Proceedings of the 1995 Winter Simulation Conference
ed. C. Alexopoulos, K. Kang, W. R. Lilegdon, and D. Goldsman

BUILDING END USER APPLICATIONS WITH EXTEND™

David Krahl

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

ABSTRACT

This document presents an overview of the Extend modeling environment, emphasizing Extend's features for building a custom user interface. Extend is a graphically oriented discrete event and continuous simulation application with an integrated authoring environment.

1 INTRODUCTION

Too many simulation models are dull. While simulation has graduated from the mainframe to the desktop, aside from animation, few simulation modeling tools are able to exploit the interactivity possible on a personal computer. Simulation models can and should be captivating, encouraging the model user to experiment and use the model as a tool to explore. Additionally, a simulation analysis is not complete until the results are communicated to others and used to support decision making. This paper will discuss how Extend can be used to create models that are engaging as well as informative.

Two model examples will be used throughout the paper to illustrate the use of Extend. The first (discussed in Sections 2 and 3) is a single server, single queue system with random arrivals and processing times. The second model is of a call center environment for medical advice and appointments. The call center model compares a proposed system to the existing call system based on the percentage of calls answered within 1 minute. Section 4 will focus on the user interface and control panel for the model of the new call center.

2 EXTEND'S MODELING ENVIRONMENT

Before looking into how Extend can be used to build interactive models, it is helpful to understand the Extend modeling environment.

Extend models are constructed with library-based iconic blocks. Each block describes a step in a process or a calculation. 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 between the Extend blocks. The first type of flow is items which represent the objects that move through the system. Items can 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. Figure 1 shows the connector symbols for the value and item connectors.

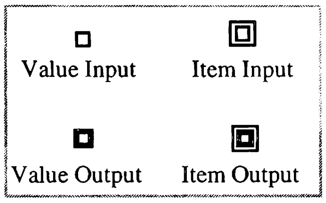


Figure 1: Value and Item Connectors

Connections are lines used to specify the logical flow from one connector to another. Item connections are represented by double lines and value connections are represented by single lines.

A model of single server, single queue system would have the following form:

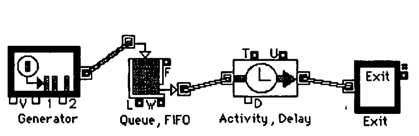


Figure 2: A Single Server Single Queue Model

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 next block. 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 connector (labeled "D") on the Activity Delay block as in Figure 3:

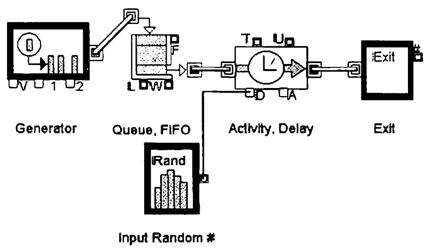


Figure 4: A Model With Random Process Times

Another feature that can be added to the model is a Discrete Event Plotter which graphically displays, in this example (Figure 5), the contents of the queue. The Discrete Event Plotter value input connector will be connected to the Queue's length (labeled "L") value output connector as follows:

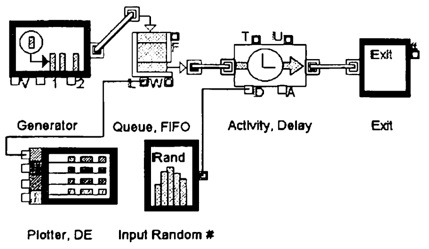
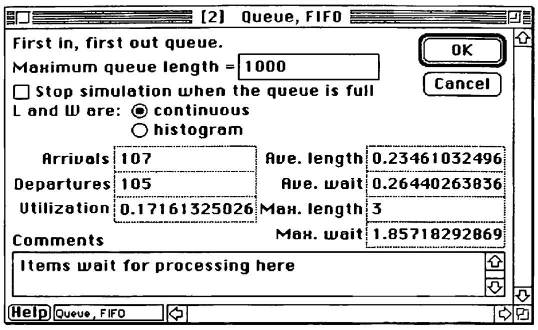


Figure 6: Discrete Event Plotter Added to Model

Simulation parameters such as the number of runs and simulation end time can be specified in the Simulation Setup dialog 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 reported 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 a dialog which shows the following information about the state of the Queue FIFO block:



First in, first out queue.

Maximum queue length = 1000

☐ Stop simulation when the queue is full

L and W are: [✓ continuous] histogram

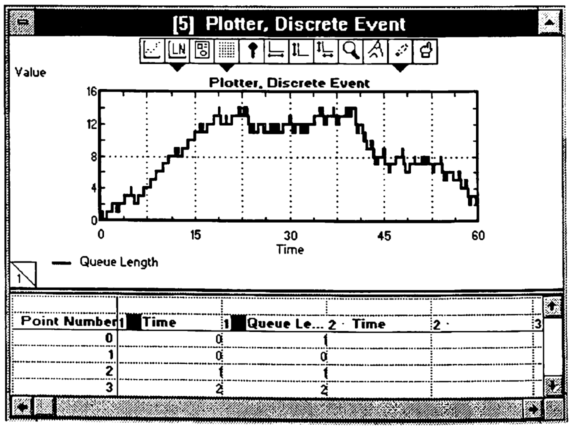
Arrivals	107	Ave. length	0.23461032496
Departures	105	Ave. wait	0.26440263836
Utilization	0.17161325026	Max. length	3
		Max. wait	1.85718292869

Comments

Items wait for processing here

Figure 7: Dialog of Queue FIFO

The Plotter block shows the number of items stored in the Queue FIFO over time in both graphical and tabular format:



Point Number	1 Time	1 Queue Le...	2 · Time	2 ·
0	0	1		
1	0	0		
2	1	1		
3	2	2		

Figure 8: 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 STANDARD EXTEND LIBRARIES

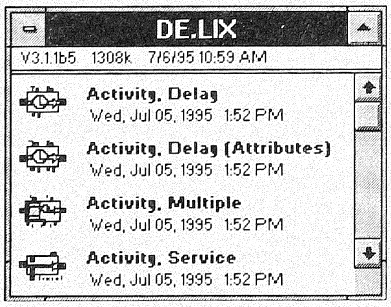
The standard Extend libraries include constructs for discrete event modeling, results plotting, generic calculations, electronics design, interprocess communication, and utilities. For discrete event modeling, the most commonly used standard libraries are the Discrete Event, Generic, and Plotter. Additional,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

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.

3.1 Discrete Event Library

The Discrete Event library contains blocks which are specific to modeling discrete event systems. In a discrete event model, the clock will update at intervals dictated by the individual events in the system. The discrete event blocks pass items to one another through their item connectors. If a discrete event block is unable to receive an item it rejects it and the item waits until it receives a downstream request.



DE.LIX
V3.1.1b5 1308k 7/6/95 10:59 AM
Activity, Delay Wed, Jul 05, 1995 1:52 PM
Activity, Delay (Attributes) Wed, Jul 05, 1995 1:52 PM
Activity, Multiple Wed, Jul 05, 1995 1:52 PM
Activity, Service Wed, Jul 05, 1995 1:52 PM

Figure 9: Discrete Event Library Window

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

- Activities - Time delays
- Batching - Combining of items
- Resources - Limits capacities
- Decisions - Chooses alternate paths

3.2 The Generic Library

The Generic library is used for both continuous 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.

3.3 Other Libraries

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

4 CUSTOMIZING EXTEND

The above discussion illustrates the highly graphical and interactive nature of Extend. However, Extend is also malleable: it can take the shape of the model application.

The most visible aspect of a custom model is the user interface. By modifying an existing interface or creating a new one, the simulation modeler is able to create a model which can be exercised by someone more familiar with the system than with the simulation tool. Models can be built that fit naturally into the conceptual framework of the person using the model. In Extend, this can be done with a range of tools, including:

- Hierarchical modeling - Models can be subdivided into logical components.
- Dialog cloning and the Notebook - Consolidate critical parameters and results to a central location. This can be used to create a custom user interface.

- Interactive controls - Add sliders and switches which provide interactive control of the model
- Pictures and animation - The model becomes the interface.
- User messages - Users can be prompted for information at any point in the simulation.
- Block development environment - Create modeling components that are unique to a specific environment.

The following sections describe how these features can help modelers build models that can be easily exercised by a wide range of people.

4.1 Hierarchy

Hierarchical blocks (H-Blocks) contain other hierarchical and/or blocks from Extend libraries. Each H-Block can have a unique icon, external connectors, and help text.

Figure 10 illustrates the hierarchical block for an appointment area. There is an item input connector for arriving phone calls and value output connectors for the utilization, number of calls completed, number of calls lost, and the percent of calls that were answered within 1 minute.

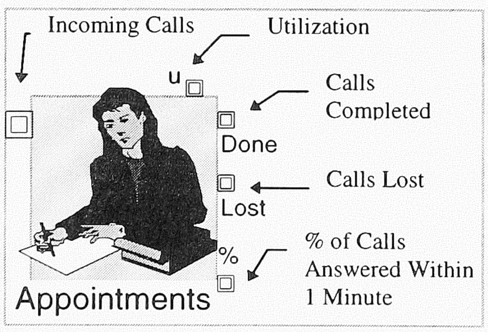
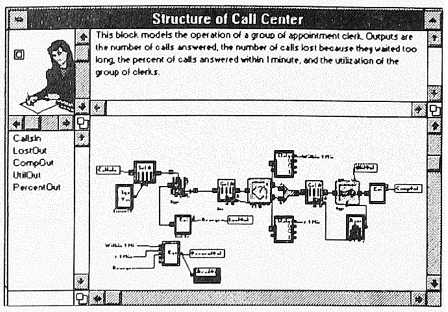


Figure 10: Hierarchical Block

H-blocks are typically created by selecting a range of blocks on the model worksheet and choosing the "Make Selection Hierarchical" menu item. The H-block is created with a default icon and connections are automatically added. H-blocks can then be edited using a structure editor as shown in Figure 11.


Structure of Call Center

This block models the operation of a group of appointment clerk. Outputs are the number of calls answered, the number of calls lost because they waited too long, the percent of calls answered within 1 minute, and the utilization of the group of clerks.

Figure 11: Hierarchical Block Structure Window

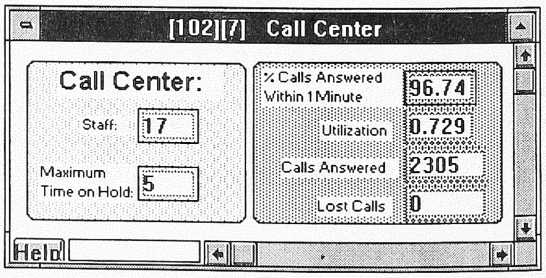
Once completed, H-blocks can be placed in a library for use in other models. Changes made to the source H-block can be reflected in the models that use it.

4.2 Cloning for a User Interface

Often used in conjunction with H-blocks, dialog cloning allows the modeler to create a copy or "clone" of an item's dialog and move it out to a different area. This can be used to create a dialog for an H-block by cloning out important dialog items to the top level of the H-block. Descriptive text and graphics can be added to enhance the user interface.

Cloning is done by selecting the clone tool (◉) from the toolbar, clicking on the dialog item to be cloned and dragging it to another location.

In Figure 12, the significant dialogs from the H-block in Figure 11 have been cloned to the upper left corner, and descriptive text and simple graphics have been added. The block structure has been hidden from the model user by shrinking the H-block window. This creates an "executive interface" that allows a model user to easily change model parameters without needing to understand the entire model structure.


[102][7] Call Center

Call Center:

Staff: 17

Maximum Time on Hold: 5

% Calls Answered Within 1 Minute: 96.74

Utilization: 0.729

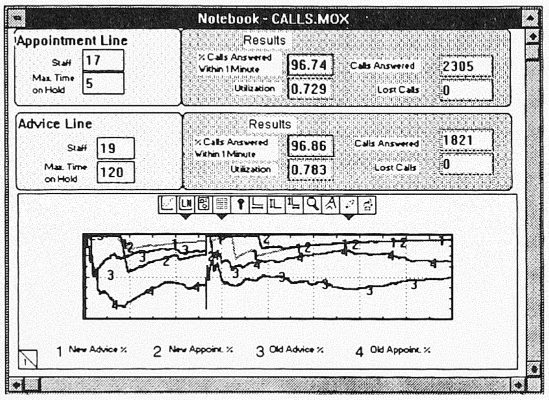
Calls Answered: 2305

Lost Calls: 0

Figure 12: Cloned Dialog in H-block

Dialog items can also be cloned to the Notebook (a separate page for storing model input and results) to create a central location for the significant model information.

Figure 13 illustrates a simple interface in Extend's Notebook for changing model parameters. Here, in a call center model, the staff level and the time that a customer is willing to wait in line can be easily changed. Once the simulation has completed, the number of calls lost (due to callers waiting too long), number of calls completed, the service level and the utilization are reported in an adjacent area. There is also a plot comparing the service level of this scenario to the base alternative.


Notebook - CALLS.MOX

Appointment Line
Staff 17
Max. Time on Hold 5

Results
% Calls Answered Within 1 Minute 96.74
Calls Answered 2305
Utilization 0.729
Lost Calls 0

Advice Line
Staff 19
Max. Time on Hold 120

Results
% Calls Answered Within 1 Minute 96.86
Calls Answered 1821
Utilization 0.783
Lost Calls 0

Figure 13: Clones Used in the Notebook

4.3 Controls

Extend includes a number of interactive controls which give the model developer flexibility to define a control panel interface. Figure 14 illustrates the three types of controls available:

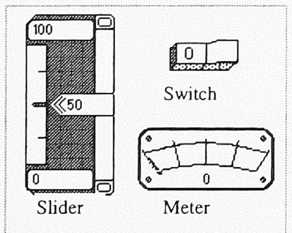


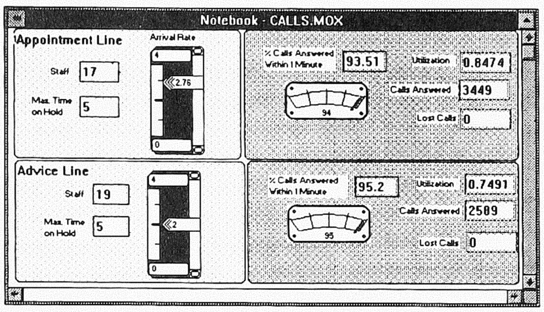
Figure 15: Extend Controls

The slider corresponds to the volume control on a radio. The model user can select the value by dragging the control on the slider to the appropriate value. Maximum and minimum values can be entered directly on the slider.

The Switch resembles a light switch with two states: on or off. Clicking on one side of the switch or the other changes its state.

The meter reports a level of a simulation variable. The dial on the meter will vary between the maximum and minimum values set for the meter proportionally to the value it represents.

Figure 16 shows controls added to the Notebook. In this example, the model user can increase or decrease an arrival rate by moving a slider and view the percentage of calls answered within 1 minute on a meter.


Notebook - CALLS.MOX

Appointment Line
Staff 17
Max. Time on Hold 5
Arrival Rate 4 / 2.76 / 0
% Calls Answered Within 1 Minute 93.51
Utilization 0.8474
Calls Answered 3449
Lost Calls 0
94

Advice Line
Staff 19
Max. Time on Hold 5
4 / 2 / 0
% Calls Answered Within 1 Minute 95.2
Utilization 0.7491
Calls Answered 2589
Lost Calls 0
95

Figure 16: Notebook With Controls Added

4.4 Pictures and Animation

Extend models are animated automatically when they are built. There are, however, specific animation blocks and functions which allow a modeler to go beyond the standard functionality. For example, pictures can be displayed to show that certain events have occurred, levels of important simulation variables can be displayed, and text can be shown which displays the model's status. Static drawings can also be added to enhance the visual impact of the model.

In Figure 17 the call center model has been enhanced with animation, pictures, and clones of dialog items to create an interface into the model itself. Clicking on one of the buttons will perform an action specific to this model, clicking on one of the icons will open the underlying hierarchical block for data entry and results, and the model animation will show the status of the call centers. This creates the effect of having a model which becomes its own custom user interface.

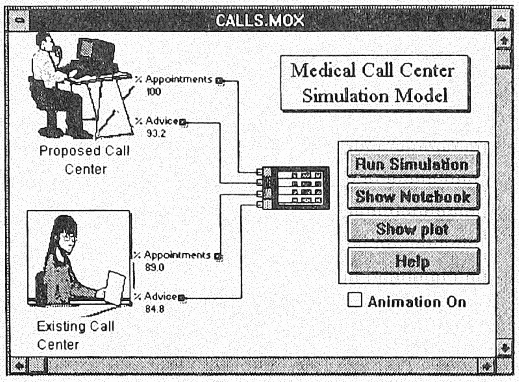
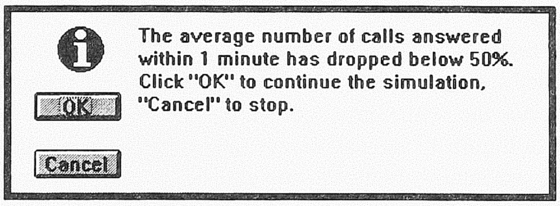


Figure 17: Model With Pictures and Animation

4.5 User Messages

Direct communication can be made with the model user through Extend's prompting messages. Prompting dialogs which request information or logical decisions, report an error, or play a system sound can be used to add a more dynamic feel to the simulation model.

Figure 18 illustrates a dialog which gives the user a choice between continuing a simulation run or stopping if a simulation metric is out of range for a feasible system design.

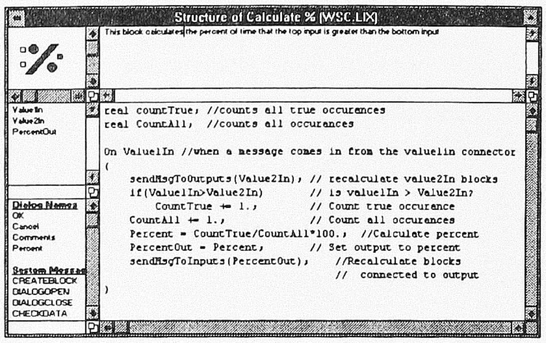


The average number of calls answered within 1 minute has dropped below 50%. Click "OK" to continue the simulation, "Cancel" to stop.

Figure 18: User Prompt

4.6 Scripting

Blocks in Extend are "soft coded". Their functionality and appearance can be modified or completely redefined by the modeler. This can be done either through hierarchy, as discussed earlier, or by programming the behavior of the block in the built-in ModL™ language. Both approaches have their advantages. Hierarchical blocks can generally be built more quickly and no programming expertise is required. Programmed blocks will have somewhat better performance and have the flexibility of a full featured programming language. An example of Extend's programming environment is shown in Figure 19. This is a block which was developed for a specific purpose: it calculates the percent of the time that the upper input is greater than the lower input. While this can be done without any programming, (see Figure 10), a few lines of ModL code in a block such as this can reduce overall model complexity.



This block calculates the percent of time that the top input is greater than the bottom input

```
real countTrue; //counts all true occurences
real CountAll;  //counts all occurances

On ValueIn //when a message comes in from the valuein connector
{
    sendMsgToOutputs(Value2In); // recalculate value2In blocks
    if(ValueIn>Value2In)        // is valueIn > Value2In?
        CountTrue += 1.;        // Count true occurance
    CountAll += 1.;             // Count all occurences
    Percent = CountTrue/CountAll*100.;  //Calculate percent
    PercentOut = Percent;       // Set output to percent
    sendMsgToInputs(PercentOut);    //Recalculate blocks
                                    //  connected to output
}
```

Figure 19: ModL Scripting Environment

ModL is a compiled language which is tightly integrated with the Extend environment. Selecting the "Open Block Structure" menu item accesses the block's script. When the block is saved, the script is automatically complied. The components of the block's structure (dialog, icon, connectors) are accessed by referring to the name of that component.

Dialog items such as block parameters, text, or check boxes are created graphically in the dialog builder (Figure 20) and are treated as variables by the ModL script. These can be changed or utilized in response to user or system events. For example, in the Input Random Number block, the name of the parameters of a distribution change when a different distribution is selected.

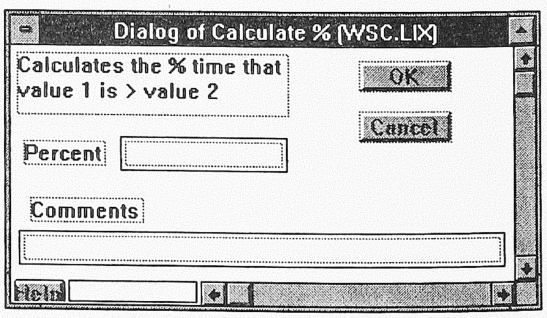

Dialog of Calculate % [WSC.LIX]
Calculates the % time that value 1 is > value 2
OK
Cancel
Percent
Comments
Help

Figure 20: Dialog Builder

Dialog items also generate block level events. These are known as messages. The script reacts to these with a message handler (function) for each dialog which is called automatically when a user selects a dialog item.

For example, since each block has an "OK" button, there is also an "OK" message handler which is called whenever the "OK" button is clicked. Typically, this message handler would check the consistency of the block data before the dialog is closed.

Connectors are also treated as script variables and message handlers. These are used during the simulation to pass information from one block to another.

Using scripting, additional animation options are available. For example, the icon of the block can be changed through animation objects. These can change the appearance of a block by modifying text, adding shapes or pictures, or displaying a level. An example of this is the Decision block which changes its icon based on the condition specified.

5 SUMMARY

The main goal for Extend is accessibility. This is achieved through low cost, a captivating model building environment, a native graphical user interface, and model development tools which make models for a specific area of application easy to build. Because Extend can be used without elaborate training or financial outlay, Extend is bringing the technology of simulation to more people than any other simulation product.

REFERENCES

Imagine That, Inc. 1995. *Extend Software Manual.* San Jose, CA.

Krahl, David. 1994. An Introduction To Extend, In *Proceedings of the 1994 Winter Simulation Conference,* ed. J. D. Tew, M. S. Manivannan, D. A. Sadowski, A. F. Seila, 538-545. IEEE Piscataway, NJ.

AUTHOR BIOGRAPHY

DAVID KRAHL is the Technical Coordinator for Imagine That, Inc. He is responsible for the technical support and block development for Extend. Mr. Krahl received a BS degree in 1986 in Industrial Engineering from the Rochester Institute of Technology and is nearing completion of a MS degree in Project and Systems Management from Golden Gate University. He has performed consulting, technical support, and development for a wide range of simulation products.