

AN EXAMINATION OF IMPLEMENTATION IN EXTEND, ARENA, AND SILK

Sid Redman
Sarah Law

P.O. Box 516
The Boeing Company
St. Louis, MO 63166-0516, U.S.A.

ABSTRACT

This paper provides an examination of different modeling situations implemented in Extend, Arena, and Silk and demonstrates how the implementation of the software impact the results and whether these behaviors can be modified. The modeler being more informed of the methods implemented can work within the software to more accurately produce the desired outcome. The methods may not be obvious and often affect the model. This influence may or may not be significant enough to bring attention to it. It ends by concluding that the assumptions in the software should be visible to the modeler to aid in model verification and validation.

1 INTRODUCTION

Default assumptions regarding values, processes, and processing order impact the results of all simulations. Unfortunately, every simulation tool has different defaults and the modeler may not realize how they affect a model. Because of this, “identical” models in different tools may produce different results. Modelers must understand the simulation language.

In Schriber and Brunner (2001) the basics of how discrete-event simulation software works is discussed. Here we expand on their work, taking a closer look at three software packages and how the implementation in the software affects models.

Each simulation tool has its own terminology for describing the concepts in discrete event simulation. To provide clarity and a common ground for comparison, the basic terms of simulation will follow Schriber and Brunner (2001).

Sections 2 to 10 use a question-answer format to examine and compare how Extend, Arena, and Silk handle different situations. The questions appear at the beginning of each section and address issues such as: How to select from multiple queues, What determines the order of simultaneous events, When do resource capacity changes take

affect, How to model line swapping, How is multiple resource allocation handled, Can an entity yield control without losing control, Can events be rescheduled, How precise is the clock, How to simulate compound conditions involving the clock. Section 12 discusses the conclusion. Appendix A-C provides an introduction to each tool; the version used, how models are built, and how the code underlying the model is generated.

2 SELECTING FROM MULTIPLE QUEUES

What is the default queue choice? Can it be modified? The situation is the grocery customer faces multiple checkout lanes. Which one does he choose to stand in? How is the decision modeled?

The Extend User’s Guide says that an entity exiting a block (the grocery aisles) connected to multiple blocks (the check out lines) with available capacity will arbitrarily go to any available block. Actually the entity checks each block based on the order in which it was connected to the previous block and goes to the first available block that it finds. Thus, the block that was connected first will receive all the items from the previous block as long as it has available capacity. This applies whether the block is a queue or any other Extend block:

- If the first block connected has infinite capacity, then other blocks will never receive items. (e.g. if lane one’s queue is unlimited, then no other lane will be chosen)
- If no connected blocks have available capacity, then the entity will be destroyed. (e.g. the customer has nowhere to stand in line to check out so he/she disappears-5)

Rather than relying on connection order to set the queue choice, you can use a Prioritizer block. Priority is set using the blocks dialog window, where the block assigned the largest value has the lowest priority.

For an entity to select from multiple queues in Arena, the modeler must specify a selection order. Predefined selection rules include: cyclic, random, preferred order rule, and largest number in queue. Specialized logic may be used to specify the queue choice when needed.

The logic used to select a queue in Silk must be user-defined. This user-defined logic may depend on queue length, entity attributes, random distributions, or other reasons. A unique feature of Silk is the ability of an entity to reside in multiple queues.

When selecting from multiple queues, Silk allows the user the most flexibility by allowing entities to reside in multiple queues and allowing entities to choose a queue based on any logic. Extend's default setting of destroying entities when no queue is available warrants a caution because entities typically do not get destroyed in actual systems.

3 SIMULTANEOUS EVENT SCHEDULING

If multiple events are scheduled to occur during the same entity movement phase (EMP), what determines their order? Can this be modified? (e.g. two drivers reach the intersection at the exact same moment. Who goes first?)

When events occur simultaneously in Extend, the event that is located leftmost in the graphical modeling window occurs first, giving it priority to resources, queues, activities, etc. If events are exactly the same distance from the left edge of the window, the event nearer to the top of the window receives priority. Even the smallest differences in alignment within the model window can entirely change the order of events in the simulation. Figure 1 shows red entities reach the Activity Delay first and thus exit first because their Generate block is slightly to the left of the Blue Generate block.

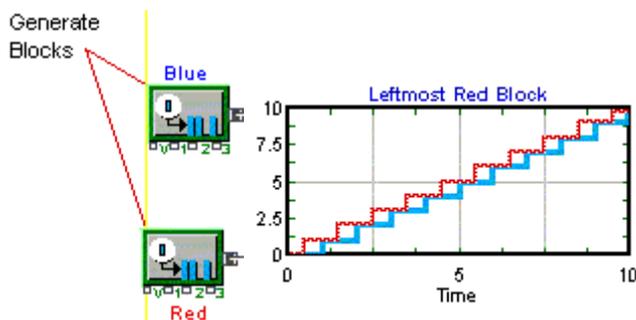


Figure 1: Extend Simultaneous Events

In Arena, the block that was placed first into the model window occurs first when “simultaneous” events occur. The model order can be checked, but not modified, by viewing the SIMAN code.

If simultaneous time-based events are scheduled to occur in Silk, the event referenced first in the code will occur

first. The order of events can be figured out by viewing the code. Simultaneous status based events (resource releases, queue changes) occurring at the same time can be prioritized with an integer priority.

This is one of the areas where the differences between Arena and Extend can be most clearly seen. Extend allows the sequence of simultaneous events to be modified by adjusting graphical position while Arena does not. The drawback with Extend is that a model may return different results simply because a block was moved slightly, unknowingly modifying the sequence.

Silk does not separate the user from the model with a visual interface, therefore it is less confusing when it comes to sequence of simultaneous events.

4 RESOURCE CAPACITY CHANGES

If a resource capacity change is scheduled to occur at a certain time, does it change before or after other events occur in the simulation? (e.g. a drive up window becomes available at the exact moment a customer arrives. Does the window open, then the customer can select a lane including the newly opened lane; or does the customer select a lane, then the new lane opens?)

Resource capacity changes are scheduled just as other events in Extend; changes or events will occur in a left to right, top to bottom order, if they are scheduled simultaneously. Extend allows a decrease in a resource to result in over-utilization until the current process completes.

A change in capacity will occur after other events scheduled during the same EMP in Arena. Figure 2 shows that an entity created at the same time a capacity is incremented results in two entities being queued versus one.



Figure 2 Arena Resource Capacity Changes

Arena allows the user to select from the following list of options to determine how an over-utilized resource is to be handled.

- “Ignore” starts the time duration of the capacity change immediately, but allows the busy resource to finish processing the current entity.
- “Wait” lets the busy resource finish processing the current entity before time duration of the schedule changes.
- “Preempt” interrupts the current processing entity, changes the resource capacity, and starts the time duration of the schedule change immediately. The

resource will resume processing the preempted entity as soon as the resource becomes available.

Resources in Silk have a “setCapacity” method that can be used to change capacity at any time during the run. Resource capacity changes occur immediately for idle resources, but not until the next idle state for non-idle resources.

Arena allows the user to select how capacity changes affect entities being serviced, while Extend has only the default setting already available. Silk and Extend allow some flexibility with regard to when the capacity change takes place, while Arena does not. As seen above, the timing of resource changes can affect the processing of entities.

5 LINE SWAPPING

Can an entity be removed from one queue and placed into another? (e.g. check out lane 5 is held up for a price check. How are customers switching to an unblocked lane simulated?)

Extend Technical Support recommended three methods of allowing line swapping. The first is to incorporate a renegeing queue. The queue can be set to renege items based either on an event or after some amount of time has elapsed. The renegeed entity can then be sent back to the multiple queue selection process to effectively simulate the type of line swapping that occurs in banks, grocery stores, fast food places, etc.

The Queue decision block or custom queue block was also recommended.

Arena allows items to be removed from a queue. Smarts example 085, Dynamic Queue Priorities, shows how to remove entities from a queue. The example checks the queue every minute to see if any entities waiting in the queue have been there for more than 20 minutes. If so, Arena removes it and places it into the shortest queue thus demonstrating line swapping through custom logic.

Silk Queues are objects with arrays of member Entity objects so entities can be added or removed from any queue in any order. Users are required to explicitly identify the entity object to be removed (by rank or object reference) at the time of removal, so line swapping is simply a matter of identifying the entity that you would like to move, and specifying where it should go.

Although not directly supported through existing modules line swapping is achievable through custom logic.

6 ALLOCATION TO MULTIPLE RESOURCE NEEDS

When queued items are waiting for multiple resources to become available, how are the resources allocated? (e.g. Can a group of two skip ahead in line at the amusement ride because there is a car with two seats available and the

groups ahead all require more seats or do they have to wait until they are at the front of the line?)

Extend provides a Queuing Resource Pool block for multiple resource allocation. The Queuing Resource Pool block is combined with one or more Resource Pool blocks that choose which entities receive their resources. Multiple Queuing Resource Pool blocks may request the same resources.

The Resource Pool searches through the queues looking for the first entity that is able to use its currently available resources. If an entity is waiting for resources in multiple pools to become available, it will only receive an allocation of resources when the resources it needs from each pool are available.

This method of multiple resource allocation can be dangerous, because if the right number of resources that an entity needs are never available, no entities may exit out of the queue. Extend’s Resource Pool does have an “Only allocate resource pool to the highest ranked Item” option; however, it is not available when more than one type of resource is requested.

Arena, like Extend, allows multiple queues to compete for resources. Entities may be queued separately if they are requesting different allocations of resources, or if entities in different sections of a model are requesting the same resource. Because of this, Arena looks through all queues requesting resources and creates a system queue for all entities requesting resource allocation.

When a resource becomes available for use, Arena only considers the first entity in all queues; thus no entity can skip over another. When an entity releases resources, Arena gives priority to any of the first entities in any queue that has a first resource request that matches the first resource requested by the releasing entity. If none is found then all the other entities in the group are evaluated again with the longest waiting serviced first. This method of multiple resource allocation can create strange patterns where one queue will empty entirely before items in other queues will have a chance at the resource.

Silk is similar to Arena if the process-oriented “push” method of resource allocation is used. However, Thread-Tec, Inc. encourages users to use the “pull” resource allocation method in which “intelligent” resource entities or a resource scheduler entity is used to determine multiple resource assignments. Thus a model with multiple resource allocation has essentially no limitation on how resources could be allocated. Resources can be allocated to the first item that can utilize all of the available resources, the first entity in a queue, the highest priority entity, etc.

Silk’s ability to have entities in multiple queues allows the user to collect statistics on how much time an entity spends waiting on a particular resource. This can help you to target bottlenecks in the system and give a more precise understanding of the system. Neither Extend nor Arena

allows this type of statistics collection and can meet with stair step processing of alternating queues.

7 YIELD CONTROL

Can an entity activate another process and wait until the process completes to be reactivated during the same EMP? While some methods have been proposed to simulate this with small delay times, the ability to yield without incrementing the clock is considered here because delaying any amount of time allows the remaining entities at the same EMP and any other entities scheduled between the small delay time to process before the yielding entity is reactivated.

Extend has a message-based architecture where yield control occurs constantly. Whenever an entity moves in or out of a block, messages pass throughout the blocks, changing and checking the system status. An entity can create another process or generate an item by posting an event and outputting some corresponding value to another block. The original entity can yield and resume using any series of control logic in Extend.

The Arena Block, Unblock and Proceed blocks are designed for yield control. The Block will remain in place until it is Unblocked. Each time an entity passes into a Block, the blockage for the Proceed is incremented by a specified value (default is one). An Unblock subtracts a specified value (default is one) from the blockage, if a blockage exists. If no blockage exists, an error is generated thus caution must be taken. A Proceed block is blocked as long as its blockage has a positive value; it is open if the blockage is zero.

Processor threads control the execution of simulated entity processes in Silk, making its yield control very similar to a thread yield within the Java program. Yield control in Silk can be managed in any number of ways with user coding, but it is typically done using wait and activate methods. An entity can halt itself for whatever reason and be reactivated later by any other class. The halt statement can be seen in Figure 3.

All three tools provide methods of yield control during the same EMP thus allowing the modeler to maintain clock time to insure that no other entities modify the simulation states during the yield. Arena is limited by offering only one method.

8 EVENT RESCHEDULING

Can events be rescheduled? (e.g. Painting a part takes two painter-hours to complete. If one painter is available then the task is scheduled to complete after two hours. But one hour later another painter becomes available. The task then needs to be rescheduled to complete in one and a half hour. Or can a resource be interrupted when servicing an entity? (e.g. a higher priority task has arrived)

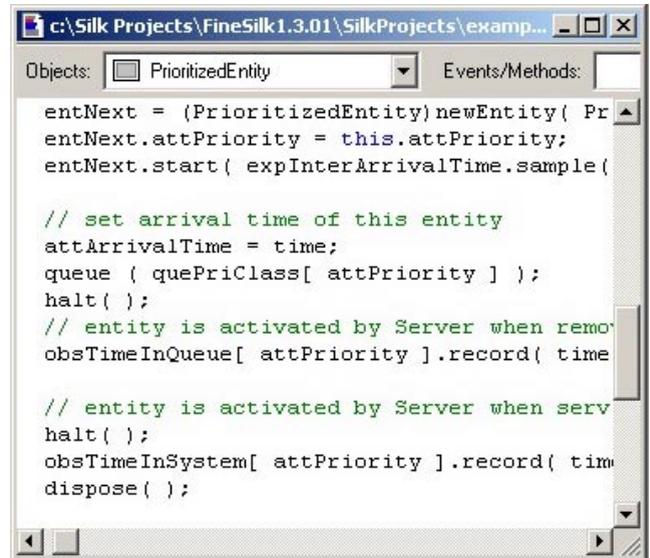


Figure 3: Silk Yield Control

Blocks rather than items are scheduled in Extend, which sends messages back and forth whenever an event occurs in the model. This makes rescheduling an event for any number of reasons a fairly simple matter. Not only can events be rescheduled, but also ongoing processes can be interrupted. Some blocks provide an option for preemption; others could be interrupted with a Downtime block or equivalent.

Many blocks have value inputs that can affect the scheduling of events in a model. For example, the Shift block can have scheduled shifts that are modified by an input. If the input is less than 0.5, the Shift block turns to off (rescheduling events for a later time), but if the input is greater than 0.5, then the scheduled shifts are followed.

The Zap block can be used to reschedule events in Arena. It removes an entity from the event calendar and sends it to a block label. The entity to be removed is specified by its identity number, regardless of where it is in the model.

In Silk, users cannot directly reschedule events. However control logic can be used to manipulate events before or after they enter into the event calendar. Silk has three lists that can create the next event, two of which are considered “Potential Events.” Potential Events are:

- A list of entities suspended at “while(condition())” waiting for a change in a state variable.
- A list of entities suspended at halt() waiting for an activate().

The third list is the list of entities at delay () or the future events list (FEL). Once an event has entered the FEL or the current events list (CEL), it cannot be rescheduled, but it may be tagged and ignored. A duplicated “rescheduled” event can then be created. Another workaround is to

prevent events from entering the FEL or CEL until no rescheduling would occur in the model, because Potential Events can be endlessly manipulated.

Extend allows the most control, ease of use, and flexibility in event rescheduling while Arena has one mechanism and Silk none.

9 CLOCK PRECISION

All three tools keep approximately fifteen digits of accuracy for time clocks and event calendars allowing for excellent time keeping.

10 CONDITIONS INVOLVING THE CLOCK

What happens if an entity waits for a compound condition involving the clock? (e.g. wait until all resources are in use, or it is exactly 12:00 PM)

In Extend an event must be scheduled to ensure that a condition is checked at a certain point in time. Conditions are only checked when events occur in the system. This occurs because the clock only stops at discrete points and it is not continuously checking the condition; thus the condition will be checked only when other events occur in the system which may or may not be at the desired time.

Arena uses a polled waiting mechanism for compound conditions that checks conditions at the end of each EMP, which not ensure a condition will be checked at the exact clock time that it becomes true, because an EMP may or may not occur at the specific time.

Silk conditions are checked every time a state variable change occurs which may affect an Entity waiting at a “while(condition())” construct. This is closer to the event-listener design pattern used in object-oriented programming. To guarantee that a condition involving the clock is checked at the specific time it becomes true, an event must be scheduled at that time.

For all the tools, special logic must be used to ensure that a compound condition involving the clock is checked at a specific time.

11 CONCLUSIONS

Silk, Arena, and Extend can all be used to generate models of the same system, but as seen, the level of control and flexibility that a modeler has varies greatly with each. Silk allows the greatest flexibility and control. Extend and Arena are removed from the programming environment, and consist of pre-built constructs that are strung together to create a model. In both Extend and Arena, the underlying code is accessible, but understanding and modifying it can be difficult.

All of the tools have been implemented a specific way which in some cases require the modeler to develop custom logic. In fact any simulation software will have a spe-

cific implementation requiring study by the user. The methods employed by the simulation software can be subtle and the impact unnoticed, making it important to understand the tool when creating or using a model. While a graphical simulation tool can be faster to generate smaller scale models, documentation was found to not fully explain implementation, thus requiring multiple conversations with technical support and test models to exploit the details. A general programming language with an open source simulation engine could be faster to review the implementation and develop appropriate modeling logic without relying on documentation, and technical support.

Regardless of simulation software chosen implementation cannot be avoided, so it is important to be aware and understand how the model is affected.

APPENDIX A: EXTEND

Extend (version 5.0.4) provides multiple libraries for both continuous and discrete event modeling shown in Figure 4 (Image That, Inc. 2000b). The libraries consist of blocks such as Generate, Delay, and Resource, each of which is available for drag and drop use in a model.

The blocks are added to the modeling window and then connected to create the modeling order. Anytime you are not certain what the simulation order is, you can check the order under the Model toolbar. Simply click on “show simulation order” and each block will be numbered to show where it fits into the simulation.

The visual order of the blocks at the run time sets the model order. But, non-visual elements may also be used for scheduling. The code underlying each block is accessible for reading and writing purposes. Blocks rather than items schedule events, and items can travel in a “push” or “pull” fashion.

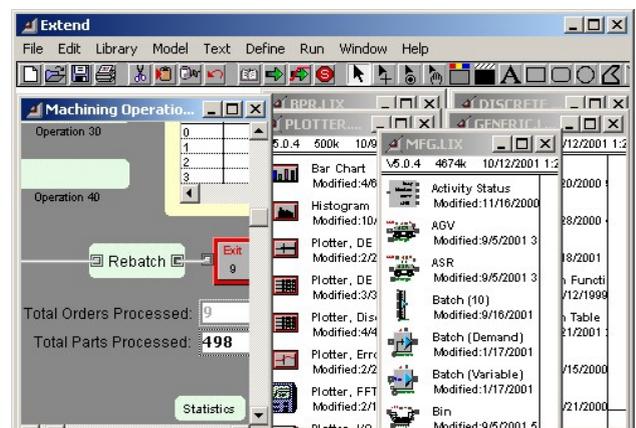


Figure 4: Extend's Modeling Environment

APPENDIX B: ARENA

Arena (version 5.0) has a visual interface similar to Extend where blocks from various libraries are put into a model window and then connected to create a modeling order (Rockwell 2000a). Each block represents code that is added to a SIMAN model.

The model code is created as blocks are added to the modeling window shown in Figure 5. This can create some strange coding orders, because once something is added, its code order never changes. And again, Non-visual elements are used for scheduling. The SIMAN code that is generated is available for viewing and can be looked at to determine the model order.

Generally, items “push” through the simulation, although some blocks (such as remove) can be used to pull items.

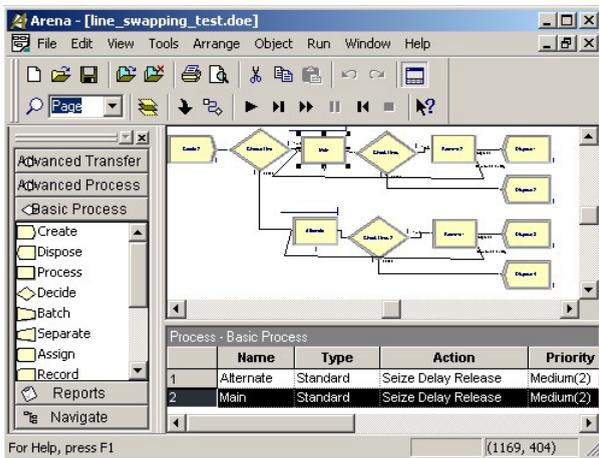


Figure 5: Arena’s Modeling Environment

APPENDIX C: SILK

Silk (version 1.3.01) is an extension to Java with constructs for generic discrete event simulation such as entity, queue, and resource (Kilgore 2000). Coding and compiling in a Java development environment using Silk creates simulation models (see Figure 6). Model order is determined by the order of the process-oriented simulation code you create in the entity “process” method. You can determine the model order by examining each class carefully, beginning with the class containing the “main” method. Entities are scheduled and managed by their own methods (delay, halt, “while(condition())”) and other entity methods (activate), allowing “push” and “pull” modeling.



Figure 6: A Silk Model Running in Visual Café 4.0

REFERENCES

- Imagine That, Inc. Extend Version 5 Programmer’s Reference. San Jose: Imagine That, Inc., 2000a.
- Imagine That, Inc. Extend Version 5 User’s Guide. San Jose: Imagine That, Inc., 2000b.
- Imagine That, Inc. Frequently Asked Questions. 31 Jan. 2002 <<http://www.imaginethatinc.com/>>
- Kilgore, Richard A. 2000. “Silk, java and Object-Oriented Simulation” Proceedings of the 2000 Winter Simulation Conference, ed. J.A Joines, R.R. Barton, K. Kang, and P.A. Fishwick, 246-252. Piscataway, NJ: Institute of Electrical and Electronics Engineers.
- Krahl, David. 2000. “The Extend Simulation Environment.” *Proceedings of the 2000 Winter Simulation Conference*, ed. J.A Joines, R.R. Barton, K. Kang, and P.A. Fishwick, 280-289. Piscataway, NJ: Institute of Electrical and Electronics Engineers.
- Morelli, Ralph. Java, Java, Java. Upper Saddle River, NJ: Prentice-Hall, 2000.
- Rockwell Software, Inc. Arena Basic Edition Reference Guide. USA: Rockwell Software, Inc., 2000a.
- Rockwell Software, Inc. Arena Professional Edition Reference Guide. USA: Rockwell Software, Inc., 2000b.
- Rockwell Software, Inc. Arena Standard Edition Reference Guide. USA: Rockwell Software, Inc., 2000c.
- Rockwell Software, Inc. Arena Variables Guide. USA: Rockwell Software, Inc., 2000.
- Schriber, T.J. and D.T. Brunner. 2001. “Inside Discrete-Event Simulation Software: How it works and why it matters.” *Proceedings of the 2001 Winter Simulation Conference*, ed. B.A. Peters, J.S. Smith, D.J. Medeiros, and M.W. Rohrer, 158-168. Piscataway, NJ: Institute of Electrical and Electronics Engineers.
- Schriber, T.J. and D.T. Brunner. 2000. “Inside Discrete-Event Simulation Software: How it works and why it matters.” *Proceedings of the 2000 Winter Simulation Conference*, ed. J. A. Joines, R. R. Barton, K. Kang,

and P. A. Fishwick, 90-100. Piscataway, NJ: Institute of Electrical and Electronics Engineers.

Schriber, T.J. and D.T. Brunner. 1996. "Inside Simulation Software: How it works and why it matters." *Proceedings of the 2000 Winter Simulation Conference*, ed. J. Charnes, D. Morrice, D. Brunner, and J. Swain, 23-30. Piscataway, NJ: Institute of Electrical and Electronics Engineers.

Systems Modeling Corporation. Arena Professional Edition Reference Guide. Sewickley, PA: Systems Modeling Corporation, 1994.

ADDITIONAL RESOURCES

The Javadocs included with Silk were an important resource along with the other documentation included with the installation. The help references included with each software package were used extensively to help understand each program.

AUTHOR BIOGRAPHIES

SID REDMAN is an engineer focusing on advanced support concepts in Boeing's Phantom Works the research and development division. He received his BSME from the University of Missouri-Columbia. He is a member of Military Operations Research Society. His email is <raymond.s.redman@boeing.com>.

SARAH LAW is a research intern at Boeing and a degree candidate in Systems Sciences and Mathematics at Washington University in St. Louis, MO. Her email address is <sell@cec.wustl.edu>.