Proceedings of the 2002 Winter Simulation Conference
E. Yücesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes, eds.

A WEB-READY HiMASS: FACILITATING COLLABORATIVE, REUSABLE, AND DISTRIBUTED MODELING AND EXECUTION OF SIMULATION MODELS WITH XML

Thorsten S. Daum

HiMASS
Hierarchical Modeling and Simulation Systems
2 Goethe Street
Binghamton, NY 13905, U.S.A.

Robert G. Sargent

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

ABSTRACT

We investigate the use of XML as an open, cross-platform, and extendable file format for the description of hierarchical simulation models, including their graphical representations, initial model conditions, and model execution algorithms. We present HiMASS-x, an XML-centered suite of software applications that allows for cross-platform, distributed modeling and execution of hierarchical, componentized, and reusable simulation models.

1 INTRODUCTION

This paper discusses the benefits of using the Extensible Markup Language (XML, see Bray et al. 2000) in a new version of HiMASS, the Hierarchical Modeling and Simulation System. An older prototype, called HiMASS-j, was completely implemented in Java (Eckel 1998) and used Java source code and proprietary binary data formats. (An introduction to HiMASS-j can be found in Daum and Sargent (1997, 1999, 2001) and an in depth discussion in Daum (1998).)

This new version of HiMASS will be called HiMASS-x and it consists of two applications. HiMASS Modeler is a powerful graphical user interface (GUI) that provides visual interactive modeling (VIM) capabilities for hierarchical discrete event simulation (DES) models. HiMASS models use the Hierarchical Control Flow Graph (HCFG) Model paradigm (Fritz and Sargent 1995, Sargent 1997.) The other application, HiMASS Engine, provides implementation of simulation algorithms for HCFG models.

Section 2 gives a brief overview of the HCFG Model paradigm, a hierarchical model paradigm for DES that includes flexible hierarchical modeling and reuse of model elements (MEs.)

Section 3 list a few properties of XML that are useful for HiMASS.

Section 4 discusses the hierarchical decomposition of HCFG models. An approach to using XML to aid the modeling process is given. As it is often desirable to specify certain aspects of model behavior in a high level programming language, an approach that integrates programming code into XML is presented. To a large extent, HiMASS-x removes language dependence from the model specification. Furthermore, HiMASS-x allows for model documentation to be easily generated.

Section 5 briefly discusses the benefits of using XML for the VIM aspects (Sargent and Daum 1998) of a model, which provides access to VIM specifications for different applications.

Section 6 discusses the impact of using XML on the implementation of Experimental Frames.

Section 7 discusses the use of XML in the HiMASS Engine, allowing for a pluggable, arbitrary combination of local and distributed resources such as Experimental Frames and ME libraries.

Section 8 contains the conclusions.

2 THE HCFG MODEL PARADIGM

HCFG Models define a modeling paradigm for discrete event simulation modeling. Conceptually, HCFG Models consist of a set of independent, encapsulated, concurrently operating (Atomic) Components where each (Atomic) Component has its own thread of control and the Components interact with each other solely via message passing. Two primary objectives for HCFG Models are: (i) to facilitate model development by making it easier to develop, maintain, and reuse models and MEs and (ii) to support the flexible and efficient execution of models.

In an HCFG Model, the model Components and their interconnections (i.e., the Channels) are specified via a Hierarchical Interconnection Graph (HIG). A HIG is a hierarchical structure which allows a modeler to specify model Components hierarchically by supporting the concept of

"coupling" together existing model Components to form new model Components. Each model has exactly one HIG.

The basic building block in the HIG is the model Component. Model Components are encapsulated entities which have an external view and an internal view. The external view is the interface of the Component, which describes how the Component can interact with other Components. The internal view describes the implementation of the Component. From the external view, all model Components have the following attributes: a name (instance name), a type (type name), a set of input ports, and a set of output ports. (Internal views are covered below.) The distinction between "instance" and "type" is significant. If multiple model Components are "instances" of the same type of Component, then those Components all share the same type definition.

HCFG Models use two different classes of model Components: Atomic and Coupled. An Atomic Component (AC) is an independent, encapsulated, concurrently operating entity whose behavior is specified via a corresponding Component behavior specification, which gives the AC's internal view.

Coupled Components (CCs) are encapsulated model Components formed by coupling together other Components (atomic and/or coupled) to form new Components. CCs do not have behavior specifications. The internal view of a CC is the view from inside the Component but outside all enclosed sub components. The internal view of a CC is specified via a "Coupled Component Specification (CCS)". A CCS specifies (i) a set of sub components which are coupled together to form a new CC type and (ii) how those sub components are interconnected.

Each AC is encapsulated and has an HCFG, a set of (local) variables including a (local) simulation clock, and a point of control (POC). The behavior of each type of AC is specified by an HCFG, which is state based. An HCFG is a hierarchical structure which allows a modeler to specify an AC's behavior by recursive decomposition of its state space into a disjoint set of encapsulated partial behaviors called Macro Control States (MCSs) (pronounced "max"). A MCS is specified via a MCS specification structure, which is an augmented directed graph where the nodes are (other) MCSs and/or control states. A control state (CS) is a formalization of the "process reactivation point" (Cota and Sargent 1992, Zeigler 1976). (The POC moves from CS to CS and the movement of the POC gives the thread of control of an AC.) Edges leaving MCSs in the augmented graphs have no attributes while edges leaving CSs have three attributes: a condition, a priority, and an event. The condition specifies when an edge can become a candidate for traversal by the POC, the priority is used to break time ties when more than one edge exiting a CS is a candidate for traversal at the same simulation time, and the event is executed whenever that edge is traversed via the POC during simulation execution. An AC changes state when that AC's POC moves over an edge exiting its current CS to an adjacent CS (causing the event on that edge to be executed) in that AC's HCFG. The simulation execution algorithm moves the POC of each AC. Sequential, parallel, and distributed simulation executional algorithms exist for HCFG Models (Cota and Sargent 1990). Priorities need to be assigned to the ACs for the sequential simulation execution algorithm to handle time ties of events among ACs. (See Daum 1998 for further discussion of AC priorities.)

Each HCFG Model has a model tree. A model tree consists of a HIG tree and a HCFG tree for each AC. The HIG tree 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. An HCFG tree contains the specification structures of the MCSs in an AC's HCFG as its nodes and its root MCS enclose the entire behavior specification of that AC. In the model tree, each leaf node of the HIG tree has that AC's HCFG tree. Thus a model tree shows the two-tiered hierarchical structure of an entire HCFG Model.

MEs in the HCFG Model paradigm include CCs, ACs, MCSs, edge conditions, and events. These MEs have type–instance relationships and share importance characteristics such as reusability.

3 PROPERTIES OF XML

XML is a data format for structured information interchange. It was developed as a restricted form of the Standardized Markup Language (SGML) with several design goals:

- Straightforward usability over the Internet.
- XML documents can be processed by readily available, standardized software.
- XML documents can be easily transformed.
- XML documents are human-readable.

One major extension of XML over SGML is that the XML specification not only specifies the syntax of XML documents, but also the behavior of programs processing XML.

Software developers seeking to access, modify, or generate XML documents can rely on software packages that are freely available for virtually any platform and that behave in predictable ways due to the standardized XML specification. Traditionally, developing and sharing new data formats frequently presented significant software engineering challenges, requiring detailed syntax specifications as well as extensive programming efforts. Using XML, it is sufficient to agree on a set of XML tags and their meaning in the given context and programming to integrate new formats is kept to a minimum.

4 HIERARCHICAL MODELING

As modelers build more complex models, hierarchical modeling becomes an important issue. Hierarchical modeling provides the ability to partition a model specification into components which in turn can be recursively partitioned into (sub) components, resulting in a hierarchical specification structure of the model (Daum 1998, Daum and Sargent 1999.)

Partitioning models into hierarchical sub components can be crucial for the manageability of complex models. Without hierarchical specification structures, it can be a challenge to develop and present large models, which may contain hundreds of components. Hierarchical modeling allows for the specification of a model at different levels of abstraction, which can help in the verification and validation of a model.

This hierarchical decomposition of models leads to a model which is specified as a number of trees of encapsulated MEs, where only the leaf nodes contain arbitrary code specifying model behavior. All other MEs merely contain sub elements, information on how these are interconnected, and mechanisms for passing values of ME variables and parameters up and down the model tree.

Traditional modeling systems typically specify such non-leaf MEs in either an internal, proprietary data format or in a high level programming language. The first approach sacrifices inter-operability with systems that do not understand the particular data format, and the second approach ties the model to a particular programming language, which may limit the choice of simulation engines, data collection and data presentation tools, etc.

The HiMASS-j prototype used both approaches. During model development, all MEs were saved by the HiMASS Modeler GUI in a binary, proprietary format. This made it extremely difficult to share models with other tools for applications such as model presentation and documentation. For model execution, the HiMASS-j GUI would generate the model specification as Java source code, which could then be compiled into an executable model. Figure 1 shows a trivial CC in the HiMASS Modeler GUI and Table 1 shows the corresponding Java source code generated by HiMASS-j.

While this approach presents some "openness" (i.e., developers can read and manipulate the Java source and do not have to use the GUI to specify MEs), the model specification is still tied to the Java language. Furthermore, using a high level programming language to describe a ME that merely serves as a container for variables, sub elements, and their interconnections, is unnecessarily complex. The need to recompile the model (or parts thereof) after every change made with the GUI can also slow rapid model development.

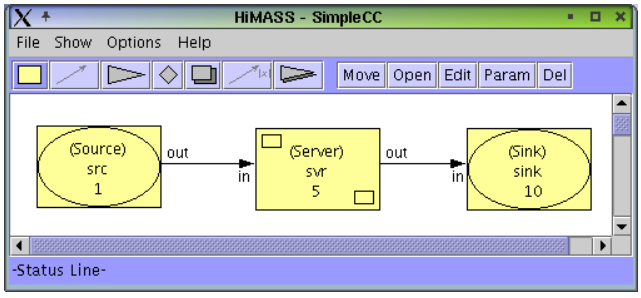

Figure 1: Simple CC with Sub Components

Table 1: CC Specification in Java Source Code

Filename: SimpleCC.java

```
import himass.sim;

public class SimpleCC extends CC {
 Sink sink;
 Source src;
 Server svr;

 public SimpleCC(int t1, int t2) {
   add("src", src = new Source(t1), 1);
   add("svr", svr = new Server(t2), 5);
   add("sink", sink = new Sink(), 10);
}

 protected void connect() {
   connect(svr.out, sink.in);
   connect(src.out, svr.in);
 }
}
```

4.1 Language Independent Model Elements

With HiMASS-x, MEs are saved in XML. Table 2 shows the same simple CC saved in XML format. It shows the type definition of a CC named "SimpleCC". The ME contains three ACs as sub elements "src", "svr", and "sink", which are connected by two channels. When an instance of "SimpleCC" is invoked, it passes two integer parameters to two of its child AC's. Figure 2 shows how these parameters can be specified in the HiMASS Modeler GUI.

In this example, the type definitions of the AC's are provided by an external ME type library (which is not shown.) It is possible to include the type definition of a sub element inline by defining a `<children>` element inside the `<ac>` element or to dynamically load a ME type definition over the network, by adding a `ref` attribute referencing the URL of the type to the `<ac>` element. This feature allows for greater flexibility than the Java source

Table 2: CC Specification in XML

Filename: SimpleCC.xml

```
<me type="cc" name="SimpleCC">
 <children>
   <ac name="src" type="Source" priority="1">
     <param name="t1" type="int" pass="t1"/>
   </ac>
   <ac name="svr" type="Server" priority="5">
     <param name="t2" type="int" pass="t2"/>
   </ac>
   <ac name="sink" type="Sink" priority="10"/>
 </children>
 <connections>
   <channel from="src.out" to="svr.in"/>
   <channel from="svr.out" to="sink.in"/>
 </connections>
</me>
```

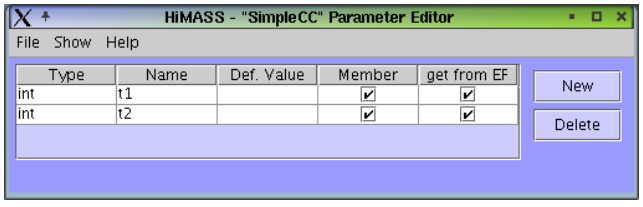

Figure 2: HiMASS Parameter Editor

code solution, as there, ME types must always be defined as separate Java class files.

This XML definition is the functional equivalent of the previous Java class definition. The XML can be created with the HiMASS Modeler GUI, but also with a standard XML editor or even a simple text editor. XML files can be shared with and modified by other developers without the need for any specialized software.

4.2 Model Documentation

One of the benefits of XML is the availability of free, standardized software that allows for the easy transformation of XML documents. By applying a style sheet to an XML document, its data can be converted into a web page, a printable document, or almost any other format. This allows one to easily generate documentation for a model, ME, or ME library–directly from the XML-encoded specification.

Figure 3 shows a partial screen shot of a web page (in HTML format) that was generated by applying an XSLT (Clark 1999) style sheet to the ME type definition in Table 2.

The XSLT style sheet for generating HTML documentation from model specifications is available online at <http://himass.com/wsc02>.

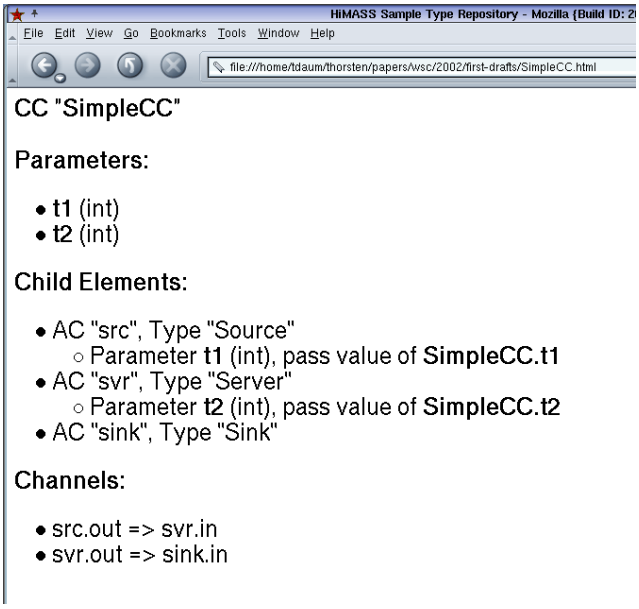

Figure 3: Web Page Generated from CC Definition

4.3 Behavior Specification

While the hierarchical structure of a model can be adequately expressed using the hierarchical properties of XML, HCFG model specifications can also contain arbitrary code to specify the behavior of edge conditions and event routines. Such code can simply be inserted between the appropriate XML elements.

Figure 4 shows how an exponential delay can be modeled as a reusable MCS using the VIM capabilities of the HiMASS Modeler GUI and Figure 5 shows how arbitrary code can be specified, e.g., for a time delay condition. Table 3 shows an XML-formatted exponential time delay edge condition.

Constructs such as class and variable declarations are still XML-encoded. Often, these constructs have different syntaxes in different programming languages and thus make cross-language development difficult. Keeping them in XML maintains language independence to a large extend. Only arbitrary expressions that, e.g., compute values, are inserted as they would appear in source code. Such expressions are often source-code compatible across different programming languages. The `return ... ;` expression in the exponential delay condition above, e.g., can be compiled both by a Java and a C++ compiler. In addition, low level routines such as this are often part of a (pre-compiled) component library and do not have to be specified by model developers.

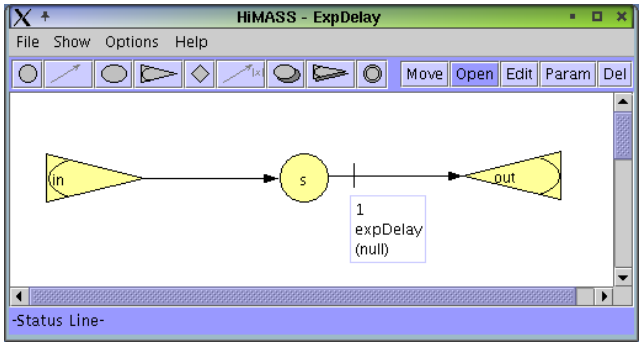

Figure 4: HiMASS Modeler GUI for Exponential Delay MCS

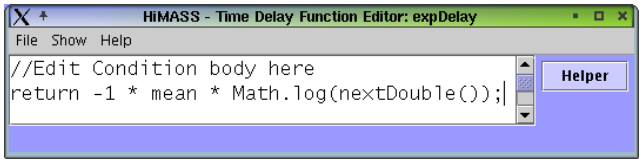

Figure 5: HiMASS Modeler GUI for Time Delay Condition

Table 3: Exponential Time Delay Edge Condition

Filename: ExpDelayCond.xml

```
<me type="time-condition" name="expDelay">
 <var type="double" name="mean"/>
 <code return="double">
 return -1 * mean * Math.log(nextDouble());
 </code>
</me>
```

5 VISUAL INTERACTIVE MODELING

HiMASS provides a powerful GUI for the VIM of MEs, as well as entire models. With HiMASS-x, the graphical information describing the layout of MEs is stored in an XML-based vector graphics format. (In HiMASS-j, GUI information was stored in a proprietary binary format, that was based on Java serialization and that was not guaranteed to work across different versions of Java.)

When GUI information is saved in XML, it can be interspersed with the structural information of the model. This is useful when executable models must display graphical information during or after the simulation run. As general tools for XML-based graphics formats exists, it is straightforward to extract graphical model information for documentation or advertising purposes. It is also possible to use other software packages to create or process HCFG models.

6 EXPERIMENTAL FRAME

HiMASS supports the Experimental Frame (EF) concept (Zeigler 1976) for the specification of a model's experimental conditions (see Daum and Sargent 2001.) An EF can specify values such as the mean rate of arrivals or the seeds for the pseudorandom number generators. EFs are usually implemented as one or several sets of *(key, value)* pairs, where *key*s are the unique identifiers of the (model and other) attributes that can be specified through the EF and *value*s are the numerical, string, or other values that are assigned to these attributes upon EF initialization. EFs are usually stored in files independent from the model which are loaded by the simulation software during model initialization. The values provided by an EF are assigned to the appropriate model attributes either as part of model initialization or before each of the attributes is accessed for the first time during model execution.

Prior to utilizing XML, HiMASS-j used a proprietary, binary data format for EFs. That practice was unsatisfactory in two regards. The EF file format was based on serialized Java objects and was not guaranteed to be usable across different versions of the Java platform. Due to the binary nature of the old EF file format, it was impossible to manually inspect or manipulate EF files.

HiMASS-x uses XML for the specification of EFs. Although a comprehensive GUI is provided that allows for the interactive specification of values for the EF, model developers can use their favorite XML or text editor to quickly change a value in the EF between simulation experiments.

Figure 6 shows how the HiMASS Modeler GUI is used to specify the EF for the trivial model used in this paper and Table 4 shows the XML representation of this EF.

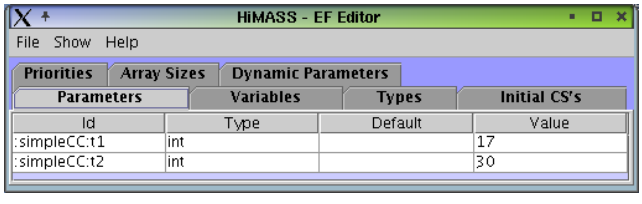

Figure 6: HiMASS EF Dditor

Table 4: EF Specification in XML

Filename: SimpleEF3.xml

```
<ef name="simple" id="3">
 <cc name="simpleCC">
  <param name="t1" type="int" value="17"/>
  <param name="t2" type="int" value="30"/>
 </cc>
</ef>
```

The structure of an EF file closely resembles the structure of a model definition file and it is thus intuitive to read and manipulate. EF files can be processed, e.g., to generate documentation, in the same way as model definition files.

7 MODEL EXECUTION

The HiMASS-x Engine is used to execute HCFG simulation models. It utilizes XML to set up and run simulation experiments. Table 5 shows the XML file for a simulation experiment.

The file specifies the model to be "simpleModel". Since the `<model>` element has no `url` attribute, the model itself will be loaded locally.

Table 5: XML Simulation Experiment File

Filename: SimpleExec.xml

```
<himass>
 <source type="me" name="simple"
   url="http://himass.com/1.0/simple.xml"/>
 <model name="simpleModel">
   <sim algo="seq" lang="java"/>
   <root type="SimpleCC" source="simple"/>
   <ef name="simple" id="1"/>
   <ef name="simple" id="2"/>
   <ef name="simple" id="3"/>
 </model>
</himass>
```

7.1 Remote ME Libraries

ME type specifications can be loaded from a remote ME library, if the model file contains no type specifications for referenced MEs. The location of the ME library is specified by the `<source>` element. It is possible to specify several ME libraries or none at all, if all ME types are fully specified in the model.

7.2 Simulation Algorithm and Experiments

The `<sim>` element selects the HCFG simulation algorithm and the execution language. Using XML makes it easy to "plug in" different simulation algorithms for the same HCFG model, without having to recompile. Specifying the execution langue is useful when ME libraries for several languages (e.g., Java and C++) are available. The `<root>` element specifies which of the available CCs should be the top-level CC, i.e., the root of the model tree. The type definition of this root CC recursively references all MEs used in the model and how they are interrelated, so that the simulation engine can dynamically build a runtime representation of the model. The `source` attribute specifies that the CC should be loaded from the ME library specified above.

For this experiment, the model will be executed three times with three different EFs as specified by the different `<ef>` elements. Instead of listing individual EFs, it is also possible to give a range of EF files that should be used. It is also possible to load one, several, or all EF files from a remote source.

8 CONCLUSIONS

We presented a new approach to modeling, managing, and executing HCFG models. The consistent use of XML reconciles formerly disparate data formats. The use of XML is beneficial in several ways.

Since XML is an open, standardized format for which several free, high-quality software packages exist, it becomes feasible to access HCFG models, ME repositories, and EF files independently from HiMASS-x. Models and other components of an HCFG system can be easily shared with other users and systems. As XML is human-readable, model data cannot get locked into any particular software package. Modelers can be confident that they will be able to access their models and data in the future.

Because XML was designed to be run over the Internet and standard XML processing software was developed with that goal in mind, HiMASS-x resources can be accessed remotely with very little effort. Distributed capability leverages the full potential of ME libraries and is crucial for the implementation of distributed simulation algorithms.

Using XML allows the specification of a standard format for HCFG models that is independent from any particular software implementation. Leveraging freely available, ready-to-use XML software finally can cut down significantly on the software development effort.

The complete files used in this paper and other relevant resources are available online at `<http://himass.com/wsc02>`.

REFERENCES

Bray T., J. Paoli, C.M. Sperberg-McQueen, and E. Maler, editors. 2000. *Extensible Markup Language (XML) 1.0* Second Edition. World Wide Web Consortium. Available online via `<http://www.w3.org/TR/2000/REC-xml-20001006>` [accessed March 31, 2002].

Clark J., editor. 1999. *XSL Transformations (XSLT) 1.0* World Wide Web Consortium. Available online via `<http://www.w3.org/TR/xslt>` [accessed March 31, 2002].

Cota, B. and R. Sargent. 1990. Simulation Algorithms for Control Flow Graphs. CASE Center Technical Report 9023. Syracuse University, Syracuse, New York.

Cota, B. and R. Sargent. 1992. A Modification of the Process Interaction World View. *ACM Trans. Model. Comput. Simul.*, 2, 2, 109–129.

Daum, T. 1998. *An Investigation into Specifying HCFG Models Using Visual Interactive Modeling.* Graduate Thesis. Otto von Guericke University, Magdeburg, Germany.

Daum, T. and R. Sargent. 1997. A Java Based System for Specifying Hierarchical Control Flow Graph Models. In: S. Andradottir, K.J. Healy, D.H. Withers, and B.L. Nelson, eds., *Proc. of the 1997 Winter Simulation Conference*, 150–157, IEEE, Piscataway, New Jersey.

Daum, T. and R. Sargent. 1999. Scaling, Hierarchical Modeling, and Reuse in an Object-Oriented Modeling and Simulation System. In: P.A. Farrington, H. Black Nembhard, D.T. Sturrock, and G.W. Evans, eds., *Proc. of the 1999 Winter Simulation Conference*, 1470-1477, IEEE, Piscataway, New Jersey.

Daum, T. and R. Sargent. 2001. Experimental Frames in a Modern Modeling and Simulation System. *IIE Trans.*, 33, 3, 181–192.

Eckel, B. 1998. *Thinking in Java.* Upper Saddle River, New Jersey: Prentice-Hall.

Fritz, D. and R. Sargent. 1995. An Overview of Hierarchical Control Flow Graph Models. In: C. Alexopoulos, K. Kang, W. Lilegdon, and D. Goldsman, eds., *Proc. of the 1995 Winter Simulation Conference*, 1347–1355, IEEE, Piscataway, New Jersey.

Sargent, R. 1997. Modeling Queueing Systems Using Hierarchical Control Flow Graph Models. *Mathematics and Computers in Simulation*, 44, 237-249.

Sargent, R. and T. Daum. 1998. Visual Interactive Modeling in a Java-based Hierarchical Modeling and Simulation System. In: P. Lorenz and B. Preim, eds., *Proc. of Simulation und Visualisierung '98*, 1–17, Society for Computer Simulation International, Ghent, Belguim.

Zeigler, B. 1976. *Theory of Modelling and Simulation.* New York, N.Y.: Wiley.

AUTHOR BIOGRAPHIES

THORSTEN S. DAUM is the principal developer of HiMASS. He was a director and co-founder at Xmlify, a Silicon Valley corporation specializing in XML-centered data conversion technology. He holds a graduate degree in computer science with a focus on simulation from Otto von Guericke University in Magdeburg. His interests include discrete event simulation, Object Oriented methodology, and XML. He was a visiting researcher with the Simulation Research Group and CASE Center at Syracuse University. His e-mail address is <tdaum@himass.com>.

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 and has been awarded the TIMS (now INFORMS) College on Simulation Distinguished Service Award for longstanding exceptional service to the simulation community. His current research interests include the methodology areas of both modeling and discrete event simulation, model validation, and performance evaluation. Professor Sargent has published extensively and is listed in *Who's Who in America*. His e-mail address is <rsargent@syr.edu> and his web page is <www.cis.syr.edu/srg/rsargent>.