Proceedings of the 1997 Winter Simulation Conference
ed. S. Andradóttir, K. J. Healy, D. H. Withers, and B. L. Nelson

A FRAMEWORK FOR THE SIMULATION EXPERIMENTATION PROCESS

Thomas C. Fall

Lockheed Martin Western Development Laboratories
1260 Crossman Avenue
Sunnyvale, California 98010, U.S.A.

ABSTRACT

Modeling and simulation of very large systems introduces a number of issues that are not seen in the modeling of simpler systems. These arise because the resource requirements for simulation rise much faster than the number of components being modeled and quickly overtake the available hardware and calendar time allocated. This implacable constraint has been met in several ways, though each has issues.

1) Decrease the number of objects by aggregating some.

Issues: Are the right objects being aggregated and how do we know? How do we mix objects of different aggregations?

2) Coarsen the temporal resolution for some portions of the model.

Issues: Are the right portions coarsened? How do we mix the coarse with the fine?

3) Build several models, each addressing different facets.

Issues: How do we combine these into a comprehensive model? How do we organize this so that for a particular enquiry, we can determine which of these facet models are applicable?

This paper will discuss an approach that addresses these issues by providing a framework that accepts different levels of aggregation and/or resolution.

1 CONSTRAIN SIMULATION SIZE

To motivate this discussion, let's start with a modeling problem that is big, yet still well within what is being modeled today. The example is a telecommunications example, an Asynchronous Transport Mode (ATM) network consisting of 40 nodes highly interconnected with 50 OC48 links, 150 OC12 links and OC3 links to the external systems (OC48 is Optical Carrier 48, at 2.488 Gbps, OC12 is 622 Mbps and OC3 is 155 Mbps) and where the typical path has an external link on each end, four nodes and three trunk links. Given that we want to calculate propagation delay on each link, bit errors, queuing at each port and switching, we need 200 floating point operations (FLO) for each object we model going through this system. If we model at the bit level and we believe the average traffic on this network will be 100 Gbps, then for each second of simulation time, we will need 20,000 gigaFLO, or 20 teraFLO. At the time of this writing, there is a major effort under way to build machines (for example, Intel's ASCI Red) capable of 1 teraFLO per second (teraFLOPS) (Pountain 1997). Even with this machine, not yet available, each second of simulated time would take 20 seconds to compute.

This factor of 20 wouldn't be so bad if we only had to simulate just certain short time intervals which have particularly interesting dynamics. However, we usually also have to let the simulation run for a while to 'settle'. That is, to let events that are temporally correlated come to an equilibrium. In the telecommunications arena, this is mainly the call duration; a call is initiated and a sequence of signalling ensues that allows the resources to be allocated to service that call. When the call is terminated, the resources are deallocated (the call is torn down). These are the two events that have the most effect on other calls and, for voice connections, they are typically separated by three minutes. Settling, if we start from an empty system, takes at least three times this length, though the system still hasn't usually completely settled that early. Even then, we are talking about 9 minutes of simulation time, which would take 180 minutes (20 times longer) on the not yet existing teraFLOPS machine and that just gets us to a settled state. We still need to add time to do whatever experiments we wish to do.

Thus, a given run for this system would take three hours and the course of runs needed to answer an important design question might take several days, just in run time. And, the system posed here is not nearly as large as some telecommunication models that have been successfully simulated, and those have been simulated on machines substantially less powerful than the teraFLOPS machine still being designed. In other words, for most real problems, we cannot rely on hardware to provide us sufficient computational capacity to perform the simulations we require. Rather, we are going to have to intelligently decide just what needs to be modeled and so constrain our simulation size.

2 ADDRESSING RESOURCE LIMITATIONS

To most modelers, the example above would seem strained, since they would almost certainly not model at the bit level. Rather, they would aggregate to the packet level (in the example above, the packets would be ATM cells which are 424 bits) achieving a compression factor of the packet size (taking our run time from 180 minutes to 25 seconds). And, in fact, they would model large parts of the network at the call level; that is, start a stream of a certain number of packets per second at one time, then terminate it at a later time. This number of packets is deducted from the capacity of the nodes the stream travels through. The action of all these packets is thus expressed with only two computer objects, the start object and the terminate object, as opposed to having a computer object for each packet. If the streams typically transfer many packets, this approach offers substantial compression. Often, some streams will be modeled at the packet level to get fine level resolution, but most of the network would be at the coarser call level. In fact, the nodes not on the paths of interest may be aggregated into a bulk resource usage representing the totality of calls originating or terminating in that subnetwork.

This telecommunications example is an example of the discrete event simulator methodology as opposed to the time difference methodology. An analogous example could be constructed for the time difference methodology and the compression approach for that methodology is to coarsen the time resolution, at least for parts of the system.

The discussion so far has seemed to indicate that the only way to address computational resource constraints is to decrease the number of objects being simulated. However, for many other large scale computational efforts, an approach being increasingly used is to distribute the processing over several CPUs. The CPUs may be in the same box (such as symmetric multiprocessors), or they could be spread over several boxes connected over LANs and even WANs. Simulations, as opposed to activities such as data base service, have the additional constraint that the simulation of different components must be synchronized. In other words, if we've divided the simulation into processes depicting different modules of the system and have distributed these processes, we need to ensure that when the true system modules would interact, the appropriate simulation processes must be able to interact. But if one of these processes runs substantially slower than expected, then all the other processes that could be affected by it must either wait for it or will have to roll back if they have proceeded on. There are, of course, some types of simulation where all the processes run at about the same speed, computational fluid dynamics being a prime example, and thus synchronization is not a problem. But where we are modeling a system consisting of different components, such as telecommunications models, the processes will likely not have the same run times and, in fact, may be very hard to predict a priori. For these type models, distributing a single simulator over several processors is still a research issue and is not an available option on current commercial simulation tools (Ho, Cassandras and Makhlouf, 1993 and Bhatti and Vakili, 1996).

Even though modelers have been forced to use a single processor, large systems are successfully simulated. In some of these, aggregation methods do not suffice to provide the requisite compression, and something like distributed processing is necessary. In these cases, instead of distributing the processes at run time, some modules are extensively simulated by themselves (or empirical data is collected for them) before run time so that we have an easily calculated method for determining that module's behavior at run time. There are two major classes: 1) analytic (for instance, the pressure-volume-temperature formula for the aggregate behavior of immense numbers of individual gas molecules) or 2) stochastic (the "combat results tables" of military board games are drawn from historic evidence, either from real life or from simulations and allow a dice toss to determine the results of a given engagement). This will only work if we can decompose the system; that is, divide it into subsystems and determine where the boundaries can be drawn between these subsystems, the 'interfaces' between them, so that we can have assurance that the coupling between them is substantially captured by one of these mechanisms, the analytic or the stochastic.

Most real systems will admit a decomposition into subsystems, and may well admit analytic expressions of their behavior. But, these analytic expressions may involve tens or hundreds of variables. In that case, we will look for the expressions that either utilize very few variables or utilize variables we would be working with anyway in other subsystems. Even though our original expressions may have been analytic, these aggregated expressions will typically be stochastic to capture the range of affects of the unexpressed variables. Our framework process uses the decomposition into subsystems along with efficient stochastic combination techniques to track our results achieved and to help guide our next efforts.

In sum, in this section we have discussed three ways to deal with the limitations of computational resources:

1) Decrease the number of objects by aggregating some.

2) Coarsen the temporal resolution for some portions of the model

3) Build several models, each addressing different facets

Each of these approaches has issues, which we'll discuss in the next section.

3 THE EXPERIMENTAL FRAMEWORK

To control our run time, as we've seen in the previous section, we try to break the system into subsystems which interact with each other utilizing a sufficiently small number of variables. In addition to the subsystems of the system, we must also consider the environment the system acts in. In the framework approach, we treat the relation of a subsystem to the system as if it were the relation of a system to its environment. So, focussing on a particular element, whether it is a system or a subsystem, there is a 'possibility' space that we work in, that is, the space of what we believe to be the likely states of the environment our element will have to work in. We typically set up our modeling program by dividing this 'possibility' space into extreme subspaces, i.e.,scenarios, where the rest of the 'possibility' space is believed to lie within the realm defined by these extreme cases. The 3-D spatial analog of this is a tetrahedron where the extremes of the 3-D figure are the 0-D extreme subspaces (the apexes), the 1-D extreme subspaces (the edges) and the 2-D extreme subspaces, the faces. If we choose a set of these subspaces correctly, the rest of the points in the possibility space are linear combinations of these. So if we run simulations on these subspaces getting estimates of behavior on each, we could combine these to obtain estimated behavior for the other points of the possibility space. Thus, we compress the experimental program because we now only have to test a much smaller number of points, those within these scenarios.

Looking at it abstractly, our possibility space is a bounded, convex body of some finite dimensionality. The different scenarios represent extreme subsets of that body having lower dimensionality. However, there are variables the simulations must consider that are not represented in this possibility space and the purpose of doing multiple runs of a given scenario is to test their affect and the effect of varying the values of the scenario variables, if the scenario subset has non-zero dimensionality. Thus, the results of a course of runs will form a distribution. What we wish to do is to combine these distributions to get estimated distributions for the points of the possibility space not in the selected test sets, the scenarios. Note that if we can break subsystems into subsubsystems, we can use the same process to compress the experimental course for them, as well.

Combining distributions has always been somewhat problematic. If the underlying relation is linear and the component distributions are symmetric, then the method of moments (that is, combine the averages and combine the variances) will work. However, if the distributions are skewed, or the relations are nonlinear, the method of moments yields a less satisfactory result.

Of course, we could use Monte Carlo techniques to combine the distributions, but this puts us pretty much back where we were. Another method, the Dynamic Focusing Approach (DFA) uses what could be thought of as a controlled Monte Carlo approach (Fall 1997). Instead of selecting from the component distributions randomly, it guides each choice. An important consequence of this is that, if there is a region of the estimated combined distribution in which we're interested, we know which choices of the component distributions got us there. From examining those choices, we can tell which components were the strongest determiners of that region, and thus we can focus on those for more detailed study.

4 OPERATION OF THE FRAMEWORK

The framework process described here assumes that we can break the system into subsystems and that, though the knowledge of the subsystem's behavior might be fuzzy, we do know how those behaviors combine to provide the system's behavior. In most cases, we have real systems that serve as starting points for the system we are studying. The historic evidence of their behaviors provide us the delineations we need. In the cases where we don't have parent systems, our course of action is determined by what our objectives are. If we are on a very short fuse, where some evidence must be quickly obtained, then we just run simulations up to the time limit, trying to spot any interesting behavior that could be focussed on, and report out what we've seen. If we have more time, we run these simulations until we can identify where the likely divisions into subsystems should be and what the relation is that combines those up into the total system. Note that for the simulation to be explainable, we really have to achieve this decomposition.

If we have been able to define appropriate subsystems, we can compress our simulation. One method, if appropriate, is to simulate the subsystems and combine using method of moments or DFA. The other is to aggregate some of the subsystems. Typically, we then present the coarse portions to those modeled at a finer level by having the coarse ones modify the stochastic choices the finer would make. That is, the coarser express themselves as 'bulk' phenomena; for instance, the telecommunications traffic we model in bulk expresses itself by taking up resources. A message stream being modeled at the finer level would be seeing buffers and switches handling their packets in a fashion corresponding to the bulk load imposed by the coarse components. This addresses the issue of how to mix levels of aggregation. The issue of whether the right objects are being modeled is an analysis issue, determined from the data that was used to delineate classification boundaries.

The temporal resolution approach has the same issues which are similarly addressed

The last approach, where we have built several models to capture the system, can use method of moments or DFA to utilize these components to form a comprehensive model. One should note that, since DFA is a controlled Monte Carlo approach, for a given enquiry, which will typically regard a particular region of the combined distribution, we can trace back to which components most determined that region. Thus, we have a mechanism that helps us focus on those component models that are most important to the enquiry.

In sum, to be able to compress our simulation experimentation sufficiently to stay within our resource constraints, several approaches are typically used that reduce the size of our simulation. Though these have issues, the framework approach described here has substantially addressed them.

REFERENCES

Bhatti, G.M. and P. Vakili. 1996. Parallel/distributed simulation for parametric study of wireless communication networks. Submitted to *IEEE Transactions on Parallel and Distributed Systems*.

Fall, T.C. 1997. Dynamic focusing approach to mixed level simulation. In *Proceedings of Enabling Technology for Simulation Science*, ed. A. Sisti, 22-30. SPIE, Bellingham, Washington.

Ho, Y.C., C.G. Cassandras and M.R. Makhlouf. 1993. Parallel simulation of real-time systems via the standard clock approach. *Mathematics and Computers in Simulation* 35(1):33-41.

Pountain, D. 1997. Parallel goes populist. *Byte* 22(3): 88NA3-88NA8.

AUTHOR BIOGRAPHY

THOMAS C. FALL is a Principal Systems Engineer at Lockheed Martin Western Development Laboratories. He is leading a team charged with simulating very large telecommunications networks. He has also been involved with various aspects of information integration for decision makers including evidential reasoning and modeling and simulation and has used models to assist the evidential reasoning process for event classification, for data fusion and for capacity management. He received his B.S. in Physical Chemistry and his Ph.D. in Mathematics, both from UC Berkeley.