*Proceedings of the 2000 Winter Simulation Conference*
*J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.*

# A JAVA-BASED SIMULATION MANAGER FOR WEB-BASED SIMULATION

Charles Marr
Christopher Storey
William E. Biles

Department of Industrial Engineering
University of Louisville
304 J. B. Speed Building
Louisville, KY 40292, U.S.A.

Jack P. C. Kleijnen

Department of Information Systems /
Center for Economic Research
Tilburg University
P. O. Box 90153
5000 LE Tilburg, THE NETHERLANDS

## ABSTRACT

This paper discusses a Web-based simulation manager program that executes an *Application Service Provider (ASP)* event for a customer who does not possess the in-house capability to program complex simulations. The utility in using this simulation manager is that the customer needs results in near real-time; that is, approximately the time to run one complete replication of the simulation model plus some overhead time to send the commands necessary to execute the simulation and to process the results. The simulation manager executes simulation studies in a parallel replications format, using either designed experiments or optimization methodologies, by sending the appropriate messages to a set of *engine processors* to cause the execution of the prescribed simulation trials. It then receives and analyzes the simulation results produced by the simulation engines, and sends a project report to the human customer.

## 1 INTRODUCTION

The approach proposed here is aimed at carrying out a simulation study in a parallel-replications mode (Heidelberger 1988), utilizing a set of *P slave processors* available to a *master processor* called *SimManager*. By "available" we mean that the processors exist somewhere in the world and that their owners have entered into an agreement to participate in a simulation consortium that we shall call the *Alliance*. Through the *Alliance* agreement, these owners maintain their processors in the "on" state, with the simulation application accessed, so that they are available for use as slave processors (hence-forth referred to as *simulation engines,* or simply *engines*). We use the term *SimManager* here to identify the master processor that is controlling the simulation study and the term *engines* to refer to the processors that are actually executing the simulation trials. Other terms that are sometimes used to refer to this relationship are *master/slave*, *supervisor/ worker*, and *client/server.*

Suppose that the simulation model involved in the project has input variables $X_i$ , $i$= 1,…, $N$ and performance measures $Y_j$ , $j$ = 1,…, $M$, and that the objective of the simulation study is to establish the best values of $\boldsymbol{X}$ and $\boldsymbol{Y}$ using an optimization scheme such as response surface methodology (RSM). The simulation study involves the execution of $R$ replications at each of $S$ system scenarios, so that the total number of simulation trials to be executed is $K = RS$. This simulation workload is assigned to *P engines* by the *SimManager* processor. *SimManager* sends a file to each simulation engine detailing that engine's work assignment, receives back a file containing the statistical results derived from that engine's efforts, and organizes these results into a form that meets the needs of the customer who has purchased the simulation study. This is often an iterative activity that takes place over several cycles. We assume here that a human analyst is available to intervene with *SimManager* to ensure proper execution, to maintain security, and to deal with those customers who want to interact with a human.

This paper focuses mainly on the following interfaces: (a) that between the *SimManager* and its *engine* processors, (b) that between *SimManager* and its customer base, and (c) that between *SimManager* and its simulation model catalog. It describes methods for allocating workload to the $P$ engine processors so that the simulation study is carried out in minimum time or at minimum cost. The objective is to give the customer the best service possible, which could entail providing personal presentations of the final report by expert simulation consultants. This "personal" contact could be a face-to-face training session to novice simulation customers, or a combination of telephone conversations and e-mail.

This paper explores the simulation-study capabilities of a Java-based simulation manager such as that discussed by Biles and Kleijnen (1999). Theoretically, a simulation study that requires $R$ replications of each of $S$ system scenarios to obtain the desired confidence intervals on each of $M$ performance measures can be achieved in the time $T_r$ to run just one complete replication on the slowest available engine processor plus overhead time $T_o$. This *overhead time* is the sum of (a) that required prior to execution in order to send the required files and performance parameters to the $P$ engine processors, and (b) that needed after execution to receive simulation output back from the $P$ engines and compile simulation results. In its most ideal performance, *SimManager* would conduct a simulation study involving $K = S*R$ simulation trials in a differs from that of the PADS format described by Fujimoto (1998) in which a simulation model is decomposed and its $P$ component modules are simulated on $P$ engines.

The Front End is the interface between SimManager and its potential customers, who may only have access to the World Wide Web (WWW) via a shared computer or one with limited processing capability. A target audience might be for example, MBA graduate students who need the capability to perform an analysis of alternatives for business practices, but who lack formal simulation programming knowledge. This web-based simulation concept might also lead to carrying out distributed, non-parsed simulations for industrial organizations seeking the time and cost-savings that this technology offers. Figure 1 shows a sequential look at the interaction between a customer using the WWW to access *SimManager* and the Front-End software. *SimManager* controls the flow of the simulation, including which engine processors are assigned which simulation models, as well as the number $S$ of different sets of inputs conditions (scenarios) and the number $R$ of replications at each scenario. *SimManager* then compiles the summary results generated through the simulation study and delivers them to the customer.

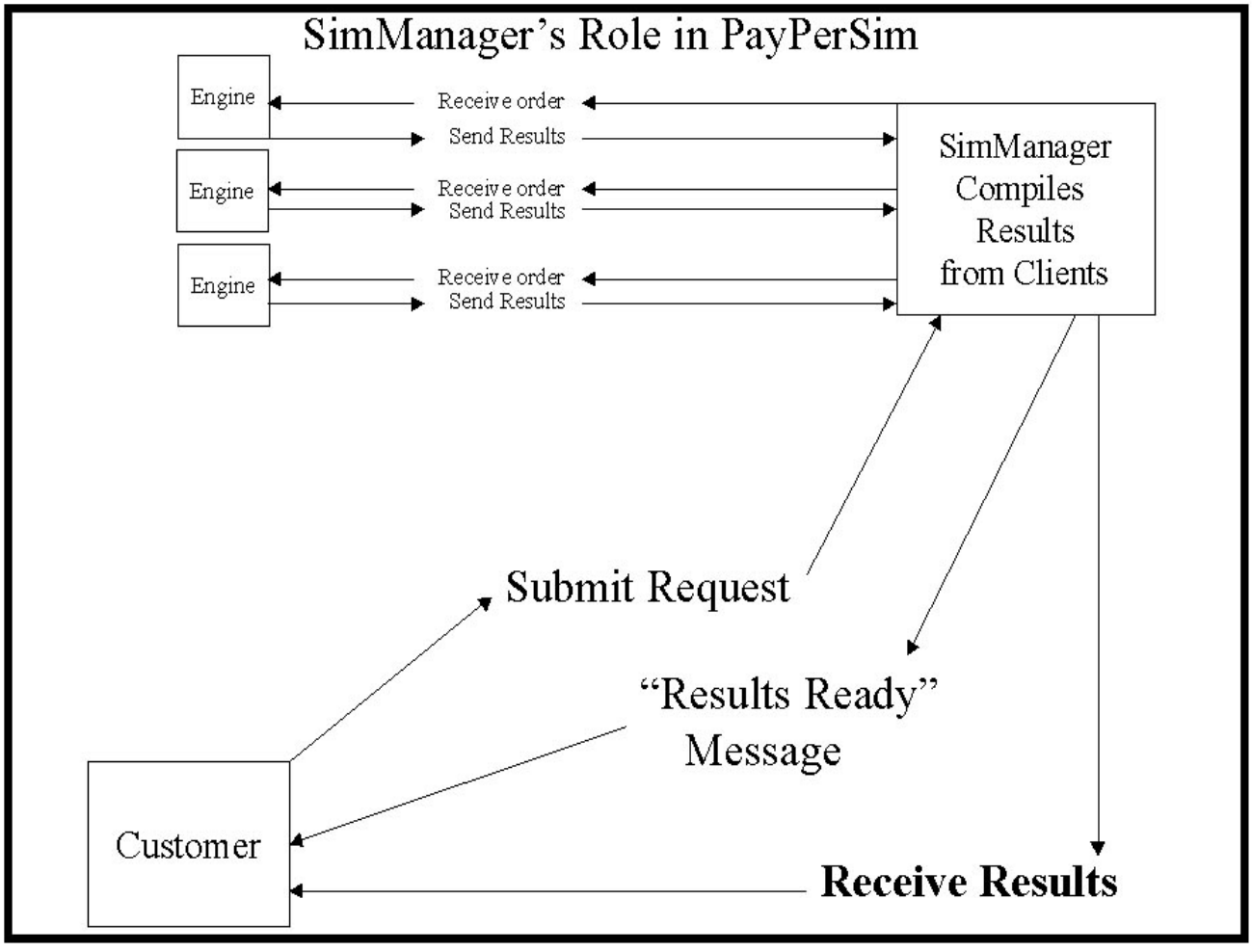


Figure 1: The Relationship Between the Simulation and the *SimManager*

## 2 AN APPLICATION SERVICE PROVIDER (ASP) CONCEPT

Figure 2 shows how *SimManager* would function in an *Application Service Provider (ASP)* environment. *SimManager* would serve as the interface between a customer seeking assistance in simulation modeling and the simulation system. As an *ASP*, *SimManager* would require a fee for service based on the total number of simulation trials executed with a given simulation model. When the customer accesses the web page, he/she would click on the *Catalog of Simulation Models* and view a description of any models that appear to meet his/her specific needs. The customer could print, at no charge, the information describing any simulation models he/she might be interested in. One of the published facts about each simulation model would be the cost of executing a single simulation trial with that model.

For example, the charge for the Silk Bake-Load model (Kilgore and Healy, 1999) might be \$0.20 per simulation trial, whereas the charge for a Silk model of a (s, S) inventory system might be \$0.10 per simulation trial. If the customer requires 100 replications of the Baked-Load model at each of 8 scenarios, the charge is C = \$0.20(8)(100) = \$160.00. If the customer finds this sum too expensive, he/she might choose to follow an iterative approach and initially order only 10 replications at each of the 8 parameter sets at a cost C = \$0.20(8)(10) = \$16.00. Then by examining the, say, 95 percent confidence intervals on the performance measures of interest and having *SimManager* compute the required number of replications using the sequential procedure discussed by Law and Kelton (2000), the customer could select a greater number of replications. In fact, the sample size models discussed in Law and Kelton (2000) would allow *SimManager* to select the minimum number of replications needed to achieve a desired relative error $\gamma$ ($0<\gamma<1$).

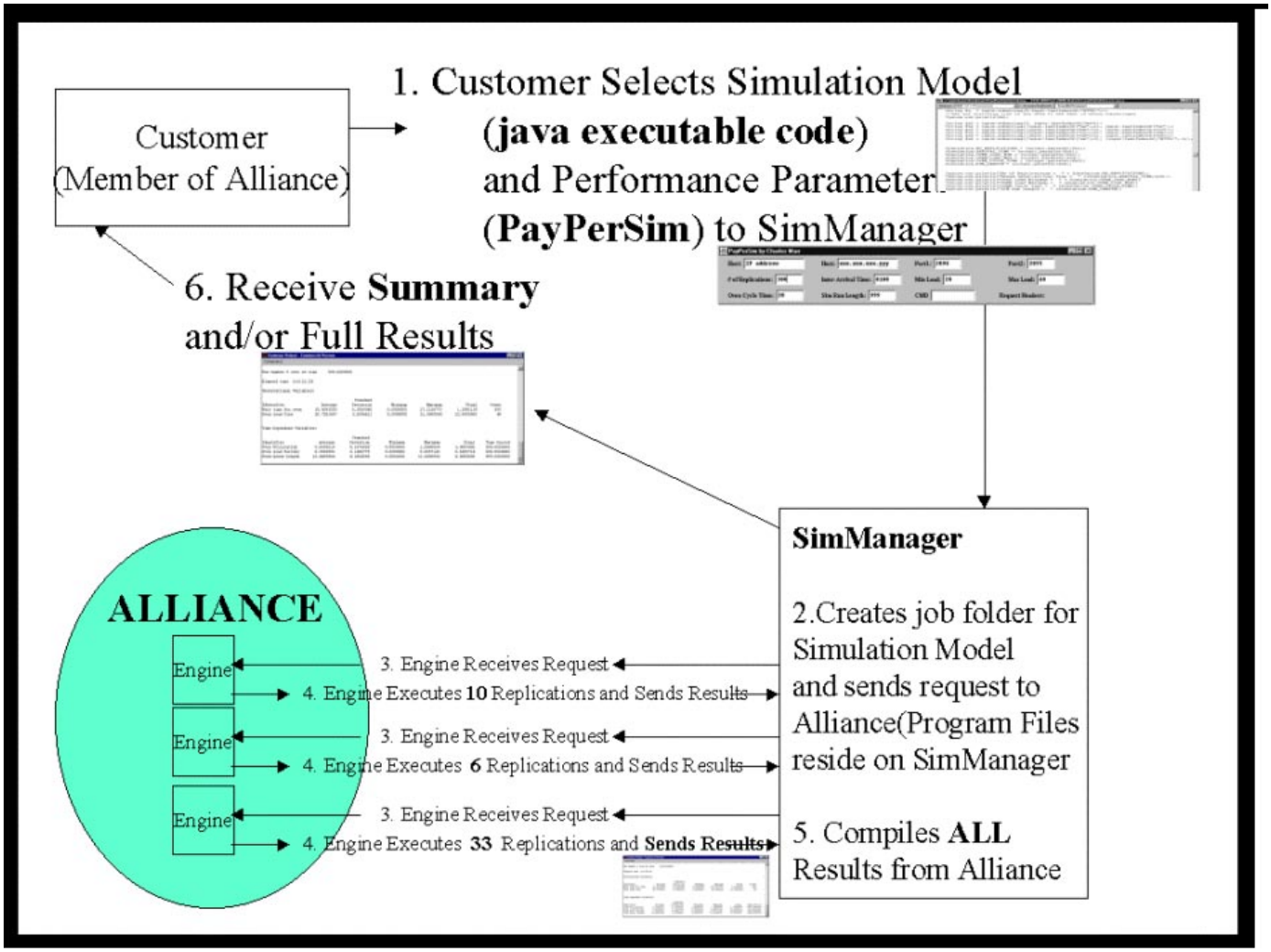


Figure 2: A Concept for *SimManager* Operation in an *ASP* Environment

## 3 AN ILLUSTRATIVE EXAMPLE

Using the Java-based Silk simulation system (Kilgore and Healy 1999) as a test platform, we were able to complete a simple simulation study for a complex single-server machine. The example involves a Silk-based simulation model of a bake oven that is used to heat treat computer chips in a microelectronics manufacturing process. This program was executed in a Visual Cafe environment (Symantec 1998). Our MBA student wants to determine the oven utilization and oven load percentage as his measures of performance for two alternatives. For both ovens the mean inter-arrival time of parts into the oven is 1.00 seconds. For Oven A the minimum oven load is 30 parts, with a maximum oven capacity of 40 parts. The cycle time for the oven is 30 seconds. The length of the terminating simulation run is 800 seconds. For Oven B the minimum oven load is 10 parts, with a maximum oven capacity of 25 parts. The cycle time for the oven is 20 seconds. The length of the terminating simulation run is 800 seconds These data were passed through the *Sim Manager* dialog box shown in Figure 3. Once we describe the execution of the simulation from this point we will return to our MBA student example to discuss the results.

## 4 *SimManager* OPERATION

The procedure for executing the distributed simulation involves four operations:

1. A customer, our MBA student, completes the data entries in the *PayPerSim* dialog box shown in Figure 3. These entries include the mean inter-arrival time, the mean service time, the minimum load size, and the maximum capacity of the oven, the simulation, and the required number of replications. This information is then sent as a customer request from the thin-client *Front-End* interface to the *SimManager* processor, currently shown as "a46324" in Figure 3.
2. *SimManager* receives this request for work and places it in the job queue. After determining what engines are available for work *SimManager* assigns this workload to the engine processor. This request is sent via a HTTP pass of "GET//...performance parameters..." It should be noted that here $S = 1$ and $R = 2$, so that there are $K = 2$ simulation trials to be carried out with one engine processor ($P = 1$). *SimManager* could have just as easily selected several engine processors (e.g., $P = 6$) and utilized multiple ports on each processor (e.g., $L = 5$), if the simulation study required it.
3. The engine acknowledges receipt of this work order by sending an echo check of the assigned set of input parameters. Had the simulation workload been assigned to several engine processors, each engine would have responded in this manner.
4. As the simulation runs, the engine collects raw data (See Figure 4), load size, wait time in queue, etc and saves them as arrays. Upon completion of the replication, this raw data is sent to the *SimManager* associated with the job number assigned.
5. Once all of the jobs associated with the customer request are complete, an e-mail message is sent telling him that the results are ready for pick up. The customer may then select to receive all the raw data, the summary results of each replication, a final report of data, or any combination.

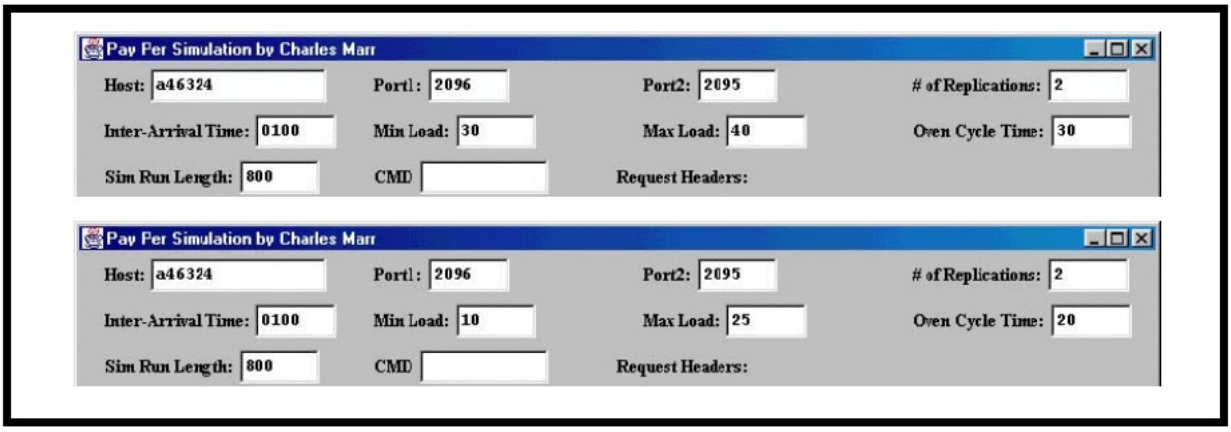


Figure 3: Front-End Dialogue Box for Input of our MBA Student Example

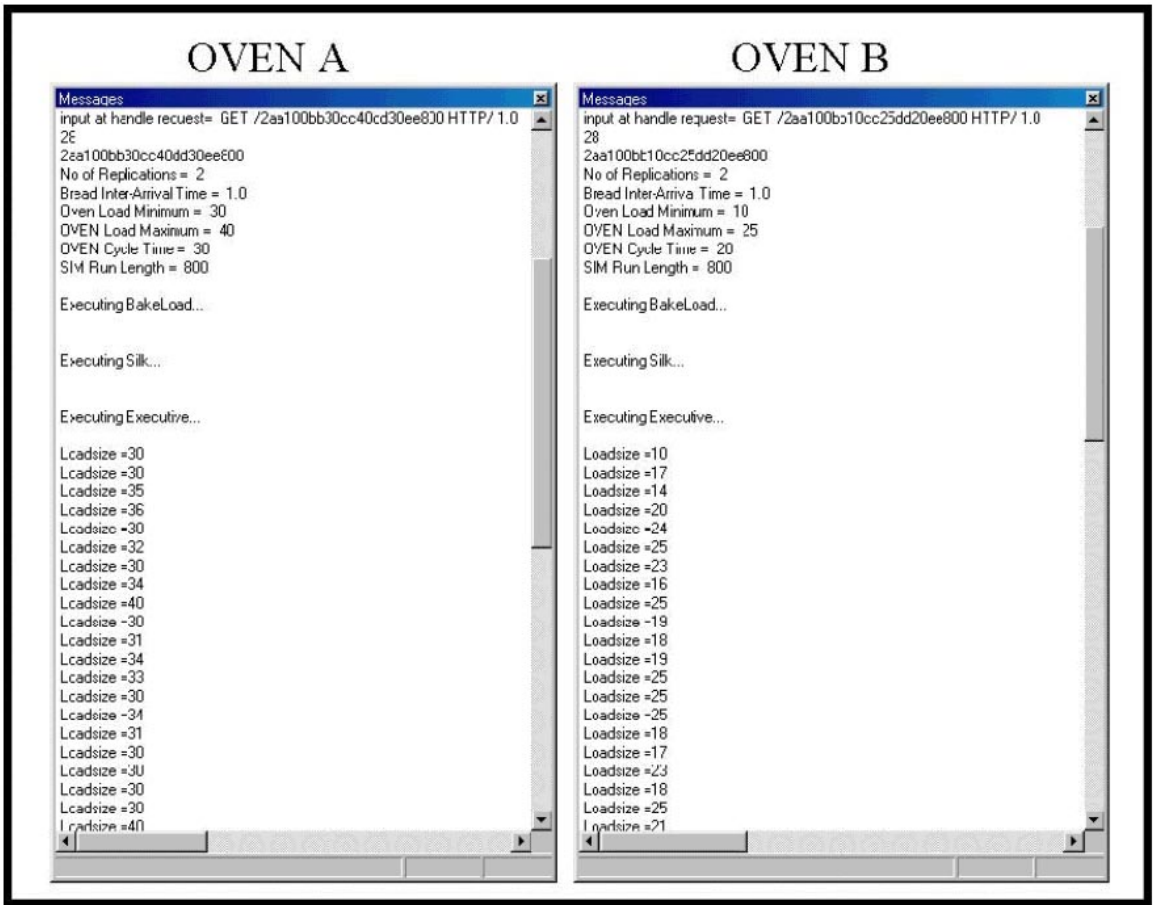


Figure 4: Engine Showing Performance Parameters and Raw Data Results for MBA Example

Detailed simulation results captured during the simulation run can also be recorded in a separate data file for each replication should the customer require it. The more likely project report, however, would consist of a set of summary statistics for *R* replications at each of the *S* sets of input values. This project report might include, for example, the mean, variance, standard deviation, minimum, maximum and a 100(1-α) percent confidence interval for each of the performance measures the customer has requested. In the *SimManager* environment, each engine computes these summary statistics at the conclusion of its work assignment, saves these results in arrays, and populates a database file back to the *SimManager*.

## 5 HOW THE *Alliance* OPERATES

The *Alliance* is simply a consortium of s*imulation engine* processors belonging to individuals and organizations that have agreed to participate in the *SimManager Application Service Provider* system. The "owner" is the individual who usually operates a specific *engine* processor. The *Alliance* of engine processors need not be co-located. They can belong either to an *intranet,* to the *internet,* or to a combination thereof. An engine is simply known to *SimManager* by its IP address; e.g., 123.456.789.111. *SimManager* also knows the performance characteristics of each engine, such as *processing speed* and *availability*. The owner of each engine has agreed to leave the processor "on" with the simulation application in an active window; e.g., VisualCafe (Symantec 1998) and Silk (Kilgore and Healy 1999). As long as the engine processor is "on" and has the simulation application active, it is available to the *SimManager* server.

There are three distinct phases of the *Alliance* operation. First, a customer initiates a request for a simulation study using the *Front-end* dialog box such as shown in Figure 3, thereby providing the specific parameters required for his/her project. These parameters will usually include the number of replications, the simulation run length, the random number seed vector, the mean entity inter-arrival time, minimum and maximum batch sizes, the mean service time, and, in the case of a steady-state simulation, a "warm-up" period. *SimManager* sends the necessary information to *P* selected engines.

An engine completes its assigned workload in some time length based on such factors as processor capability,

speed, and the current utilization by its "owner." For example, a high-end Pentium III 500Mhz computer is capable of performing a greater number of replications per unit time than is a 486-66 processor. However, if the PIII-500 processor is often busy (i.e., in use by its owner), it could well take longer to complete a simulation assignment due to its reduced *availability.* That is, when the owner is using the engine processor for some other application (e.g., word processing or surfing the web) the simulation task continues to run in the "background," but at a slower pace. Figure 5 illustrates how the execution time for a simulation trial is affected by the owner's use of the engine processor. The simulation activity is not curtailed, but its execution time per replication is lengthened.

## 6 KEY ENABLERS

The initial version of SimManager, later to become the Front-End portion, was simply a hand-held COMPAQ 810 that send a "GET" command from the pocket windows browser through a "hard link" serial cable to a DELL Insprion 7000. The Dell ran the SILK simulation "Baked Load" as an application running within Visual Cafe Professional Edition 3.0. The original "Baked Load" model was modified by adding in a WebServer.class and HTTP.class to the project. As well, the simulation class was modified to allow "passing" the performance parameters from the COMPAQ 810 to the Dell via an HTTP request. The request was echoed to the COMPAQ and the simulation began, with a user-required acceptance of the academic license from SILK. Once the simulation ran to completion, the Summary Results would post to the screen on the DELL. The "customer" COMPAQ 810 did not receive any feedback other than the initial echo of input parameters. The next step was to write a JAVA thin-client Front-End. This took the form of a stand-along executable file that would allow passing the performance parameters to any machine running the Baked Load or sSInventory Model, as long as the IP address and port # were known by the customer. The same "GET" command was sent to the simulation models, just in a more distributed form. On the DELL, the simulation ran within Visual Cafe 3.0 on different ports. This began our replications of many machines, many ports, demonstrating the capability of the Front-End to send jobs, work, to the Alliance. Each Port replicated a different simulation engine. Again the summary results were seen on the screen of the DELL, but no feed back to the "customer," now any computer with an Internet access.

Our next advance came with storage of RAW data within the simulation itself. For example, having the actual loadsize data instead of the mean and standard deviation allowed calculation of MIN, MAX and determination of the actual distribution of the loadsize, as well as the required warm-up time for steady-state. This data was saved as an array and later saved as a ZIP file on the simulation engine computer. This file was e-mailed to the SimManager for compilation with other simulation engines results to create the complete simulation study and creation of the response surface depicting the process over a range of performance parameters, (see Figure 6).

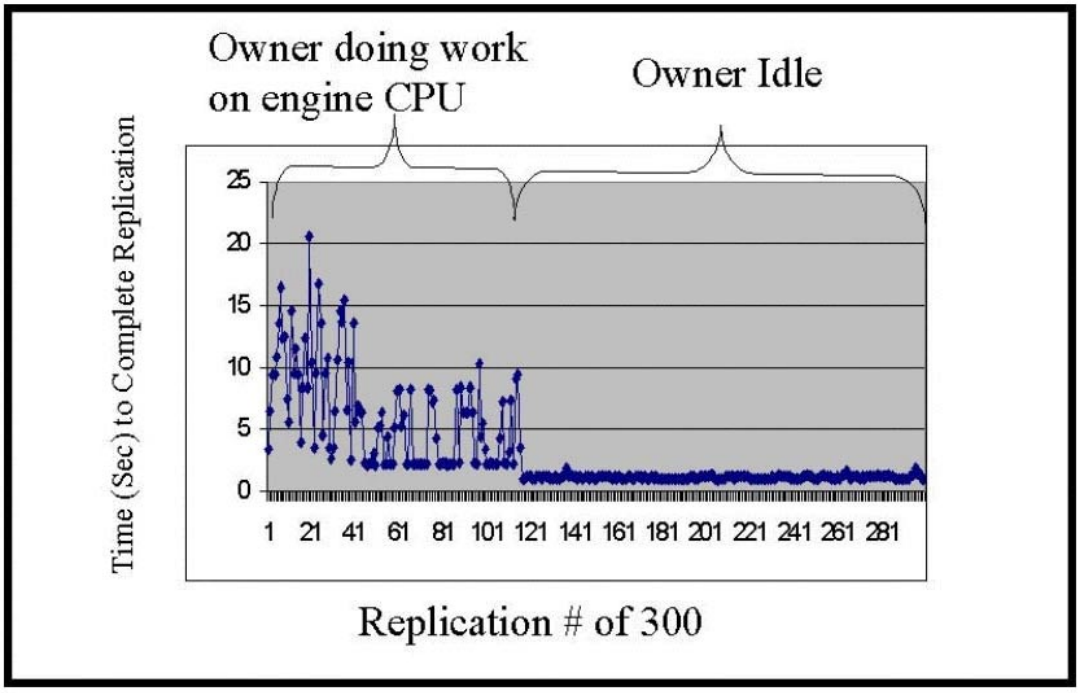


Figure 5: Illustration of Simulation Execution Time as a Function of "Owner" Activity

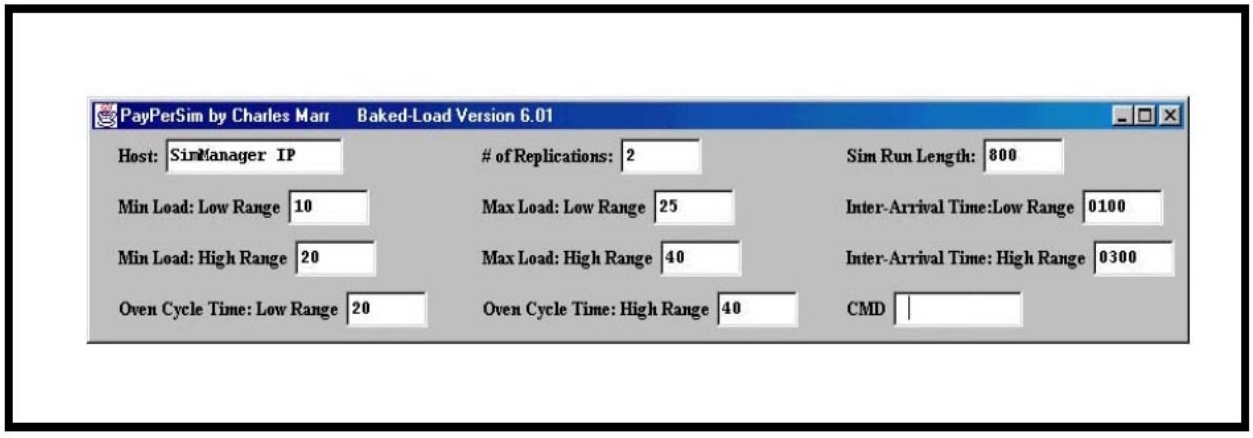


Figure 6: Front-End for Executing Baked-Load across a Range of Performance Parameters

The next phase of the project was to create an effective three-tier system. The first tier is the Front End. This became an applet running within a WebPage resident in a Cobalt Cube file Server. It has one simple task and purpose. Provide an entryway for choosing a simulation and filling in the applicable performance parameters. The next tier is the SimManager itself. The SimManager holds the available workforce or simulation engines current doing and available for work. As well, the SimManager determines the number of replications sent to each machine. Currently, this is done very simply and the control strategies must be further developed. The SimManager also is responsible for keeping track of the job status and sending a completion message to the customer. Additionally, the Cobalt Cube also holds the database of information. This is where each Jobs results, raw data and summary results are stored. Once the SimManager determines that the job has been completed a message is sent to the customer to retrieve the results. The customer may have selected to receive the raw data, summary results, or a complete report. Each requires more processing and thus incurs a greater charge. The third tier is where all the real work is done. The simulation engines send UDP messages to the SimManager upon activation and at periodic intervals "telling" the SimManger that it is ready to accept work and its current status (used to determine true availability).

## 7 SUMMARY AND CONCLUSIONS

The purpose of this article was to provide proof of principle of a concept for carrying out simulation studies on the World Wide Web. The main concept presented here was that web-based simulation can be exploited to run simulation trials in a parallel replications format on multiple engine processors, thereby significantly shortening the time required to complete a simulation study. Such a project can be completed in approximately the time required to complete a single replication.

There are three distinct phases of *SimManager* operation. First, a customer initiates a request for a simulation study. This request can be initiated on a web page with a dialog-box type of input form. The customer fills in the blanks of a *Front-End* dialog box with the performance parameters required for his/her simulation model; i.e., the number of replications needed, the simulation run length, the mean entity inter-arrival times, minimum and maximum batch sizes, and mean service time. These performance parameters are sent to a set of $P$ simulation engines and the replications begin. An engine processor completes its assigned workload in some time length based on such factors as processor capability, speed, and concurrent utilization by its "owner."

The greatest benefit of the *SimManager* concept is that a simulation replication runs from start to finish on one computer; that is, there is no need to engage in *distributed simulation* (Fujimoto 1998). Theoretically, if a sufficient number of simulation engines are available (i.e., $P \geq K$), the simulation study should require only the time of one replication plus the overhead time of sending the request and results across the Internet.

The next step in this research is to create a web interface to enable customers to actually gain access to commercial applications of Java-based computer simulation. A second phase would be to program the *SimManager* application in a more widely used development language, such as C++ or Visual Basic, thereby enabling the use of a greater variety of simulation models. In this way it may even be possible to gain access to simulation models coded in such languages as Arena (Kelton, Sadowski and Sadowski 1998) and Promodel (Benson 1997).

Other areas of future study include the following: (a) completing the development of the control strategies needed to determine the number of replications sent to an

engine processor (processor type, speed, current utilization, historical utilization, etc); (b) exploring variance reduction strategies on different ports of the same engine; and (c) completing an analysis of using different optimization strategies in connection with more complex simulation studies (e.g., Tabu search, simulated annealing, response surface methodology, and genetic algorithms). Further research will investigate the effect of running the same simulation model on different ports of the same engine, as compared to running it on the same port on each of several engines, in terms of the time required to report results to the *SimManager*.

## REFERENCES


Banks, J. 1998. Software for Simulation. In *Handbook of Simulation,* ed. J. Banks, 813-836.

Biles, W. E., and J. P. C. Kleijnen. 1999. A Java-Based Simulation Manager for Optimization and Response Surface Methodology in Multiple-Response Parallel Simulations. In *Proceedings of the 1999 Winter Simulation Conference.*

Fujimoto, R. M. 1998. Parallel and Distributed Simulation. *Handbook of Simulation,* ed. J. Banks, 429-464.

Heidelberger, P. 1988. Discrete-event simulation and parallel replications: statistical properties. *Scientific and Statistical Computing* (9):1114-1132.

Kelton, W. D., R. P. Sadowski and D. A. Sadowski. 1998. *Simulation with Arena,* McGraw-Hill, New York.

Kilgore, R., and K Healy. 1999. Introduction to Silk: A Java-based, process-oriented simulation system. Threadtec, Inc., St. Louis, MO, 1-15.

Law, A. M., and W. D. Kelton. 2000. *Simulation Modeling and Analysis.* McGraw-Hill, New York.

Pappalardo, D. 2000. ASPs still searching for the right mix. *Network World Fusion: News.*

Symantec. 1998. *Visual Cafe Version 3: User's Guide*. Symantec Corporation, Cupertino, CA.


## AUTHOR BIOGRAPHIES


**CAPTAIN CHARLES A. MARR** is a Graduate Student at the University of Louisville. He received his B.S.M.E and commission in the U.S. Army from Rose-Hulman Institute of Technology in 1989. He is simultaneously pursuing his M.S. and Ph.D. in I.E under the Army's Advanced Civil Schools program. His email address is <charles-marr@us.army.mil>.

**CAPTAIN CHRISTOPHER B. STOREY** is a Graduate Student at the University of Louisville. He received his B.S. in the United States Military Academy at West Point in 1990. He completed the M. S. in Computer Science from the University of Southern California and is currently pursuing his M.S. and Ph.D. degrees in I.E from the University of Louisville. His email address is <storeyc@knox-rotc.army.mil>.

**WILLIAM E. BILES** is the Edward R. Clark Chair of Computer Aided Engineering in the Department of Industrial Engineering of the University of Louisville. He received the BS degree in Chemical Engineering from Auburn University, the MS in Industrial Engineering from the University of Alabama in Huntsville, and the PhD in Industrial Engineering and Operations Research from Virginia Polytechnic Institute and State University. Dr. Biles is currently engaged in teaching and research in the areas of simulation methodology, rapid prototyping in product design, and automated manufacturing. He has authored more than 100 journal articles and conference papers, two books, and 15 chapters in books and handbooks. He is a registered Professional Engineer in Indiana and Kentucky, and a member if INFORMS, SME and NSPE and a Fellow of IIE. Dr. Biles recently spent four months on sabbatical leave at Tilburg University in the Netherlands, where he engaged in joint research with Dr. Jack P. C. Kleijnen on Web-based simulation. His email address is <webile01@gwise.louisville.edu>.

**JACK P. C. KLEIJNEN** is Professor of Simulation and Information Systems in the Department of Information Systems and Auditing of Tilburg University (Katholieke Universiteit Brabant) in the Netherlands, where he is also associated with the Center for Economic Research (CentER). He received his PhD in Management Science from Tilburg University. His research interests are in simulation, mathematical statistics, information systems, and logistics. Dr. Kleijnen has published six books and more than 130 articles. He has lectured at numerous conferences throughout the USA, Europe, Israel and Turkey; served as a consultant for numerous industrial and government organizations; and is a member of several editorial boards. He has spent several years with different universities and companies in the USA. Dr. Kleijnen has been awarded a number of fellowships, both nationally and internationally. His email address is <kleijnen@kub.nl>.