Proceedings of the 1993 Winter Simulation Conference
G.W. Evans, M. Mollaghasemi, E.C. Russell, W.E. Biles (eds.)

EXTEND: A LIBRARY-BASED, HIERARCHICAL, MULTI-DOMAIN MODELING SYSTEM

Bob Diamond

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

ABSTRACT

Extend™ is a graphical, interactive, general purpose simulation program for both discrete event and continuous modeling. Extend incorporates the capabilities of both a data-driven simulator and a simulation language. Because it is both object-oriented and extensible, Extend offers a powerful yet easy-to-use simulation environment for modeling in any field or discipline.

This paper presents an overview of Extend and discusses its use for both continuous and discrete event modeling.

1 INTRODUCTION

Simulation software has traditionally been both expensive and difficult to learn. This has resulted in limited distribution and even more limited usage. As simulation consultant Andrew Siprelle of Siprelle Associates notes: "In my experience with other industrial engineers, I have found that very few actually use simulation. Those who do, use it rarely. Of the industrial engineers I knew at Alcoa, who are all graduates of Georgia Tech, University of Tennessee, Virginia Tech or Purdue, only 1 or 2 actually had the fortitude to write one or two models. I can think of perhaps 5 or 6 different *occasions* upon which an engineer had to run a model. These models were predominantly written in SLAM or SIMAN, and all were developed, without an exception I can think of, by consultants."

Historically, simulation programs fall into two distinct classes: languages and data-driven simulators. Simulation packages such as SIMAN and SLAM tend to be general purpose as to what they can simulate, but are specialized languages. Simulators such as ProModel and Witness allow users to build models using graphical elements which are pre-determined by the package (Law 1992). Simulation languages, while providing flexibility, have consistently proven to be a barrier to usage due to the high level of learning and implementation effort required (Thomasma 1988). On the other hand, simulators, while easier to use than a language, are typically domain-restricted, most often in the field of manufacturing (Fegan 1991). Those that are not domain-restricted are usually specific to discrete event or continuous modeling and are therefore not suited for cross-functional use. This limitation of simulators also restricts the acceptance of simulation, since within any company each department using simulation would have to learn to use a completely different package.

Recent attempts to overcome these limitations have resulted in hybrid products which combine the graphical element interface of the simulator with a simulation language (Pegden 1992). However, often the hybrid product is the result of enhancements which have been added to an existing architecture not originally designed to support them. In other cases, the existing architecture might not be robust enough for today's requirements. For example, the result might be a graphical interface on top of a language, but not fully integrated with it so the user has to program in the language and then build a separate interface. Or the graphical interface might be integrated with the language, but the language is limited in scope such that there is no ability to add primitives or higher level elements without going to an outside compiler.

This paper presents the Extend simulation application for personal computers, developed by Imagine That, Inc. Extend was created to provide a complete simulation environment, combining the ease-of-use of a simulator, the power of a built-in C-like language, and a library-based authoring environment for extensibility unrestricted by domain.

2 BACKGROUND

The author's own experiences and frustrations developing and maintaining simulation models written in FORTRAN for NASA's Project Apollo provided roots for the development and evolution of a modular simulation system that would integrate a graphical interface with a library of object-oriented elements (blocks) and a full-featured underlying language.

Extend version 1 (1988) was based on the concept of a general purpose simulation engine for both continuous and discrete event modeling. It featured an integrated graphical interface of object-oriented blocks and a built-in authoring environment for user-extensibility. This provided a medium

in which users could easily build models in any discipline as well as create their own libraries of blocks which did not have any fixed domain limitations. This has led to an ever enlarging base of libraries in a wide variety of fields (electronic engineering, paper production, biology, control systems engineering, manufacturing and industrial operations, chemistry, business process reengineering, and so forth).

Extend version 2 (1992) supports advanced modeling features such as unlimited hierarchy, real-time animation, interactive controls and data, sensitivity analysis, notebooks for model control and documentation, real time plotting, inter-block messaging, and full connectivity with other programs and the outside environment through import/export, file and serial I/O, and driver functions. To accommodate non-C programming, Extend also supports commands for calling external code resources written in other languages.

Extend's modularity provides re-usability as well as the ability to focus on any domain. The integrated graphical environment means that a modeler can use the model both to do the work and to communicate the results. The internal language means the user does not have to choose between the limitation of the primitives and blocks that come with the package and the aggravation of working with an outside compiler.

3 EXTEND'S MODELING FRAMEWORK

The key to Extend is its library-based iconic-block modeling concept. Extend allows the user to create models from building blocks stored in libraries. Extend blocks graphically represent their real-world counterparts and also serve as the main method of entering and reporting model parameters.

Blocks can be as simple as a single function or as complex as many thousands of lines of code. Blocks contain the behavior and user interface necessary to enter data, automatically connect with the rest of the model, and process data during a simulation run. As discussed in section 7, blocks have an icon, a dialog, script, and connectors. Icons can be customized to represent exactly what is being modeled, for example an element, a person, or a process.

The Extend package includes several libraries of blocks for general purpose continuous and discrete event modeling. There are also libraries of blocks for domain-specific modeling, such as for manufacturing and engineering.

Because Extend blocks are written using the same built-in C compiler and dialog editor that Imagine That! uses to develop its pre-built libraries, the user can modify a pre-existing block and add detailed functionality to it. For example, the user can add matrix fluid dynamics calculations to an activity or have a block call an external modem to send and receive data. Users can also create new blocks with custom icons, dialogs, and scripted behavior without having to go outside the Extend program.

3.1 Overview of the Model Structure

Extend models consist of a worksheet with blocks connected together showing the flow of items or values. Inputs and outputs are handled by the blocks on the worksheet.

3.1.1 Message Sending Architecture Runs the Simulation

Extend's simulation engine uses an object-oriented methodology to send messages to blocks. These messages initialize the blocks, then run the simulation. Some of the messages are:

- CHECKDATA - allows the block to check data validity. Warns the user if data is not valid.
- STEPSIZE - allows the block to arbitrate a stepsize between different blocks in continuous simulations.
- INITSIM - allows the block to initialize any variables or arrays that are necessary for the simulation.
- SIMULATE - sent to all the blocks periodically during a continuous simulation to recalculate their results. Used to process events during a discrete event simulation.
- ENDSIM - sent to all blocks at the end of a simulation to dispose of unwanted data.

There are other messages which are sent when a user has clicked on a button or typed in some new data. There are also user defined messages which allow custom behavior for each block, or for the model as a whole when a particular block is present in the model. For example, Discrete Event blocks pass messages back and forth with interconnected blocks to show their status, item availability, and capacity.

3.1.2 Method of Simulation Changed by Blocks

This architecture allows the blocks to choose the method of simulation. For example, if blocks from the Discrete Event library are present, the model uses a discrete event based architecture (event queue) when it is run. Mixed mode (discrete and continuous) simulation occurs when continuous blocks from the Generic library are used with discrete event blocks in the model. If no discrete event blocks are present, the model works in continuous (time advance) mode.

The user can even define new libraries which change the way simulations run by having blocks send messages to other blocks, either through connections or globally using an index. This also means that the independent variable does not have to be time, but can be any desired variable.

3.2 Discrete Event Models

In Extend's discrete event models, *items* change state only as *events* occur (see section 3.2.1). It is the information about model items (for example, their quantity and condition) which represents the state of the model at any time. Simulated time advances from one event to the next and the amount of time between events is usually not periodic.

Extend's Discrete Event and Manufacturing libraries contain general purpose discrete event blocks such as statistical distribution generators, queues, delays, routing, and batching, as well as domain-specific blocks such as machines and conveyors.

3.2.1 Items and Events

An item is whatever is being processed in the model. Items can be parts, people, data, or any other real-world entity. An event is anything that occurs in a model, such as a machine becoming available or a switch being turned on.

3.2.2 Attributes and Priorities

Attributes, priorities, and numeric values are used to make items unique so they can be sorted, routed, and tracked throughout the model. Attributes are characteristics of an item, such as its quality, age, or size. Attributes can have any name and can have real or integer values. Items can also have a numeric value greater than one so that they are effectively cloned. Items can be assigned many attributes, a value, and a priority. These properties can then be used for any model purpose, such as to determine cost, define activity durations, or specify downtimes.

3.2.3 Batching and Unbatching

Batching, or combining of many unique items into one new item for processing, is fully supported. For example, to specify that a software package requires 5 disks *and* 2 manuals *and* 1 installation sheet, the user only needs to enter the numbers 5, 2, and 1 in the Batch block. Uniqueness of item attributes and priorities is maintained even if an item is subsequently unbatched into its original components. By selecting a checkbox, strategic resource items (such as a scarce part or a manager) will not be called upon until the other resources have arrived.

3.2.4 Pull or Push System

By default, discrete event blocks are set up as a "Pull" system. For example, a queue holds items until an activity (server) is free to pull items from the queue. Blocks also contain a set of controls to override this and set up a push system, if desired.

3.2.5 Values in Discrete Event Models

Most information about the items in a model is calculated automatically by the blocks. For example, queue blocks present the average wait, maximum queue length, and other values related to the items which have passed through that queue. Additional information, such as the total of all the items which have passed through all the queues, can be calculated explicitly, using blocks which add, subtract, etc.

3.3 Continuous Models

In continuous models (also called process or flow models), values change based directly on changes in time (or based on some other independent variable such as temperature). These values represent the state of the model at any particular time. As the model runs, simulated time advances from one time step to the next. The time between steps is usually periodic and model values usually change at each step.

After the initialization messages and during the main part of the simulation run, blocks are sent SIMULATE messages (see 3.1.1) by the engine. This message is used by each block to recalculate its output values based on its input values.

Extend's Generic and Engineering libraries contain general purpose continuous blocks such as decisions, accumulators, and transfer functions, as well as domain-specific blocks such as a low pass filter or phase-locked loop.

3.3.1 Adjusting Stepsize (dt)

Stepsize in continuous modeling can be determined explicitly or by algorithm. Users can specify a stepsize or delta time for the simulation by entering the number in a dialog. Stepsize can also be automatically calculated and arbitrated between different blocks using the STEPSIZE message in a block's code. This eliminates human error when determining stepsize in models where delta time is a critical factor.

3.3.2 Passing Arrays or Matrices

Blocks can pass values, arrays of values, or arrays of arrays (structures) from block to block or globally in the model.

3.4 Mixed Mode Models

Extend supports mixed mode simulation since the discrete event architecture does not care if the time variable is changed by continuous blocks. Events occur at discrete times, but extra events caused by continuous blocks are ignored by discrete event blocks. Continuous blocks are not affected by discrete events, as they treat them as part of the time-advancing simulation process.

4 BUILDING AN EXTEND MODEL

The most important technique to remember when building a model is to start small, test the model against known or hypothetical data, refine the model, validate it, refine it again, and so forth. Extend provides numerous tools for testing and debugging models, from readout and timer blocks that can be inserted at any point in the model, to the ability to step through a simulation while the model displays all messages.

The steps in building an Extend model are:

- Launch Extend (opens a model worksheet)
- Open libraries of blocks
- Select blocks from a library
- Move the blocks to the desired position
- Connect the block's connectors with the mouse
- Add data to the block's dialog, if required
- Determine what data you want to see graphed and connect to one or more plotter blocks
- Specify model timing in the Simulation Setup dialog
- Run the simulation

4.1 Opening a Library and Selecting a Block

Using the mouse, drag a block and place it on the model worksheet.

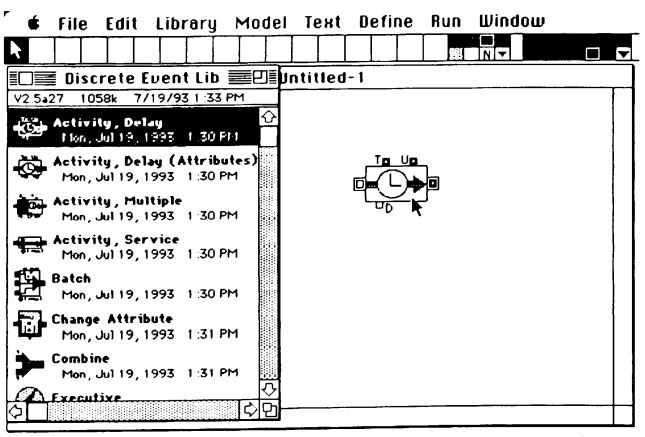


Fig. 1 - Placing an Activity Block on the Model

The "Activity Delay" block on the model worksheet above causes inputs to be delayed for the specified time, which may be constant or random. It also gives information such as number of items entering and leaving, utilization, etc.

Extend has extensive libraries of blocks for quick model building. Users can also build their own custom blocks and save them in libraries for reuse in other models.

4.2 Connecting the Blocks

Connecting blocks tells the model the path the data will take. To connect blocks, use the mouse to drag a connection line from the output of one block to the input of another.

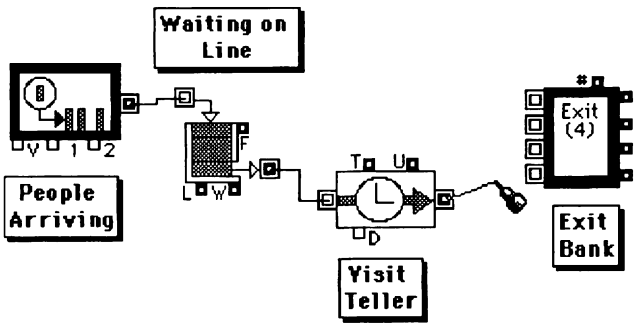


Fig. 2 - Connecting the Activity Block to the Exit Block

Extend blocks have pre-defined input and output ports, which are the small connector boxes on the perimeter of the blocks. Pre-defined outputs mean Extend automatically handles the flow of data, speeding up model building. The model above shows a connection being made from the output connector of the Activity block (which represents a bank teller in this model) to the input connector of an Exit block. Extend does not allow the user to connect incompatible connectors.

4.3 Entering and Changing Data

There are many ways to get data into an Extend model. Extend supports copy/paste and importing and exporting of text files either at the beginning of the simulation or while the simulation runs. However, the primary and most intuitive place for entering model parameters is directly in the dialog of the blocks.

To access a block's dialog to enter data, double-click on the block. For example, the Generator block (the block at the left in the model in Figure 2) requires the user to enter the desired arrival time distribution. Its dialog is seen in Figure 3.

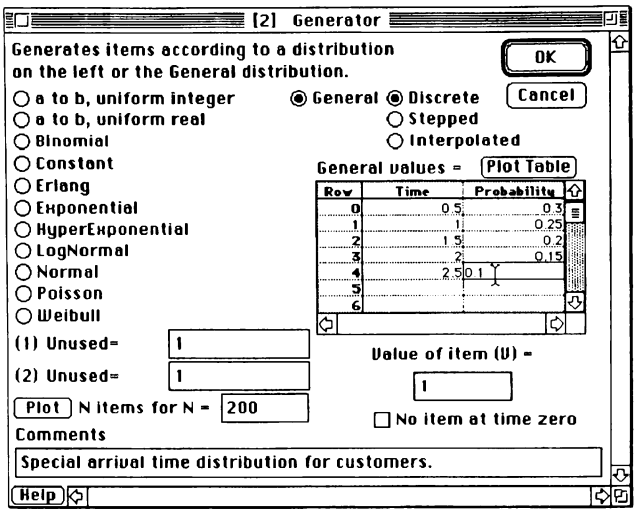


Fig. 3 - Specifying Custom Arrival Distribution of People

In this case, a table of arrival times and their associated probabilities is entered as a General distribution. Users can also select a distribution by clicking on a button and entering the associated arguments, such as the mean or the standard deviation.

Block dialogs can have radio buttons, parameters, checkboxes, data tables, slide controls, meters, and switches. Dialogs can also change appearance depending on what button the user clicks. Extend includes a built-in dialog editor, so users who build blocks can build custom dialogs.

4.4 Connecting Plotters

Plotter blocks, which can be placed anywhere in the model, provide real-time graphs as well as tabular data.

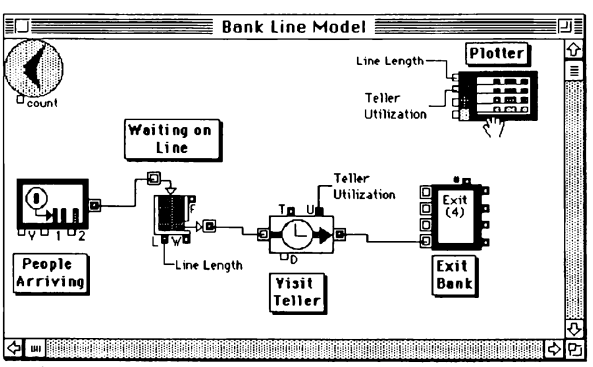


Fig. 4 - Adding Plotter to Plot Results over Time

By connecting a block's output connector to a plotter, a user can specify what gets plotted. For example, in the model above, information about the length of the queue and the utilization of the teller is plotted. Instead of connecting by drawing a connection line, Extend's *named connection* feature is used. Named connections act as variables on the model worksheet, eliminating the need to draw connection lines from block to block for every connection.

4.5 Setting Simulation Times and Running the Model

Models run for the length of time entered in the Simulation Setup dialog. Multiple runs for sensitivity analysis or Monte Carlo simulations may also be specified by choosing a random seed for repeatable distributions. The plot of the Bank Line model in Figure 4 shows the queue length and utilization of the teller as the simulation runs:

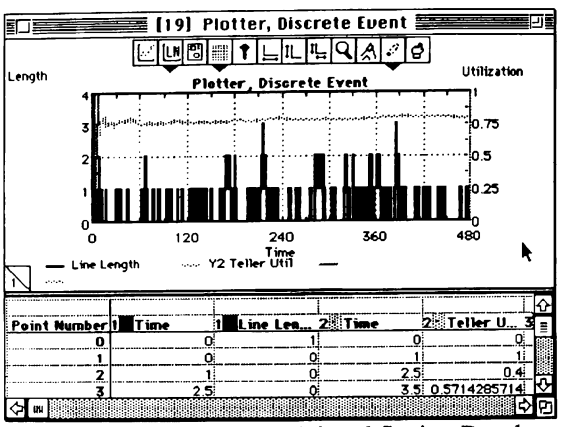


Fig. 5 - Running the Model and Seeing Results

Users can interactively change model parameters during the run to see any effects on intermediate results. They can also set up automatic sensitivity analyses of a model parameter, varying the value based on a range of values, a distribution, or an ad hoc entry read from a file. Menu commands can be used to turn animation and reporting features on and off as the simulation runs, depending on the user's requirements.

5 CAUSTIC MIXING - A CONTINUOUS MODEL

Extend's continuous modeling capabilities are illustrated by a Caustic Mixing model. This models a tank used for mixing water and sodium hydroxide. The tank level is supposed to be maintained at 50% in spite of a changing outflow. In addition, the tank has to regulate how much water and how much sodium hydroxide (at SG 1.50) is being drawn in so that the specific gravity of the solution is maintained at about SG 1.12.

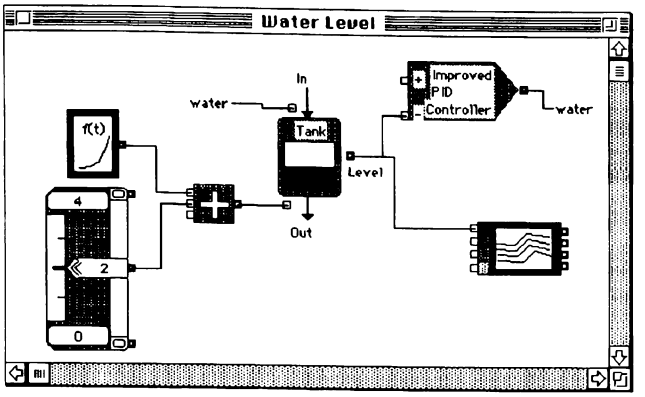


Fig. 6 - Water Tank with PID Controller

The model above represents the water tank maintaining its 50% level with changing outflow. The water level sensor output is processed using a PID controller with a fixed set point. This amplifies the error signal (level-setpoint), feeding it back to the water intake valve, thereby controlling its on/off state.

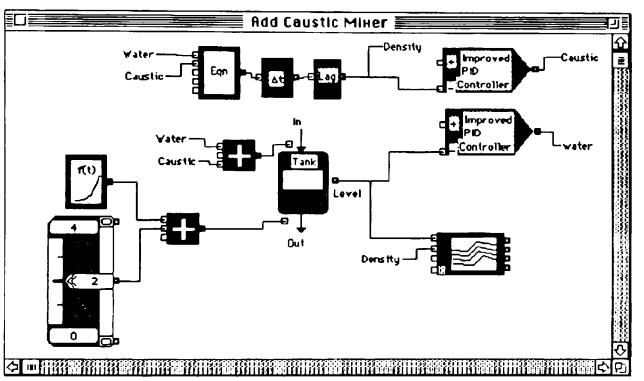


Fig. 7 - Add Mixing Density Controller

At the top of this model is a series of blocks representing the behavior of the sodium hydroxide input system. An Equation block is used to monitor how much water and sodium hydroxide (caustic solution) is going into the tank, calculating the current density of the mixture. The output of this calculation is then delayed and "low pass" filtered by the Lag block. The PID controller represents the actual caustic valve going into the tank, and adjusts the amount of sodium hydroxide to maintain the desired specific gravity. Both PID controllers have to be set to compensate for the feedback loop characteristics, keeping the system stable and responsive to changing output flows.

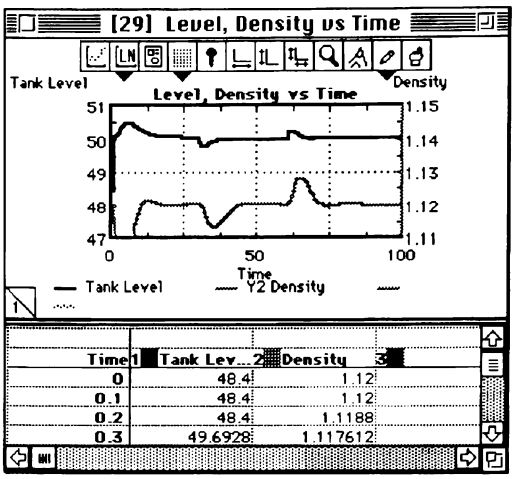


Fig. 8 - Plotting the Results of Transient

The output of the model shows the level of the mixture in the tank and its density. Because this is a continuous model, the time steps are equally spaced.

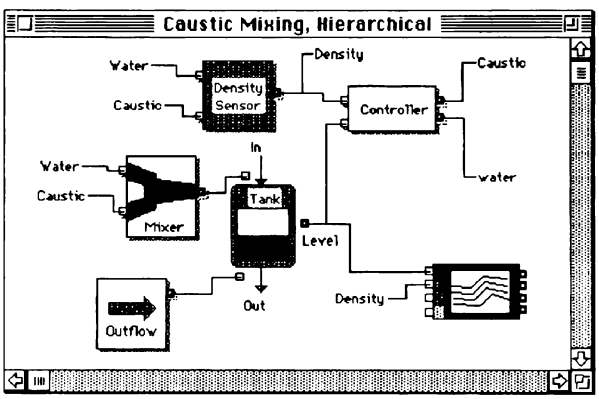


Fig. 9 - Making the Model Hierarchical

Figure 9 shows how hierarchical blocks can be used to condense the model shown in Figure 7. A menu command is used to select a group of blocks to make them one hierarchical block. Hierarchical blocks facilitate presentation and user navigation within the model.

6 PACKING LINE - A DISCRETE EVENT MODEL

The discrete event Packing Line model (Fig. 10) represents the packaging, movement, and transportation of consumer products in response to orders from the central warehouse.

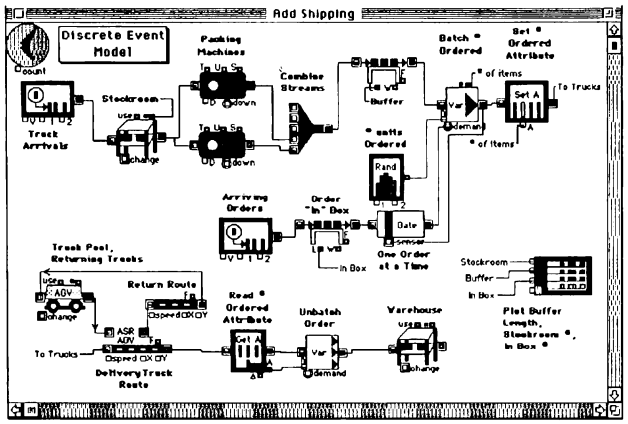
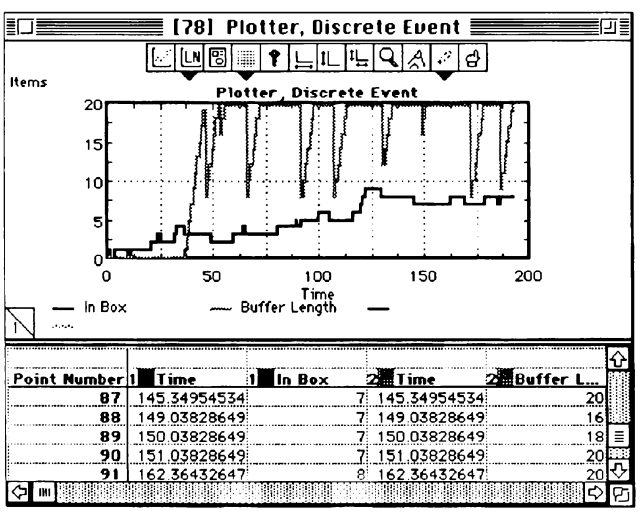


Fig. 10 - Packing Line Model with Attributes, AGVs

Figure 10 shows product arriving by truck at a statistically bound rate. The items are pulled into a Stock block which accumulates them. The machine blocks pull items out of stock and pack them one by one, taking a specified time for each.

A buffer with a limited size holds the finished products until receipt of an order for movement to the warehouse. The orders arrive at random times and are for between 5 and 15 units. If trucks are available in the trucking (AGV) pool, the finished orders are shipped to another warehouse where they are stored. The trucks then return via another route to the truck pool.



Point Number	Time	In Box	Time	Buffer L...
87	145.34954534	7	145.34954534	20
88	149.03828649	7	149.03828649	16
89	150.03828649	7	150.03828649	18
90	151.03828649	7	151.03828649	20
91	162.36432647	8	162.36432647	20

Fig. 11 - Plot showing Order Backlog, Buffer Length

The plot of the simulation in Figure 10 shows how many orders arrived (In box) and the backlog of orders waiting for trucks (Buffer Length). Notice that time is discrete (not equally spaced) between events.

7 CREATING A NEW EXTEND BLOCK

Every Extend block has 5 parts:

- Dialog - the information you see when you double-click on the block's icon to change parameters. Extend has a built-in dialog editor. You specify all the buttons, text, and entry boxes that go into the dialog, determining the dialog's size.
- Script - Extend has a built-in, compiled, C-like language called ModL. It is the ModL program that makes a block work. The program reads information from the connectors, the dialog, and the model environment and produces output that can be used by other blocks. ModL contains over 250 built-in functions (math, matrix, I/O, arrays, statistics, distributions, and so forth), as well as user-defined functions and random number generation. Users can use XCMDs to call external code in any language.
- Icon - the block's picture you see in the model. You can draw the icon with Extend's drawing tools or copy clip art and paste it in. You can also add custom animation to the icon using Animation functions.
- Connectors - the input and output connectors on the block. These appear in the icon and transmit information to and from the script. There are 4 distinct types of connectors.
- Help text - the text that appears when you click the Help button in the block's dialog. You can enter custom help for the blocks you build.

These parts are interconnected. For example, the script reads information from the connectors and dialog, the help text is displayed through the dialog, and so on.

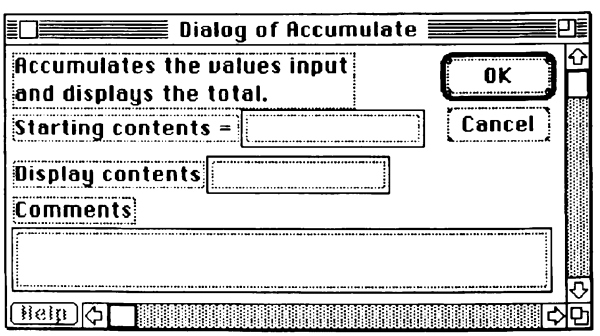


Fig. 12 - Dialog Design Window

Figure 12 is an existing block's dialog window. There are 11 types of dialog items including buttons, checkboxes, data tables, parameters, and controls. Dialog items have names which are used by the script to interact with the dialog.

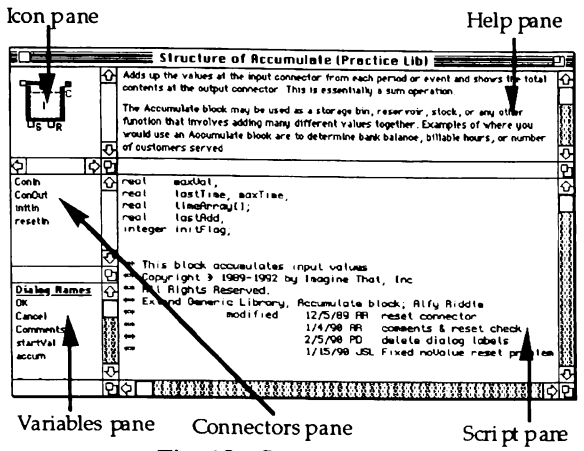


Fig. 13 - Structure Window

The structure window (Figure 13) is where you draw (or paste) the icon, add connectors, enter help text, type the script, and compile the block.

8 EXTEND FEATURES

8.1 Extend is not Domain-Limited

Extend can be used to model any system or process in business, science, or engineering. Both continuous and discrete event modeling are supported. New libraries can be user-developed for specific disciplines.

8.2 Different Levels of Use

- Users unfamiliar with simulation can use models assembled by others. They can enter data in block dialogs, run the simulation, and observe the results.
- Entry-level users can build models using the extensive libraries of pre-built blocks included with Extend, or by using blocks supplied by third parties such as advanced users at their company.
- Intermediate-level users can create custom effects by entering equations or formulae into the Equation block or by using hierarchy to combine the function of several blocks into one.
- Advanced users can create new blocks with custom dialogs and icons using the built-in C-like language, dialog editor, and icon drawing tools.

8.3 Integrated Simulator with Language

- Iconic modeling means the picture is the model. There is no need to build a separate graphical interface. You can also have custom icons so that each part of the model looks exactly like what is being modeled.
- Extend has extensive libraries of pre-built blocks for quick model building. Blocks can be modified or new blocks can be created using Extend's built-in language and authoring environment.

8.4 Advanced Features

- Hierarchy – Hierarchical blocks are actually nested models or submodels. Select a group of blocks and have one block represent them. Or open a new hierarchical block and build a model from the bottom level up. There are unlimited levels of hierarchy and hierarchical blocks can be saved in libraries for use in other models.
- Animation – Both built-in and scriptable, animation is turned on or off through menu commands. Animation supports changing colors, levels, text, and shapes, and showing pictures and movies.
- Sensitivity Analysis – A controlled way to view parametric effects over multiple runs of a model.
- Cloned (copied) dialog parameters – Drag a copy of any dialog parameter to any place in the model or notebook. The copy acts just like the original.
- Notebooks – Control model parameters and document the model from one location.
- Controls – Interactively control a model with sliders and switches. Get readouts from meters.
- Drawing and text tools – Add graphic objects and text to models. Paste background layout pictures.
- Interapplication communication allows direct linking from Extend to spreadsheets as the simulation runs.

8.5 Additional Packages

In addition to the basic Extend package, there are two bundled companion packages distributed by Imagine That, Inc.

- Extend+Manufacturing™ is used for discrete operations research, industrial engineering, or manufacturing systems analysis. This package combines the Extend package with an additional manual, library, and diskette of examples. It is used to model paper flow, material handling, queueing systems, service industries, networks, manufacturing plants, information processing, distribution systems, transportation, etc.
- Extend+BPR™ is developed for business process reengineering. The models and documentation support redefining business processes to maximize efficiency and quality. This package is used by managers and staff to analyze a new process for technology insertion, to model organizational changes, and to provide metrics for strategic planning.

Several third parties have also developed libraries for control systems engineering, environmental decision making, optics and spectroscopy, paper production, etc.

9 CONCLUSION

Extend was developed to provide professional-level computer simulation which is accessible on the desktop for any level of user in any discipline. The main purpose of the product is to allow users to try out ideas quickly, because if it takes too long to explore ideas, users will learn not to. For example, it should take less than 15 seconds to make any change and re-run an Extend model.

Extend's graphical interface and object-oriented nature provide for rapid model development. The built-in authoring environment and C-like language add flexibility and power.

REFERENCES

Fegan, J.M., G.M. Lane, and P. J. Nolan. 1991. Introduction to Simulation Using Intelligent Simulation Interface (ISI). In *Proceedings of the 1991 Winter Simulation Conference*, SCS, San Diego, CA.

Law, A. M. and M. G. McComas. 1992. How to Select Simulation Software for Manufacturing Applications. *Industrial Engineering*, July 1992:29-35.

Pegden, C. D., and D. A. Davis. 1992. Arena™: A SIMAN/CINEMA-Based Hierarchical Modeling System. In *Proceedings of the 1992 Winter Simulation Conference*, SCS, San Diego, CA.

Thomasma, T. and O. Ulgen. 1988. Hierarchical, Modulation Simulation Modeling in Icon-based Simulation Program Generators for Manufacturing. In *Proceedings of the 1988 Winter Simulation Conference*, SCS, San Diego, CA.

AUTHOR BIOGRAPHY

BOB DIAMOND is the founder and president of Imagine That, Inc. He has been creating simulation models and products for almost 3 decades. In the past he developed the Rocket-Drop simulations for NASA Project Apollo and, more recently, has developed the DesignScope and Extend commercial simulation applications.