Proceedings of the 2004 Winter Simulation Conference
R .G. Ingalls, M. D. Rossetti, J. S. Smith, and B. A. Peters, eds.

SOME RECENT ADVANCES IN THE PROCESS WORLD VIEW

Robert G. Sargent

Department of Electrical Engineering and Computer Science
L.C. Smith College of Engineering and Computer Science
Syracuse University
Syracuse, NY 13244, U.S.A.

ABSTRACT

We discuss a modification of the process world view, a graphical modeling representation language for the modified process world view called Control Flow Graphs (CFGs), an extension to CFGs called Hierarchical Control Flow Graph (HCFG) Models, and a simulation system that uses HCFG Models called HiMASS.

1 INTRODUCTION

According to Nance and Sargent (2002), Lackner (1962) identified the concept of a simulation model having a world view, Krasnow (1967) identified different world views for continuous, discrete, and combined (discrete and continuous) simulation models including the activity, event, and process representations for the discrete world view, and Kiviat (1969) provided the first detailed comparative analysis of the basic discrete world view representations that consisted of the activity, event, process, and transactional world views. A detailed description of the activity, event, and process world views using system theory is given in Zeigler (1976). (We note that sometimes the process world view is defined as consisting of (i) an "active resource" view and (ii) a "transactional" view that characterizes only the dynamic objects as processes. In the remainder of this paper we will use the term "process world view" to mean the "pure" (or active resource) process view and the term "transactional world view" when processes are used only for dynamic objects, e.g. when processes are used for only transactions.)

Graphical representations to aid in model specification for the different world views have been developed. Activity cycle diagrams (Paul 1993) are used to aid model specifications that use the activity world view. The transactional world view simulation languages usually contain a graphical transaction flow oriented language with their introduction, e.g., GPSS (Schriber 1991). Event Graphs were introduced by Schruben (1983) and further developed by Sargent (1988) and Som and Sargent (1989) as a graphical model specification language for models using the event world view. Control Flow Graphs (CFGs), a graphical model representation language for the (modified) process world view, was developed by Cota and Sargent (1990c) (also see Cota, Fritz, and Sargent 1994 and Sargent 1994) and was later extended by Fritz and Sargent (1993, 1995) to Hierarchical Control Flow Graph (HCFG) Models (Sargent 1996) to aid in the control of representational complexity. (These latter two graphical representations are discussed in Sections 3 and 4, respectively.)

In this paper we present some advances that have occurred for the process world view. In the next section we discuss a modification to the process world view. In Section 3 we describe the graphical representation language CFGs, which is for the modified process world view presented in Section 2, and in Section 4 we describe HCFG Models. In Section 5, we briefly discuss the Hierarchical Modeling And Simulation System (HiMASS) that uses HCFG Models and in Section 6 we present the paper summary.

2 MODIFIED PROCESS WORLD VIEW

Cota and Sargent (1992) developed a modification of the process world view. This modified view has (i) "modularity," which includes encapsulation and locality, and (ii) a modified definition of control states. Modularity is very important in simulation because, as discussed in detail by Cota and Sargent (1992), modularity is required in order to have hierarchical modeling, to reuse model components, and to perform parallel simulation, and makes it easier to model complex systems and to modify models. This modified view also requires model components to interact through a strictly defined interface.

In order to have modularity, the "cancel" construct used in modeling must be eliminated. This requires a new definition of the control state. (Recall that a control state is a formalization of the "process reactivation point" (Zeigler 1976).) In the process world view, each process is defined by a procedure. Each procedure includes instructions for suspending execution for one or more intervals of time. Each interval of time may be specified explicitly, e.g., the time when a service will be completed, or may last until

some specified condition is met, e.g., the time when a server becomes available. In the classical (historical) process world view, only one situation may be specified for each suspension, i.e., either one specific time or one specific condition can be specified. In the modified view, any number of alternatives based on conditions and/or times can be specified for each suspension. The alternative that becomes "true" the earliest in simulation time is the alternative selected to reactivate the suspended procedure. (Each alternative has a unique priority specified to handle any earliest time ties.) This means a control state in the modified process world view may have multiple exiting branches with each having a unique priority; whereas, a control state in the classical view is allowed to have only one exiting branch. (See Cota and Sargent (1992) for a detail discussion and examples.)

3 CONTROL FLOW GRAPHS

Cota and Sargent (1989, 1990a, 1990b, 1990c, 1990d) developed the graphical representation language CFGs for the modified process world view discussed in Section 2. In CFGs each system component (or process) is specified by a CFG. The interactions between components are specified by message passing over directed channels connecting model components. Messages leave a component via an output port and enter a component via an input port. Each port connects to only one channel and each channel carries only one type of message, which implies that there may be multiple channels between components. Messages queue at the input port of each channel until the CFG decides to receive them; i.e., CFGs are active receivers. (This contrasts with the passive receiver model used, e.g., in object-oriented programming and simulation, where components react when information is received.) The time-stamps on the messages are their sending times. (This contrasts with the method generally used with parallel and distributed simulation, where the time-stamps are the times the messages are to be received.) The specification of channels is accomplished via an Interconnection Graph, which is a directed graph whose nodes represent the components and whose directed edges represent the channels. Thus, one Interconnection Graph and a set of CFGs (one for each model component) are required to specify a simulation model using the CFGs representation.

Each CFG has a set of (component) variables, a (component) simulation clock, and an augmented directed graph whose nodes represent the control states of a component and whose directed edges specify possible component state transitions. Each edge has three attributes: a condition, a priority, and an event. The condition specifies when an edge can be a candidate for traversal, the priority is used to break ties when more than one edge is a candidate for traversal at the same simulation time, and the event specifies the actions to be taken if that edge is traversed. An event may include receiving a single message waiting at a channel's input port, changing values of (component) variables, and sending messages over channels. There are three types of conditions: (i) an input port, (ii) a boolean expression on the values of the (component) variables, and (iii) a time delay. The input port condition is *true* if there is at least one unreceived message waiting at its associated input port, the boolean expression condition is *true* if it evaluates to true, and the time delay condition becomes *true* after a simulation time delay specified by its associated time delay function. The simulation execution algorithm examines all edges leaving the current control state and selects the edge whose condition is *true* first, i.e., at the earliest point in time starting from the current simulation time. If there is more than one such edge, then the edge with the highest priority is selected. Each CFG has its own "thread of control," which is a point of control (POC) that traverses over the selected edge from its current control state to its next current control state. When a POC traverses over the selected edge, it advances the simulation clock, if necessary, to the time at which the condition of the selected edge becomes *true*, and then executes the event on that edge. Each CFG operates independently of other CFGs except for message passing interaction.

Different simulation execution algorithms for CFGs have been developed for use on sequential, parallel, and distributed computers. A special feature of CFGs is that they make information explicit that can be used for execution algorithms. This avoids requiring the modeller to add additional information (such as lookahead information) as is usually required for parallel and distributed simulation.

There are two simulation execution algorithms (Cota and Sargent 1990b, 1990c, 1990d) available for sequential computers: synchronous and asynchronous. The synchronous algorithm uses the standard approach of executing the event with the lowest time event across all CFGs. (This algorithm has been implemented in the various versions of HiMASS.) The asynchronous algorithm allows certain events to be executed out of time sequence when such changes do not affect the simulation results in order to eliminate some event list operations, thereby reducing the simulation execution time. Both of these algorithms require priorities to be assigned to each of the CFGs (model components) to handle event time ties (Cota and Sargent 1990b, 1990c; Daum 1998; and Fritz and Sargent 1993, 1995).

For parallel and distributed simulation there are four simulation execution algorithms available (Cota and Sargent 1989, 1990a, 1990b, 1990c, 1990d): two conservative algorithms—one using null message passing and the other using deadlock detection and resolution; an optimistic algorithm, which is more efficient than the usual optimistic algorithm because rollback occurrences are based on the control states of a CFG instead of always rolling back if a "late message" is received at a CFG; and an optimistic Only When Necessary (OWN) algorithm, which is a combined optimistic and conservative algorithm. These algorithms automatically obtain information from CFGs to,

e.g., identify unconditional events and compute "internal" lookaheads, thereby eliminating the need for a modeller to add such knowledge. Cota and Sargent (1989, 1990a) also developed some algorithms to automatically generate "external" lookahead information from CFGs to be used in conjunction with these simulation execution algorithms. Zarei and Pidd (2001) empirically evaluated three of the automatic "external" lookahead algorithms by using them in conjunction with the null message conservative algorithm on a CRAY computer to simulate queueing networks. They found that the three automatic lookahead algorithms performed better than the best manually inserted lookaheads (including the computation time to calculate the lookaheads automatically). This work found that the single-pass algorithm was the best performer of the three automatic lookahead algorithms tested and in some cases performed considerable better than the other two algorithms. Zarei and Pidd also found these lookahead algorithms to be robust and application independent.

We now present some observations on CFGs. CFGs are the first graphical representation language for the process world view (excluding the transactional world view), are based on a theoretical foundation (the modified process world view), constitute a general purpose discrete event model paradigm, and allow the use of experimental frames. Furthermore, as Zarei and Pidd discuss, CFGs (i) with their graphical representation are easy to understand, are easy to communicate, and aid in model validation, (ii) provide an intermediate tool between the conceptual model and the simulation program, (iii) separate the model from the execution (processing) layer, and (iv) allow the information needed for the various simulation execution algorithms for use on different types of computer architectures to be automatically extracted from CFG models, thereby making the model execution transparent to the modeller, and that the methods used are application independent. A desired modeling feature that CFGs lack is hierarchical modeling capability (Sargent 1993).

4 HCFG MODELS

We give an overview of HCFG Models in this section. HCFG Models is a graphical hierarchical modeling (specification) language developed by Fritz and Sargent (1993, 1995) for modeling complex systems and are based on the use of CFGs. The primary objectives for HCFG Models are (i) to facilitate model development by making it easier to develop, maintain, and reuse models and model elements and (ii) to have the same or similarly flexible and efficient execution as CFGs. The first objective was accomplished through the use of encapsulated model elements and providing for the use of two types of independent and complementary hierarchical structures to CFGs. The second objective was accomplished through the use of the same simulation execution and automatic lookahead algorithms that are used for CFGs. (A system theoretic description of HCFG Models and proofs that the two structures are hierarchical are found in Fritz and Sargent (1993) and Sargent and Fritz (1993). Sargent and Daum (2004) contains a description of the development of HCFG Models.)

The Hierarchical Interconnection Graph (HIG) is one of the hierarchical structures in HCFG Models, and it permits the coupling of components. A HIG is a directed graph where the nodes are components and the edges are channels. There are two types of components: (i) Atomic Components (ACs), which are the components in CFGs, and (ii) Coupled Components (CCs), which are formed by coupling together ACs and/or other CCs. CCs are specified via a Coupled Component Specification (CCS), which is a directed graph with nodes being components and edges being channels. There is a HIG tree that contains the hierarchical relationships of the components where the leaf nodes are ACs, the internal nodes are the CCSs of the CCs, and the root node is the CCS of the top CC that encloses the entire HCFG Model.

The behavior of each AC in HCFG Models is specified by an HCFG and this is the other hierarchical structure. An AC's behavior can be recursively partitioned into a disjoint set of encapsulated partial behavior specifications called Macro Control States (MCSs). A MCS (pronounced "max") is an augmented directed graph where the nodes are other MCSs and/or Control States, the directed edges leaving MCSs have no attributes, and the directed edges leaving Control States have attributes as in Control Flow Graphs. There is an HCFG tree that contains the hierarchical relationships of the MCSs, where the root node MCS encloses the behavior of that entire AC. There is an HCFG Model tree that shows the two-tiered hierarchical structure of an entire HCFG Model, where the top tier is the HIG tree and each leaf of the HIG tree has that AC's HCFG tree.

The model elements of HCFG Models are ACs, CCs, MCSs, events, and edge conditions. Each model element and each HCFG Model have "types" and "instances." Types represent the specifications of models or model elements, and instances represent concrete representations of models or model elements. Libraries of types can be created, and this allows for reuse of model elements and models (Daum and Sargent 1999). Scaling of model elements is possible in HCFG Models by using arrays of model elements (Daum and Sargent 1999). Furthermore, MultiChannels and MultiEdges can be used to connect arrays or single entities of components and MCSs, respectively (Daum and Sargent 1999).

Numerous HCFG Models have been developed. This includes several "toy models." A set of components for modeling queueing systems has been developed for a library (Sargent 1997). Two large-scale models have been reported in the literature: a surveillance radar system (Farr et al. 1995) and a traffic intersection (Daum 1997). The latter model had over 400 AC instances from 14 AC types and over 60 CC instances from 20 CC types.

We now evaluate HCFG Models. We first note that HCFG Models specify a graphical hierarchical modeling language that uses the process interaction world view. This modeling language has all of the positive features mentioned above for Control Flow Graphs, and it addresses its main weakness, namely the lack of hierarchical modeling capability. HCFG Models have two powerful hierarchical modeling capabilities: the coupling of components and the recursive partitioning of the behavior of each AC through the use of MCSs. Since each of these hierarchical structures is independent of the other and in its use, different levels of hierarchicalness can be used in the coupling of components and for specifying the behaviors of ACs. We note that while both of these structures are simple to understand and use, the use of coupling of components is much simpler than the use of MCSs. HCFG Models use the layered approach to modeling (Sargent and Daum 1998) and have a separate layer for the simulation execution algorithms. The layered approach allows at any level of model granularity (i) the use of (existing) model elements from libraries, (ii) the specification of new model elements, and (iii) the examination of the specifications of existing model elements. HCFG Models allow the use of scaling, reuse of model elements, and the use of an Experimental Frame (Daum and Sargent 1999, 2001).

Several observations about modeling with HCFG Models have been developed from modeling systems using HCFG Models. First, messages can be used as part of the "modeling process" when modeling using active components (active resources). In the modeling of queueing systems, for example, messages can be used to represent customers when active components are used to represent queueing subsystems (Sargent 1997). Second, modeling can occur from top-down, bottom-up, or as a mixture of the latter two (Daum and Sargent 1999). Third, there is considerable flexibility in specifying HCFG Models, e.g., in how to divide a model into Components and into MCSs. Fourth, *simple* ACs are almost always used, which usually result in only a few levels of hierarchicalness being used in the MCSs. Fifth, CCs are commonly used and frequently with several levels of hierarchicalness. Sixth, many instances and few types are generally used for both ACs and CCs. Seventh, modeling using ACs and CCs seem more straightforward and simpler than the modeling of the behaviors of ACs. Eighth, it appears that ACs are best modelled by keeping events and boolean edge conditions "simple," an approach that aids model verification and validation. Ninth, the number of control states usually increases as *simpler* events and boolean edge conditions are used. Tenth, understanding the modeling of systems using HCFG Models appears to be a rich area for research.

5 HiMASS

Since 1993, three major versions of HiMASS have been developed for the use of HCFG Models: a simple prototype implemented in C++ called HI-MASS (Fritz, Daum, and Sargent 1995; Fritz, Sargent, and Daum 1995; Sargent and Fritz 1995), a major system with an elaborate model development environment implemented in Java called HiMASS-j (Daum 1998; Daum and Sargent 1997, 2001), and an improved system implemented in Java and XML (Bray et al. 2004) called HiMASS-x (Daum and Sargent 2002) (and recently named HiMASS with additional capabilities to be added). The development of new versions was motivated by advances in software engineering that allowed for significant improvements in the implementation and from the lessons learned from working with existing versions. Only an overview of the various versions of HiMASS will be presented. (A more detailed description of the various versions of HiMASS with observations is given in Sargent and Daum 2004.)

5.1 HI-MASS (C++ Version)

From 1993 to 1995, a C++ based prototype called HI-MASS was developed under sponsorship of the U. S. Air Force's Rome Laboratory. The major purposes of this prototype were to demonstrate the feasibility of and to evaluate the usability of HCFG Models as a way of specifying hierarchical models for discrete event simulation (Sargent and Fritz 1995). This prototype was designed to run on Sun-OS/Solaris workstations but also ran on other Unix and Linux based systems that had a compatible C++ compiler. HI-MASS was implemented using the freely available GNU C++ Compiler (G++) and C++ library (libg++) and is object oriented. The Graphical User Interface (GUI) for specifying HIGs via Visual Interactive Modeling (VIM) was implemented using the InterViews toolkit (Linton et al. 1992). HI-MASS consists of 60 C++ classes and approximately 25,000 lines of code.

HCFG Models are specified in HI-MASS using two complementary specification structures. The HIG is specified via VIM using a GUI. The behavior of each AC is specified by writing C++ code using predefined classes and functions. A model's initial conditions are specified in the simple Experimental Frame contained in HI-MASS. HI-MASS requires the executable models to be completely specified in C++. The sequential synchronous simulation execution algorithm is used in the simulation engine.

This prototype showed that a powerful model development environment for specifying HCFG Models is possible. Such an environment would include (i) VIM capability using GUIs and dialog boxes for specifying both HIGs and MCSs, (ii) a model navigator to display and move through the model tree, (iii) scalability and reuse of model elements, (iv) libraries of model elements, (v) parameters in model elements, and (vi) an Experimental Frame with additional capabilities.

5.2 HiMASS-j (Java Version)

Beginning in 1996, a new implementation of HiMASS was started from scratch as a completely Java-based software

system called HiMASS-j. This system has an elaborate model development environment with a modern approach to the Experimental Frame (Daum and Sargent 2001). The GUI toolkit features built into the Java Development Kit (JDK) allowed for the implementation of advanced, cross-platform GUIs. The use of the JDK and Java GUI features made it possible to develop GUIs and dialog boxes to specify via VIM the HIG of a model and the MCSs that define the HCFG of each AC, and also to display the model tree with interactive capability. HiMASS-j has over 160 Java classes and runs on a wide variety of computer architectures including UNIX-like systems and various versions of Windows operating systems.

HiMASS-j provides for the use of parameters and variables in model elements. The values for these can be specified when building the model elements or in the Experimental Frame (EF) that is used in HiMASS-j. When specifying these entities in model elements via VIM, those whose values need to be specified in the EF are automatically entered into the EF.

HiMASS-j also provides for the use of model element libraries, which can easily be used when specifying model elements via VIM. HiMASS-j uses the layered approach to modeling, where one can model at any level using pre-specified model elements or specify new model elements from scratch, and also look inside model elements to see their internal specifications.

In HiMASS-j, a very strict object-oriented approach has been applied to every aspect of a model. This allows model entities (e.g., ACs and parameters) to be easily used or changed when specifying models and in the EF. Properties that can be changed through the EF are strictly encapsulated (i.e., saved with the element to which it belongs). This means, e.g., that if a modeller changes a variable name somewhere in the model, the change is automatically made in the EF. Also, if a modeller adds a new model element to a model from a model element library, the EF automatically has the appropriate properties of that model element.

Reuse of model elements is one of the major features of HiMASS-j. HiMASS-j allows for the reuse of both customized and uncustomized model elements. Allowing model elements to have parameters and variables provides for considerable reuse of model elements. HiMASS-j also provides for scaling through the use of arrays of components, MCSs, channels, and edges. An array can be static (i.e., the number of elements in the array cannot change) or dynamic where the number can be specified, e.g., through the EF. Combining scaling with reuse and an integrated EF provides an efficient way for the building and customization of models.

5.3 HiMASS-x (XML Version)

Starting in 2001, another major version of HiMASS called HiMASS-x was developed. While the software is still implemented entirely in Java, all data formats are now based on XML. This includes the graphical representations of the models, (reusable) model elements, component libraries, EFs, and executable models. Basing HiMASS data formats on XML allows modellers to easily access shared model element libraries on other systems. Parties in different locations can collaboratively build, manipulate, and use models.

The emerging technology of XML allows disparate data formats to be consolidated into a standardized format that is human-readable, portable, and easily processed by software following standardized rules. Previous versions of HiMASS focused primarily on how to implement desired properties of a modern modeling system. Basing these properties in XML allows for easier interaction between HiMASS model elements and other facets of technology, resulting in more flexibility and increased ease of use.

HiMASS-x has the same capabilities as HiMASS-j except that HiMASS-x has superior "communication" capabilities because of the use of XML. (We note that HiMASS-x has been renamed HiMASS and is available as a commercial product with planned additional capabilities. See the web page <`http://www.himass.com`>.)

6 SUMMARY

We presented some recent advances in the process world view. A modified version of the process world view that provides for modularity, which has numerous advantages, was described. We discussed CFGs, which has numerous features and was the first graphical language for the process world view. An overview of HCFG Models, which is an extension of CFGs to provide hierarchical modeling, was presented. Lastly we discussed the various versions of HiMASS. The last two versions of HiMASS provide an extremely powerful modeling environment for developing HCFG Models.

REFERENCES

Bray, T., J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau, eds. 2004. *Extensible Markup Language (XML) 1.0*, 3rd ed. World Wide Web Consortium Available online via <`www.w3.org/TR/REC-xml/`> [accessed August 18, 2004].

Cota, B. A. and R. G. Sargent. 1989. Automatic lookahead computation for conservative distributed simulation. CASE Center Technical Report No. 8916, Syracuse University, Syracuse, New York.

Cota, B. A. and R. G. Sargent. 1990a. A framework for automatic lookahead computation in conservative simulation. In *Proceedings of 1990 Distributed Simulation Conference*, ed. D. Nicol, 56-59. San Diego, CA: Society for Computer Simulation.

Cota, B. A. and R. G. Sargent. 1990b. Simulation algorithms for control flow graphs. CASE Center Technical Report No. 9023, Syracuse University, Syracuse, New York.

Cota, B. A. and R. G. Sargent. 1990c. Control flow graphs: A method of model representation for parallel discrete

event simulation. CASE Center Technical Report No. 9026. Syracuse University, Syracuse, New York.

Cota, B. A. and R. G. Sargent 1990d. Simultaneous events and distributed simulation. In *Proceedings of the 1990 Winter Simulation Conference,* ed. O. Balci, R. P. Sadowski, and R. E. Nance, 436-440. Piscataway, NJ: IEEE.

Cota, B. A. and R. G Sargent. 1992. A modification of the process interaction world view. *ACM Transactions on Modeling and Computer Simulation* 2 (2): 109-129.

Cota, B. A., D. G. Fritz, and R. G. Sargent. 1994. Control flow graphs as a representation language. In *Proceedings of the 1994 Winter Simulation Conference*, ed. J. D. Tew, M. S. Manivannan, D. A. Sadowski, and A. F. Seila, 555-559. Piscataway, NJ: IEEE.

Daum, T. 1997. An HCFG Model of a traffic Intersection specified using HiMASS-j. In *Proceedings of the 1997 Winter Simulation Conference*, ed. S. Andradottir, K. J. Healy, D. Withers, and B. L. Nelson, 158-165. Piscataway, NJ: IEEE..

Daum, T. 1998. An investigation into specifying HCFG Models using visual interactive modeling. Graduate Thesis, Department of Simulation and Graphics, Otto von Guericke University, Magdeburg, Germany.

Daum, T. and R. G. Sargent. 1997. A Java based system for specifying Hierarchical Control Flow Graph Models. In *Proceedings of the 1997 Winter Simulation Conference*, ed. S. Andradottir, K. J. Healy, D. Withers, and B. L. Nelson, 150-157. Piscataway, NJ: IEEE.

Daum, T and R. G. Sargent. 1999. Scaling, hierarchical modeling, and reuse in an objected-oriented modeling and simulation system. In *Proceedings of the 1999 Winter Simulation Conference*, ed. P. A. Farrington, H. Black Nembhard, D. T. Sturrock, and G. W. Evans, 1470-1477. Piscataway, NJ: IEEE.

Daum, T. and R. G. Sargent. 2001. Experimental frames in a modern modeling and simulation system. *IIE (Institute of Industrial Engineering) Transactions* 33 (3): 181-192.

Daum, T. and R. G. Sargent. 2002. A web-ready HiMASS: Facilitating collaborative, reusable, and distributed modeling and execution of simulation models with XML. In *Proceedings of the 2002 Winter Simulation Conference*, ed. E. Yucesan, C.-H Chen, J. L. Snowdon, and J. M. Charnes, 634-640. Piscataway, NJ IEEE.

Farr, S. D. A. F. Sisti, D. G. Fritz, and R. G. Sargent. 1995. A simulation model of a surveillance radar data processing system using HI-MASS. In *Proceedings of the 1995 Winter Simulation Conference*, ed. C. Alexoppoulos, K. Kang, W. R. Lilegon, and D. Goldsman, 1364-1370. Piscataway, NJ: IEEE.

Fritz, D. G., T. Daum, and R. G. Sargent. 1995. Users Manual for HI-MASS. Simulation Research Group, Syracuse University, Syracuse, NY.

Fritz, D. G. and R. G. Sargent. 1993. Hierarchical Control Flow Graph Models. CASE Center Technical Report No. 9323, Syracuse University, Syracuse, NY.

Fritz, D. G. and R. G. Sargent. 1995. An overview of Hierarchical Control Flow Graph Models. In *Proceedings of the 1995 Winter Simulation Conference,* ed. C. Alexoppoulos, K. Kang, W. R. Lilegon, and D. Goldsman, 1347-1355. Piscataway, NJ: IEEE.

Fritz, D. G., R. G. Sargent, and T. Daum. 1995. An overview of HI-MASS (Hierarchical Modeling and Simulation System). In *Proceedings of the 1995 Winter Simulation Conference*, ed. C. Alexoppoulos, K. Kang, W. R. Lilegon, and D. Goldsman, 1356-1363. Piscataway, NJ: IEEE.

Kiviat, P. J. 1969. Digital computer simulation: computer programming languages. RAND Memo RM-5883-PR, RAND Corporation, Santa Monica, CA.

Krasnow, H. S. 1967. Dynamic representation in discrete interaction simulation languages. In *Digital Simulation in Operational Research*, ed. S.H. Hollingdale, 77-92. New York: American Elsevier Publishing Company.

Lackner, M. R. 1962. Toward a general simulation capability. In *Proceedings of the AFIPS 1962 Spring Joint Computer Conference*, 1-14. San Francisco, CA.

Linton, M. A., P. Calder, J. Interrante, S. Tang, and J. Vlissides. 1992. *Interviews reference manual*, 3.1 ed. CSL, Stanford University, Stanford, CA..

Nance, R. E. and R. G. Sargent. 2002. Perspectives on the evolution of Simulation. *Operations Research* 50 (1): 161-172.

Paul, R. J. 1993. Activity cycle diagrams and the three phase method. In *Proceedings of the 1993 Winter Simulation Conference*, ed. G. W. Evans, M. Mollaghasemi, E. C. Russell, and W. E. Biles, 123-131. Piscataway, NJ: IEEE.

Sargent, R. G. 1988. Event graph modelling for simulation with an application to flexible manufacturing systems. *Management Science* 34 (10): 1231-1251.

Sargent, R. G. 1993. Hierarchical modeling for discrete event simulation. In *Proceedings of the 1993 Winter Simulation Conference*, ed. G. W. Evans, M. Mollaghasemi, E. C. Russell, and W. E. Biles, 569-572. Piscataway, NJ: IEEE.

Sargent, R. G. 1994. Control flow graphs as a paradigm for discrete event simulation. In: *Proceedings of the 1994 IEEE Mohawk Valley Section Technologies & Applications Conference*, I33-I36. Piscataway, NJ: IEEE.

Sargent, R. G. 1996. An introduction to Hierarchical Control Flow Graph Models. In *Proceedings of the 18th International Conference on Information Technology Interfaces*, ed. D. Kalpic and V. Hljuz Dobric, 21-23, Pula, Crotia.

Sargent, R. G. 1997. Modeling queueing systems using Hierarchical Control Flow Graph Models. *Mathematics and Computers in Simulation* 44 (3): 233-249.

Sargent, R. G. and T. Daum. 1998. Visual interactive modeling in a Java-based hierarchical modeling and simulation system. In *Proceedings of the Simulation und Visualisierung '98 Conference*, ed. P. Lorenz and B. Preim, 1-17, Magdeburg, Germany. Ghent, Belguim: Society for Computer Simulation International.

Sargent, R. G. and T Daum. 2004. Hierarchical Control Flow Graph Models and HiMASS. In *Proceedings of the 2004 Operational Research Society Simulation Workshop*, ed. S. C. Brailsford, L Oakshott, S. Robinson, and S. J. E. Taylor, 83-92. Birmingham, United Kingdom: The Operational Research Society.

Sargent, R. G. and D. G. Fritz. 1993. Hierarchical modeling and simulation, Final Report for the AFSOR Summer Research Extension Program.

Sargent, R. G. and D. G. Fritz. 1995. Hierarchical modeling and simulation system (HI-MASS), Final Technical Report RL-TR-95-184, Rome Laboratory USAF, Rome, NY.

Schriber, T. J. 1991. *An introduction to simulation and GPSS*. New York: John Wiley and Sons, Inc.

Schruben, L. W. 1983. Simulation modeling with event graphs. *Communications of the ACM* 26 (11): 957-963.

Som, T. K. and R. G. Sargent. 1989. A formal development of event graphs as an aid to structured and efficient simulation programs. *ORSA Journal on Computing* 1 (2): 107-125.

Zarei, B. and M. Pidd. 2001. Performance analysis of automatic lookahead generation by control flow graphs: Some experiments. *Simulation Practice and Theory* 8 (8): 511-527.

Zeigler, B. P. 1976. Theory of modelling and simulation. New York: John Wiley & Sons, Inc.

AUTHOR BIOGRAPHY

ROBERT G. SARGENT is a Professor Emeritus of Syracuse University. He received his education at The University of Michigan. Dr. Sargent has served his profession in numerous ways including being the General Chair of the 1977 Winter Simulation Conference, serving on the WSC Board of Directors for ten years, chairing the Board for two years, being a Department Editor for the *Communications of the ACM*, and currently holding the presidency of the WSC Foundation. He has received several awards including the INFORMS-College on Simulation Lifetime Professional Achievement Award and their Distinguished Service Award. His current research interests include the methodology areas of modeling and of discrete event simulation, model validation, and performance evaluation. Professor Sargent has published extensively and is listed in Who's Who in America. His web and e-mail addresses are <www.cis.syr.edu/srg/rsargent/> and <rsargent@syr.edu>.