Proceedings of the 1994 Winter Simulation Conference
ed. J. D. Tew, S. Manivannan, D. A. Sadowski, and A. F. Seila

AN INTRODUCTION TO EXTEND™

David Krahl

Imagine That, Inc.
6830 Via Del Oro, Suite 230
San Jose, California 95119

ABSTRACT

This document presents an overview of Extend, a multidomain, hierarchical simulation modeling tool. Extend is designed around a modern graphical user interface and is extensible into many different areas of application including both discrete event and continuous systems.

1 INTRODUCTION

In the past ten years, simulation software has made great strides in power and ease if use. The power available on an analyst's desktop today exceeds what was available on mainframes ten years ago. Additionally, new user interface developments have made that power far more accessible. What has been achieved, however, is only the beginning. With features such as hierarchical modeling, object orientation, libraries of modeling constructs, and a graphical user interface, Extend represents the next generation of simulation software. The low cost and ease of use of Extend is allowing many analysts to place it along side their spreadsheet and word processor as a necessary business tools. Extend is giving simulation power and flexibility to those who would normally be unable to commit the time and money required for traditional simulation programs. Analysts no longer need to become experts in a particular modeling tool or hire consultants to utilize simulation technology.

Extend bridges the gap between a simulator and a simulation language. Models are constructed in a network fashion by specifying the logical flow through the Extend blocks with connectors similar to a simulation language. The Extend blocks, however, can be as rich in features and utilize a similar parameter style input as typically found in simulators. An Extend block can be as simple as an addition function or as complicated as thousands of lines of code will allow. The blocks themselves are built with a compiled simulation specific language, ModL, which is built into the Extend authoring environment should they want to enhance existing or develop new blocks. This gives modelers the capability of building custom libraries (block sets) of high performance blocks for a specific application or industry. Currently, in addition to the libraries of blocks produced by the Extend developer, Imagine That, Inc., no fewer than 12 third party libraries are available for various industries. They range from libraries designed to enhance the Imagine That's own Discrete Event library to highly industry-specific libraries, in areas such as designing paper mills or modeling environmental systems. The true elegance of Extend is that each of these libraries are built with the same block development tools as are used by Imagine That, Inc. to develop their own blocks. In addition to the ModL language, help, custom icons, and dialog parameters are specified graphically. This results in a truly integrated product which is specific to a particular industry, like a simulator, but because modelers still have access to the full Extend product has the power of a simulation language.

Extend was designed around a graphical user interface. This interface was integrally designed from the start, not grafted onto a development of a DOS or mainframe based program. The method of model entry and analysis as well as specific graphical support tools create an environment uniquely suited to modern "point and click" interfaces. This results in a consistent, easy to use simulation product that is a natural environment to analysts who are accustomed to a windowing interface. Extend is available for Microsoft Windows and the Macintosh. These operating environments have become industry standards supporting a wide range of applications which can be used to support the modeling effort with simple "cut and paste" or automatic information exchange. Some examples include: spreadsheets that can be used to store model input and output data without elaborate import or export routines, and drawing programs whose images can be pasted into Extend as an icon.

Extend is on the leading edge of the "state of the art". Other simulation vendors are only now beginning to develop and release their "next generation" simulators with features that Extend has supported since 1988. Specifically, features such as hierarchy, industry specific libraries, object orientation, and authoring environments are only now beginning to emerge from the traditional simulation software vendors (Collins and Watson 1993) (Hendriksen 1993). In any case, the cost, in terms of dollars and time for these products, continues to be much higher than that of Extend. Extend is bringing simulation to those who need this technology, not just a select

few who have had the resources for traditional simulation tools.

2 BUILDING AN EXTEND MODEL

Extend models are constructed with library based iconic blocks. Each block describes a step in a process or a calculation. As stated earlier, blocks reside in libraries. Each library represents a grouping of blocks with similar characteristics such as Discrete Event, Plotters, Electronics, or Business Process Reengineering.

There are two types of logical flows to and from the Extend blocks. The first type of flow is for items which represent the objects that move through the system. Items have attributes and priorities. Examples of items include parts, patients, or a packet of information on a network. The second type of logical flow is for values or information. Values represent a single number. Examples of values include: the number of items in queue, the result of a random sample, and the level of fluid in a tank. Each block has connectors which are the interface points of the block. Connections are lines used to specify the logical flow from one connector to another. Item connectors and connections are represented by double lines and value connections and connectors are represented by single lines. A single server, single queue would have the following form:

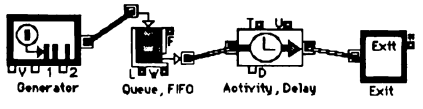

Figure 1: A simple model, single server single queue.

The block on the far left represents a Generator which periodically creates items. Following this is a Queue block which holds items until requested by the following block, the Activity Delay. The Activity Delay represents a limited capacity of one processing unit and delays an item for a fixed amount of time. The last block in the model is an Exit block which removes items from the system. An enhancement to this model would be to specify that the delay in the Activity Delay is determined by a specific random distribution. This can be done by connecting the output of an Input Random Number block to the delay (labeled "d") connector on the Activity Delay block as follows:

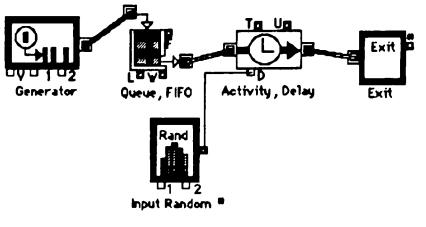

Figure 2: A simple model with random process times

Another feature that can be added to the model is a Discrete Event Plotter which plots, in this example, the contents of the queue. The first Plotter Discrete Event value input connector will be connected to the Queue's length (labeled "L") value output connector as follows:

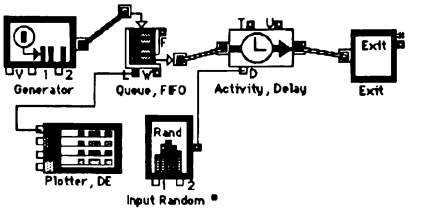

Figure 3: Discrete Event Plotter added to simple model

Simulation parameters such as number of runs and simulation end time can be specified in the Simulation Setup menu item under the Run menu. The simulation can then be run by selecting the Run Simulation menu item from the Run menu.

During the run, the current simulation status is displayed in a bar near the bottom of the monitor screen. This displays the estimated time before the run will be completed, the current simulation run time, the number of simulation steps completed so far, and the current simulation run number.

Once the simulation run has completed, the results of the simulation are contained within the blocks. Double clicking on each block reveals the information collected from the simulation run. For example, double clicking on the Queue FIFO block opens the dialog which shows the following information about the state of the Queue FIFO block:

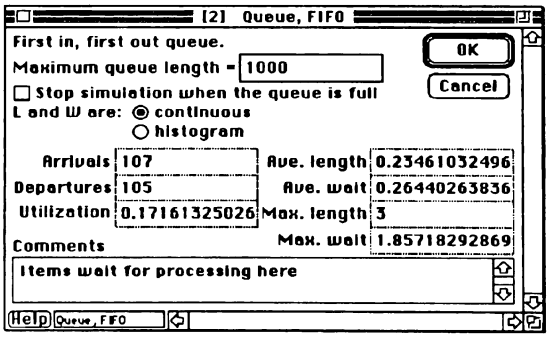

Figure 4: Dialog of Queue FIFO

The plotter block shows the number of items stored in the Queue FIFO over time:

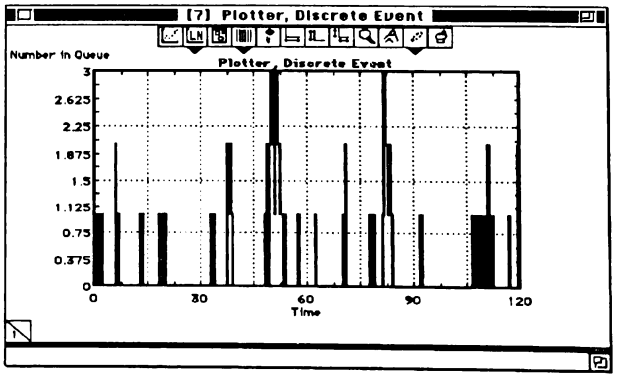

Figure 5: Plot of queue length

Simulation results may be stored in a table, plotted, cloned to a different area of the worksheet, exported to another program such as a spreadsheet or database, displayed in an animation, or even used to control some aspect of the computer's operation through external device drivers.

3 HIERARCHY

There are two methods for building custom blocks in Extend. One is hierarchy, a second is through the ModL scripting language which will be discussed later. Hierarchical blocks represent a collection of blocks grouped under a single icon. Hierarchy can be any number of levels deep and hierarchical blocks can be stored in libraries for use in other models.

One method for building a hierarchical block is to build a model on the worksheet. Select the blocks to be contained within the hierarchical block and choose "Make Selection Hierarchical" from the Model menu. Extend automatically creates connectors on the new hierarchical block where the interface points are to the remainder of the model. Double clicking on the hierarchical block brings up the individual blocks so that individual parameters can be modified. The structure of the block, including unique help text, icon, and connectors can be edited by option double clicking on the hierarchical block. At any point, the block structure may be edited to change the external connectors, blocks within the hierarchy, the appearance of the block icon, or any other customizable feature of the block. Once a block has been created it can be optionally stored in an Extend library and reused in other models.

Hierarchy is important in graphical modeling for large models. Without hierarchy, large models are difficult to comprehend or maintain. Hierarchy allows the modeler to flexibly group logical sections of the model. This feature also supports team modeling. As each hierarchical block is independent, blocks created by different modelers can be easily combined. Because any values and names used within a hierarchical block are local to that block, all that needs to be defined are the block external interface requirements.

4 STANDARD EXTEND LIBRARIES

The standard Extend libraries include functions for discrete event modeling, results plotting, generic calculations, electronics design, interprocess communication, and utilities. For discrete event modeling, the most commonly used standard libraries are the Discrete Event, Generic, and Plotter. Additional, optional, discrete event libraries include the Business Process Reengineering and Manufacturing libraries which will be discussed in detail later.

Extend supports the following general modeling functionality for discrete event modeling:

- Attributes - Unique variables which are local to the items moving through the simulated system.
- Priorities - A unique value, local to a given item, which can be used to rank items in a queue or interrupt items in process.
- Values - The number of items represented by a single item. Setting a value will create clones of an item when that item arrives to a queue, resource, or exit block.

The most commonly used block types in the Discrete Event library are as follows:

- Activities - These represent limited capacity time delays and include the Activity Delay, Activity Attributes, the Activity Service, and the Activity Multiple blocks.
- Batching - Batching in Extend allows multiple items to be combined into a single item and then later restored to their original items. The Batch and Unbatch block perform this functionality.
- Resources - Resource blocks hold a limited number of items which can represent scarce resources in the model. Typically, resources are batched with an item at the start of a process and unbatched from the item when the process is complete. In the Discrete Event library, the Resource block serves this function.
- Decisions - The Select DE (Discrete Event) blocks in Extend allow items to choose one of a number of paths based on a value input to the Select block. The BPR and Manufacturing libraries contain more sophisticated versions of this block.

A simple Extend model using these blocks:

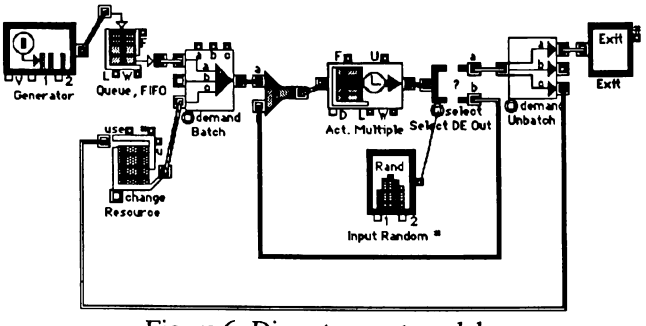

Figure 6: Discrete event model

This models a single queue, multiple server system with a rework loop. The resource is acquired before processing and not released until the item has successfully completed processing.

The Generic library is used for both continuous modeling and discrete event modeling. In the continuous mode, calculations are performed at each evenly spaced clock step. In the discrete event mode, calculations are made in response to a request (message) from a discrete event block.

When used with Discrete Event library, the generic library is typically used to provide values for inputs or operate on the value outputs of the discrete event blocks.

Typical examples of using the Generic library in this mode include using a Decision block to compare the length of two queues or using an Input Random Number block to generate a random time delay for an Activity.

There are a number of classes of generic blocks. These include: mathematical calculation, integration, file operations, logical calculations, integration, statistical calculations, error reporting, simulation events (such as playing a sound or displaying a dialog), accumulation, and threshold detection.

In addition to the above libraries, Extend also includes libraries for statistics, animation, plotters, utilities, electronics, filters, digital circuits, controls, and apple events. Libraries are available from third party developers for control systems, paper manufacturing, neural networks, biology, and signal processing.

5 EXTEND MODELING SUPPORT FEATURES

Extend is a complete modeling environment which not only gives the simulation modeler complete model building and analysis tools, but also allows the modeler to communicate with other software for data input and analysis. In addition, there are many thoughtful features designed specifically to support the hierarchical, graphical, object oriented environment. Following is a list of some of the most interesting Extend modeling support features.

5.1 The Worksheet and Drawing Tools

Extend models are built by dragging the appropriate blocks from the library window or pull down menu and placing them on a main worksheet. The Extend modeler can zoom and scroll this worksheet as well place explanatory text on the screen. There is also a set of graphic tools which allow the user to draw various shapes and lines on the screen to further enhance model presentation. Of course, because Extend is a native graphical user interface application, the clipboard can be used to paste graphical objects in to Extend as well as copy Extend models, inputs, and outputs into other applications. For example, a symbol can be drawn in a sophisticated drawing package and then pasted into Extend as an icon or on the background screen.

5.2 The Notebook and Cloning

In addition to the main worksheet, Extend also makes use of a Notebook which can contain auxiliary model information. Perhaps the most interesting use of the Notebook is using it to contain cloned dialog items (a cloned dialog item is a copy of a block dialog item which can be placed in a hierarchical block the model worksheet, or in the model notebook). This feature solves the problem of giving the user easy access to an important dialog variable which may be buried under many levels of hierarchy. The notebook can also contain graphical objects and text. This gives the Extend modeler the capability to build a customized user interface with cloned dialog items, label them with industry specific text, and use graphic objects to enhance the presentation. The example below represents access to important model parameters from over 20 blocks and was built by the author in less than an hour. It includes the ability to modify all important simulation input parameters, reporting simulation results and graphing variable levels.

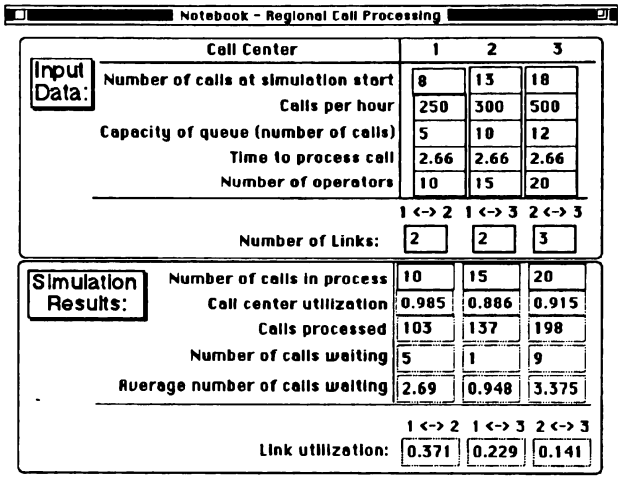

Figure 7: Using the Extend notebook to create a custom interface

5.3 Spreadsheet Access

Extend can access spreadsheets in a number of ways. As mentioned earlier, simulation data can be cut and pasted to and from GUI compliant spreadsheets. There are, however, other, more automatic methods which include Apple Events, which allows the "live" transfer of data, and the publish and subscribe function on the Macintosh which allow entire tables of data to be transferred at the beginning or end of a simulation run.

5.4 Named Connections

Named connections allow the modeler to transfer the logical model flow from one point in the model to another. This is useful in reducing the "spaghetti" of connections particularly in a large model where some connections may span the entire worksheet. Extend has a "Show Named Connections" feature which allows the modeler to trace the flow through the named connections.

5.5 Equations

The equation functions, used in the Equation block, provide a block level method of evaluating complex equations. This functionality gives access to the ModL language and functions in a runtime format. Full access to the ModL language, system variables, and up to 5 user input variables is available.

5.6 Controls

There are three controls in Extend which can be used for interactive user input. These are:

- Slider - Provides a continuous range value inputs controlled by a sliding mechanism.
- Switch - Provides a 0/1 input.
- Meter - Reports a continuous range of values.

Each one of these controls is represented as a tool on the screen (or within a block) that the user can manipulate either before, during, or after the simulation run.

6 PLOTTERS AND TABLES

Extend has a number of plotters and tables that can be used to represent simulation results, store simulation input and output and provide ASCII file transfer capabilities.

For discrete event modeling, the following plotters are useful:

- Plotter Discrete Event - Displays the value of up to four variables throughout the run. The plot data is stored in a table which can be viewed or transferred to another application.
- Plotter DE Multisim - Plots one variable over four simulation runs.
- Plotter DE Error Bars - Records an observation of a variable's mean over a period of time and over multiple runs plots the mean and standard deviation for each batch.

7 BPR AND MANUFACTURING LIBRARIES

The BPR (business process reengineering) and Manufacturing libraries are vertical market libraries produced by Imagine That, Inc. They are discrete event libraries which provide additional blocks to be used in conjunction with Extend's standard libraries.

The BPR library is designed for modeling business processes. Each block in the BPR library uses business terminology. Examples include the Transaction, Decision, and Labor blocks. The blocks are represented by standard flow-charting symbols which make models easier to understand by those unfamiliar with Extend.

The Manufacturing Library focuses on industrial modeling. It includes constructs such as conveyors, AGV's, flexible batching and unbatching operations, and enhanced queuing and preempting operations.

8 MODELING IN EXTEND

Model building in Extend is a "drag and drop" operation. The modeler selects a block from the appropriate library window (or menu) and drags it with the mouse onto the Extend worksheet. The modeler can connect the connectors from one block to the next, and set the block dialog values, defining the system logic.

The following illustrates two different models built in Extend. The two approaches presented are, in the author's opinion, two of many that could be used by a modeler. Both are intended to illustrate graphical model building, hierarchy, and presentation to the end user.

8.1 A Call Center Model

The first model is a call answering center for airline reservations. There are three regional call centers, each one with its own queue of calls. Should a queue become full, the call can be transferred to another call center if a link to that call center is available and there is space in the queue at that call center (otherwise the caller will not get through and will be counted as lost business).

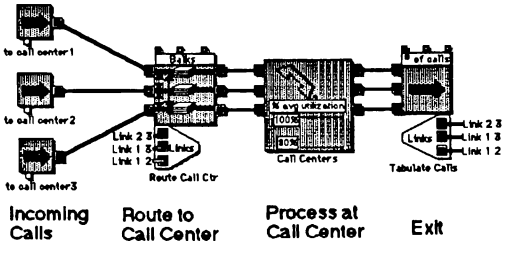

Figure 8: Call Center model

At the top level, only hierarchical blocks are visible. This model represents two levels of hierarchy (some of the hierarchical blocks contain hierarchical blocks themselves). This illustrates a complex model being represented only by its four most major steps. The actual number of library blocks is 104. While certainly not a large model, even this number of blocks, displayed without hierarchy, would be impossible to display here. Below is the hierarchical block which describes the arrivals to the system. This block includes an Import block which generates the item, two Constant blocks and a Divide block which calculates a time between arrivals based on a user specifies rate of arrivals, and an Animate Item block which will show an animation action at the top (worksheet) level when an item passes through it. This hierarchical block also includes a clone of the dialog value of the lower constant block which represents the rate at which arrivals occur. Additional text has been

placed next to the clone and a box drawn around it to enhance clarity.

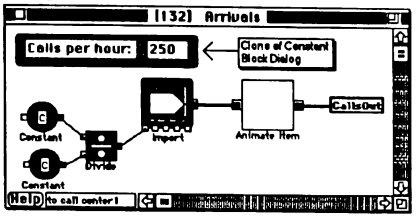

Figure 9: Hierarchical Block in Call Center Model

Significant model parameters such as the initial number of calls, the average number of calls per hour, the capacity of the queue, the number of operators on duty, the average time to process a call, and the number of links are cloned to the Notebook (see Figure 7). The Notebook also contains the simulation results of the total throughput, the average number in the queue, the current number of calls in process, the utilization of the operator, and the utilization of the links. In addition, cloned plots of the number of calls waiting and the number of calls in process are in the Notebook as well. This is an example of a special purpose simulation created from an Extend model with only "cut and paste" operations. The user is able to manipulate the critical input parameters and view simulation results from the Notebook alone. If desired, virtually any Extend model parameter can be brought forward to the Notebook.

Another possible approach to this model would be a more modular model. This would allow the user to vary the number of call centers by building hierarchical blocks representing the links, operators, queues, decisions, and inputs to the model. A model constructed in this fashion would yield a more complex model at the top level, but would allow the modeler to easily change from three to four call centers by adding additional hierarchical "primitives".

8.2 An Environmental Model

The following model is drawn directly from Pritsker (1986) (referred to as "Cedar Bog Lake"). It models a simple ecosystem for consisting of solar energy, plants, herbivores, carnivores, organic material, and energy transfer. The equation below (1) indicates the relationship between each of these factors.

$$
\begin{aligned}
solar &= 95.9(1.0 + 0.635\sin(2\pi time)) \\
\frac{d(plants)}{dt} &= solar - 4.03\,plants \\
\frac{d(herbivores)}{dt} &= 0.48\,plants - 17.87\,herbivores \\
\frac{d(carnivores)}{dt} &= 4.85\,herbivores - 4.65\,carnivores \\
\frac{d(organic)}{dt} &= 2.55\,plants + 6.12\,herbivores + 1.95\,carnivores \\
\frac{d(energy)}{dt} &= 1.00\,plants + 6.90\,herbivores + 2.70\,carnivores
\end{aligned}
\tag{1}
$$

At the highest level, the model appears as follows:

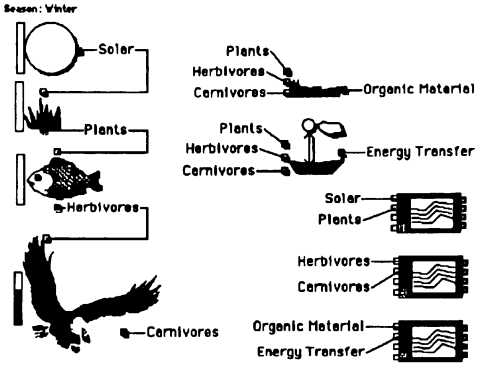

Figure 10: Cedar Bog Lake

The actual equations are defined graphically within each of the hierarchical blocks. For example, the block which calculates level of plant life would be:

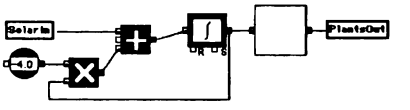

Figure 11: "Plants" block

The Notebook has also been used to display the plots for each of the environmental variables:

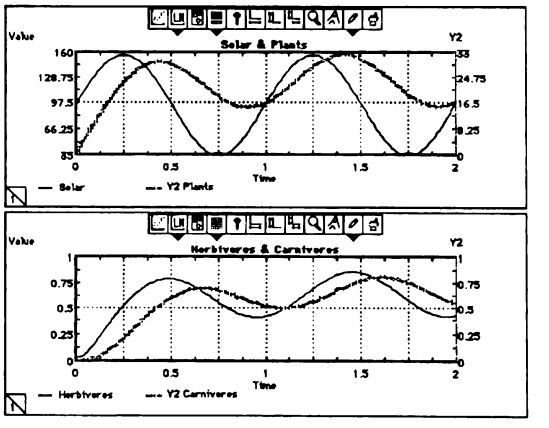

Figure 12: Graph of Cedar Bog Lake variables

These examples illustrate the modeling flexibility of Extend. The Call Center model is derived from an actual system (differing only in the numbers used) and the Cedar Bog Lake model is an implementation of a classic continuous simulation example. Note that neither of these models required any programming to create either the model or custom user interface.

9 MODL

ModL is the programming language which defines the logic of each Extend block. The ModL language is similar to the c programming language. This allows Extend users who are familiar with the c language to be able to use ModL easily. Each block has a series of

ModL message handlers and procedures within it which provide the primary method of interfacing with other blocks.

The message handlers are functions which are called in response to system events. Typical message handlers are:

- on initsim - Called at the start of the simulation and is typically used for initializing the block variables.
- on checkdata - Called before initialization to check the consistency of the simulation parameters and construction.
- on simulate - Called at each simulation event. Most discrete event blocks ignore this event unless an event is scheduled to occur within that block at the current time.
- on endsim - Called at the end of the simulation and is usually used to record a last data point and report simulation results.
- on <connector name> - This message handler responds to a signal to a given connector. This is the main method by which one block communicates to another. For example, when an Activity Delay releases an item, it will send a message to the previous block requesting another item, if this block can not hold an item, it will send a message upstream through the connections until a block is reached that can return a message indicating whether or not an item is available.

Procedures are functions which are called by the message handlers. They are included in ModL for convenience and language completeness.

There is a range of support procedures available in Extend as well. Some examples are: mathematical operations (such as max and min), matrix manipulations, integration, animation, string manipulation, block messaging, attribute management, and system information.

The ModL language is available within the Extend authoring environment. This environment also includes an icon builder for creating a graphical representation of the completed block, a text editor for creating the help text associated with the block, a list of system and user variables available to the ModL programmer, and a graphical builder for defining the dialog associated with the block.

In the Icon Builder, a block developer can either use Extend's built in drawing functions or paste a drawing in from the system clipboard. This is also where the connector position and types are specified. An Extend block can have up to 255 connectors per block (although it is generally not recommended to have more than 10 or 15). The name of the connector specifies if it is an input or an output from the block. A connector that ends in "in" represents an input and "out" represents an output. These connector names are used as variables in the script. Connectors can be used to pass single values, arrays, or messages.

The text editor for the help provides a rich text format for providing the help which is unique to this block.

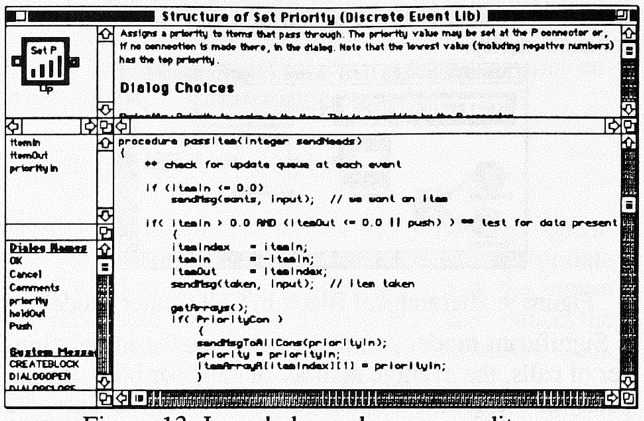

Figure 13: Icon, help, and program editors

The dialog can include check boxes, radio buttons, buttons, text fields, numeric fields, and more. The programmer can set each one of these to display only or editable items. In addition, any parameter can be treated as a program variable. One example of this is in the Input Random Number block where the names of the parameters displayed on the dialog are changed by the model program depending on which distribution has been chosen.

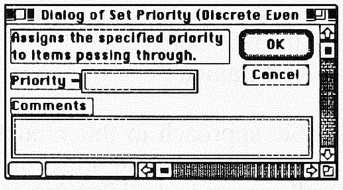

Figure 14: Dialog editor

10 SUMMARY

Extend is a sophisticated modeling tool for both continuous and discrete event simulation. It provides multiple methods for developing application specific components and interfaces and is suitable for rapid prototyping as well as detailed analysis. Most importantly, however, is that Extend's low cost, multidomain capability, powerful modeling features, and elegant user interface define Extend as the leader in the next generation of simulation software.

REFERENCES

Collins, Norene and Christine M. Watson. 1993. Introduction to Arena, in *Proceedings of the 1993 Winter Simulation Conference*, ed. G. W. Evans, M. Mollaghasemi, E. C. Russell, and W. C. Biles, 205-212. IEEE Piscataway, NJ.

Hendriksen, James O. 1993. SLX, the Successor to GPSS/H, in *Proceedings of the 1993 Winter Simulation Conference*, ed. G. W. Evans, M. Mollaghasemi, E. C. Russell, and W. C. Biles, 263-268. IEEE Piscataway, NJ.

Imagine That!, Inc. 1994. Extend Software Manual. San Jose, CA.

Pritsker, A. A. B. 1986. *Introduction to Simulation and SLAM II*. 3rd ed. New York: Halsted Press.

AUTHOR BIOGRAPHY

DAVID KRAHL is the Technical Coordinator for Imagine That! Inc. In this position he is responsible for the technical support and block development for Extend. David received a BS degree in Industrial Engineering in 1986 from the Rochester Institute of Technology. Since then, David has performed consulting, technical support, and development for a wide range of simulation products.