AGENT-BASED SIMULATION OF DISCRETE-EVENT SYSTEMS

G. Allen Pugh

Indiana University Purdue University Fort Wayne; Email:pugh@ipfw.edu

1. BACKGROUND

Manufacturing facilities can often be modeled as a collection of queues. However, once queueing system complexity rises beyond the trivial, analytical methods become intractable. In these cases, discrete-event simulation is essentially the only tool capable of modeling such systems. In this paper, two approaches to modeling manufacturing queueing systems are explored.

While there is a plethora of simulation software tools available, consideration here will be constrained to those most suitable for (and available to) education. The following criteria were used to narrow the selection.

- The software must be inexpensive, or be available to to students at a substantial discount. The rationale for this criterion is that students can complete the course with their own copy of the tool.
- The interface must be graphical so that models can be readily constructed. Less time consumed in model construction translates into greater time for analysis and fine tuning.
- The software should allow for (optional) animation and interaction while running. These features, while computationally expensive, encourage experimentation.
- Numerous example and help files should be available.
- Computational demands should be reasonable. The software should run on the types of machines students are likely to have access to.

2. WORLD VIEWS

Two very different types of world views are examined. The first is the traditional discrete-event simulation modeling environment exemplified by Extend (http://www.imaginethatinc.com/). The second is a less traditional model based upon the interaction of agents, and demonstrated with NetLogo (http://ccl.northwestern.edu/netlogo/). Agent-based modeling has been well documented as a useful simulation pedagogical tool (Colella, 2001). As an example a conventional M/M/1 queueing system is constructed in both environments. This stochastic system consists of an arrival process which generates entities (representing customers, machine failures, phone calls,

etc.), a single first-come, first-serve waiting line (queue), and a server. The time between arrivals and the service time are both described by a negative exponential distribution. It is a useful demonstration because a closed form solution also exists, allowing for another measure of comparison.

2.1 A traditional modeling environment.

As shown in Figure 1, the M/M/1 system can be modeled with six blocks. These blocks represent a discrete-event clock, an arrival process that generates entities, the waiting line, the server (or service process), a block to generate the random service time, a graph, and an exit. Additional blocks may be added to illustrate such things as the change of queue length over time. Blocks are incorporated into the simulation by selecting them from a library and then drawing the appropriate connections between them.

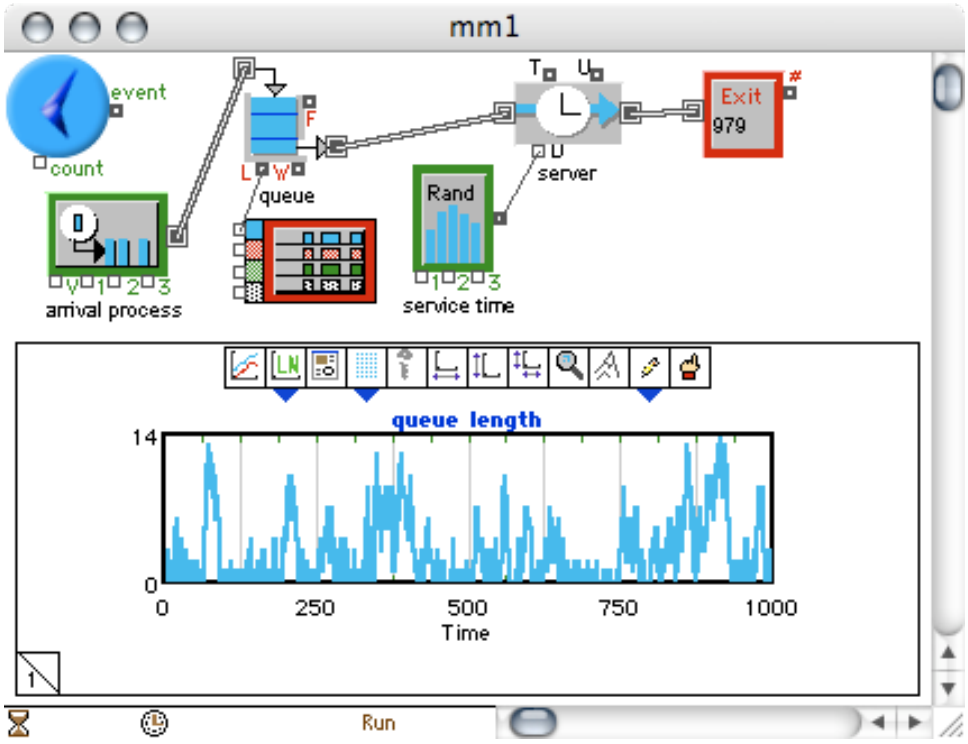

Figure 1: An M/M/1 queueing system in Extend.

Entities generated by the arrival process flow through the system in much the same manner as in an actual process. Note the focus is on process, not entities. Students can usually successfully build a model of this complexity within 2 weeks of the start of the course.

2.2 An agent-based environment.

Figure 2 illustrates the same M/M/1 model constructed with NetLogo. Note the interface emphasis on entities over process, especially in the primary (bottom) pane. NetLogo is typical of agent-based environments in that there is no direct provision for such entities as queues or service blocks and thus they must be created. Fixed constructs such as the arrival generator and service provider are developed as patches, which constitute the background of the primary display window. Agents move about the background. There is no formally designed queue – the agents are simply allowed to stack up in front of the server patch.

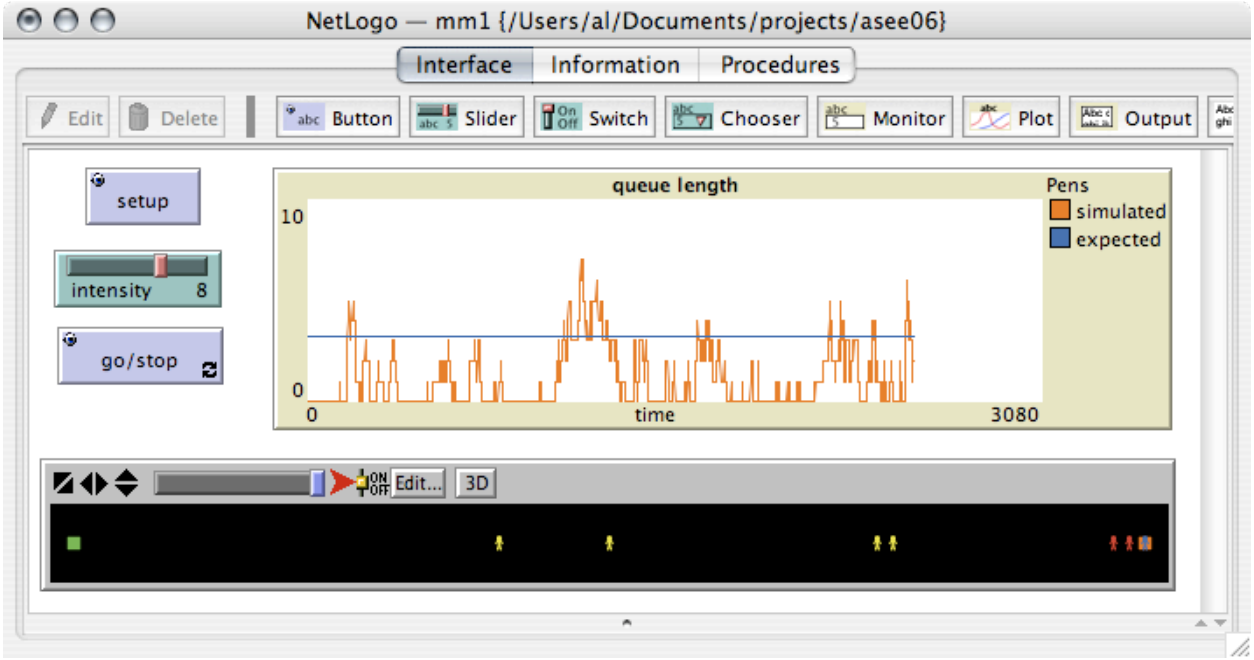

Figure 2: An M/M/1 queueing system in NetLogo.

Because of the emphasis on entity mobility in agent-based environments, the placement of background patches representing generators and servers becomes very important. There must be sufficient room for agents to queue up.

The single most important factor in using agent-based modeling of discrete-event systems simulation is the issue of discrete verses continuous time progression. Simulated time on digital computers is not truly continuous – rather, the choice is between time marching monotonically (in uniform steps), or leaping between events (requiring an event queue). The latter has the advantage of greater ease and precision in discrete-event modeling. Monotonically advancing time requires a careful choice of units (the granularity problem). Smaller units yield slow, but more accurate, models. The increased accuracy is due to a greater tolerance for larger variability in mean times between events. For example, in a simple queue with a mean exponential inter-arrival time of 2 minutes and a normal (3, 0.1) service time would require small time units to usefully model

A constant probability (corresponding to a Poisson arrival) of an event occurring may be modeled in a monotonic time environment by simply generating a uniform random deviate (URD). For example, if the expected value of an arrival process is 20 parts/minute (Poisson), then the expected time between arrivals is 1/20 minutes (exponential), and the chance of an arrival during any given minute (cycle) is likewise 1/20. So a discrete URD in the range 0-19 can be generated and the event considered complete if said URD yields a 0 (or any one of the possible 20 values). This system works well for exponential inter-arrival and service times provided the ratio of their expected values is rational. For example, consider an M/M/1 queueing system with mean inter-arrival time of 12 minutes and a mean service time of 10 seconds. the required URD's would have ranges of 0-11 and 0-9 respectively. More troublesome is the case where mean service time is 9.40, where the time interval must be shrunk by an order of magnitude to insure full precision - which renders the simulation slower by a like amount. Note resorting to a continuous URD (even if available) does not address the problem as the comparison is only made at integer intervals.

3. CONCLUSION

Both the traditional discrete-event modeling and the agent-based approaches represent the example M/M/1 system well, but branching into more complex models is somewhat problematical in the agent-based environment. This will likely improve as agent-based environments continue to evolve. A specialized agent-based simulation environment for production lot-sizing has already already been developed (Dessouky, 2002).

Two (very) preliminary observations regarding student use are:

- they do not care to be exposed to such diverse modeling environments in the same course, and;
- agent-based models appear more difficult to construct - but this may well be an artifact of their being offered after traditional models.

There remains much more to explore in the choice of discrete-event simulation modeling metaphors.

REFERENCES

Colella, V., Klopfer, E., and Resnick, M. (2001). *Adventures in Modeling: Exploring Complex,Dynamic Systems with StarLogo*. Teachers College Press, New York.

Dessouky, M. M. J. Rickel, and N. Sadagopan (2002), An Agent-based Learning Approach for Teaching the Relationship Between Lot Size and Cycle Time, *INFORMS Transactions on Education*, **3**, (1), 1-19.