Proceedings of the 2000 Winter Simulation Conference
J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.

JAVABEANS-BASED FRAMEWORK FOR CONSTRUCTION SIMULATION

Anil Sawhney
Hemant Deshpande
André Mund

Del E. Webb School of Construction
Arizona State University
PO Box 870204
Tempe, AZ 85287-0204, U.S.A.

ABSTRACT

The modeling and analysis of construction processes is gaining recognition in the construction industry. Recent developments such as Java-based simulation are giving a unique opportunity for improvements in the modeling and analysis of construction processes. Component-based architecture such as JavaBeans can be used to develop modular simulation environments supporting high reusability of software components. This paper describes a prototype component-based architecture for construction simulation. It highlights the work performed by the authors in using JavaBeans for the simulation of construction processes.

1 INTRODUCTION AND BACKGROUND

Simulation has been used extensively in many different areas of the construction industry. It started with the introduction of CYCLONE by Halpin (1977). Over the last two decades computer simulation has established its usefulness in the area of modeling and analysis. A user can enter the fields such as number of entities in the system, delay time in the Normal activity etc. of construction processes for planning, scheduling, and control of functions, operations, and resources of a construction project (Sawhney and Mund 1999). Recent developments in object-oriented methods (Liu and Ioannou 1992) in general and Java-based programming in particular (Sawhney et al.. 1999) and construction project level simulation (Sawhney et al. 1998 and Odeh et al. 1992) are being reported by researchers in the field of construction simulation (Sawhney et al. 1999).

Simulations are usually performed in a standalone environment for testing, experimenting, and studying a system or its prototype. However, as the ability to share knowledge across wide areas increased dramatically with the help of the Internet, simulation experts began to use the web to perform simulation (Sawhney et al. 1999). Motivated by these developments and recommendations resulting from a workshop that suggested that simulation modeling systems should be made "friendlier" by using more graphics (Ibbs 1986), the authors have undertaken to develop a simulation framework geared towards construction using a JavaBeans-based, object-oriented approach.

This paper provides an overview of the ongoing work on this project. In the next section a brief introduction to the state-of-the-art of Java-based simulation is given. The concepts of Java-based simulation and JavaBeans-based simulation are outlined in the following two sections. The fifth section of the paper describes the way in which the Silk simulation modeling language was extended using JavaBeans. The sixth section gives implementation details. Conclusion and future directions are provided in the last section of the paper.

2 WHAT IS JAVA - BASED SIMULATION?

Generally, a simulation model programmed in the Java language can be defined as a Java-based simulation. However, the term "Java-based simulation" is used more broadly to point to simulation models that can be accessed over the Internet. This is also commonly referred to as "WebSim" (Powersim 1998). WebSim programs can be divided into the following two major categories (Page et al. 1997): (1) Simulation programs that can be accessed remotely through web browsers in which a single copy of the simulation runs on a server and passes the simulation results to the invoking client (Page et al. 1997); and (2) a variation of the first, but with the added feature of code mobility afforded by such network programming languages as Java. The simulation executes on the client rather than the server (Page et al. 1997). In essence the modeler develops a Java-based applet that is embedded in a web page. The construction process simulation tool described in the paper belongs to this category.

Overall, it is felt that Java-based simulation will expose the benefits of computer simulation to a larger audience of problem-solvers, decision-makers and trainers since models can be distributed and executed over the Internet using standard browser software on any operating system and hardware platform. Additionally, the use of Java as a programming language for simulation provides a number of direct benefits such as: (1) the capability of producing hierarchical, modular, and reusable simulation applications; (2) a "write once, run anywhere" platform independence that allows simulations developed using Java to be distributed and shared freely; and (3) native support for networking and common Internet protocols, database connectivity, multi-threading, distributed objects, and graphical user interfaces (Healy and Kilgore 1997).

Java-based simulation can be used beneficially in a number of ways. The following list provides some potential applications of Java-based simulation:

- Simulation for Teaching and Training: The use of simulation has the potential to significantly alter the current teaching and training methodologies (Page et al. 1997, Fishwick 1997). The primary objective of the authors in the development of the JavaBeans-based construction simulation template is to enhance the undergraduate construction engineering education.
- Web-Based Collaborative Decision-Making: Java-based simulation that can be accessed over the Internet can be effectively used as a collaborative decision making tool when the team involved in decision making is present at different locations.
- Simulation-Based Marketing: Manufacturers of products and processes can use WebSim to demonstrate their technologies to prospective users over the Internet. (Powersim 1998).
- Simulation Research Methodology: The ability to rapidly disseminate models, results, and publications on the Internet permits new approaches to the conduct of simulation research (Page et al. 1997).

3 STATE-OF-THE-ART

A number of discrete event simulation toolkits programmed in Java have been developed recently. The following paragraphs provide an overview of the important Java based simulation toolkits available.

SimProd, a tool for developing flexible simulation models of production systems on the World Wide Web (WWW), is an extension of Simjava developed by Ross McNab and Fred Howell (Kapuno and Nagarur 1999). SimProd provides different objects that represent entities existing in the production system, like machines, AGVs, conveyors and others, supporting many types of distributions, which are found useful in the manufacturing environment. Models using objects from SimProd can be implemented as applets and executed in a Web browser.

Simjava, a Java-based discrete event simulation package authored by Fred Howell and Ross McNab (Howell and McNab 1998), is conceptually based on the HASE++ simulation library (Coe et al. 1998) and the Sim++ library for C++ (Jade Simulation International 1992). Based on a discrete event simulation kernel, Simjava includes facilities for representing simulation objects as animated icons on a screen. Also, Simjava simulations may be incorporated as "live diagrams" into web documents (Howell and McNab 1998). Currently efforts for the integration of Java-based simulations written (using Simjava) and Virtual Reality Modeling Language (VRML) based animations are being conducted.

Silk™, a general-purpose simulation language written using the Java programming language, combines process-oriented modeling structures with powerful object-oriented language features in an intelligent design. This encourages model simplicity and reusability (Healy and Kilgore 1997). In Silk, models are developed directly in the Java programming language using an Application Programming Interface (API) composed of classes which consist of relatively few powerful process-oriented modeling features (Healy and Kilgore 1997). Silk also allows a modeler to develop domain specific simulation objects using the JavaBeans-based methodology. This is a key feature that makes Silk very attractive to developers. The authors used Silk to develop the JavaBeans-based construction simulation tool described in this paper.

JSIM, a simulation toolkit in which models can be built using either the event package (Event-Scheduling Paradigm) or the process package (Process-Interaction Paradigm) (Nair et al. 1997), supports a good graphical environment for displaying queues. A Java database is used for storing results.

JavaSim is a set of Java packages for building discrete event process-based simulation, similar to that in Simula and C++SIM (Little 1997). SimKit is another Java based discrete event simulation toolkit that is currently being developed (Buss and Stork 1996).

4 JAVABEANS-BASED SIMULATION

JavaBeans is a portable, platform-independent software component written in Java. It enables developers to write reusable components once and run them anywhere benefiting from the platform-independent power of Java (DeSoto 1997).

JavaBeans are essentially Java classes that follow a pre-defined property and event interface convention. These JavaBeans can be utilized to write any type of Java program. Normally, JavaBeans can be manipulated and incorporated into a Java program in any visual

development environment such as Symantec Visual Café, IBM VisualAge for Java, or Sun Java Workshop.

Java offers several features that are ideally suited to the implementation of advanced discrete-event simulation architectures and reusable simulation software components. Java based simulation results in creation of "objects".

The JavaBeans technology is based on Java and provides a means of creating and using Java classes as software components. The term software component (or simply component) differs from the term object in several subtle ways. An object is generally thought of as being the runtime incarnation of a class within a large system. Components are referred to as specific objects that are packaged and intended for reuse. Component software is a type of software that is designed heavily around the idea of code reuse and compartmentalization. (DeSoto 1997).

Component software is a very popular and powerful concept throughout the software industry to increase development efficiency. Software components are designed and built so they can be accessed and used in a variety of different development and runtime scenarios.

During the 1990's, simulation software has utilized the advantages of Object-Oriented Programming (OOP). The next step in simulation software development is to use software components. Software component begins where OOP left off and adds capabilities to maximize software reuse and facilitate rapid development through assembling components.

JavaBeans are part of this component-based architecture that support the following features:

- Introspection–a process of exposing the properties, methods, and events of a JavaBean.
- Customization–ability of objects, external to the JavaBean, to customize the appearance or behavior of the JavaBean.
- Persistence–mechanisms that result in saving an instance of the JavaBean to a disk or other storage device.
- Platform Independence–ability to work on any computing platform.
- Web-based–ability to be deployed on the Internet because of the underlying Java language.

The above features of JavaBeans can be beneficially used in the development of simulation programs in general and industry specific simulation frameworks in particular. JavaBeans that mimic the functioning of industry specific "objects" can be designed on top of a layer of a general-purpose discrete event simulation toolkit. Users can then utilize the special purpose JavaBeans to develop simulation models. Advanced users will have the option of extending the JavaBeans, developing new JavaBeans or directly accessing the general-purpose discrete event simulation toolkit to write advanced simulation models. The authors have utilized this concept to develop and test a JavaBeans based construction simulation framework.

5 JAVABEANS-BASED CONSTRUCTION SIMULATION FRAMEWORK

The authors have utilized Silk to develop a prototype JavaBeans based construction simulation framework. Silk is a collection of Java classes for discrete-event simulation. These classes include a unique "entity-thread" based simulation engine and a set of process-oriented modeling methods for true object-oriented simulation design (Healy and Kilgore 1997). Simulation models are developed directly in the Java programming language using Silk classes (Sawhney et al. 1999). The package of classes can also be extended to develop user defined JavaBeans. The hierarchy of classes used for the current research is as depicted in Figure 1.

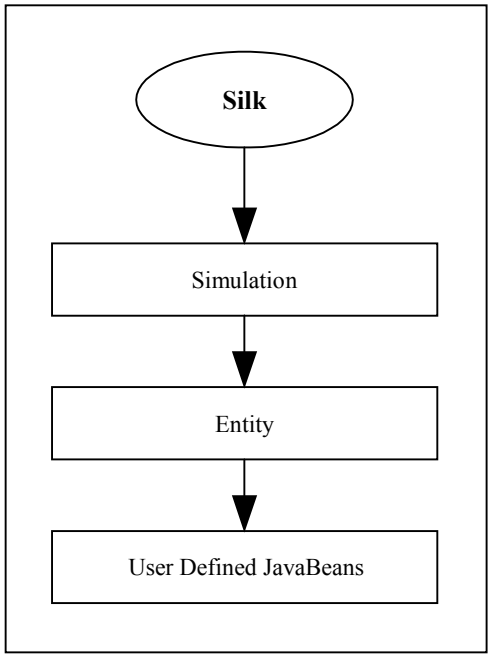

Figure 1: Hierarchy of Silk Classes

Normally a Silk user can write a simulation program in Java by directly extending the Silk classes. However, an improved approach will be to develop JavaBeans that model common scenarios encountered in construction simulation and allow the user to develop simulation models for construction processes on the "fly" by using these JavaBeans.

The authors have developed JavaBeans that can be used in the simulation of construction processes. The user-defined JavaBeans are developed by extending the Silk "entity" class.

6 IMPLEMENTATION DETAILS

The JavaBeans based construction simulation framework developed by the authors is based on the CYCLONE methodology. CYCLONE, developed by D. W. Halpin (1977) specifically for modeling of construction operations, is a popular construction modeling and simulation tool. It uses a graphical modeling format consisting of five prime elements as shown in Figure 2.

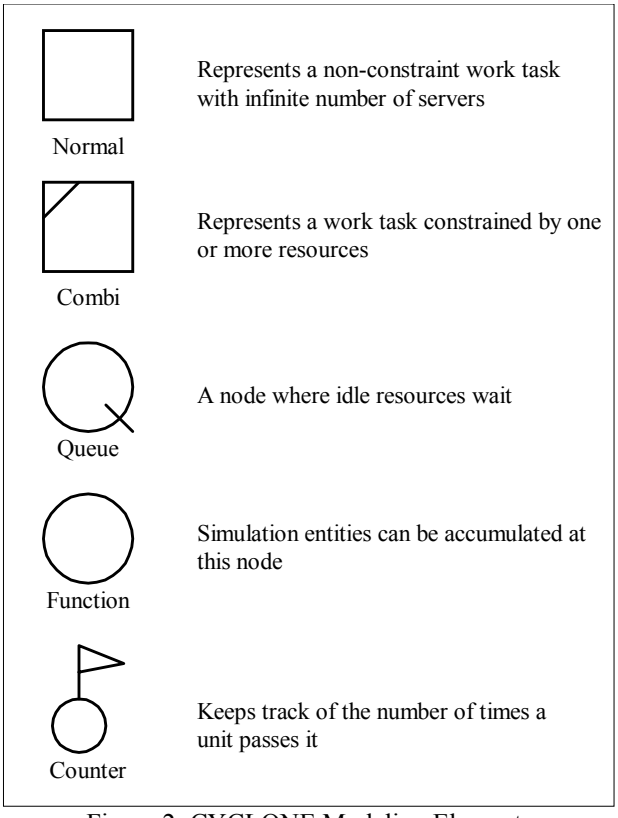

Figure 2: CYCLONE Modeling Elements

The first element, the NORMAL represents a non-constraint work task while the COMBI represents a work task that is constrained by the availability of resources. Idle resources wait at a QUEUE node and new elements/resources can be created at a FUNCTION node. Finally the COUNTER is used to keep track of the number of times a unit passes it. Simulation models for construction operations are developed using these five CYCLONE elements.

In this study JavaBeans that provide the functionality of the CYCLONE modeling element were developed. The Silk modeling language can be extended for development of these JavaBeans. Figure 3 depicts the JavaBeans developed for the CYCLONE modeling elements. The structure shows that a prototypical "ModelBean" was first developed by extending the Silk "entity" class. The JavaBeans representing the CYCLONE modeling element are then developed by extending the ModelBean. The key properties associated with the CYCLONE JavaBeans are described in Table 1.

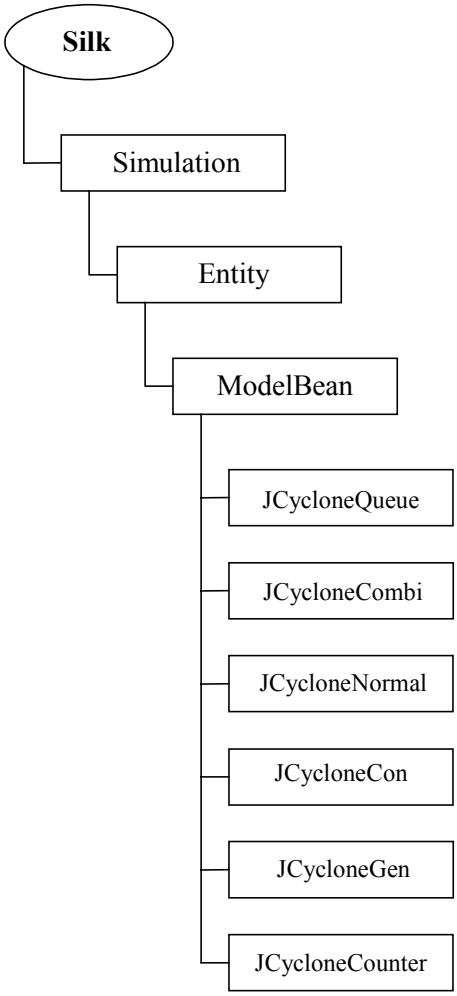

Figure 3: JavaBeans for the CYCLONE Modeling Elements

Table 1: Properties of the CYCLONE Beans

JAVABEAN	PROPERTY
JCycloneQueue	Capacity
JCycloneNormal	Service Time
JCycloneCon	No. of Entities
JCycloneGen	No. of Entities
JCycloneCount	No. to Count
JCycloneCombi	Service Time, Priority

Figure 4 shows graphically the ModelBean. The ModelBean "extends" the Silk entity class and "implements" the outputListener interface. In addition to inheriting properties, methods and events from the Silk entity class the ModelBean provides the default constructor, the "output" method and the "notifyOutput" method.

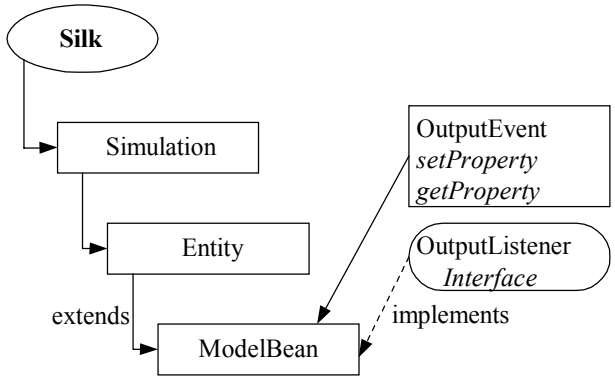

Figure 4: Graphical Representation of ModelBean

The "ModelBean" as shown in the hierarchy has all the methods defined in it. Hence, the beans in the lower level can inherit the methods implemented in the ModelBean. This is possible because of the inheritance property, which the object-oriented programming provides. As can be seen in Figure 4, the ModelBean makes use of "OutputEvent" class and "Outputlistener" interface. The "OutputEvent" class consists of a constructor, a *"setObject"* and "getObject" methods. The "setObject" method allows to set the OutputEvent to a specific object, in this case a ModelBean.

The "OutputListener" interface has an output method that needs to be implemented in all the classes, which implement this interface. An interface just defines a method, which is to be implemented by every class, which "implements" the interface. The main methods in the ModelBean are the "output" and "notifyOutput" methods. This is depicted in Figure 5.

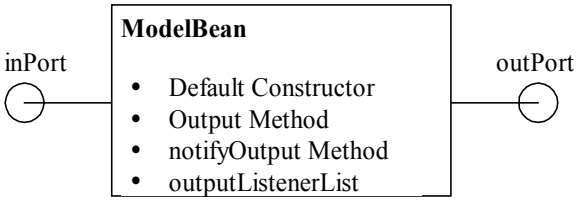

Figure 5: Detail of ModelBean

The "notifyOutput" method transfers the entities from the "inPort" of a modeling JavaBean to its "outPort". This is depicted in Figure 6.

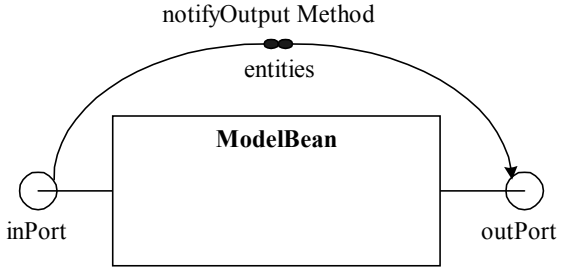

Figure 6: Working of "notifyOutput" Method

The "output" method of the ModelBean is used to transfer the entities from the "outPort" of an upstream ModelBean to the "inport" of the current (downstream) ModelBeans. In performing this transfer the "output" method uses an "OutputListenerList". The "OutputListenerList" is a vector that stores a list of downstream ModelBeans to which a ModelBean is connected. The actual transfer of entities is shown in Figure 7.

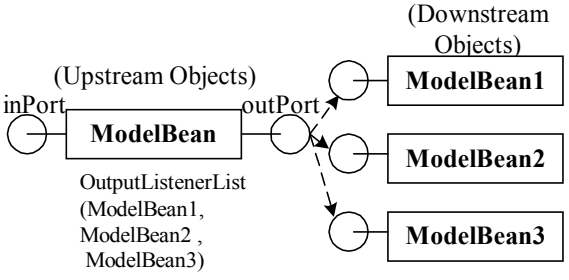

Figure 7: Working of Output Method

As seen from the figure the Upstream ModelBean is connected to three downstream ModelBeans. These downstream ModelBeans are stored in the "OutputListenerList" of the Upstream ModelBean. The "output" method of the ModelBean1 transfers an entity from the "outPort" of the Upstream ModelBean to its "inPort". Similarly, the "output" methods of the ModelBeans2 and ModelBeans3 are called to transfer the entities. An "OutputEvent" object is passed as an argument to this method in order to identify the associated downstream object. The output method uses this "OutputEvent" object to transfer the entities from the "outPort" of the upstream bean to the "inPort" of the current bean. This is depicted in Figure 7.

JavaBeans such as "JCycloneNormal" implement "push-type" logic. From a modeling standpoint, a bean implementing push-type logic, is passive, as the individual processing steps it models are performed by the entities that invoke its *pushProcess* method. The only active processing performed by a bean is to activate the entities that are passed from an upstream bean. A PushModelBean interface is used in this case. It has a *pushProcess* method. JavaBeans implementing the "PushModelBean" defines this method. In case of "JCycloneNormal" bean the entities come into the "inPort" of the bean. The entities are then delayed by the time specified using the pushProcess method and the "notifyOutput" method is called. This method transfers these entities after the specified delay time to the outPort and calls the "output" method.

For the "JCycloneQueue" bean, there could be two scenarios. Firstly, it may be the starting object and requires generation of entities. The user specifies the number of entities to be generated. It generates the required number of

entities and transfers them to the "inPort" of the Queue. Secondly, it may be an intermediate bean, which receives entities from an upstream object. In this case the entities are just transferred from the "inport" to the "outport" using the "notifyOutput" method.

Using the same process logic all the beans were developed. Now in order to develop a simulation model, any Java based visual/graphic developing environment such as Visual Café can be used. These beans can be dropped on to the work area as per the logic. The user can specify details such as number of entities in the system, delay time for the Normal activity, etc. Once all the connections are set, the user can run the model and observe the results obtained. The user has to write the Java code for simulation superclass and an applet that calls the frame that has all the connected beans forming a model.

The simulation then can be published on to a web server. A client connected to the Internet can then view the published simulation with the help of standard web browser.

Once the simulation is completed the output window displays the standard output.

7 CONCLUSIONS

This paper provides information about the development of JavaBeans based simulation for construction processes. By utilizing component-based technology such as JavaBeans the environment is built up from reusable software components that can be dynamically assembled using visual development tools. Cyclone modeling elements were used for developing the user-defined beans. Silk classes were extended for this purpose.

ACKNOWLEDGMENTS

This project was supported in part by the National Science Foundation (NSF) (Grant No. DUE 9996399) and Arizona State University's Center for Research on Education in Science, Mathematics, Engineering and Technology (CRESMET). Opinions expressed are those of the authors and are not necessarily those of NSF or CRESMET.

REFERENCES

Buss, A.H. and K.A. Stork. 1996. Discrete event simulation on the world wide web using JAVA. In *Proceedings of the 1996 Winter Simulation Conference*, ed. J.M. Charnes, D.J. Morrice, D.T. Brunner, and J.J. Swain, 780-785, Institute of Electrical and Electronics Engineers, Piscataway, New Jersey.

Coe, P.S., F.W. Howell, R.N. Ibbett, and L.M. Williams. 1998. A Hierarchical Computer Architecture Design and Simulation Environment. *ACM Transactions on Modeling and Computer Simulation*, 8(4):431-446.

DeSoto, A. 1997. Using the Beans Development Kit 1.0 – A Tutorial. JavaSoft, November 1997. 1-1.

Fishwick, P.A. 1997. Web-Based Simulation. In *Proceedings of the 1997 Winter Simulation Conference*, ed. A. Andradottir, K.J. Healy, D.H. Withers, and B.L. Nelson, 100-102. Institute of Electrical and Electronics Engineers, Piscataway, New Jersey.

Halpin, D.W. 1977. CYCLONE: Method for Modeling of Job Site Processes. In *Journal of the Construction Division*, ASCE, 103(3): 489-499.

Healy, K.J. and R.A. Kilgore. 1997. Silk™: A Java-Based Process Simulation Language. In *Proceedings of the 1997 Winter Simulation Conference*, ed. A. Andradottir, K. J. Healy, D. H. Withers, and B. L. Nelson, 475-482. Institute of Electrical and Electronics Engineers, Piscataway, New Jersey.

Howell, F. and R. McNab. 1998. Simjava: a discrete event simulation package for Java with applications in computer systems modeling. *First International Conference on Web-based Modeling and Simulation*, San Diego, CA, Society for Computer Simulation.

Ibbs, C.W. 1986. Future directions for computerized construction research. *Journal of Construction Engineering and Management*, ASCE, 112(3): 326-345.

Jade Simulations International. 1992. Sim++ User Manual. Jade Simulations International Corporation.

Kapuno Jr., R.R. and N.N. Nagarur. 1999. SimProd: A Web-Based Flexible Simulation Package for Production Systems. *EJSAT*, July 1999, <http://www.sat.ait.ac.th/ej-sat/articles/1.2/ng.html>.

Little, M.C. 1997. JavaSim World Wide Web Home Page. <http://javasim.ncl.ac.uk/>.

Liu L.Y. and P.G. Ioannou. 1992. Graphical Object-Oriented Simulation System for Construction Process Modeling. In *Proceedings of the Eighth Conference on Computing in Civil Engineering*, ASCE, Dallas, Texas, 1139-1146.

Nair, R.S., J.A.Miller, and Z. Zhang. 1997. Java-Based Query Driven Simulation Environment. In *Proceedings of the 1996 Winter Simulation Conference*, ed. A. Andradottir, K.J. Healy, D.H. Withers, and B.L. Nelson, 786-793. Institute of Electrical and Electronics Engineers, Piscataway, New Jersey.

Page E.H., R.L. Moose, and S.P. Griffin. 1997. Web-based Simulation in Simjava Using Remote Method Invocation. In *Proceedings of the 1997 Winter Simulation Conference*, ed. A. Andradottir, K. J. Healy, D. H. Withers, and B. L. Nelson, 468-474. Institute of Electrical and Electronics Engineers, Piscataway, New Jersey.

Powersim. 1998. Metro JX Sever Documentation. Powersim Corporation, Herndon, VA.

Sawhney, A., S.M. AbouRizk, and D.W. Halpin. 1998. Construction Project Simulation Using CYCLONE.

Canadian Journal of Civil Engineering, Canadian Society of Civil Engineering. 25(1): 16-25.

Sawhney, A. and A. Mund. 1998. Simulation based Construction Management Learning System. In *Proceedings of the 1998 Winter Simulation Conference*, ed. D.J. Medeiros, E.F. Watson, J.S. Carson , M.S. Manivannan, 1319-1324. Institute of Electrical and Electronics Engineers, Piscataway, New Jersey.

Sawhney, A., O. Abudayyeh and T. Chaitavatputtiporn . 1999. Modeling and Analysis of a Concrete Production Plant using Colored Petri Nets. *Journal of Computing in Civil Engineering*, American Society of Civil Engineers (ASCE), 13(3):178-186.

Sawhney, A. and A. Mund. 1999. Hierarchical and Modular Modeling of Structural Steel Erection Process Using Petri Nets. *Journal of Civil Engineering and Environmental Systems*, Gordon and Breach Publishers, 17: 63-88.

Sawhney, A., A.Mund, J. Manickam, and J. Marble 1999. Java-Based Simulation of Construction Process Using Silk. In *Proceedings of the 1999 Winter Simulation Conference*, WSC Part 2 (of 2), ed. P.A.Farrington, H.B. Nembhard, D.T. Sturrock and G.W.Evans, 985-991. Institute of Electrical and Electronics Engineers, Piscataway, New Jersey.

AUTHOR BIOGRAPHIES

ANIL SAWHNEY received his bachelor of Engineering degree from India in 1987 and a Master of Building Engineering and Management degree from School of Planning and Architecture, New Delhi in 1990. He completed his Ph.D. studies at the University of Alberta in 1994. He is currently working as an Associate Professor in the Del E Webb School of Construction at Arizona State University. His research interests are mainly focused on construction simulation techniques and use of computers in construction education. His email address is <anil.sawhney@asu.edu>.

HEMANT DESHPANDE received his Bachelor of Engineering degree from the University of Pune in India in 1997. Currently, he is pursuing his Master of Science studies in Industrial Engineering at Arizona State University. He has been working as a research assistant in the Del E Webb School of Construction at Arizona State University. His research interests centers on Java programming, simulation modeling and software development. His email address is <hemant.deshpande@asu.edu>.

ANDRÉ MUND received his Bachelor of Engineering degree from UAL in Portugal in 1994. He worked for a contractor in Berlin, Germany, from 1994 to 1997. He completed his Master of Science studies at Western Michigan University in June 1999. Currently he is pursuing a Ph.D. at Arizona State University and working as a research associate in the Del E. Webb School of Construction. He is interested in the area of heavy construction equipment selection and computing and information technology applications in construction. His email address is <amund@asu.edu>.