Proceedings of the 1999 Winter Simulation Conference
P. A. Farrington, H. B. Nembhard, D. T. Sturrock, and G. W. Evans, eds.

"MODEL, THEN BUILD": A MODERN APPROACH TO SYSTEMS DEVELOPMENT USING CSIM18

Herb Schwetman

Mesquite Software, Inc.
P.O. Box 26306
Austin, TX 78755, U.S.A.

ABSTRACT

Developing a behavioral model of a system prior to implementing that system provides significant benefits. In some cases, errors in the design can be detected and corrected prior to implementation. In other cases, having a means for predicting the performance of the system prior to its deployment may aid in the design, implementation and configuration of the system. This paper presents a case for developing simulation models of systems early in the design and implementation process. A number of examples illustrate the benefits which can accrue.

1 INTRODUCTION

New systems are becoming increasingly more complex, offering more functionality, greater speed, and higher reliability, all at reduced costs. Along with these new requirements, there is also the fact that market windows are decreasing, time-to-market is decreasing, and the number of product cycles is increasing All of these factors mean that it is more important than ever that all facets of a newly designed system be validated and optimized as early in the design and implementation cycle as is possible and as quickly as is possible. One technique that has been used to facilitate this design process is to develop a model of the system as it is being designed and prior to its implementation. This paper presents computer-based simulation as an effective vehicle for developing the models which will speedup and improve the development process for complex systems.

In this context, system refers to an assemblage of interacting components. While familiar examples include computer systems, and communications systems, there are many other kinds of systems, ranging from consumer products, telephone call centers, manufacturing facilities and transportation and logistics organizations. Issues common to these systems include:

- identifiable components which interact,
- strict specifications for both correct and timely operation which must be met, and
- a range of choices regarding the constituent components which impact both the price and the performance of the system.

Computer-based simulation models can help address these issues and can provide a quantitative basis for making decisions in the design, implementation and configuration of these kinds of systems.

2 COMPUTER BASED SIMULATION MODELS

As stated above, many systems can be represented as a collection of components. Along with these components, there are entities which visit these components, seeking and receiving service at the components. Process-oriented, discrete-event simulation is a well known technique for creating computer-based models of these kinds of systems. In this paper, CSIM18 (Schwetman, 1996) is the simulation tool used to implement the examples which illustrate the "model, then build" approach to systems development.

A simulation model of a system is a computer program constructed to "mimic" the operation of the system. Typically, a simulation model will have software components which correspond to the major components of the system. In addition, the model will have entities, which model the behavior of the active entities found in the system. The goal is for the behavior of these entities in the model to accurately portray the operation and performance of the real system.

Through the development of this kind of simulation model, the designers can get a early look at the key

features of their design. This early look can impact the design and development process in many beneficial ways:

- the logical relationship between entities accessing different components can be investigated;
- the availability of critical status information for entities can be tested,
- the presence or absence of deadlocks can be studied, and
- the expected processing rates and response times can be estimated.

The results of studying this model can be extremely useful in qualifying a system design before implementation is begun. Furthermore, detecting design deficiencies early in the design-implementation cycle and correcting these deficiencies can lead to earlier delivery of the final product and can lower development costs (when compared to correcting these deficiencies later in the cycle).

3 EXAMPLES

3.1 Airline Ticket Counter

This example was presented in a paper by Pritsker and O'Reilly (Pritsker and O'Reilly, 1998). Briefly, there is a check-in counter with six agents, and there are two queues of passengers: priority passengers and coach passengers. The first two agents at the counter handle only priority passengers; the last two agents handle only coach passengers, and the two agents in the middle handle passengers from both queues, but they always check the priority queue first (Figure 1).

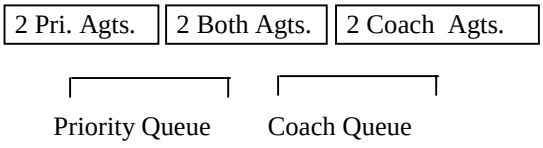

Figure 1: Passenger Check-in Counter

The parameters describing the two classes of passengers appear in Table 1.

Table 1: Passenger Parameters

Parameter	Priority	Coach
Service times	unif(2.0, 20)	tri(3.0, 12,0, 6.0)
Interarr times	exp(5.0)	exp(2.0)

This model illustrates the kind of problems that can be discovered and corrected early in the design cycle of a system. This model of the check-in counter, as described, shows that the queue of coach passengers grows without bound. A cursory analysis would suggests that this would not happen, that the check-in counter would operate in a reasonable manner. However, the simulation model shows that the priority passengers get enough of the two middle agents' time, so that the coach passengers are "short changed" and, as time passes, the line of coach passengers grows longer and longer and response times for coach passengers increase as well.

One problem with this model is that because of the high utilizations of the agents, the results for average response times are likely to vary from run to run, especially if the runs of a short duration. For the configuration with two priority agents, two agents in the middle and two coach agents, the response times summarized over 100 runs, each simulating 18 hours of operation, are shown in Table 2.

Table 2: Passenger Response Times for 2 Priority Agents, 2 Middle Agents, and 2 Coach Agents (minutes)

	Priority	Coach
Avg resp time	12.980	59.303
Min resp time	11.422	17.742
Max resp time	15.016	121.888

Once the model is constructed, it is easy to experiment with different configurations of check-in agents and parameter values for the passenger classes, and to determine the impact of these changes on the behavior of the system. For example, reassigning one of the middle agents to deal exclusively with coach passengers reduces the response times, but this does not solve the problem of long delays for the coach passengers. The results for this configuration of agents summarizing 100 runs, each simulating 18 hours of operation, are shown in Table 3. Adding another check-in agent for coach passengers does eliminate the problem completely. The results for this configuration of agents summarizing 100 runs, each simulating 18 hours of operation, are shown in Table 4.

Table 3: Response Times for 2 Priority Agents, 1 Middle Agent and 2 Coach Agents (minutes)

	Priority	Coach
Avg resp times	16.642	38.868
Min resp times	12.178	14.119
Maximum response times	23.693	91.617

Table 4: Response Times for 2 Priority Agents, 2 Middle Agents and 3 Coach Agents (minutes)

	Priority	Coach
Avg resp times	12.486	11.452
Min resp times	10.976	8.391
Max resp times	14.982	21.094

It can be noted that each of the studies in the previous tables (100 runs each simulating 18 hours of operation) consumed less than three seconds of CPU time on a 300 MHz PC. The model was implemented using the CSIM18 package (Mesquite, 1997).

This example demonstrates the potential value of developing a model of a system prior to its implementation. The first configuration, with six agent stations will probably be judged to be unsatisfactory. Additional studies show that with the parameter values for passenger arrival and service patterns (Table 1), no arrangement of six agents, giving preferential treatment to priority passengers, has acceptable response times for both priority and coach passengers. With the addition of a seventh agent station, acceptable levels of service can be offered to both classes of passengers. It is probably less expensive to build the simulation model and experiment with the different configurations and numbers of agents than it is to build a check-in counter with six agent stations and then have to modify it, to add the seventh station.

3.2 Web Servers

A World Wide Web (WWW) server is a computer system which is attached to the Internet and responds to requests to supply HTML pages and other files in response to GET requests from remote browsers (Figure 2). The rate of requests to a web server varies from one request every few seconds to hundreds of requests per second, depending on the popularity of the site, and whether or not it has been mentioned in the media recently. As an example, a site could experience 10,000,000 page requests in a week; assuming five days in a week and 12 hours in a day (the expected hours of heavy access rates for some servers), this is nearly 50 page requests per second.

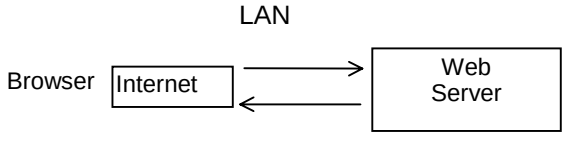

Figure 2: Web Server Model

A simulation model of a web server (modeling both the hardware and the web server software) can be useful in determining the capacity of the server required to handle a projected request rate. In addition, such a model can help assess the impact on performance of adding additional functionality to the server. It is pointless to advertise a web site or offer new services if the server is unable to deliver acceptable performance to the visitors to the site.

In a recent study, a simulation model of a web server was used to analyze the benefits of caching files in the server. The model has the capability of modeling a wide variety of web-based transactions. In the study, a fairly simple transaction was used to model the behavior of a stream of typical browser requests for pages. Each request consists of a request for a HTML page, quickly followed by a sequence of requests for additional files, to fill in the contents of the page; these additional files supply items such as graphics, images, maps, etc. which enhance the appearance and content of the original page. These "bursty" arrivals mean that traditional techniques for modeling arrivals of requests will probably not be sufficient to accurately model the behavior of the web server.

The study developed a model of the browser submitting page requests and the web server operating on its host computer system. The results of the study show that an amount of memory producing a cache hit rate of 0.50 will reduce the time required for a complete browser session (with averages of eleven page requests per browser session and six additional files per page). The results are summarized in Table 5.

Table 5: Response times (seconds) for Web Server Model

Cache hit rate	Browser resp. tm.	Page resp. tm	Add. file resp. tm
0.00	103.201	1.294	0.542
0.25	66.302	0.432	0.142
0.50	57.257	0.177	0.061
0.75	54.534	0.079	0.030
1.00	50.451	0.013	0.009

The original question dealt with the configuration of a web server which can handle 10,000,000 page requests per day, or 50 page requests per second. Analysis of the server data suggests that each page results in a number of additional files being requested, with the number of these additional files ranging from one to five. Additional analysis shows that a file cache hit rate of 0.1 can be expected for both the initial pages and the additional files.

The simulation model described above was modified to mimic the behavior of the server operating in this different environment. In particular, since the focus is on page requests, the browser was modified to submit just one

page request per session. The results of the simulation runs are summarized in Table 6.

Table 6: Web Server Response Times (sec) for 50 Page Request Per Second

Num. of Nodes	Req. Resp. Tm
1	- -
2	5.8
3	1.4
4	1.0
5	0.9

The model predicts that with one server node, the server cannot keep up. With two or more nodes, the server can keep up, however, it appears that at least three nodes would be required to yield acceptable response times for this page request rate.

In this case, having a simulation model allows the managers of the web server to estimate response times as the server deals with varying page request rates. The result is the opportunity to configure a cost-effective server which will offer acceptable response times.

3.3 Graphics Processor Chip

In a modern, high performance personal computer (PC), the graphics card is an important component, especially for PC's used for playing the latest interactive games with 3D animation. Each new generation of graphics chips (which are the key components in graphics cards) must offer higher levels of performance. In simple terms, a graphics chips renders a description of a scene into a screen full of colored and textured triangles, with each triangle positioned at the correct place on the screen. The speed and the degree of realism of the game is directly influenced by the rate at which these triangles can be rendered and positioned on the screen. With higher processing rates, more and smaller triangles can be rendered in less time, resulting in higher resolutions, more fidelity and faster animations.

One tool that some graphics chips vendors have used is a simulation model of the each new chip. By using such a model of a new chip, the designers are able to verify some aspects of the operation of the chip, as well as the triangle rendering rate of the chip.

At a high level of abstraction, a graphics chip can be viewed as a collection of processing elements and memories. Many of the processing elements, in turn, are constructed from different kinds of components such as multiplexers, pipelines, and buffers. Most of the computation (rendering steps) takes place at stages in these components.

A pipeline is a collection of processing stages; a complex computation can often be organized so that each of the steps of the computation can be accomplished in one stage of a pipeline execution unit The entire computation is completed when it has passed through the sequence of stages in the pipeline. If a new task can be started in the pipeline on each cycle and if each stage of the pipeline can completes its sub-computation in one cycle, then a number of tasks equal to the number of stages in the pipeline can be in process simultaneously, greatly increasing the task throughput rate (when compared to a single, unpipelined processing unit). This can be thought of as a task processing assembly line, where each stage in the pipeline corresponds to a station on the assembly line.

In many cases, some of the sub-tasks at a stage in the pipeline cannot be completed in a single cycle, and the pipeline must be stopped (stalled is the term used) until the incomplete sub-task can finish. Because of these stalls and many other issues, predicting the performance of a pipeline, and hence of components which contain pipelines can be difficult. Some chip design and implementation projects have developed and used extensive simulation models to predict chip performance before and during the implementation of these components.

In many simulation models, it is common to represent a system as a collection of facilities, where a facility consists of one ore more servers and a queue for waiting entities. Unfortunately, a processing pipeline, as described above, cannot be readily represented as sequence of facilities. The problem is that a task at one stage cannot vacate that stage until it can move to the next stage. In particular, if a task has stalled in a stage, none of the tasks behind that stage can move forward until the stalled task advances.

The ten-stage pipeline in Figure 3 is a simple example which illustrates the use of a simulation model. In this pipeline, the probability of having stall cycles and the number cycles per stall are as shown in Table 7. One measure of the performance of a pipeline is the cycles per element (the measure is really cycles per instruction - CPI - because pipelines were originally developed for central processing units processing instructions in computers), with 1.0 being the best possible value. The simulation model predicts that the CPI for the pipeline as described is about 2.8 cycles per instruction. If the number of cycles in stage six can be reduced from two to one, the CPI can be improved to about 1.5 (Table 8).

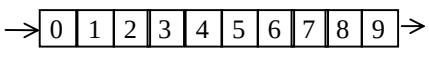

Figure 3: Ten-Stage Processing Pipeline

Table 7: Stall Probabilities and Number of Stall Cycles per Stage

Stage	Prob. Stall	Cycles/Stall
0	0.0	0
1	0.2	1
2	0.0	0
3	0.0	0
4	0.0	0
5	0.6	2
6	0.0	0
7	0.0	0
8	0.0	0
9	0.1	1

Table 8: Cycles Per (CPI)

Stall Cycles Stage 6	2	1
Cycles per Element	2.801	1.608

A model of a full chip (such as a 3D graphics chip) is made up of many components including components such as this pipeline. The ability to construct a realistic model of the chip using building blocks such as this pipeline enable the designer to determine which components must be improved. For example, redesigning this pipeline, to reduce the number of stall cycles in stage six from two to one could be such an improvement. Discovering the need for this kind of improvement early in the design (rather than after the chip is implemented) can materially reduce the both the cost of the chip and the time to market.

4 BENEFITS OF SIMULATION

The benefits of developing a simulation model can accrue in two major areas:

- the process of developing a model of a system often provides insight into some of the major issues which impact the design of a system,
- having an operational model of the system can give a quantitative basis for making many of the cost/performance trade-off decisions that are part of all development projects.

In addition, being able to detect and remedy problems early in the design and implementation cycle often produces significant cost savings.

In the examples in Section 3, a simulation model of the system being analyzed served as a predictor of performance. Each of the systems was so complex that simple techniques could not be used to predict performance. Each of the models incorporated the structure of the system and mimicked the behavior of the workload and the system as it processed elements of the workload. As a result, the model could produce useful estimates of the performance of the system.

All of the models described in Section 3 were implemented using CSIM18. Each of these models was composed as a C++ program and made use of the features of CSIM18. One of the advantages of creating models with a programming language is that there are almost no constraints on complexity and level of detail in the model. Furthermore, many subtleties in the behavior of the system and the workload can be incorporated into the model. In addition, the models are efficient and produce performance estimates quickly.

5 SUMMARY

Developing a computer-based simulation model of a system prior to its implementation has been shown to be cost-effective. Examples of three different kinds of system projects have shown how design and configuration problems can be detected, analyzed and fixed early in the development cycle. Doing this can result in reducing the total time required to design, implement, test, configure and deploy many kinds of systems.

ACKNOWLEDGEMENTS

CSIM is copyrighted by Microelectronics and Computer Technology Corporation (MCC). CSIM18 is supported and marketed by Mesquite Software, Inc. under license from MCC.. Dr. Jeff Brumfield developed the new data collection and presentations functions including the run-length control algorithm in CSIM18

REFERENCES

Law, A. and D. Kelton, 1991. *Simulation Modeling and Analysis*. (Second Edition) MacGraw-Hill.

Mesquite Software, Inc., 1997. *User's Guide, CSIM18 Simulation Engine*. Austin, TX.

Pritsker, A. and J. O'Reilly, 1998. Awesim: The Integrated Simulation System, In *Proceedings of the 1998 Winter Simulation Conference*. Ed. D. Medeiros, E. Watson, J. Carson, and M. Manivannan, 249, 255. Washington, DC.

Schwetman, H., 1996. CSIM18 - The Simulation Engine. In *Proceedings of the 1996 Winter Simulation Conference*. ed. J. Charnes, D. Morrice, D. Brunner, and J. Swain, 517 - 521. San Diego, CA.

Schwetman, H. and J. Brumfield, 1997. Data Analysis and Automatic Run-Length Control in CSIM18. In

Proceedings of the 1997 Winter Simulation Conference. Ed. S Andradottir, K. Healy, D. Withers, and B. Nelson, 687 - 692. Atlanta, GA.

Schwetman, H, 1998, Model-Based Systems Analysis Using CSIM18. In *Proceedings of the 1998 Winter Simulation Conference*. Ed. D. Medeiros, E. Watson, J. Carson, and M. Manivannan, 309-313. Washington, DC.

Shannon, R, 1998. Introduction to the Art and Science of Simulation. In *Proceedings of the 1998 Winter Simulation Conference*. Ed. D. Medeiros, E. Watson, J. Carson, and M. Manivannan, 7-14. Washington, DC.

AUTHOR BIOGRAPHY

HERB SCHWETMAN is founder and president of Mesquite Software, Inc. Prior to founding Mesquite Software in 1994, he was a Senior Member of the Technical Staff at MCC from 1984 until 1994. From 1972 until 1984, he was a Professor of Computer Sciences at Purdue University. He received his Ph.D. in Computer Science from The University of Texas at Austin in 1970. He has been involved in research into system modeling and simulation as applied to computer systems since 1968.