MODELING WITH EXTEND™

Dave Krahl

Imagine That, Inc.
6830 Via Del Oro, Suite 230
San Jose, CA 95119 U.S.A.

ABSTRACT

This document presents an overview of the Extend modeling environment. Extend is a general-purpose, graphically-oriented, discrete event and continuous simulation application with an integrated authoring environment and development system. Examples of applications will illustrate the product's broad appeal. Specific features will be demonstrated by examining a simple model of a single server, single queue system to which detail and enhancements will be added.

1 INTRODUCTION

In the last few years, there has been a convergence in the simulation industry. Simulation languages have become easier to use, often adding a user interface layer similar to that traditionally found in simulators. Simulators have added functionality to the point where their power and flexibility rival that of traditional languages. Because of this, it has become difficult for someone to determine the advantages of one product over the others based strictly on a feature comparison.

In this confusing marketplace Extend stands out as a product whose basic design provides a combination of unparalleled ease of use, power, and extensibility. It exists as:

- A stand-alone simulation tool which can be used to create complex discrete event and continuous models without programming.
- A simulation authoring package where model interfaces can be easily created, without programming, to enhance productivity and ease of use.
- A development environment for building customized models of unique types of systems. The programming environment allows the modeler to create a simulator for a specific industry.

2 WHAT MAKES EXTEND UNIQUE

Extend provides features and capabilities not found in other simulation software. This allows the modeler to concentrate on the modeling process and quickly produce a model that is easy to manipulate and communicate to others. These features include:

- Graphical logic making the model easier to understand and communicate.
- An unparalleled level of interactivity. Model parameters can be changed and results viewed during simulation execution. This is done through the graphical user interface; there is no need to enter a debugging mode or type in cryptic debugging commands.
- Superior hierarchy. Extend's hierarchy allows for animation and reuse, and can be any number of levels deep. This gives modelers an excellent tool for organizing large models and reusing model segments in other models.
- An innovative discrete event architecture which makes model building more intuitive.
- Graphical model building for discrete event, continuous, and combined models. Extend is usable across an organization's entire range of process simulation applications.

3 APPLICATIONS

Since Extend is a general purpose simulation program, it has been applied in a wide range of areas. The application examples here include supply chain, high speed manufacturing, and chemical processing.

3.1 Supply Chain

The U.S. Marine Corps is undergoing a revolution in the way they conduct combat operations. There are many ideas regarding how the tactical supply chain must change to

provide the necessary logistical support. The TLoaDS model has been developed to explore the ability of existing, evolutionary, or revolutionary methods and equipment for this challenging mission. This application (Figure 1) has been described as "warehouses that move", referring to the changing location of supply ships and depots.
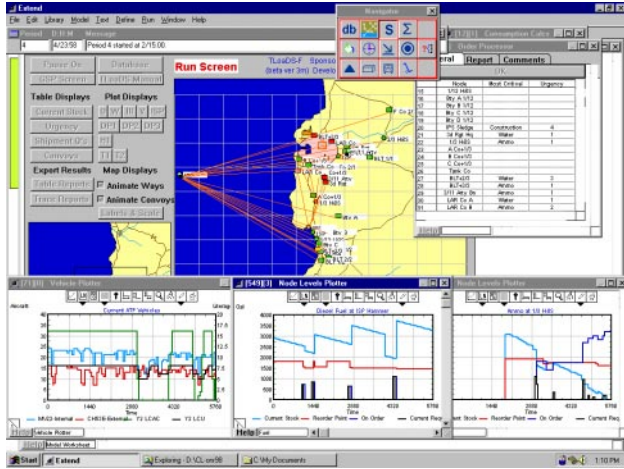

Figure 1: Supply Chain Model

3.2 High Speed Processing

This model of a packaging line takes bulk material (cookies) in pounds, packages the material in bags, packages bags into cartons, and then cartons into cases. The model is used to understand the dynamics and capacity of the overall system. This model (Figure 2) shows the effect of changing the speed of a piece of equipment, the failure and repair rates for that piece of equipment, the size of in-line storage, and even the mixing of different products.
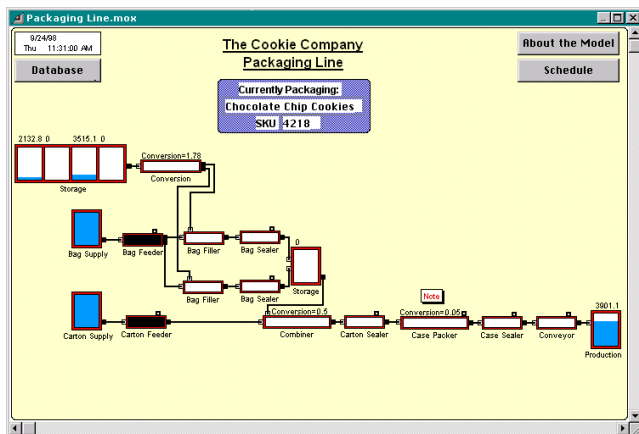

Figure 2: High Speed Processing

This application was built using Simulation Dynamics' "Industry" product that utilizes the Extend simulation engine.

3.3 Pulp and Paper Processing

The model represents an integrated pulp and paper facility in New Zealand's central North Island.

The model is a steady-state representation of an integrated pulp mill, recycle facility and sack kraft machine. It allows the engineers to perform "what ifs" to determine the optimum mix of products and grades for specific economic conditions. It is also capable of material and energy balances. Without a tool such as a model, these types of highly flexible operations, with hundreds of permutations and combinations, are difficult to optimize.

This mill now boasts bleached softwood kraft production costs in the bottom quartile worldwide.

This application was built in H. A. Simons IDEAS simulation software which is based on the Extend simulation engine.
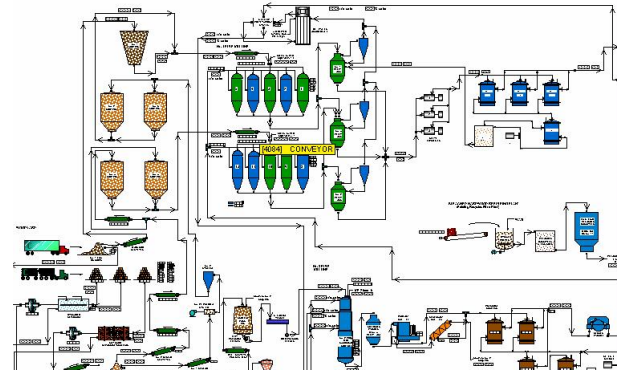

Figure 3: Pulp and Paper Processing

4   EXTEND'S MODELING ENVIRONMENT

Before looking into how Extend can be used to build models such as these, 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. The dialogs associated with each block define the behavior of the block as well as report on block status and results.

Blocks reside in libraries. Each library represents a grouping of blocks with similar characteristics such as Discrete Event, Plotters, Electronics, or Business Process Reengineering. Blocks are placed on the model worksheet by dragging them from the library window onto the worksheet. The flow is then established between the blocks.

There are two types of logical flows between the Extend blocks. The first type of flow is that of "items," which represent the objects that move through the system. Items can have attributes and priorities associated with them. Examples of items include parts, patients, or a packet of information. The second type of logical flow is "values,"

which will change over time during the simulation run. 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 that are the interface points of the block. Figure 4 shows the connector symbols for the value and item connectors.

Value Input   Item Input
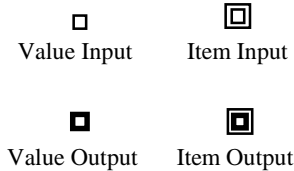
Value Output   Item Output

Figure 4: Value and Item Connectors

Connections are lines used to specify the logical flow from one connector to another. Double lines represent item connections and single lines represent value connections.

5   CAR WASH EXAMPLE

The following example is of a single server, single queue system. For the purpose of illustration, the model of a car wash will include one wash bay and one waiting line. The model for this car wash is shown in Figure 5.
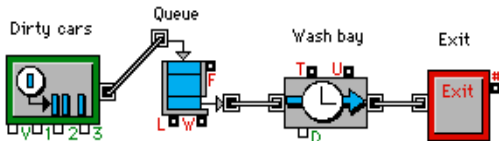
Figure 5: A Single Server Single Queue Model

The block on the far left is a Generator block and periodically creates items (in this case dirty cars). Following this is a Queue, FIFO block that holds the cars until requested by the next block. The Activity Delay has a limited capacity of one processing unit and delays the car for a fixed amount of time. This block represents the wash bay. The last block in the model is an Exit block that removes the cars from the system.

5.1 Random Processing Time

Suppose that the processing time for the wash bay is best represented by a specific random distribution. This can be modeled by connecting the output of an Input Random Number block to the delay connector (labeled "D") on the Activity Delay block as in Figure 6. Every time a car enters the wash bay, a new processing time is requested

from the Input Random Number block. For each request, the Input Random Number block generates a new processing time from the specific random distribution defined in the block's dialog.
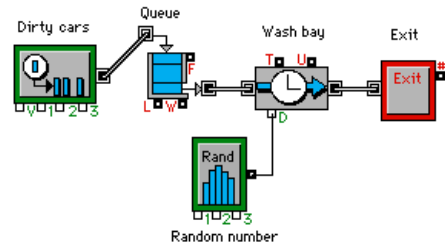
Figure 6: A Model with Random Process Times

5.2 Graphical Output

A Discrete Event Plotter graphically displays model metrics. In this example (Figure 7), the Plotter will graph the contents of the Queue, or the number of dirty cars waiting in line, over time. To accomplish this, connect the Queue's length value output connector (labeled "L") to the input connector of the Discrete Event Plotter as follows:
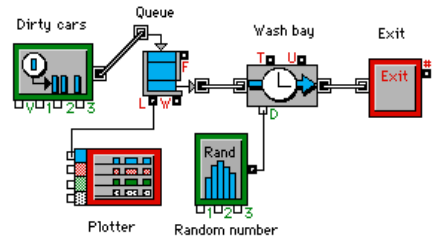
Figure 7: Discrete Event Plotter Added to Model

5.3 Attributes

Assume that the car wash offers two types of washes, basic and deluxe, and that the processing time is dependent upon the type of wash requested. To differentiate between the two different types of wash requests, attributes are used. The Set Attribute Block adds an attribute called "type" to each car and randomly sets the value of this attribute to 0 (basic) or 1 (deluxe) using another Input Random Number Block, as shown in Figure 8. As the dirty cars leave the queue and enter the wash bay, the Get Attribute block reads the "type". A Conversion Table block converts this number to a value representing the mean processing time for washes of that type. The mean value can then be fed into the Input Random Number Block that is already connected to the delay connector of the Activity Delay (Figure 8).
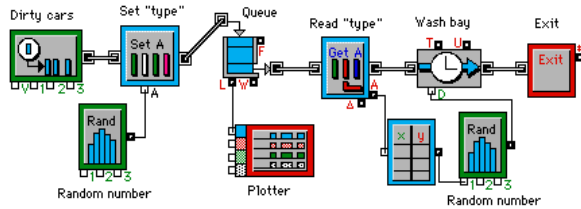
Figure 8: Setting "Type" Attribute

5.4 Resources

When the dirty cars are ready to be washed, they are driven through the wash by one of the car wash attendants. The attendants are modeled as resources by adding a Resource Pool block. Within this block the number of attendants are specified. The Queue, FIFO block is replaced with the Queue, Resource Pool block. Within the Queue, Resource Pool Block the type and number of resources required before the item may be released to the next block are specified. Therefore, dirty cars will enter the Queue, Resource Pool block and wait until an attendant is available. If an attendant is available and the wash can accept another car, the number of attendants in the Resource Pool block is decremented by one and the car is allowed to proceed into the wash bay. Upon exiting the wash bay, the attendant is no longer needed and may be released back to the Resource Pool with the Release Resource block as shown in Figure 9.
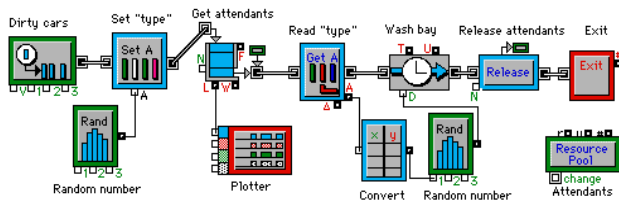

Figure 9: Modeling Resources

5.5 Activity Based Costing

Now that there is a basic model of the car wash, the model can be enhanced to calculate the average cost of washing each car. The following information is available:

- Attendants are paid $8.50 an hour.
- Cars use $1.25 in soap
- Electricity and water used by the wash bay cost $1.50 per minute.

The cost of the attendant is defined within the Resource Pool block and the cost of the soap, water and electricity in the Activity Delay block (Figure 10). As the model is run, the accumulated cost of each vehicle is automatically calculated and stored in an attribute. The Cost By Item Block can be added to read the cost attribute,

sort the items by an attribute (such as the "type" attribute) and report on the throughput, total cost and average cost by type of wash requested. The Cost Stats Block can also be added to report the total cost generated in each of the blocks, for example the total cost generated by the attendants (Resource Pool) or the wash bay (Activity Delay).
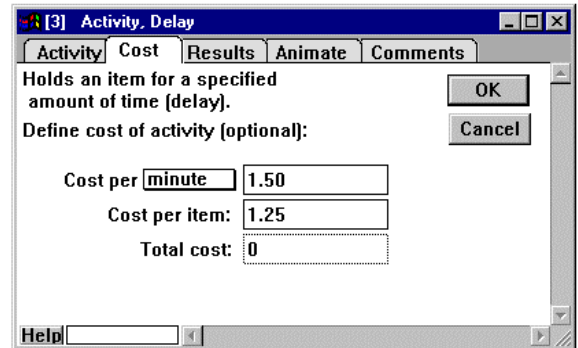

Figure 10: Cost Tab of Activity Delay Block

5.6 Interprocess Communication

The term interprocess communication (IPC) describes the act of two applications communicating and sharing data with one another. This feature allows the integration of external data and applications into Extend models. Automatic communication between Extend and other applications can take one of three forms:

- "Paste-Link" where the information is automatically updated between Extend and Excel.
- Blocks which utilize the IPC functions to communicate directly with other applications. The Extend IPC library is an example of this which allows models to send data to, get data from, and execute macros within Excel spreadsheets.
- Imbedded objects (ActiveX or OLE). These retain their native user interface, but reside with the Extend model worksheet or blocks. All of the features and interface of the imbedded application are directly available within Extend.

5.7 Model Results

Once the simulation run is complete, 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, Resource Pool block (Figure 11)opens a dialog showing the following information about the state of that block:
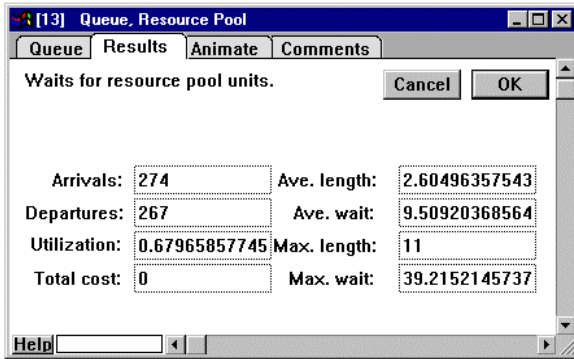
Figure 11: Dialog of Queue Resource Pool

The Plotter block (Figure 12) shows the number of items stored in the Queue, Resource Pool over time in both graphical and tabular format:
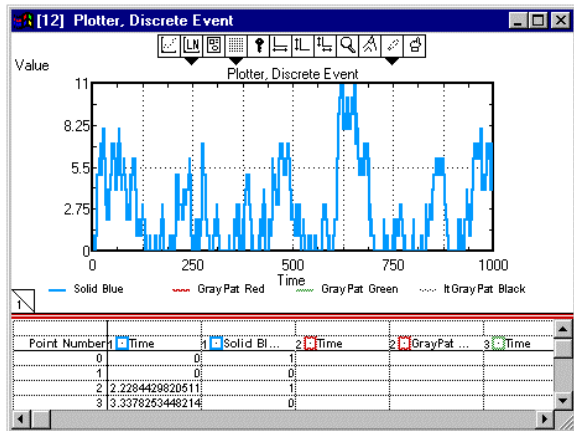

Figure 12: Plot of Queue Length

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 outside world through external device drivers.

5.8 Data Analysis

Extend offers a number of methods for analyzing both input and output data. These range from internal analysis features to built-in interfaces with other applications.

An interface is provided to distribution-fitting programs that aid users in selecting the appropriate statistical distributions based on empirical data collected in the field.

In addition, sensitivity analysis can be performed to determine how sensitive a system is to changes in specific input parameters. For example: to determine how sensitive the car wash is to changes in the inter-arrival time of dirty cars, sensitivity analysis can be performed on the inter-

arrival mean parameter of the Generator Block. By selecting the inter-arrival time dialog item and choosing Sensitize Parameter from the Edit menu, the change in the parameter value from one run to the next is defined. 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. By cycling through different inter-arrival times for the dirty cars and comparing the results from the different runs, an understanding of how sensitive the car wash is to the arrival rate of dirty cars can be obtained.

The Statistics library helps users to collect and analyze output data. Blocks from the Statistics library automatically gather data from the appropriate blocks and calculate confidence intervals.

6   CUSTOMIZING EXTEND

The above discussion illustrates the highly graphical and interactive nature of Extend. However, Extend can also take the shape of the modeled system. Interfaces, components, and graphics can be used which tailor the model to a specific application area.

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. The following sections will describe some of the tools provided in Extend that support customization.

6.1 Animation

Animation is a powerful presentation and debugging tool that can greatly increase model clarity. In Extend, animation icons moving from block to block represent the flow of items through the system. Users can choose from a number of icons provided with Extend or create their own in an external drawing package.

For example, adding animation to see cars traveling from block to block in the car wash model is done by selecting the appropriate icon in the Animate tab of the Generator block. From here, the picture representing all of the items created by the Generator can be defined. In addition, any block that the items pass through has the capability of changing the item's animation icon. For example every item exiting the Generator block can be represented with a picture of a dirty car. As the items pass through the wash bay, the Activity Delay block changes each item's animation picture to a clean car, thus providing visual cues of how the items are changing as they progress through the model.

In addition, custom animation can be added to display pictures and text, level indicators, pixel maps, and QuickTime movies.

An interface also exists to Wolverine Software's Proof Animation package. Activities, Resources, Generators, and Exit blocks each have specific functionality to send information to the Proof animation during simulation execution. Additional animation features in Proof can be accessed through the Proof library of blocks. This allows Extend modelers to easily utilize the industry's most sophisticated animation package.

6.2 Hierarchical Modeling

Hierarchy allows models to be subdivided into logical components or sub-models. A single descriptive icon can represent each sub-model. Double clicking on the hierarchical block will open a new window displaying the sub-model. This can greatly simplify the representation of a model and allow the user to hide and show model details as appropriate for the target audience.

In the car wash model (Figure 9), as detail was added the number of blocks increased. As a result, the representation of the model became slightly encumbered with model details.

Using hierarchy, the model can be represented by the system's most basic elements:

- the arrival of dirty cars
- the queue of dirty cars waiting for availability of the wash bay
- the wash bay
- the departure of clean cars

By selecting a group of blocks and choosing Make Selection Hierarchical from the Model menu, a section of the model is encapsulated within a hierarchical block. This block can be saved to a library to be used again in other models. The icon for the hierarchical block can be modified by using the built-in icon editor or by importing an existing picture. The number of hierarchical layers allowed in Extend is unlimited. Figure 13 shows the car wash model with hierarchical blocks representing some of the basic elements of the car wash. While the representation of the model is more intuitive and simple than Figure 9, all of the detail of the model can still be accessed by double clicking on any of the hierarchical blocks to display the underlying sub-model.
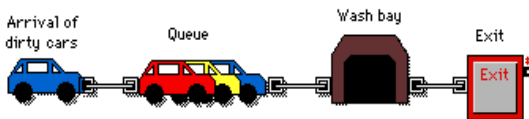

Figure 13: Car Wash Model with Hierarchical Blocks

6.3 Dialog Cloning and the Notebook

As noted in the previous section, input and output parameters associated with the model can be found in the dialogs of the appropriate blocks. While this provides an intuitive association between system metrics and the constructs used to model them, it can make searching for specific data more difficult. This can be especially cumbersome when working with large models containing many layers of hierarchy. An effective way of dealing with this is to use the notebook and cloning feature. With the notebook, a single custom interface can be created that consolidates critical parameters, results, and model control to a central location.

The notebook is a separate window associated with each model. Initially the notebook is a blank worksheet to which text, pictures, and clones can be added. Clones are direct links to dialog parameters and can be created by selecting the Cloning Tool from the tool bar and dragging a dialog parameter from a block dialog to the notebook or model worksheet. Once a clone is created, any changes to the clone are immediately reflected in the block and visa versa. Therefore it is no longer necessary to access the block's dialog to change an input parameter or view updated results. Creative use of the notebook can result in a simple yet effective interface for a large, complex model. Figure 14 shows the notebook for the car wash model as an illustration of how the notebook can be used to consolidate important parameters into one location.

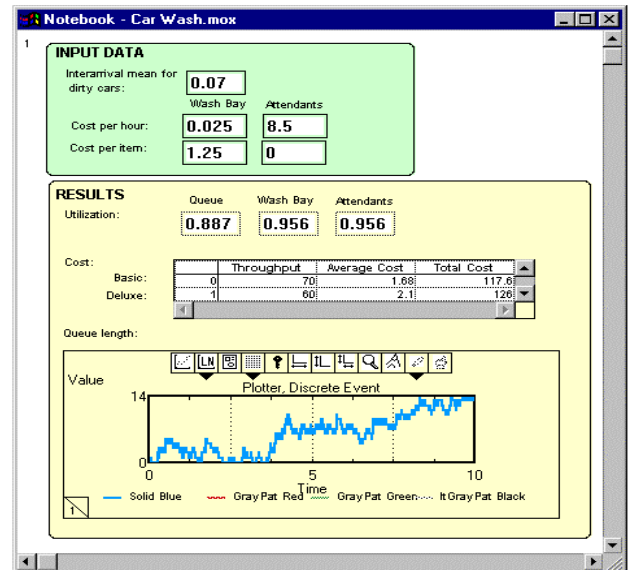Figure 14: Notebook for Car Wash Model

6.4 Block Development

The block development environment is one of Extend's most powerful features. While the majority of Extend users find the pre-built constructs sufficient for their needs, the

193

block development environment provides a way for users to expand the modeling capabilities to perform unusual or highly specialized tasks.

Extend's open source architecture allows access to the structure of any block that is shipped with Extend. By opening the structure of a block, its icon, dialog, help text, and programming code can be edited. The interface and functionality of any block can be modified or a new block created from scratch.

ModL is the powerful and flexible language used to define the behavior of the block. This language provides high-level functions and features while having a familiar look and feel for users with experience programming in C. In addition, external XCMDs and DLLs can be called from within ModL giving the option of programming in any language which supports this feature (such as C or Pascal).

This level of extensibility has prompted many users to develop libraries of custom blocks for specific industries. Users and third-party developers have created libraries for modeling many systems including neural networks, control systems, high-speed systems, chemical processes, silicon wafer fabrication, pulp and paper mills, and radio and microwave communication systems.

6.5 Scripting

Since Extend was created from the ground up as a graphical simulation tool, much of the process of defining a model was originally dependent on user interaction. For example, the user places blocks on the model worksheet, connects blocks together by drawing a connection between the two, and defines the block's behavior by double-clicking the block to open its dialog and filling the appropriate parameters. Scripting is a feature that allows models to be created and/or modified through a suite of ModL functions. With this functionality, users can create objects that can automatically build and modify models. With scripting, users can develop their own model building "wizards" or self-modifying models. Without having to rely on general-purpose "wizards" provided by the software vendor, users can develop "wizards" specific to their needs and can have complete control over the level of detail and accuracy resulting from automated model building.

Coupled with Extend's ability to communicate with other applications using interprocess communication (IPC), scripting provides an easy way to allow other applications to control every aspect of Extend including building the model, importing/exporting data, and running the simulation.

7   DISCRETE EVENT ARCHITECTURE

Extend utilizes a message based architecture which allows for more natural model building than is possible in other simulation tools. Messages are used to pass information to connected blocks about the state and actions of the block sending the messages. For example, as soon as a queue receives an item it will send a "wants" message to the downstream blocks to see if any of them can accept an item. The messaging system is applied to the item as well as the value connectors. Because of this, complex models and logic can be built without resorting to "dummy" resources, "logical" workstations, or programming that is specific to a given model.

7.1 Modeling Enhancements

A more advanced architecture makes modeling easier. In using a modern, message-based system Extend allows the modeler to focus on the modeling task rather than the simulation tool.

- Complex model segments can be built from simple, elemental blocks. These can then be saved in a library for use in other models. This type of model construction eliminates the need for "kitchen sink" modeling components in which every possible permutation must be programmed by the developer (making the interface unnecessarily complex) or requiring programming to enhance the capabilities of the modeling component.
- Easier rescheduling of events. Because blocks, not items (entities), are entered into the event calendar, changing an event time is a simple assignment. In other simulation tools, the event calendar must be searched for a specific item before the change can be made.
- Events do not have to be item based. Blocks can post themselves on the event calendar even if they do not handle items. This reduces the overhead in the model because items do not have to be generated or processed when an event occurs.
- Blocking through decisions. Extend automatically determines which path an item takes before it arrives at the decision point. The alternative to this would be adding "dummy" resources to prevent the item from moving forward if space was not available.

- Queues can be separated from activities. Any number of blocks which do not hold items (passing blocks) can be between a queue and the next activity.
- Conditions do not need to be "time checked". Messages are sent to connected blocks whenever a condition changes and the condition is evaluated immediately.
- Model logic is represented graphically and is visible as part of the model structure.

8   SUMMARY

As demonstrated above, Extend's design provides a superior simulation environment. By incorporating an intuitive interface and extensive authoring and development environment, along with more advanced simulation technology, Extend has succeeded in defining its position as the leader in simulation software.

REFERENCES

Hamber, Robert. 1999, CloaDS & TloaDS, *1999 Simulation Solutions Conference.* Institute of Industrial Engineers, Norcross, GA

Imagine That, Inc. 1998. *Extend Software Manual.* San Jose, CA.

Rivera, Jim. 1998. Modeling with Extend, *1998 Winter Simulation Conference Proceedings,* ed. D. J Medeiros, E. F. Johnson, J. S. Carson, M. S. Manivannan, 257-262. IEEE, Piscataway, NJ.

Wolverine Software Corporation. 1995. *Using Proof Animation.* Annandale, VA.

AUTHOR BIOGRAPHY

DAVID KRAHL, a simulation engineer with Imagine That, Inc., is responsible for block development and technical support. He received an MS in Project and Systems Management in 1996 from Golden Gate University and a BS in Industrial Engineering from the Rochester Institute of Technology in 1986. Mr. Krahl has worked extensively with a range of simulation programs and is actively involved in the simulation community.