Proceedings of the 2008 Winter Simulation Conference
S. J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson, J. W. Fowler eds.

EXTENDSIM 7

David Krahl

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

ABSTRACT

ExtendSim 7 is a proven simulation environment capable of modeling a wide range of systems. ExtendSim 7 is used to model continuous, discrete event, discrete rate, and agent based systems. ExtendSim's design facilitates every phase of the simulation project, from creating, validating, and verifying the model, to the construction of a user interface that allows others to analyze the system. Simulation tool developers can use ExtendSim's built-in, compiled language, ModL, to create reusable modeling components. All of this is done within a single, self-contained software program that does not require external interfaces, compilers, or code generators. This paper will introduce ExtendSim 7, demonstrate its advanced technology and features, and explain sample simulation applications.

1 INTRODUCTION

At its heart, creating a simulation program is relatively easy. All that is needed is a clock, events list, random number generator, and some functions to create, route, store, schedule, and dispose items as they move through the system. The challenging part of developing simulation software is developing a system that can easily, reliably, and intuitively model a wide range of systems.

In ExtendSim 7, Imagine That has created a simulation system that allows modelers to build a wide variety of models efficiently and elegantly. This unique combination of ease-of-use and modeling power are accomplished through a variety of modeling technologies and techniques:

- Messaging architecture that seamlessly updates the status of related simulation objects.
- Compile-as-needed development environment that has the speed of a compiler and the convenience of an interpreter.
- Integrated relational database for data centralization and organization.
- Hierarchical modeling for model presentation and component re-use.
- Encapsulated block data and behavior.
- Interactive model execution.
- Linear programming based discrete-rate simulation engine.

These features allow ExtendSim to be applied to the most challenging simulation problems.

ExtendSim 7 represents the most advanced simulation system available. Over the last 20 years, Imagine That has been quietly innovating, introducing technologies such as a message-based simulation engine, integrated relational database, discrete rate simulation, hierarchical model structure, and a compile-as-needed development environment. ExtendSim continues to be enhanced by Imagine That's stable and dedicated team of software developers.

2 EXTENDSIM HISTORY

ExtendSim is the latest iteration of the Extend family of simulation products (Krahl 2001). Extend was originally released in 1988 and pioneered many concepts and features found in modern simulation environments. Specifically Extend was the first simulation program that included the following:

- Specifically designed for a graphical user environment.
- Incorporates a development environment for building elemental simulation constructs (blocks).
- Provides a hierarchical modeling structure.
- Includes a message-based discrete event architecture.
- Graphical connections for values as well as items.
- Built-in linear programming solver to more accurately model rate-based systems.

These are some of the most copied features in the simulation software industry and ExtendSim's trademark innovation is seen everywhere. The innovation continues, with new "firsts" coming with each release.

978-1-4244-2708-6/08/$25.00 ©2008 IEEE

3 BASIC MODELING CONCEPTS

Before looking into ExtendSim's technology, it is helpful to understand the ExtendSim modeling environment (Imagine That Inc. 2007).

ExtendSim models are constructed with library-based iconic blocks. Each block describes a calculation or a step in a process. Block dialogs are the basic mechanism for entering model data and reporting block results. Blocks reside in libraries where each library represents a grouping of blocks with similar characteristics. 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.

In a typical discrete event model, blocks from the Item and Value libraries will make up most of the model. A typical block from the Value library would be a Lookup Table block:

Figure 1: Lookup Table

Blocks from the Value library perform mathematical calculations and have value connectors for their inputs and outputs. Figure 1 shows a Lookup Table block that performs a lookup, converting one value input (□) to any number of outputs (▣) based on its internal table. The pull down inverted triangle (▼) on the output connector indicates that it can be expanded to report multiple results.

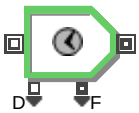

Figure 2: Activity

Blocks from the Item library operate on items (entities). Figure 2 shows an Activity block from the Item library. The Activity has an item input (▣) and an item output (▣) to define the flow of the item through the block. It also has value inputs (□) and outputs (□) for the parameters and results of the activity. Activities have a limited or infinite capacity, delay items for a fixed or random amount of time, and may be shut down, interrupted, or suspended. This particular Activity has only some of its connectors visible. Other connectors become available when the modeler selects the appropriate options in the block or if they pull down on a connector's inverted triangle.

Table 1: ExtendSim's standard libraries

Library	Description
Item	Item processing blocks
Value	Value processing blocks
Plotter	Plots and charts
Animation 2D 3D	Animation for 2D and 3D environments
Rate	High-speed, high-volume, or rate-based processes
Utilities	Blocks that perform utility functions
Electronics	Electronic components

Table 1 shows the main libraries available with ExtendSim from Imagine That. Other example libraries that illustrate modeling and programming techniques are included as well.

3.1 Building a model

Building a model involves dragging a block from the library into the model window, connecting it to the other blocks in the model, and then setting the block's dialog parameters. This connection may be made by using the block's connectors or it may be a logical connection such as choosing the name of a resource in a Queue block.

Consider the following example. A 4 CPU server is responsible for processing real-time tasks. Tasks arrive with different priorities. A higher priority task will interrupt a lower priority task that is currently in process. The interrupted tasks are returned to the processing queue. In addition, tasks that wait more than a few seconds will time out (renege).

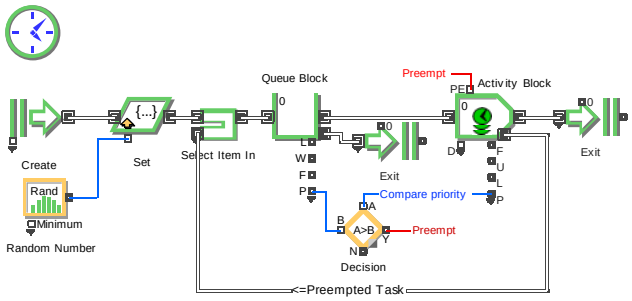

Figure 3: Server with preemption and reneging

The model in Figure 3 illustrates a number of ExtendSim features:

- The use of the connection lines to transfer items (tasks) as well as information (priorities) through the model.
- The "Preempt" named connection establishes a connection through two identical text objects as if they were physically attached. This helps to organize and document the model.

- The logic for preemption can be clearly seen. The priority of the item in the Activity is compared with the Priority of the item in the Queue. If the item in the Queue has a better priority, the item in the Activity is preempted.
- Paths for the items through the model are clearly visible.

Imagine if value connectors were not part of ExtendSim's architecture. The modeler would have to write a program that would be evaluated every time an item entered the queue. This program would need to search through all of the items in the Activity, compare the priority of these items to the priority of the item entering the queue, and potentially remove the item from the Activity and place it in the Queue. There would be no visual representation of the logic, and more importantly it would be difficult to save and re-use this model segment, communicate the logic to others, or debug.

3.2 Hierarchy

Even in this example, it is apparent that a model with a large number of blocks on the same worksheet could be cumbersome to work with. To organize models and for reuse of model segments, ExtendSim includes hierarchical modeling—combining multiple blocks into a single block. Hierarchical blocks can contain any number of other blocks. Hierarchical blocks are created with a menu command and can be stored in libraries for reuse in the same or other models.

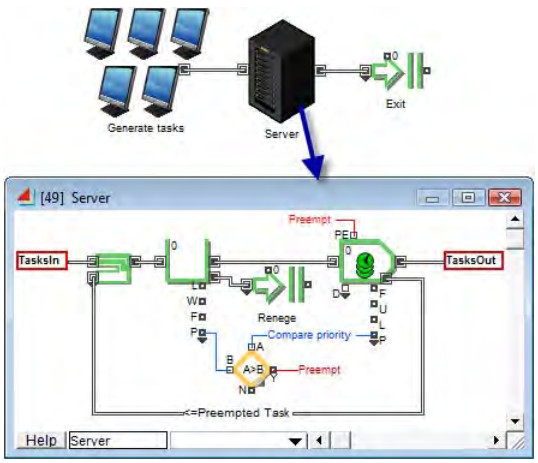

Figure 4: Server with hierarchy

In Figure 4 it can be seen how the top level of the model is simplified by combining the 5 blocks that represent the server, queue, and decision logic into a single, hierarchical block. A more descriptive icon and help have been added.

3.3 Cloning

When hierarchy is used in a model, some of the important parameters and results tend to get pushed down into lower levels so that they are not easily accessible. ExtendSim incorporates an innovative feature called cloning that allows the modeler to pull important inputs and outputs to the interior of a hierarchical block, the model worksheet, or the model notebook. Cloning is done by selecting the Clone tool ( ) and dragging any dialog item to a new location.

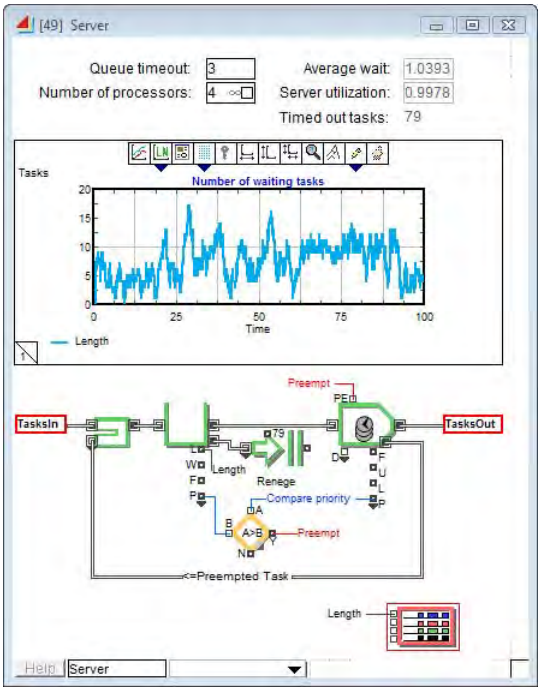

Figure 5: Cloned dialogs added to hierarchical block

Cloned dialog items can be either inputs or outputs. In Figure 5 inputs and results from a variety of blocks have been cloned to the hierarchical block's worksheet. Now, anyone can easily see or change the important model parameters without needing to locate them in the hierarchical block structure.

3.4 Animation

ExtendSim incorporates animation for model communication, debugging, and presentation. By default, models are animated in 2D on the worksheet, providing a quick view of the workings of the simulation. A more sophisticated worksheet animation can be created by animating the hierarchical blocks and changing the pictures that represent the items. This is all done within the standard ExtendSim drag and drop interface.

A quick 3D animation can be created by opening the 3D window, enabling "QuickView" mode. In this mode the item handling blocks in the model will have a representa-

tion in the 3D world and items will be animated as they move from block to block.

A more lifelike 3D animation can be created by changing to the concurrent or buffered modes. In these modes a constant ratio between simulation time and real time is maintained. Additional animation features such as mounting items together to combine them, adding scenery to show background objects, hiding selected blocks, and defining paths for the items to follow can be added to improve the look and accuracy of the animation. 3D animation actions are built into ExtendSim's Item Library blocks. These can be augmented by blocks from the Animation 2D-3D library, equations, or custom built blocks.

Figure 6: 3D animation tanker transport

Figure 6 shows a sample 3D model incorporating terrain and texture features illustrating a coastline. The 3D environment is completely integrated with the simulation environment. However, 3D animation can be added at any stage, but is not a required part of the logical model building process.

4 TECHNOLOGY

ExtendSim's underlying technology is the key to its unique combination of flexibility and ease of use. Technological features such as integrated databases, data linking, discrete rate modeling, value and item messaging, and a built-in development environment make ExtendSim stand out among simulation programs.

4.1 Database

ExtendSim's Graphical Simulation Database (GSDB) is unique. It is the only fully integrated relational database available for general purpose simulation software.

The value of a database as a data organizational tool in simulation is evidenced by the number of simulation software programs that advertise an interface to an external database (such as ODBC). However, benchmark tests reveal that models can run up to 50x faster when ExtendSim's internal database is used instead of Microsoft's state-of-the-art COM interface.

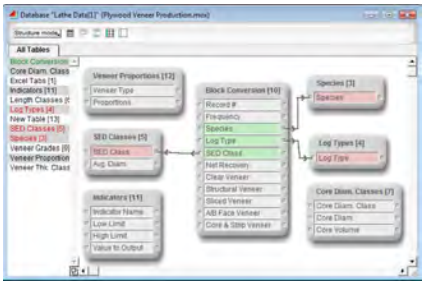

Figure 7: ExtendSim database structure

ExtendSim's integrated, relational database (Figure 7) provides data transparency and control. This is done without incurring the overwhelming performance degradation that occurs when communicating with outside database applications.

The advantages of a relational database over other data storage mechanisms are numerous:

- Database fields are typed (integer, string, date…). This helps to prevent errors and presents the information in a more readable format.
- Fields are labeled. This helps with readability and allows the modeler to insert columns (fields) without requiring an adjustment to data references.
- Parent-child relationships can be established between the tables. This helps to organize and reduce the total volume of data. In addition, model logic can traverse these relationships, making the models more compact, scalable, and maintainable.
- The information is visible. Database values can be viewed at any time. Changes to the records are seen immediately during the simulation run.
- Data can be more easily re-used in other models.

ExtendSim's relational database has many features unique to the requirements of simulation modelers:

- Records can be specified as random distributions.
- Tables or records can be linked to block dialog inputs and outputs.
- Each model can contain multiple databases.
- Link-Alert technology propagates messages to linked blocks when a database value changes.
- High-speed in-memory data storage.
- No programming is required to import, export, or access data.

With a right-click, the modeler can link a block's dialog value or table to a database record or table. If one changes, the other changes to the same value. In addition, a message can be sent to any linked blocks notifying them immediately of a data change. These link messages can be

turned off to improve performance if the modeler determines that this notification is not necessary. The messaging avoids the cumbersome and inefficient "end-of-event" evaluations that is required in other simulation software.

ExtendSim's database as been identified as a critical feature by modelers needing to simulate large and complex systems. In particular, the performance, organization, and maintainability of supply chain, agent-based, and other data intensive models benefit from the database architecture.

4.2 Discrete Rate

Discrete rate simulation uses discrete event clock scheduling to simulate high-speed or rate based systems (Damiron and Nastasi 2008). Rather than individual items moving from one block to another, discrete rate simulates a flow of material moving through the model. This differs from a continuous model in that the simulation clock moves from one event to the next and is not based on a fixed step size. Events in a discrete rate model might include a tank filling up, a rate change due to a product change-over, an equipment breakdown, or the arrival of a discrete event item representing a batch.

To provide global oversight and calculate the effective rates in a discrete rate model, ExtendSim uses linear programming (LP) technology. After all the rules for storage capacity and movement have been declared in the model, ExtendSim uses an LP calculation to maximize the flow through the system. This calculation is handled automatically and internally. The LP engine ensures mass balance, resolves zero time conflicts, and ensures that feedback loops are calculated accurately without propagation delays.

ExtendSim's discrete rate technology is fully integrated with discrete event simulation. Items can be converted into flow (for example, when a container of flour is dumped into a mixer) and back again.

4.3 Messaging

Messages are the fabric of an ExtendSim simulation. They are used to communicate information between the application and the blocks and from one block to another. Messages are sent to blocks at specific model events (such as model initialization), travel over connections, or are sent directly from one block to another. Messages ensure that each block has the most up-to-date information and that blocks update at appropriate times.

A message is interpreted by a block within the context of that block. For example, the clear statistics message causes a different set of actions in a Queue than in an Activity.

Without messaging, other simulation engines must rely on checking conditions at each step of the simulation, slowing performance and potentially skipping important zero-time state transitions.

4.4 Development Environment

ExtendSim uses a dedicated, compiled programming language for block development called ModL. There are numerous advantages to an embedded development environment that are specific to simulation modeling:

- Interactivity. Building an interactive model using a general purpose language requires that any interactive parameters be predefined and programmed into the model. In ExtendSim, a change to a dialog parameter automatically becomes a change to a variable in the underlying ModL code. In addition, the dialogs that contain the result values are updated as the simulation runs.
- Performance. The overhead required to call an outside program such as a routine written in Visual Basic or C++ can have a significant impact on performance. In addition, in other compiled simulation programs there is a compile phase that must be done every time the model is changed. In ExtendSim, the blocks are precompiled, and recompiling not necessary.
- Additional simulation-based functionality. Generic programming languages have no inherent simulation capabilities. Hundreds of functions and features have been added to ModL that are specific to continuous, discrete event, discrete rate, and agent-based simulation.
- More robust programming environment. ModL performs additional error checking and includes built-in data structures such as queues and linked lists.
- Integrated debugging. The ModL debugger is built into ExtendSim. ExtendSim's system variables and function calls are visible from within the debugging environment.
- User interface development. The ModL language is "live" from the time that ExtendSim is started to when the application is closed. Because of this, ModL can be used whether or not the simulation is running. This is invaluable for creating user interfaces that are used before and after, as well as during, the simulation run.

When compared to a simulation program that generates code (either in a simulation specific or general purpose language), ExtendSim's technology eliminates the need for the time consuming linking and compiling steps. Also, by definition, code generators cannot include the level of interactivity provided by ExtendSim.

Note that ExtendSim does have sophisticated methods for utilizing subroutines, methods, and programs developed in outside languages. This can be done through a standard COM or DLL interface. However, because of the power

and flexibility of ModL, it is rarely necessary that the modeler would need to resort to an external language.

5 APPLICATIONS

ExtendSim has repeatedly proven itself capable of modeling large complex systems and the range of applications where ExtendSim has been used is truly impressive. The following samples illustrate the flexibility and applicability of ExtendSim.

5.1 Plywood Manufacturing

The plywood manufacturing process begins with turning a log on a lathe and stripping off a thin layer of veneer, much like pulling toilet paper off the roll. The amount and type of veneer produced depends on a number of factors including the species of the tree, the location on the tree where the log was located, and the diameter of the log. The model works out to be an interesting combination of discrete event (movement of the logs) and rate based (continuous production of veneer) simulation.

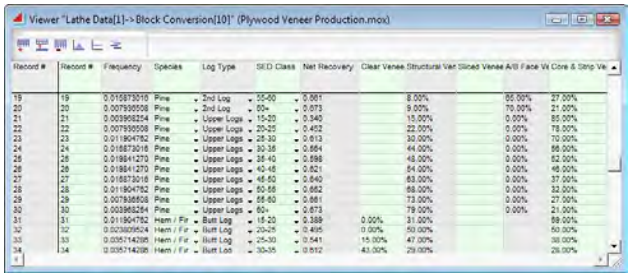

Figure 8: Database table of log properties

In this model, ExtendSim's database is used to store the empirical distribution of the logs and their properties. The probability and record number fields are linked to an empirical distribution in a Random Number block. The record of a specific log is assigned to an attribute. At any later point in the model, the properties of this log can be accessed. Figure 8 shows the database table with the log probabilities and properties.

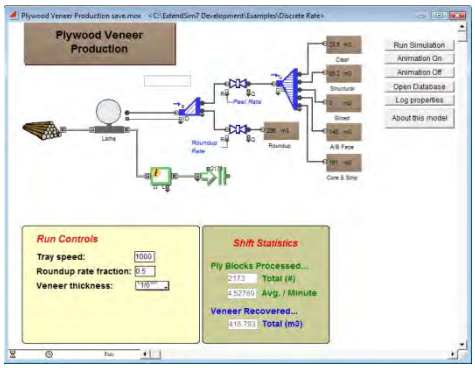

Figure 9: Plywood veneer production model

The plywood production model in Figure 9 shows logs arriving as items at the left and being placed on the lathe. The logs are then converted to a continuous flow of veneer. At the start of each cycle, the log is turned on the lathe, removing any excess material (called roundup). Once the roundup has been removed, veneer is produced until the diameter of the log reaches a preset size. Blocks from ExtendSim's Rate library convert an item representing a log to a rate-based flow of plywood veneer. These blocks then distribute the veneer to various products based on the log properties.

5.2 Agent-Based Airport Security

Simulating the effectiveness of airport security against a variety of threats helps to determine and improve the safety of the national transportation system. In this model passengers progress through the airport terminal (Weiss 2008). Airport security is performed by defense agents. Attackers attempt to pass through a series of sensors and barriers, reacting to them and modifying their actions based on previous experiences. Defenders react to the attackers based on rules of engagement and sensor detection.

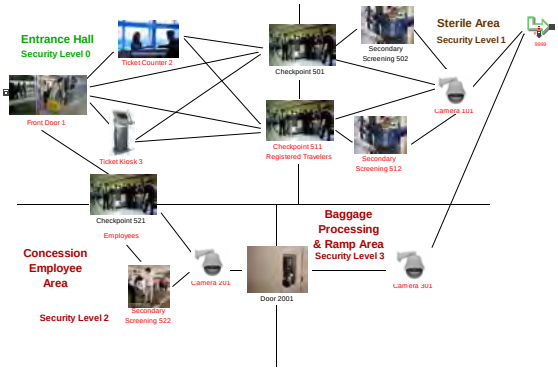

Figure 10: Airport security

Each icon in Figure 10 is a hierarchical block containing other blocks that represent the logic of the model. The hierarchical blocks represent the sequence of sensors and barriers that passengers pass through as they travel to the aircraft. ExtendSim's relational database is used to store model inputs and results such as the number of successful detections and the number of threats that were undetected.

6 SUMMARY

At Imagine That, we believe in creating a solid foundation, innovating where it makes a difference, and providing service and software that is accessible to the widest range of people for the widest range of applications. ExtendSim has been demonstrated to be both easy-to-use and powerful. The Imagine That web site provides a list of

applications from aeronautics to waste water treatment (Imagine That Inc. 2008), demonstrating the flexibility of the ExtendSim application.

REFERENCES

Damiron, C, and Nastasi, A, *Discrete Rate Simulation Using Linear Programming.* In Proceedings of the 2008 Winter Simulation Conference, S. J. Mason, R. R. Hill, L. Moench, O. Rose. IEEE, Piscataway NJ. To appear.

Imagine That Inc. 2007. *ExtendSim User Guide*. San Jose, CA.

Krahl, D. 2001. *The Extend simulation environment*. In Proceedings of the 2001 Winter Simulation Conference, ed. B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, 217-225. IEEE, Piscataway, NJ.

Weiss, W. 2008. *Dynamic Security: An Agent-Based Model for Airport Defense.* In Proceedings of the 2008 Winter Simulation Conference, S. J. Mason, R. R. Hill, L. Moench, O. Rose. IEEE, Piscataway NJ. To appear.

AUTHOR BIOGRAPHY

DAVID KRAHL is Vice President of Technical Sales with Imagine That Inc. 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 including ExtendSim, SLAM II, TESS, Factor, AIM, GPSS, SIMAN, XCELL+ and MAP/1. A few of the companies that Mr. Krahl has worked with as a consultant and educator are Chrysler, Ford, Williams International, Tefen, Raytheon, and Boeing. He is actively involved in the simulation community. His email address is <davek@extendsim.com> and the Imagine That Inc. site is <www.extendsim.com>.