Proceedings of the 1997 Winter Simulation Conference
ed. S. Andradóttir, K. J. Healy, D. H. Withers, and B. L. Nelson

A JAVA BASED SYSTEM FOR SPECIFYING HIERARCHICAL CONTROL FLOW GRAPH MODELS

Thorsten Daum
Robert G. Sargent

Simulation Research Group
439 Link Hall
Syracuse University
Syracuse, New York 13244, U.S.A.

ABSTRACT

The portion of the Hierarchical Modeling And Simulation System-Java (HiMASS-j) used for specifying Hierarchical Control Flow Graph (HCFG) Models is described. The specification of HCFG Models in HiMASS-j is by visual interactive modeling through the use of graphical user interfaces and dialog boxes. HCFG Models are specified using two complementary hierarchical specification structures, one to specify the components that comprise a model and how these components are interconnected, and the other to specify the behaviors of the individual atomic components.

1 INTRODUCTION

This paper describes the portion of the Hierarchical Modeling and Simulation System-Java (HiMASS-j) used for specifying Hierarchical Control Flow Graph (HCFG) Models. A brief overview of the HCFG Model paradigm is given in this introduction; however, a deeper understanding of the HCFG Model paradigm will provide more and better insights into the work described here. A short description of the HCFG Model paradigm can be found in "An Overview of Hierarchical Control Flow Graph Models" (Fritz and Sargent 1995) and a more detailed description can be obtain in an expanded version of this Overview paper located at `http://www.cat.syr.edu/~srg/`.

An HCFG Model can conceptually be viewed as consisting of a set of independent, encapsulated, concurrently operating (atomic) components where each (atomic) component has its own thread of control and the (atomic) components interact with each other solely via message passing. Two kinds of encapsulated components are used to specify an HCFG Model: an atomic component (AC), which is the most elementary type of component, and a coupled component (CC), which couples together ACs and/or other CCs. The CCs provide hierarchical relationships among components in HCFG Models. Each CC is specified by the use of a CC Specification (CCS). A CCS is a directed graph whose nodes represent components and whose directed edges represent channels between components. Channels carry messages between the input and output ports of components. Each channel connects exactly one output port to one input port and each port is connected to only one channel. The hierarchical relationships of all components and their interconnections are contained in the Hierarchical Interconnection Graph (HIG). A HIG tree shows the hierarchical relationships among the components but not their interconnections. The root (or top) node of the HIG tree is an instance of a CC and is usually called root with type name Root.

Each AC contains a set of (local) variables including a (local) simulation clock, a set of input ports, a set of output ports, a set of parameters, and an HCFG, which describes the behavior of that AC. The basic building block of an HCFG is the Macro Control State (MCS). A MCS is a state-based behavior specification structure and is an augmented directed graph whose nodes are control states (CSs) and/or other MCSs and whose directed edges give the set of possible control state transitions. A CS is a formalization of the "process reactivation point" (Cota and Sargent 1992). Edges originating from CSs have three attributes: a condition, a priority, and an event. The condition specifies when an edge can become a candidate for traversal, the priority is used to break ties when more then one edge is a candidate for traversal at the same simulation time, and the event specifies a state transition for the AC which is executed whenever that edge is traversed during simulation execution. Three different kinds of edges, which depend on the condition attribute, leaving CSs are used: TimeEdges, BoolEdges, and PortEdges. To specify an edge, a priority and event is specified along with an edge type and either a time function, a boolean function, or an input port, depending on the type of edge. Edges originating from MCSs do have not attributes. Each HCFG has a point of control (POC), which moves from CS to CS to indicate the state that the HCFG (i.e., the AC) is in. An HCFG is a hierarchical organized set of MCSs (which contain all of their interconnections). An HCFG tree shows the hierarchical relationships among the MCSs but not their interconnections.

The HCFG Model paradigm supports and HiMASS-j implements the use of types and instances of model elements. The model elements are the ACs, CCs, MCSs, event routines, time delay functions, and boolean functions. The types of model elements are specified and instances of the types are used to specify HCFG Models. Libraries of the types can be established and this provides for reuse. Having libraries of types allows the "layered" approach to modeling to be used where ACs and CCs can be used for model specification if the appropriate ones are available, and if not, then the needed components can be build.

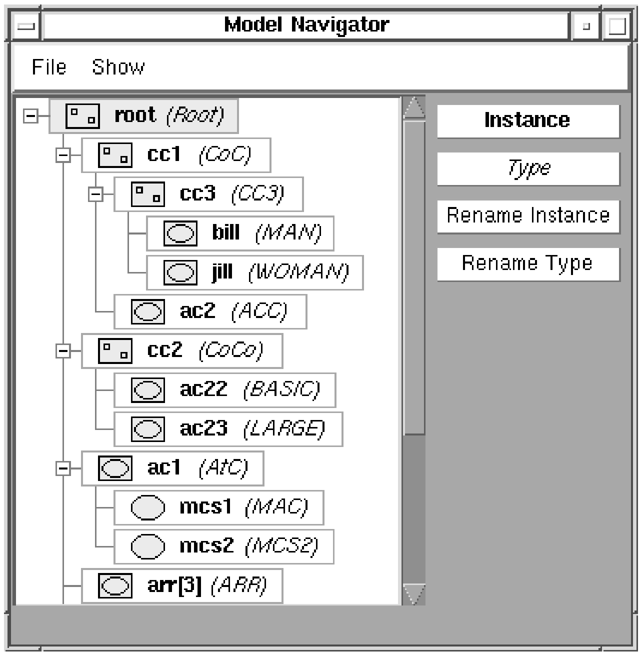
The specification of an HCFG Model requires one HIG and one HCFG for each type of AC in the model. HCFG Model specification in HiMASS-j is via visual interactive modeling (VIM) through the use of graphical user interfaces (GUIs) and dialog boxes. The HCFG model paradigm supports and HiMASS-j implements the use of experimental frames (EFs) (Zeigler 1984). The use of EFs allows the values of the parameters of the model elements, the model's initial conditions, etc. to be specified separately from the HCFG Model specification.

The HCFG model paradigm favors an "active resource" view of modeling over an "active transaction" view. Modeling from an active resource view means that the system is modeled from the point of view of the system's resources by describing the behaviors and interactions of those resources. We use the active resource view for modeling in this paper.

The remainder of this paper is organized as follows: Section 2 discusses how one specifies HCFG Models, Section 3 gives a brief description of the Java software that is used for specifying HCFG Models in HiMASS-j, and Section 4 contains the summary.

2 SPECIFYING HCFG MODELS

In this section we discuss how HiMASS-j can be used to specify HCFG Models. HCFG Models are specified in HiMASS-j via VIM using GUIs and dialog boxes. Our emphasis will be on specifying HCFG Models top down and from scratch, i.e., not using libraries of model elements. A simple way to specify and view HCFG Models is to use the Model Navigator dialog box. The Model Navigator contains the HCFG Model tree and is used to navigate, i.e. to move, among the model's CCs, ACs, and MCSs. The model tree consists of the HIG tree and the HCFG tree of each AC in the model. (The model tree does not show the top MCS of an HCFG since it is the internal view of an AC.) The Model Navigator containing a sample model tree is shown in Figure 1. Each node of the model tree has a symbol (defined below) to indicate the kind of model element, the instance name of the model element, and the type name of the model element given in parentheses. For example, the model tree in Figure 1 has a coupled component with instance name cc1 of type CoC. At each level of hierarchy of the model tree there is a 'switch' which when clicked either displays or discontinues the display of the lower levels of the model tree. The top node (named root unless renamed) of the model tree is always displayed and this is the HIG root (top) node.



root *(Root)*
cc1 *(CoC)*
cc3 *(CC3)*
bill *(MAN)*
jill *(WOMAN)*
ac2 *(ACC)*
cc2 *(CoCo)*
ac22 *(BASIC)*
ac23 *(LARGE)*
ac1 *(AtC)*
mcs1 *(MAC)*
mcs2 *(MCS2)*
arr[3] *(ARR)*

Figure 1: Model Navigator Dialog Box

The Model Navigator has four buttons on its right; Instance opens a GUI window containing the instance of the selected model element in the model tree, Type opens a GUI window containing the type of the selected model element, and the other two buttons open dialog boxes to rename either a selected model element instance or type name. HiMASS-j has two different GUI windows: the CCS GUI window for working with the CCs of a HIG and the MCS GUI window for working with the MCSs of the HCFGs. The CCS GUI window opens if an instance of a component (AC or CC) or if a type of CC is selected; otherwise a MCS GUI window opens.

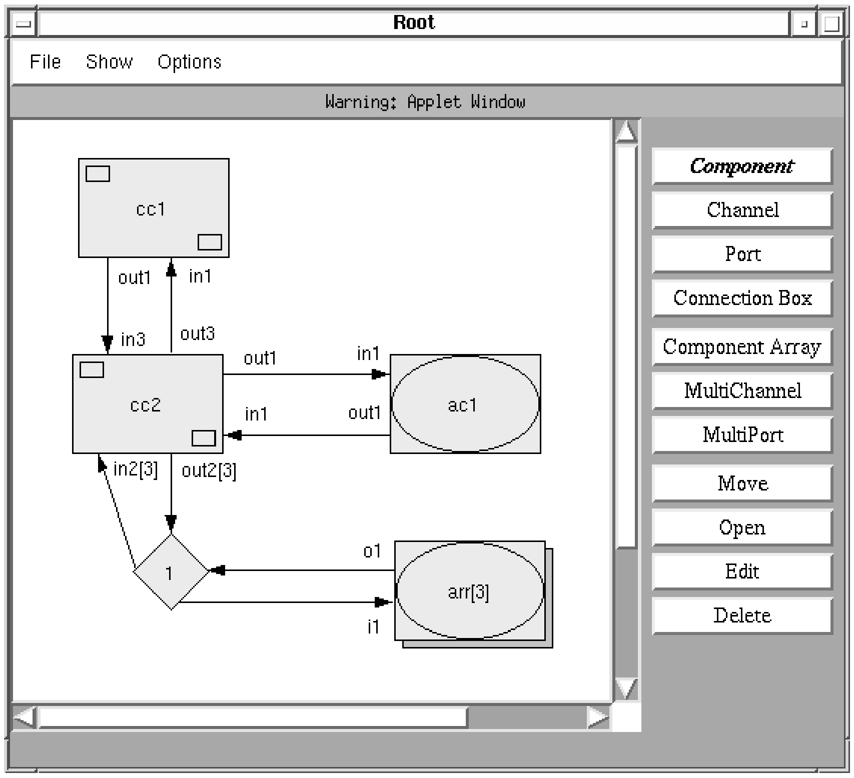


Figure 2: CCS Graphical User Interface Window

2.1 HIG Specification

The first step in specifying an HCFG Model from top down is to open the Model Navigator. The model tree in the Model Navigator will contain one node which is a CC with the instance name ‘root’ of type ‘Root’. This is the top (root) node of the model tree (and HIG tree). A modeler opens a CCS GUI window for Root by selecting the Type button while the root node is selected (highlighted). Figure 2 shows the CCS GUI window with a sample Root CC in it. (The canvas area would be blank when Root is initially opened.) Note that the type name of the CC is given at the top of the window and that there are several tool buttons on the right side of this GUI. These tools are used to either specify a CC (which contains ACs and/or other CCs), modify a CC, or change the layout of the CC on the canvas. A user selects a tool by clicking on the appropriate button, and then uses the tool in the canvas area. To create a new component, a modeler first selects the Component button and then clicks on the canvas at the desired location for the new component. A Component dialog box will open for the modeler to enter the component instance name and type name, and to select the kind of component. (A Component dialog box is similar to the Array dialog box shown in Figure 3.) In Figure 2 there are two CCs called cc1 and cc2 and an AC called ac1. Note the symbols used for these two different kinds of components and that instance names are given on the components. As new components are specified in a CC, they are automatically added to the model tree in the Model Navigator. Note how the components contained in Root CC are in the model tree in Figure 1.

In Figure 2 there are, e.g., two channels between ac1 and cc2. Channels are specified by using the Channel tool. A new channel between components is specified by first selecting the Channel tool and then clicking inside the component where the channel originates. A Port dialog box will open to enter the name of that component’s new output port for connection to the new channel. Next, click inside the component where this new channel terminates. A Port dialog box will open to enter the name of that component’s new input port for connection to the new channel. The new channel and the names of the new ports automatically appear.

HiMASS-j provides scaling for components, channels, and ports. A component array consists of a homogeneous array of components. (A component is equivalent to a component array of size one.) A typical element of the array is specified and shown. After the Component Array tool is selected and the mouse clicked at the desired location for the component array on the canvas, the Array dialog box shown in Figure 3 opens. A modeler selects the kind of component that the array has and enters the array's instance name, type name, and either a numerical value or a variable whose value is specified in the EF for the array size in the dialog box. In Figure 2 there is an array containing 3 ACs called arr. Note that the array symbol indicates whether the components are ACs or CCs and that the size of the array is given with the instance name. As new arrays are specified, they are automatically added to the model tree. Multichannels are arrays (or bundles) of channels that have their size specified by a modeler and are created similarly to channels. The major differences are that multichannels are connected to multiports and that their size must be specified in the MultiPort dialog box.

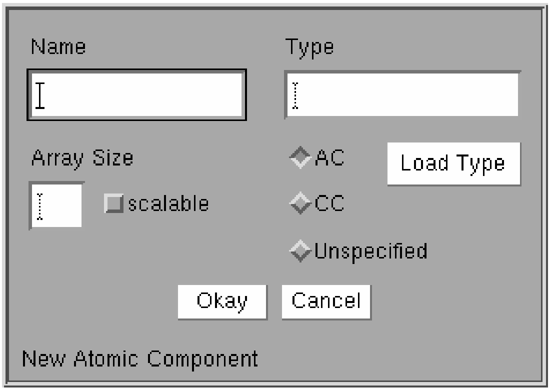


Figure 3: Array Dialog Box

Because model elements in a HIG can be of different sizes, not all channels between model elements can be clearly represented using a purely graphical notation. A "connection box" that is represented by a diamond can be used to connect channels and multichannels of different sizes to model elements. A Connection Box tool is provided to specify connection boxes on the canvas similar to how components are specified. The major difference is that a connection box is automatically given a numerical name and thus no dialog box is needed to specify a name. A Connection Box dialog box (obtained by using the Edit button) is used to make the connections of the channels entering and leaving a connection box. The ports that the channels entering and leaving the connection box are connected to are automatically given by HiMASS-j, and it is straightforward and simple to make the appropriate connections. In Figure 2 note the connection box named 1, the multichannels of size 3 between the multiports of CC cc2 and the connection box, and the channels between the (three) elements of the array arr and the connection box.

The purpose of the Port and MultiPort tools is to specify ports and multiports when specifying model elements bottom up instead of top down. The Move, Delete, and Edit tools provide the common editing capabilities. The Open tool opens up a GUI window of the model element's type to either view or specify that model element. If the model element is a CC, then a CCS window is opened; otherwise a MCS window is opened.

If a CCS window is opened for a CC other then the top CC, then the ports for that CC are shown. (The top CC has no ports.) For example, the canvas area of a CCS window for a sample CC of CoC (which is the type for the CC instance cc1 of Root) is shown in Figure 4. If no components had yet been specified for CoC, then only the input port in1 and output port out1 would be shown. In top down modeling these ports (with their names) would have been specified in an instance of the CC in the CCS window; e.g., the ports of CoC were specified in the instance cc1 in the CC Root. Note that the components in CoC are in the model tree in Figure 1.

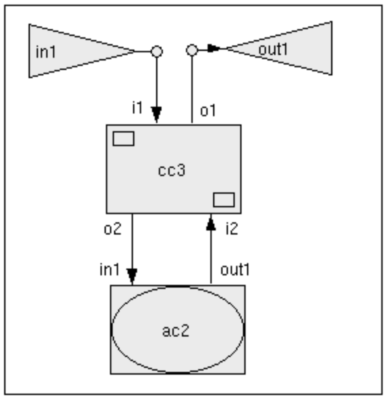


Figure 4: Internal View of a CC

2.2 HCFG Specifications

The MCSs in HCFGs are specified in HiMASS-j by using a MCS GUI window and dialog boxes. To specify an HCFG top down, the modeler first opens a MCS window for the top MCS of an HCFG for an AC. This is accomplished by either (a) using the Model Navigator to select the appropriate AC and then clicking on the Type button, or (b) using the Open tool on the appropriate AC in a CCS GUI window. Figure 5 shows the MCS GUI window containing a sample top MCS for the AC AtC.

One can readily see that there are a set of tool buttons on the right side of the MCS window. The Control State and MCS tools are used to specify CSs

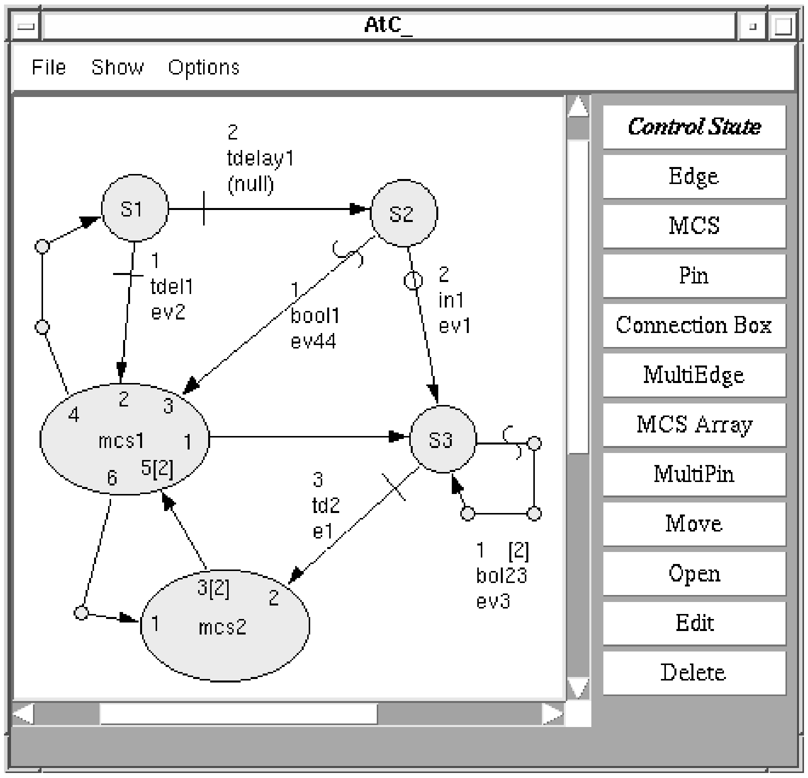


Figure 5: MCS Graphical User Interface Window

and MCSs. In Figure 5 there are 3 CSs (S1, S2, and S3) and two MCSs (mcs1 and mcs2). These tools operate similar to the Component tool in the CCS window. New MCSs are automatically put into the model tree as they are specified. Note the MCSs mcs1 and mcs2 in the model tree in Figure 1. (The MSC AtC is not shown in Figure 1 since this is the top MCS of the AC AtC and thus the internal view of this AC.)

The Edge tool operates similar to the Channel tool. If the (new) edge being specified starts at a MCS, a Pin dialog box opens when the MCS is clicked asking for the name of the (new) pin that the (new) edge will be connected to. If this edge terminates on a CS, then the edge appears when that CS is clicked, and if the edge terminates on a MCS, then a Pin dialog box opens when that MCS is clicked to specify the pin name prior to the edge appearing. (See the edges leaving MCS mcs1 and the names of the pins of MCS mcs1 in Figure 5.) If the edge specified starts at a CS, then an Edge dialog box opens. The Edge dialog box, shown in Figure 6, provides for several entries and selections. A modeler first selects the edge condition for the type of edge being specified. If a TimeEdge or a BoolEdge is selected, then the name of the time delay function or boolean function is entered into the top input box. If a PortEdge is selected, then the associated port is selected from the list of input ports given in the second input box (or added if building from bottom up). The name of the event routine is given in the third input box and the edge priority is given in the bottom input box. (A TrueEdge is a BoolEdge that is always true and a null Event is an event that does nothing and these can be specified by clicking on the appropriate button.) See the edges leaving the CSs in Figure 5. Note that the edge attributes are given in an attribute box located near each edge. The top entry in each edge attribute box gives the edge priority, the second entry gives either the time delay function, the boolean function, or the input port name depending on the kind of edge, and the last entry gives the name of the event routine. In Figure 5, the two edges leaving CS S1 are TimeEdges, the edge going from CS S2 to mcs1 is a BoolEdge, and the edge going from CS S2 to CS S3 is a PortEdge.

To specify a time delay function, a boolean function, or an event routine, one first selects the Edit tool and then selects the desired entry in an edge attribute box. A dialog box for that entity will then open. See Figure 7 for the Event dialog box, which contains a sample event for event ev1 of AtC (Figure 5). The event ev1, which is for a PortEdge, receives the message waiting at input port in1, stores the message

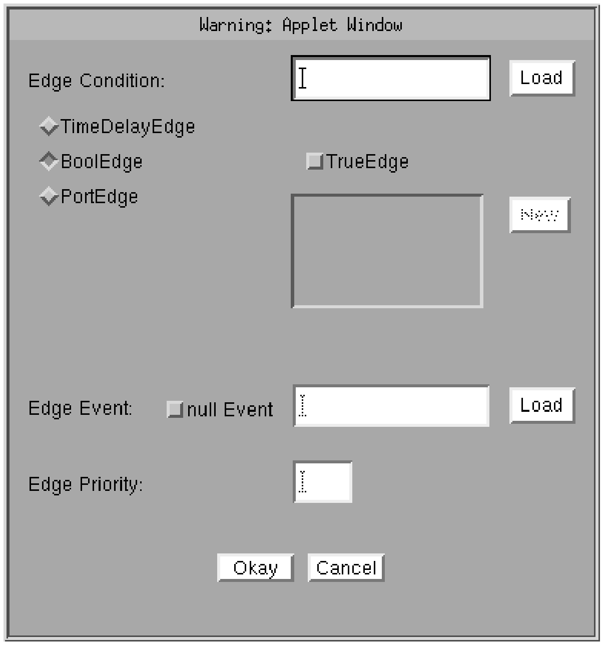


Figure 6: Edge Dialog Box

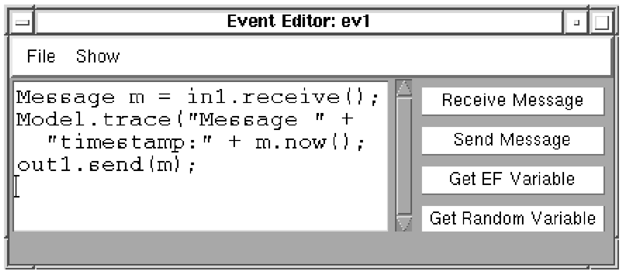
timestamp in the trace file, and sends the message to output port out1. Helper functions are provided to assist in the specification of events and these are the buttons on the right side of the dialog box. (Examples of functions and event routines are given in Daum (1997), which is a companion paper to this paper.)

HiMASS-j provides scaling for edges, MCSs, and Pins. A multiedge is an indexed array of edges. There are two multiedges in Figure 5. One connects between multipins on MCS mcs2 and MCS mcs1, and is size 2. The other multiedge is from CS S3 back to itself, and is also size 2. Note that the size of multiedges leaving CSs are given in square brackets in the attribute box. (The edge attributes of multiedges leaving CSs are also indexed.) A MCS array is a homogeneous array of MSCs similar to a component array. Connection boxes are also available for connecting multiedges.

The purpose of the Pin and MultiPin tools is to specify pins and multipins when specifying model elements bottom up instead of top down. The Move, Delete, and Edit tools provide the common editing capabilities. The Open tool opens up a GUI window of a MCS.

2.3- Experimental Frame

HiMASS-j currently provides EF support for specifying the initial CSs of the ACs, the initial values of the instance and type variables, the sizes of the arrays of model elements that use variable scaling, and the types of model elements at runtime which allows the



```
Message m = in1.receive();
Model.trace("Message " +
  "timestamp:" + m.now();
out1.send(m);
```

Figure 7: Event Dialog Box

model structure to be changed. These specifications are aided by dialog boxes and require no programming. A modeler can specify default values for all of these EF parameters.

3- HiMASS-j SOFTWARE

HiMASS-j is an object oriented software system written entirely in Java (Arnold and Gosling 1996). It has been developed and tested on both SUN workstations running Solaris and Pentium based personal computers running Linux and Windows 95. Due to the platform independent nature of Java, HiMASS-j can run without the need to recompile on a wide variety of machines, including several Unix architectures and Apple Macintosh computers. Hi-MASS-j makes extensive use of object-oriented programming features supported by Java such as encapsulation, inheritance, polymorphism, and overloading, and specific Java features such as serialization and platform independence.

HI-MASS-j was implemented using the SUN Java Development Kit (JDK) version 1.1.2, available from `http://java.sun.com`, and the Java Generic Library (JGL). JGL is currently free and available at `http://www.objectspace.com/jgl`.

A user of HI-MASS-j needs to be familiar with modeling using the HCFG Model paradigm and a basic understanding of the Java syntax. HiMASS-j can be run with any JDK 1.1.2 or newer compatible Java enabled web browser that connects to our site at `http://www.cat.syr.edu/~srg/`. Alternatively, HIMASS-j can be downloaded from the above site and run with any JDK 1.1.2 or newer compatible Java development system that includes a Java compiler, such as the SUN JDK.

3.1- Design

HI-MASS-j includes over 800 kilobytes of sources in over 160 Java classes in three packages (Arnold and Gosling 1996). Package srg.himass.vim contains the portion of HiMASS-j used for specifying a HCFG

Model via VIM, package srg.himass.sim contains the simulator, and package srg.util contains various utility classes.

The design of the VIM part of HiMASS-j follows (somewhat freely) the Model-Viewer-Controller (MVC) paradigm (Gamma et al. 1995). A Model is the logical representation of an object. In HiMASS-j, a Model contains the information that will be needed by the simulator. A Viewer is the visual representation of a Model. Viewers in HiMASS-j are, e.g., the CCS canvas and the ellipse representing a MCS instance. Controllers register changes in a Viewer, such as a mouse click, and modify the corresponding Model accordingly. Java's new Delegation Event Model that was introduced with JDK 1.1 supports this approach.

HiMASS-j makes an important distinction between instances and types of model elements (e.g. CCs, ACs, and MCSs) in the HCFG Model paradigm as discussed in Section 2. In this section we will refer to these as Element Type and Element Instance. Recall that every Element Instance has exactly one type that describes it and that an Element Type can have many Element Instances (which provides for reuse).

The HiMASS-j design combines the MVC paradigm and the Element Type/Instance distinction. In using VIM, a modeler interacts with Element Instance Viewers and Element Type Viewers. Clicking the mouse, e.g., in a CC, will cause the Component Instance Viewer (CompIV) to generate an event that will be delivered to the Component Instance Controller (CompIC). The CompIC takes the appropriate action (e.g., add a port) and notifies the Component Instance Model (CompIM) which in turn changes its state, changes the CompIV to visually represent the change, and notifies the Component Type Model (CompTM). The CompTM then changes state (e.g., adds an external port), visualizes the change in the Component Type Viewer (CompTV) which the modeler sees as the CC canvas, and notifies its other Instances to update their Viewers as well. Figure 8 shows part of the relationship among Component Instance and Type Viewers, Controllers, and Models.

Element Type Models and the model files used by the simulator are closely related but not identical. An Element Type Model contains all the information that is necessary to completely specify this element. However, this information can not be used directly by the simulator. The method body of an event routine, e.g., is saved as a String by an Event Model Type. To be understood by the simulator as actual code, it has to be written to a file, compiled into binary code, and then loaded.

When a modeler saves a model, all Element Models are written into files (serialized) along with their

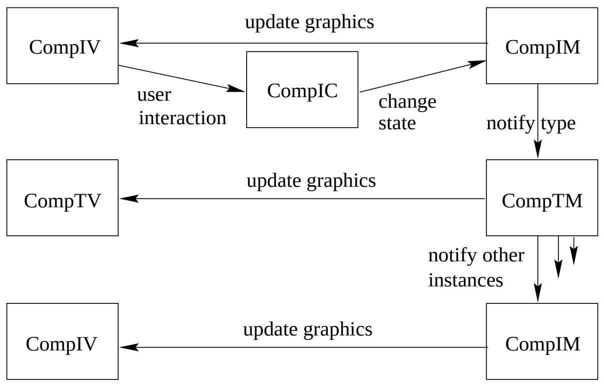


Figure 8: Component Instance and Type Models, Viewers, and Controllers

Viewers. When the model is reloaded, new objects of the various Model Type and Instance classes will be created and initialized with the saved fields that specify the different elements. When the executable model is generated, HiMASS-j generates a new Java class for each Element Model Type of the model: Strings describing event routines, time and boolean conditions become actual functions that can be executed, Strings representing Type names become part of the new class name, etc.

The produced Java class files are thus independent of the VIM part of HiMASS and can be executed by any software that implements a simulator for HCFG Models and that accepts condition and event routines specified in Java syntax.

3.2 Edge Conditions and Event Routines

A modeler can specify the HIG and HCFGs of a model by VIM. The HiMASS-j system uses the information provided to the GUI to automatically generate Java classes that the simulator uses to initialize and execute the model. No knowledge of a programming language is required for this. However, there is no way to use VIM to specify the different edge conditions and event routines used in models. Method bodies, i.e. the code of time and boolean functions and event routines, must be specified by text. Since the simulator is implemented in Java, it makes sense to specify the method bodies of condition functions and event routines in Java as well and that is what is implemented in HiMASS-j.

The specification of condition functions and event routines is aided by helper functions, such as for sending and receiving messages, declaring and retrieving variables, and creating pseudo random numbers. Since helper functions print their outputs in the Event and Condition dialog box windows, the modeler can modify this code as required. Figure 7 shows a simple event routine that was developed using the helper functions. Alternatively, a modeler can write the code from scratch; thus maximum flexibility is maintained.

4 SUMMARY

A brief description of most of the basic VIM capabilities of HiMASS-j for specifying HCFG Models was presented. Some of features not discussed include specifying messages, the model initial conditions, and data collection, and specifics on specifying event routines, time delay functions, and boolean functions. A companion paper by Daum (1997) illustrates the use of HiMASS-j in modeling a specific system, including the specification of event routines and edge condition functions. Some other papers on modeling using the HCFG Model paradigm are Farr et al. (1995) and Sargent (1997).

A brief overview of the design of HiMASS-j was presented. HiMASS-j differs from an earlier prototype called HI-MASS (Fritz, Sargent, and Daum 1995) for specifying and simulating HCFG Models. In HiMASS-j both the HIG and the HCFGS are specified via VIM, there is a clear distinction between types and instances of model elements, the model tree is displayed, there is a model navigator, the use of parameters is permitted in the model elements, and reuse of model elements is easy. In HI-MASS the HIG can be specified via VIM (with less capabilities than in HiMASS-j), the HCFGs are specified via text, no model tree is displayed, parameters are only permitted at the HCFG level and within the HCFG (e.g., in event routines), and reuse of model components is limited. The computer language of HI-MASS is C++ which differs from Java used in HiMASS-j.

Much has been said about the sometimes substantial speed sacrifice one has to accept when executing programs in Java instead of C++. However, we believe that Java's disadvantage in this regard will diminish in the future. Next generation just-in-time compilers have a potential for significant speed increases of Java applications. In addition, efforts are under way to build a Java front end to the GNU compiler (Bothner 1996) that would allow ahead-of-time compilation of Java applications into native code providing for performance equal to C++ applications.

REFERENCES

Arnold, K. and J. Gosling. 1996 *The Java Programming Language.* Reading, Mass.: Addison Wesley.

Bothner, P. 1996. A Gcc-based Java Implementation. Available at `http://www.cygnus.com/~bothner`.

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. 1997. An HCFG Model of a traffic intersection specified using HiMASS-j. In: S. Andradottir, K. Healy, D. Withers, and B. Nelson, eds., *Proc. of the 1997 Winter Simulation Conference.*

Farr, S., A. Sisti, D. Fritz, and R. Sargent. 1995. A simulation model of a surveillance radar data processing system using HI-MASS. In: C. Alexopoulos, K. Kang, W. Lilegdon, and D. Goldsman, eds., *Proc. of the 1995 Winter Simulation Conference*, 1364–1370.

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.

Fritz, D., R. Sargent, and T. Daum. 1995. An overview of HI-MASS (Hierarchical Modeling and Simulation System). In: C. Alexopoulos, K. Kang, W. Lilegdon, and D. Goldsman, eds., *Proc. of the 1995 Winter Simulation Conference*, 1356–1363.

Gamma, E., R. Helm, R. Johnson, and J. Vlissides. 1995. *Design patterns: Elements of Reusable Object-Oriented Software.* Reading, Mass.: Addison-Wesley.

Sargent, R. 1997. Modeling queueing systems using hierarchical control flow graph models. Forthcoming in *Mathematics and Computers in Simulation.*

Zeigler, B. 1984. *Multifacetted Modelling and Discrete Event Simulation.* London: Academic Press.

AUTHOR BIOGRAPHIES

THORSTEN DAUM is a graduate student at Otto von Guericke University in Magdeburg who is working towards a degree in simulation and computer graphics. His interests include the development of visual interactive modeling systems for simulation and Java software. He is a visiting researcher with the Simulation Research Group and CASE Center at Syracuse University.

ROBERT G. SARGENT is a Professor in the L. C. Smith College of Engineering and Computer Science at 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 research interests include the methodology areas of modeling and discrete event simulation, model validation, and performance evaluation. Professor Sargent is listed in *Who's Who in America.*