is a non object-oriented language.

The author proposed the development of an interactive environment for simulation model generation that allows the decision maker to develop discrete event simulation models with no programming. Major requirements for such a system (adapted from Wales and Luker 1986) are as follows:

1. Allow users to define the system in terms familiar to them not in terms of the simulation language used.
2. Guide users in the steps required for model development.
3. Checks users responses for consistency and completeness.
4. Creation of new models by the adaptation of existing ones.
5. Provide a visual representation of the system in a form understandable by decision makers.
6. Generate the simulation model code automatically.

The proposed system will function as two layers between the user and the simulation language being used. The first layer will be language-independent. An additional layer will change the output from the first layer to the simulation language employed. The language independent module is composed of two submodules RESPEC and REACT (Oloufa 1992). While the RESPEC module remains identical to the one reported before, the module REACT has been extended. An example demonstrating its utility will be described in a later section.

3.1 RESOURCE SPECIFICATION MODULE [RESPEC]

The resource specification module (explained in Oloufa 1992) is used to define the various resources used in the construction project. This definition is a complete representation of the resource that includes its attributes, capabilities, type, and the number of units available. Resources represent equipment, material and labor. Equipment resources belong to families of specialized equipment types. For example, both trucks and loaders may belong to the resource type "Vehicle" which is capable of translational motion. However each type has its own attributes and specialized capabilities. The object-oriented approach offers a very elegant mechanism for the definition of resources due to the hierarchical nature of their description. Resources belonging to a company may be saved in a "Resource Library". The module REACT explained in the next section will access this library and retrieve information belonging to the definition, methods and attributes of various resources interacting in the model.

3.2 RESOURCE ACTION MODULE [REACT]

The Resource Action Module (Fig. 1) is used to define the interactions between the entities in the simulation. The data model for its implementation is shown in Fig. (2).

Resource Action Module										
Resource			Attribute			Capability				
#	Actor	Capture	Capability	Owner	Attribute	of	Seq.	Release	Create/Dis.	Next

Fig. 1

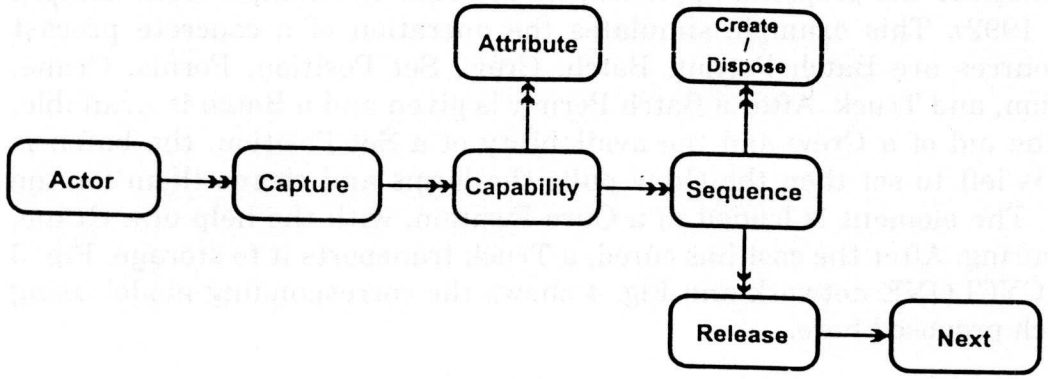

Fig. 2

An actor is an object that captures one or more resources and then proceeds to implement one or more of its own capabilities or capabilities belonging to "*captured*" or other resources. For every captured resource, the user can ask several capabilities to be invoked by entering the "*owner*" resource of these capabilities. Each of these capabilities may need input from the resources seized. This input is in the form of attributes belonging to the actor, captured resources or global variables. For example, a loader may need to inquire about the truck's capacity to determine how long it will need to load it.

Invoked capabilities may occur simultaneously or sequentially depending on the sequence number given in the Seq. column. At the completion of a capability, one or more resources may be created or disposed. (*Resources may also be created or disposed without the need of a capability invoking them*). At the completion of each capability, one or more resources may be released.

After the resource is released, it can have one of two options. It may move back and join the pool of idle resources, or it may become an actor and invoke other capabilities. A resource that is an actor and was not released continues to play that role. Each resource released may have only one or no entries in the *Next* column. If no entry is entered in the *Next* column when a resource is released, this means the resource will rejoin its pool. An entry is the *Next* column means that the resource will start capturing zero or more resources. A *Next* entry that has no corresponding resource released in the *Release* column, refers to an Actor that continues to be involved in the simulation. Parallel activities are shown as an identical number in the # column that corresponds to the number entered in the *Next* column. An entry of 0 in *Next* column refers to the first line in REACT (i.e. start of simulation). (*The proposed approach is still lacking in Start, Terminate and Branch mechanisms. These features will be added in future implementations of the system*).

4. EXAMPLE

To demonstrate the proposed approach, we selected an example from (Halpin and Riggs 1992). This example simulates the operation of a concrete precast plant. Resources are Batch Permit, Batch, Crew, Set Position, Forms, Crane, Cure Position, and Truck. After a Batch Permit is given and a Batch is available, and with the aid of a Crew and the availability of a Set Position, the batch is poured. It is left to set then the Crew pulls the forms and cleans them for the next batch. The element is loaded in a Cure Position, with the help of a Crane, for steam curing. After the cast has cured, a Truck transports it to storage. Fig. 3 shows the CYCLONE network and Fig. 4 shows the corresponding model using the approach proposed here.

5. CONCLUSION

This paper reported ongoing research aimed at simplifying the development of the simulation model. The research attempts to integrate the power of object-oriented modeling and programming to enhance the ease by which the model is built and coded in a simulation language. The interface proposed will be language independent. A "*translator*" will be needed to change this specification to a language dependent implementation.

The proposed approach is being applied in a variety of building construction scenarios to evaluate its expressive power. Future work will include mechanisms for specifying starting and stopping criteria of the simulation.

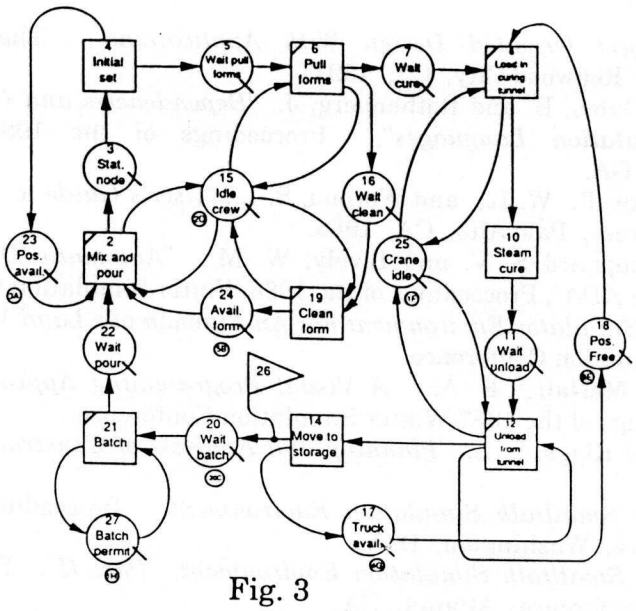

Fig. 3

Resource Action Module

Resource			Attribute			Capability				
#	Actor	Capture	Capability	Owner	Attribute	of	Seq.	Release	Create/Dis.	Next
	Batch Permit	Batch	Batch	Batch Permit				Batch Permit		
										1
1	Batch	Crew	Mix & Pour	Crew			1	Crew		
		Form								
		Set Position	Initial Set	Batch			2	Set Position		
										2
2	Form	Crew	Pull Form	Crew				Crew	+ Cast	
										3
3	Form	Crew	Clean Form	Crew				Form		
								Crew		
3	Cast	Crane	Load	Crane			1	Crane		
		Cure Position	Steam Cure	Cure Position			2			
										4
4	Cast	Crane	Unload	Crane				Crane		
								Cure Position		
										5
5	Cast	Truck	Go Storage	Truck				Truck	- Cast	
								Batch		
										0

Fig. 4

6. REFERENCES

Booch, G.: "*Object Oriented Design With Applications*", The Benjamin/Cummings Publishing Company, Redwood City, CA, 1991.

Cammarata, S., Gates, B. and Rothenberg, J.: "*Dependencies and Graphical Interfaces In Object-oriented Simulation Languages*", Proceedings of the 1987 Winter Simulation Conference, Atlanta, GA.

Conway, R., Maxwell, W. L., and Worona, S. L.: "*User's Guide to Xcell Factory Modeling System*", Scientific Press, Palo Alto, CA, 1985.

Davis, C. K., Sheppard S. V. and Lively, W. M.: "*Automatic Development of Parallel Simulation Models In ADA*", Proceedings of the 1988 Winter Simulation Conference.

Glicksman, J.: "*A Simulator Environment For An Autonomous Land Vehicle*", Proceedings of the 1986 Winter Simulation Conference.

Gordon R.F. and McNair, E. A.: "*A Visual Programming Approach To Manufacturing Modeling*", Proceedings of the 1987 Winter Simulation Conference.

Halpin, D. W. and Riggs, L. S.: "*Planning and Analysis of Construction Operations*", John Wiley, 1992.

Knapp, V. : "*The Smalltalk Simulation Environment*", Proceedings of the 1986 Winter Simulation Conference, Washington, D.C.

Knapp, V. : "*The Smalltalk Simulation Environment, Part II*", Proceedings of the 1987 Winter Simulation Conference, Atlanta, GA.

Odeh, A. M., Tommelein, I. D. and Carr, R. I.: "Using Design Drawings And Project Plans To Construct Discrete-Event Simulation Networks", Proceedings of the 8th International Symposium on Automation and Robotics In Construction, Stuttgart, Germany, June 1991.

Oloufa, A. A.: "*Object-Oriented Simulation of Earthmoving Operations*", Proceedings of the 7th International Symposium on Automation and Robotics In Construction, Bristol, England, May 1990.

Oloufa, A. A.: "*Intuitive Simulation Modeling Using Object-Oriented Constructs*", Proceedings of the 8th International Symposium on Automation and Robotics In Construction, Stuttgart, Germany, June 1991.

Oloufa, A. A.: "*Visual Simulation Programming For Construction Operations*", Proceedings of the 9th International Symposium on Automation and Robotics In Construction, Tokyo, Japan, June 1992.

Ozden, M. H.: "*Graphical Programming of Simulation Models In an Object-Oriented Environment*", Simulation, February 1991.

Riggs, L. S.: "*Interactive Graphing of Simulation Networks*", ASCE Journal of Computing, Vol. 3, No. 2, April 1991.

Roberts, S. D. and Heim, J.: "*A perspective on Object-oriented Simulation*", Proceedings of the 1988 Winter Simulation Conference.

Rothenberg, J.: "*Object-Oriented Simulation: Where Do I Go From Here?*", Proceedings of the 1986 Winter Simulation Conference, Washington, D.C.

Shannon, R.E.: "*System Simulation: The Art and Science*", Prentice Hall, Englewood Cliffs, N.J. 1975.

Shu, N. C.: "*Visual Programming*", Van Nostrand Reinhold, New York, 1988.

Thomasma T. and Ulgen, O.: "*Hierarchical, Modular Simulation Modeling In Icon-Based Simulation Program Generators For manufacturing*", Proceedings of the 1988 Winter Simulation Conference.

Tseng, F. T., Zhang, S. X. and Wolfsberger, J. W.: "*Automatic Programming Assistant For Network Simulation Models*", Proceedings of the 1988 Winter Simulation Conference.

Tumay, K.: "*Factory Simulation With Animation: The no Programming Approach*", Proceedings of the 1987 Winter Simulation Conference, IEEE, Atlanta, GA.

Ulgen, O. J.: "*Simulation Modeling in an Object-oriented Environment using Smalltalk-80*", Proceedings of the 1986 Winter Simulation Conference, Washington, D.C.