

Collaborative DEVS Modeler Users-Guide

*AI & Simulation Research Group
Electrical & Computer Engineering Department
The University of Arizona*

James Nutaro
Hessam S. Sarjoughian
Bernard P. Zeigler

August 16, 1999
Version 1.0

Copyright 1999
The University of Arizona

TABLE OF CONTENTS

1	WHAT IS THE COLLABORATIVE DEVS MODELER?	4
2	INSTALLATION	4
2.1	WINDOWS 95/NT.....	4
2.2	UNIX.....	4
3	NETWORK CONNECTIVITY	5
4	THE CDM SERVER	6
5	THE CDM CLIENT	6
5.1	SETTING UP CLIENT.....	6
5.2	CONNECTING TO A SERVER	7
5.3	SESSION-MANAGER.....	8
5.4	SYSTEM INFORMATION	8
5.5	SESSION.....	9
6	DEVS MODELER	9
6.1	MODEL CONSTRUCTION	11
6.2	AN EXAMPLE.....	12
7	MODEL NAMING CONVENTION/REPOSITORY	16
7.1	CLIENT FILES AND FILE NAMING CONVENTIONS	16
7.2	SERVER FILES AND NAMING CONVENTIONS	16
8	CDM KNOWN BUGS/LIMITATIONS	17
9	LISTING OF THE MOST COMMON ERROR MESSAGES	17
10	REFERENCES: RELATED/RELEVANT DOCUMENTATION	17

LIST OF FIGURES

Figure 1: CDM Starter Window5
Figure 2: Client’s Entry Point to a Server Window.....5
Figure 3: Server Window6
Figure 4: Install Directory Window.....7
Figure 5: Session Manager Window.....7
Figure 6: System Information9
Figure 7: Session Window.....11
Figure 8: `genr`, `proc`, `MP-coord`, and `transd` models13
Figure 9: `mproc` model.....14
Figure 10: The experimental frame15

1 What is the Collaborative DEVS Modeler?

The **Collaborative DEVS Modeler** (CDM) allows geographically dispersed modelers to collectively compose modular, hierarchical models (Grudin 1994; Zeigler, Sarjoughian et al. 1997; Sarjoughian, Nutaro et al. 1999). Dispersed subject matter experts and modeling professionals alike can collaborate synchronously and asynchronously via shared structured diagrams. CDM enforces the composition rules that are necessary to ensure that models constructed adhere to the general and powerful Discrete-event System Specification (DEVS) modeling paradigm (Sarjoughian and Zeigler 1997; Zeigler 1997; Zeigler, Kim et al. 1999)

This document focuses on using CDM. For a technical discussion of CDM and related topics refer to the list of references at the end of this document.

It is strongly recommended that you refer to Sections 8 and 9 before trying to use CDM. While the software is robust, it can generate some obscure warnings and error messages that are intended primarily for debugging and not for consumption by the user. An awareness of what sorts of messages can be expected could prevent needless frustration on your part, the user.

2 Installation

CDM installers are available for *Windows 95/NT* and common *Unix* operating systems. Installers¹ are available from the AIS website at <http://www.acims.arizona.edu/SOFTWARE>. The installers were created using InstallAnywhere™. **Note:** this Users-Guide (Version 1.0) is for CDM software Version 2.0.0.

You need to install Java 1.2.x before you can use CDM. The java runtime environment can be downloaded from <http://java.sun.com>.

2.1 Windows 95/NT

First, download the Windows installer from the AIS website, then double-click `cdm.exe`. The installer will lead you through the installation process. Once the installation is complete, you can run CDM by clicking on its icon in the Windows Start bar's Programs menu.

2.2 Unix

First, download the Unix installer from the AIS website. After downloading, type

```
sh cdm.bin
```

Notes: `cd` to the directory where you downloaded the installer before you run the installer.

To run CDM, `cd` to the installation directory and enter 'cdm'.

¹ CDM software is COPYRIGHT by the University of Arizona. Please (1) read the copyright file that comes with this software before the use of this software and (2) send an email to cdm@ece.arizona.edu containing your point of contact information.) We take no responsibility, explicit or implicit, in the "proper" functionality or proper use of the software.

3 Network Connectivity

The *CDM Starter* is the first window that you will see when you run CDM (see **Figure 1.**) Shown in the window are the port number that will be used by CDNS (Park 1998) and a model name that will be used if the single user option is selected. Check the appropriate boxes to choose a collaborative session server (i.e., select Start CDNS server), collaborative session client (i.e., select Start CDNS client), or a single user session (i.e., select the Single user). Both the port number and model name can be edited by selecting their text areas and typing in the preferred values. After making your choices, click *Go* to start CDM. Clicking the *Cancel* button will cause the application to exit immediately.

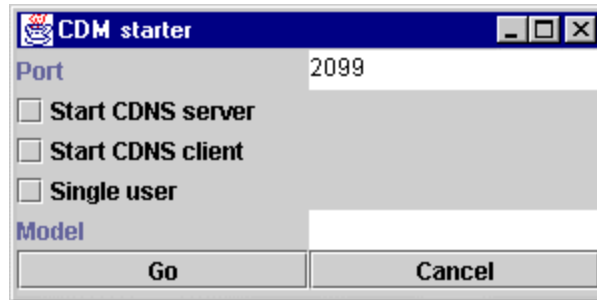


Figure 1: CDM Starter Window

The port number indicates the communication port ID that CDM (via CDNS) will use to talk to the outside world. Users who want to collaborate with one another need to use a common port ID. On most computers, port IDs (numbers) above 1024 are available for user applications. The default port is 2099, and this should be sufficient in most cases.

In order to join a collaborative modeling session, select the ‘Start CDNS client’ mode. **Figure 2** depicts the window for a user who has clicked *Enter* to connect across a network to the server called *ais7*. (Of course, both the client and the server could be on the same machine.)

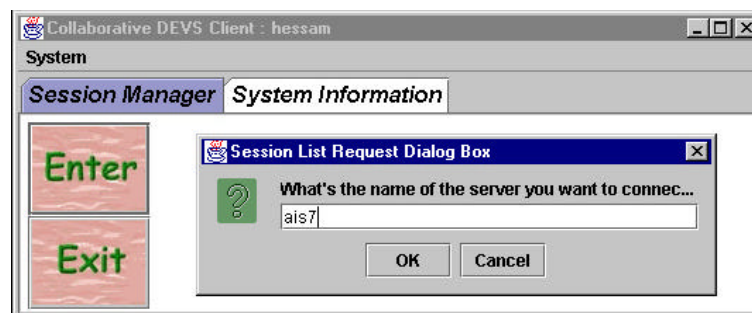


Figure 2: Client’s Entry Point to a Server Window

For each collaborative modeling session, a session server must be available to host it. A session server may host multiple modeling sessions. If you will be hosting a collaborative modeling session on your computer, the ‘Start CDNS server’ option should be selected. Upon a successful session server start up, the CDNS server window will be displayed (see **Figure 3.**) Clients who want to connect to the server will need to know its name or IP number and the port the server is using.



Figure 3: Server Window

The 'Single user' mode starts up CDM without support for collaboration. The user is able to work off-line, with the model being saved to a local hard drive. These models can later be incorporated into a collaborative session model.

4 The CDM Server

The CDM server plays the part of a librarian and referee. Clients who are participating in a collaborative modeling session communicate directly with the server; the server, in turn, passes messages and results to other clients. The server is also responsible for maintaining a persistent copy of the model on its host machine. The server should be hosted by a computer that is remotely accessible and has a fixed address.

The server maintains a dedicated session server for each model. The session server is responsible for coordinating the modelers' efforts and for storing/retrieving the model to/from its host machine. A single server can have multiple session servers, with each session have its own unique name.

A client that wants to participate in a collaborative session has to connect to the server that is responsible for that session. The client must refer to the server by name or IP number, and the port that was specified when the server was started.

To stop a server, press the *Kill server* button (see **Figure 3.**) When the server is stopped, all of the clients connected to the server are disconnected. Sessions maintained by the server will not be available until the server is restarted. When the server is started again, all of the models maintained by the server will be loaded automatically and ready for use.

5 The CDM Client

The CDM client provides a means for the user to connect to CDM servers and to manipulate and/or participate in the sessions maintained by them. The client application also provides a graphical, DEVS-based, modeling tool that can operate in a standalone or collaborative mode.

5.1 *Setting Up Client*

When the client is run for the first time (refer to **Figure 1**), the user will be asked to specify the installation directory. In most cases, the proposed directory will be correct (see **Figure 4.**) Just click *Okay* to continue.

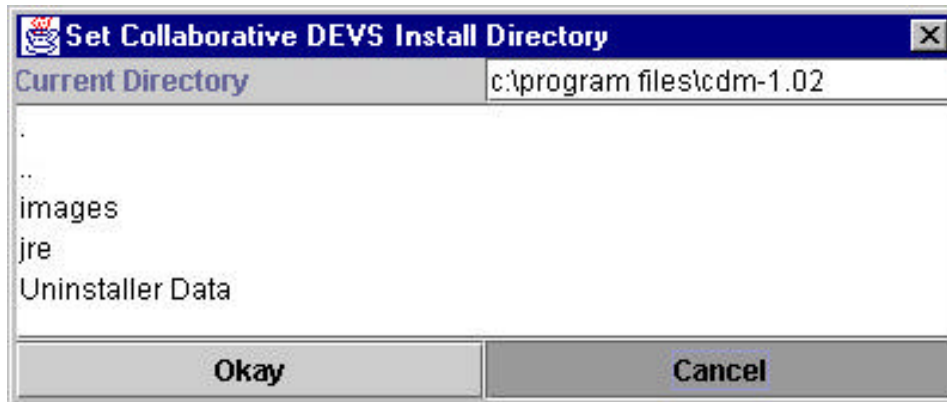


Figure 4: Install Directory Window

5.2 Connecting to a Server

To connect to a CDM server, click the *Enter* button (see **Figure 2.**) A dialog will appear asking you to enter the name of a server. After entering the server name, click *Ok*. If the connection is successful, a list of available model sessions on that server will appear (see **Figure 5**); otherwise an appropriate error message is displayed (refer to Section 9.) The Collaborative DEVS Client window is composed of three displays: *Session Manager*, *System Information*, and *Session: session name(host name)*. These are described next.

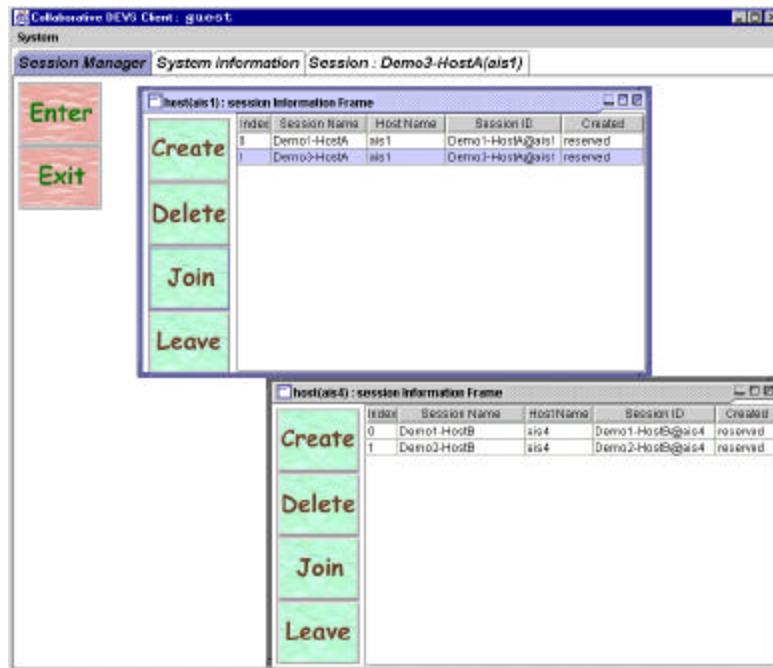


Figure 5: Session Manager Window

5.3 Session-Manager

After connected to a server, a client can create, delete, join, or leave a modeling session. In order to perform an operation on a session, select the session's name from the table, then click the appropriate button.

Creating Session (**CREATE**)

When the *Create* button is clicked, a dialog box will appear asking you to provide a name for the new session. If an unused name is provided, a new session with that name is created. If the name of a previous session is entered, the newly created session manager will search for a saved session with that name. If one is found, the session will be loaded and started. Names can be any string free of white space (i.e. Demo-HostA).

Deleting Session (**DELETE**)

To remove a session, select it from the table and click the *Remove* button. This will remove the session from the list of those available. The session can be restored by *Creating* a session with the same name. The model will not be physically removed from its host computer (session server). **Note:** A default session is created automatically by the CDM server. *The default session should not be removed.*

Joining Session (**JOIN**)

To join a collaborative session, select a session from the table, then click the *Join* button. After a few seconds, a modeling window will appear with a given name such as *default(128.196.33.1)*.

Leaving Session (**LEAVE**)

To leave a session, select the *Session-Manager* tab from the tabs at the top of the Collaborative DEVS Client window. Then, select the session you want to leave from and click the *Drop* button. The modeling window for the session will disappear shortly.

5.4 System Information

The System Information window provides system-related information such as operating system, Java version, host name, etc. (See **Figure 6**.)

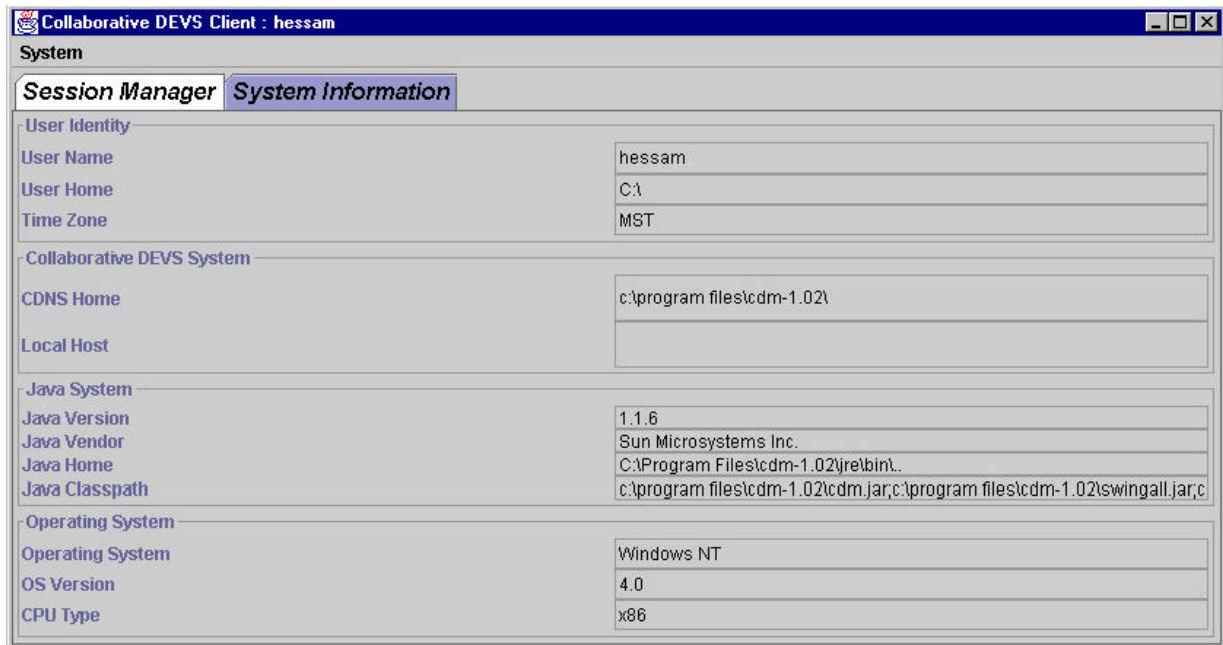


Figure 6: System Information

5.5 Session

Figure 5 shows a session window for a given Session-ID (*Demo3-HostA(ais1)*) which is composed of three parts as shown in **Figure 5**. “Demo3-HostA” is the session name and “ais” is the host name. The DEVS Modeler session provides its user with a tool bar (bottom) and two windows (influence diagram (right), and decomposition diagram (left)) as shown in **Figure 7**. The decomposition diagram is an expandable tree that shows the decomposition relationships for the entire model. The influence diagram shows how the components of a coupled model are interconnected to one another and their parent.

6 DEVS Modeler

The DEVS Modeler supports the construction of hierarchical models in accordance with the axioms of the System Entity Structure (SES) (Zeigler 1984). A SES is comprised of atomic and coupled models and their interconnections. Each coupled model can contain (or decompose into) one or more atomic or coupled models². The atomic models are leaves in the tree, and represent the smallest building blocks of a simulation model. When discussing a coupled model, its children are referred to as components of the coupled model. The ports belonging to these models are called internal (they are internal to the coupled model), while the ports belonging to the coupled model itself are called external (they define the coupled model’s external interface). Similarly, coupling between ports is referred to as internal (internal to internal port coupling) and external (internal to external and external to internal coupling). Briefly,

² Full SES (Zeigler 1984) supports hierarchical model decomposition, specialization, and aspects. The current version of CDM captures a variant of SES that does not support specialization. CDM supports decomposition and aspect features of SES. Aspects represent multiple variations of models given different perspectives and modeling objectives – aspect and specializations are different concepts. For example, a variety of computer architectures such as Divide&Conquer, Multi-Processor, and Pipeline can be represented within a model base (see Section 6.2.)

the relevant SES axioms are:

1. *Valid brothers.* No two models with the same parent can have the same name. This is accomplished in the DEVS modeler by attaching an index number to the end of the model's base name for each instance of the model that is added to a given coupled model. For example, if an instance of model **Engine** is added to coupled model **car**, the label of the model instance associated with **Car** will be **Engine0**.
2. *Strict Hierarchy.* A model may not appear more than once along any path in the SES tree. In other words, a model can not have an instance of itself as either an ancestor or descendent.
3. *Uniformity.* Any two models with the same name are identical (e.g. they have the same ports, couplings, and components).

Models influence one another through their ports. Each model has a set of input and output ports, and these ports can be coupled together to create influence relationships. Input port names for a component must be unique to that component. Similarly, for any component, its output port names must also be unique. Input (output) ports always appear to the left (right) of a component diagram (see **Figure 7.**) All couplings are unidirectional. Four rules govern the use of coupling in a DEVS model

1. *Internal output ports can be coupled to internal input ports.* All internal coupling will link output ports to input ports.
2. *No model can be directly coupled to itself.* A model can not send itself input (direct feedback is better represented as an internal transition in DEVS atomic models).
3. *External input ports can be connected to internal input ports. Similarly, internal output can be sent to external output ports.* Events originating from outside of the coupled model are passed to its components via the coupled models external input ports. Similarly, events generated inside of the coupled model are made visible to the outside world via its external output ports.
4. *Coupling can only occur between components of a coupled model and the coupled model itself.* Coupling can not go across levels in the SES tree, and it is confined to the boundaries of the coupled model in question.

The DEVS modeler maintains a model base (**mbase** in **Figure 7**) that holds a list of atomic and coupled models³. An atomic or a coupled model that appears in the left panel of the Session Window (see **Figure 7**) is called a *base model*. All atomic and coupled base models appear directly underneath the **mbase** folder – that is no base model (atomic or coupled) can be part of another base coupled model. An instance of a base model is called an *instance model*. Instance models appear in the right panel of the Session Window and/or in the left panel of the Session Window. In particular, all instance models (atomic and coupled) must have an index number. Instance models can be used to construct other, hierarchical instance model types while conforming to the axioms discussed above. Or alternatively, instance models can be used to declare other base models. Changes to a base model type will be reflected in all of its instances. Similarly, changes to a model instance (e.g., renaming) are reflected in the base model, as well as all other instances of that type in other base and instance models.

³ While CDM allows for the construction of numerous SESs by mixing and matching other sub-SESs in the **mbase**, the **mbase** should be associated with a single abstract model. All of the sub-SESs that form the **mbase** should be associated with that abstract SES model.

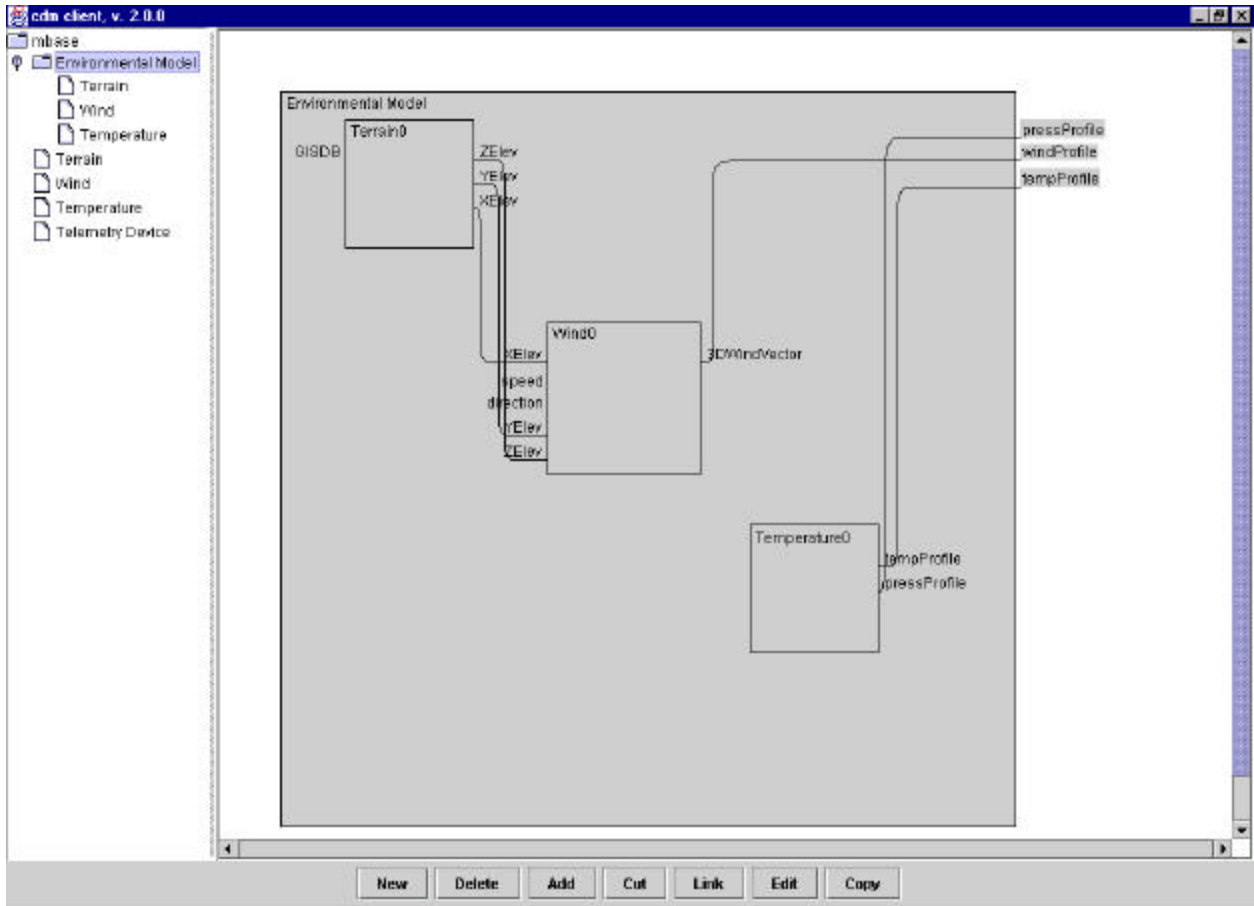


Figure 7: Session Window

To clarify, **Figure 7** depicts a base model called Environmental Model that is composed of the following base model types (left panel): Terrain, Wind, and Temperature. A more detailed inspection of the base Environmental Model (right panel) shows that any instance of the environmental model will be composed of the following instance models: Terrain0, Wind0, and Temperature0. Also, the coupling between instance models and a (potential) instance of the base Environmental Model is also depicted.

6.1 Model Construction

A model is selected for editing by clicking its base model representation in the decomposition tree (left panel in **Figure 7**). In this section, we describe how to edit the **mbase** (model) with the functions provided in the tool bar (bottom panel in **Figure 7**).

NEW

Create a new base model. When *New* is clicked, a dialog box will appear asking for the name of the new model. Enter a name and press *Ok* (or *Cancel* to abort the action). The new base model will appear in the **mbase**.

DELETE

Delete removes a base model **and all of its instances** from the **mbase** and any coupled models that have an instance of the model as a component. When *Delete* is clicked, a list of base model names will be presented. Select the base model to delete and press *Ok*. Click *Cancel* to abort the action.

ADD

Add is used to add a component to a coupled model. When the *Add* button is clicked, a list of base models will be presented. Select the type of model that you want to add, then press *Ok*. An instance of the model will then be added as a component of the coupled model in question.

CUT

Cut is used to remove coupling, ports, or components from a coupled or atomic model. To remove an instance atomic or coupled model, click the *Cut* button, then click on the instance to be cut in the influence diagram. **Changes to the effected models will be reflected by all instances of the models in question.** Note that *Cut* cannot be used to remove any base model; it can be used to remove an instance model only.

LINK

To link two ports, click the *Link* button. Then click on the source and sink ports (in that order) in the influence diagram.

EDIT

Edit is used to add input and output ports to a model, rename ports, and rename models. To rename a model or port, click *Edit* and then click on the name field that you want to change. A cursor will appear and the name field can then be changed to the new value. (Note: the keyboard focus can stick, causing a name change not to take effect. To make sure the name change is completed, click the *Edit* button after entering the new name). To add a port to a model, click on the models left side for an input port or right side to create an output port. A dialog box will appear asking for the name of the new port. Enter a name and click *Ok* to add the port to the model.

COPY

Copy will create a new base model that is an exact copy of the currently selected model with a different name. When *Copy* is clicked, a dialog box will ask for the name of the new model. After providing a name, click *Ok*. The model will appear as a new base model in the **mbase**.

6.2 An Example

In this example, our goal will be to create a simple experimental frame for a multiprocessor architecture. The generator (`genr`) will be an atomic model that has two input ports, `start` and `stop`, and an output port `out`. The multiprocessor (`mproc`) will consist of a coordinator (`MP-coord`) and two simple processors (`proc1` and `proc2`.) The multiprocessor will have one input port `job_in`, and an output port `job_complete`. The coordinator will have a single input port `job_in` and two output ports, `p1` and `p2`. Each processor will have an input port `in` and an output port `out`. Finally, a transducer (`transd`) will compile statistics such as throughput and turnaround time. The transducer will have one output port, `stop`, and two input ports, `job_started` and `job_complete`

First, the generator can be created in the following manner

1. Click the *New* button. Enter the name `genr`, then click *Ok*.
2. Select the generator model by double clicking on the `mbase` icon and then clicking on the `genr` icon in the decomposition tree.
3. Click the *Edit* button, then click on the left side of the `genr` model in the influence diagram. When the dialog box appears, enter `start` and then click *Ok*. Repeat to add the `stop` port.
4. To add the output port, click the *Edit* button then click on the right side of the `genr` model. Enter `out` for the port name and click *Ok*.

The `proc`, `MP-coord`, and `transd` models are created similarly.

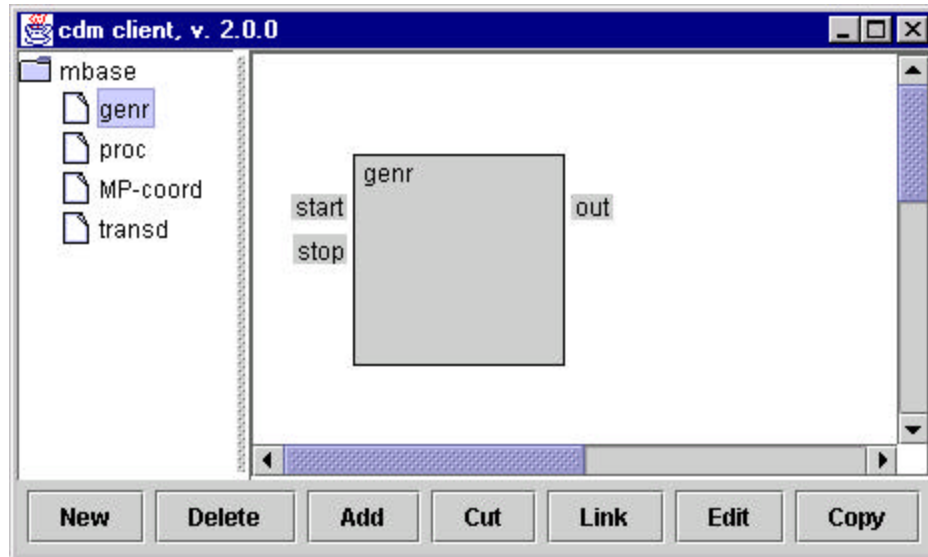


Figure 8: `genr`, `proc`, `MP-coord`, and `transd` models

The multiprocessor can be created with the following steps.

1. Click *New* and enter the name `mproc`, then click *Ok*.
2. Select the `mproc` model from the decomposition tree.
3. Add the `mproc` input port (`job_in`) and output port (`job_complete`) as before.
4. To add the first processor model, Click the *Add* button. Next, select the `proc` model type from the list dialog and click *Ok*. The model `proc0` should appear as a component of `mproc`.
5. Add the second processor in the same way. The model `proc1` should now be shown as a component of `mproc`. Note that models will appear in the order in which they were added from top left corner to bottom right corner.
6. Finally, add the `MP-coord` model to the `mproc` model. The `MP-coord` instance should be labeled `MP-coord0`.

Now, the coupling between the `mproc` model and its components can be established. The `mproc` coupling will consist of the following pairs: (`mproc.job_in`, `MP-coord.job_in`), (`MP-coord.p1`, `proc0.in`), (`MP-coord.p2`, `proc1.in`), (`proc1.out`, `mproc.job_complete`), (`proc0.out`, `mproc.job_complete`). The following steps will create the desired coupling:

1. Click the *Link* button. Click on `mproc`'s `job_in` port, then click on `MP-coord`'s `job_in` port. An arc from `mproc.job_in` to `MP-coord.job_in` should appear.
2. Repeat step 1 for each of the port pairs listed.

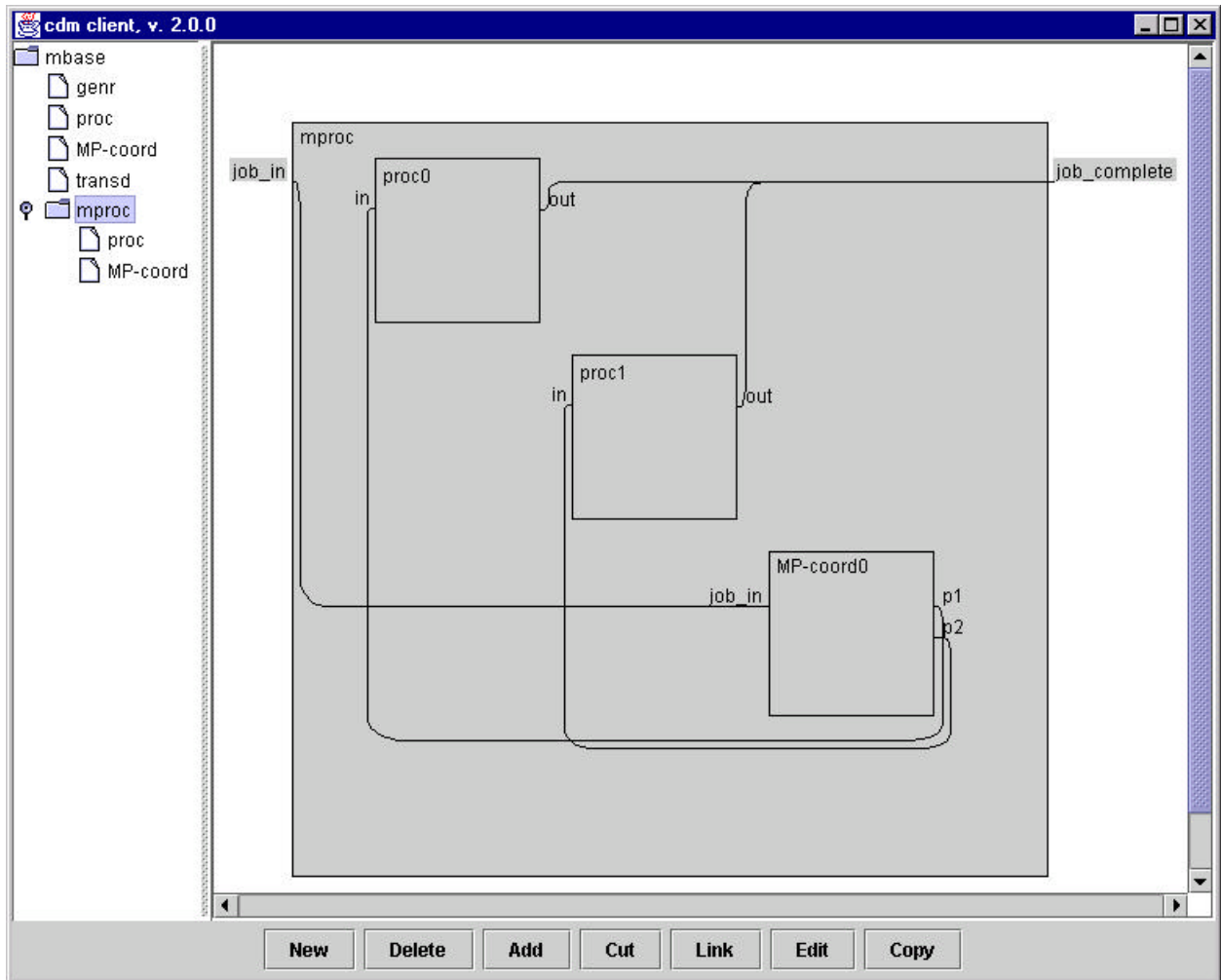


Figure 9: `mproc` model

Next, the `genr` and `transd` models will be used to create an experimental frame that can be attached to the `mproc`. To create the experimental frame, use the following steps:

1. Click the *New* button, enter the name `ef` and then click *Ok*.
2. Select the `ef` model from the decomposition tree.
3. Click the *Add* button, select the `genr` model from the list, and then click *Ok*.
4. Click the *Add* button, select the `transd` model, and then click *Ok*.
5. Add the output port `job_out` and the input ports `start` and `job_in`.
6. Create the following couplings: (`ef.start`, `genr.start`), (`ef.job_in`,

```
transd.job_complete), (genr.out, ef.job_out), (genr.out,
transd.job_started), (transd.stop, genr.stop).
```

The result is show in Figure 9.

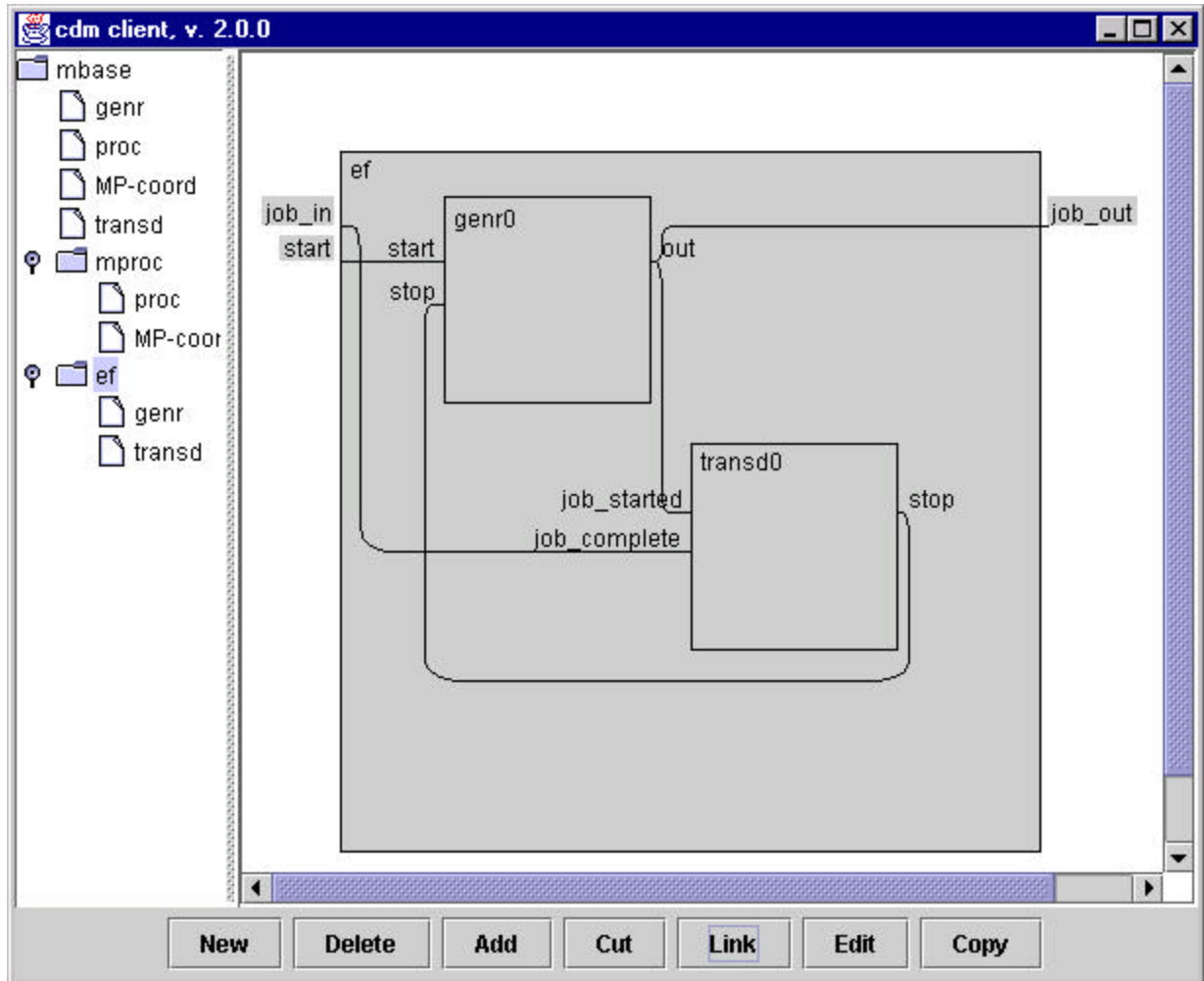


Figure 10: The experimental frame

Finally, the `ef` and `mproc` models will be coupled to form the simulation experiment. The following steps complete the example:

1. Click the *New* button and enter the name `MP-ef`. Click *Ok* to create the base model.
2. Select the `MP-ef` model from the decomposition tree.
3. Click the *Add* button. Select `ef` from the base model list and then click *Ok*.
4. Click the *Add* button. Select `mproc` from the list of models and then click *Ok*.
5. Click the *Link* button. Click on the `ef.job_out` port and the `mproc.job_in` port (in that order) to create the `(ef.job_out, mproc.job_in)` coupling.
6. Repeat 5 to create the `(mproc.job_complete, ef.job_in)` coupling.

The final results are shown in **Figure 11**.

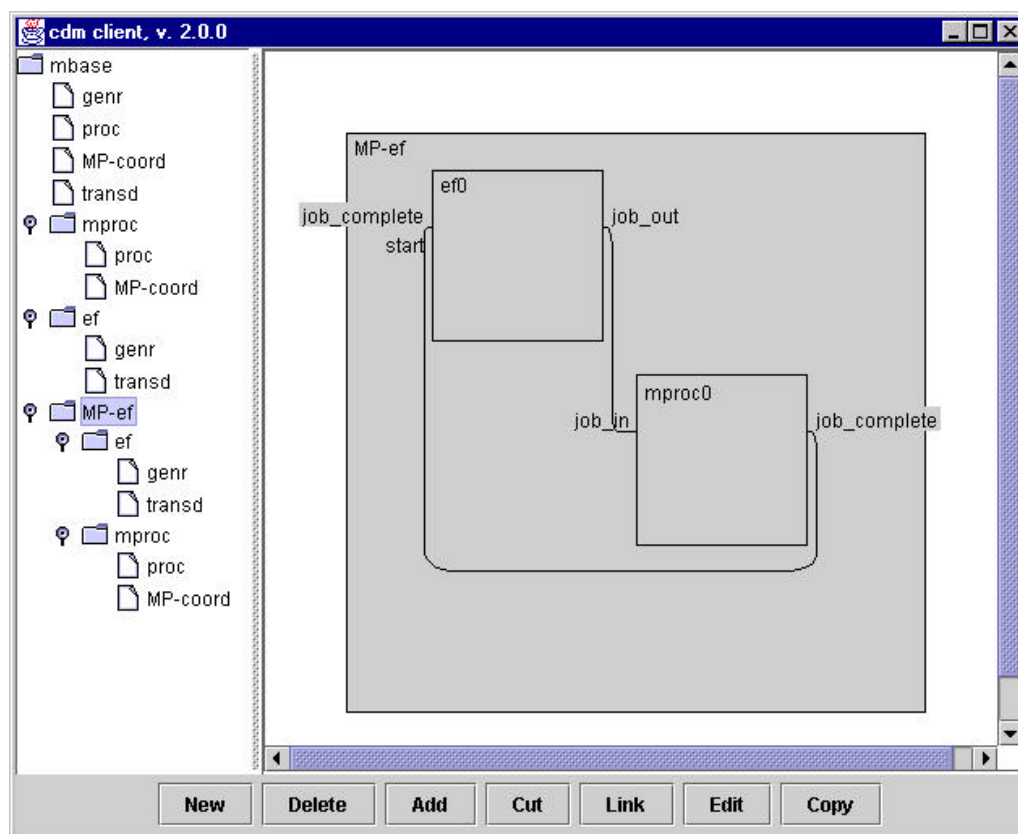


Figure 11: The MP-ef experimental frame

7 Model Naming Convention/Repository

7.1 Client files and file naming conventions.

The client keeps information about the user's installation directory in the `.clientProps` file that is located in the installation directory. Deleting this file (if it is misplaced) will allow the installation directory to be reset (see section 5.1).

7.2 Server files and naming conventions.

The server keeps information about the installation in the `.serverProps` file in the installation directory. Deleting this file (if it is misplaced) will allow the installation directory to be reset. Other files of interest include the `.cdns\sessions` (under the installation directory). This file contains the names of the sessions that will be started automatically when the server is started. Finally, the `.cdmModels` directory contains the model files produced by CDM. Files that are to

be loaded by the server have the format <session name>@<server address>. Files used in the single user mode can have any name.

8 CDM Known Bugs/Limitations

- If a cut component fails to disappear, names get chopped, or other anomalies should occur in the influence diagram, select a different model then go back to the diagram of interest. This will redraw the diagram and should resolve the problem.
- The keyboard focus can ‘stick’ to the model and port names. To ‘unstick’ the focus, click on a button in the tool bar. A side effect can be unreadable characters in a component’s name as it appears in the **mbase** tree.
- Error messages appearing as dialog boxes may be pushed to the bottom of the window stacking order. While the error message is hidden and its *Ok* button has not been pressed, the application will be frozen.
- To stop CDM in the single user mode, select the command window (or xterm window) and press Ctrl-C.
- When a new session is being started, the server window may report an error trying to load the model file. This occurs because the file in question does not yet exist (the model file will not be created until you start to edit the model).
- While using CDM, the command window (or xterm) will produce a continuous stream of output. This output is solely for the use of the developers and should only be of concern when reporting errors.

9 Listing of the Most Common Error Messages

1. *Can't connect to the server <server name>*. Indicates that the server is unreachable. Could be the result of mismatched port numbers or an IP address used where a name was expected and vice versa.
2. *Can't join session (<session name>) at the server <server name>*. Indicates that the session has not been created, or the server has become unreachable.

10 References: Related/Relevant Documentation

- Grudin, J. (1994). “Computer-Supported Cooperative Work: History and Focus.” *IEEE Computer* **27**(5): 19-26.
- Park, S. (1998). Collaborative Distributed Network System Architecture: Design and Implementation. Electrical & Computer Engineering Department. Tucson, AZ, USA, University of Arizona.
- Sarjoughian, H. S., J. Nutaro, et al. (1999). Collaborative DEVS Modeler. International Conference on Web-Based Modeling and Simulation, San Francisco, SCS.
- Sarjoughian, H. S. and B. P. Zeigler (1997). DEVSJAVA: Basis for a DEVS-based Collaborative M&S Environment. SCS Western Multi-Conference, San Diego.
- Zeigler, B. P. (1997). Objects and Systems: Principled Design with C++/Java Implementation. New York, NY, Springer-Verlag.
- Zeigler, B. P., H. Praehofer, T.G. Kim (1999). Theory of Modeling and Simulation, Academic Press.
- Zeigler, B. P., H. S. Sarjoughian, et al. (1997). An Architecture For Collaborative Modeling and Simulation. 11th European Simulation Multiconference, Istanbul, Turkey.