SIMALYTIC MODELING: A HYBRID TECHNIQUE FOR CLIENT/SERVER CAPACITY PLANNING

Tim R. Norton, Doctoral Candidate
Department of Computer Science
Colorado Technical University
Colorado Springs, CO 80907
tim.norton@simalytic.com

Keywords**: Simalytic, hybrid modeling, queuing theory, simulation, capacity planning**

Abstract

The Simalytic™ Modeling Technique* (from **Sim**ulation/An**alytic**) is a hybrid technique that addresses modeling and predicting the capacity requirements of computer systems in complex enterprise-wide client/server multiple-platform applications. This technique uses a general purpose simulation tool as an underlying framework and an analytic tool to represent application response times at individual nodes. The bridge between the two techniques is a transform function that will adjust the service time for a given server depending on the load at that server. It combines both platform-centric tools (limited features but detailed platform information) and general purpose tools (rich low level features) to address today's large heterogeneous enterprises. This methodology takes advantage of features in the different techniques (simulation vs. analytic queuing theory) as well as features in the different tools (platform-centric vs. general purpose) by defining the interface between the simulation framework and queuing theory node models. The benefits of using a hybrid technique are discussed and results are presented to show the validity of the Simalytic technique.

1. INTRODUCTION AND GENERAL BACKGROUND

The world of computing is changing at a very rapid pace. Systems that once would have been a single computer are now multi-platform. What were once batch applications are now on-line transaction processing client/server systems with GUI (graphical user interface) front-ends on PWS's (programmable work-stations) attached to departmental servers and mainframe repositories. These new application designs utilize the features and services of different types of computers (mainframe, mid-range, desktop) running different operating systems (MVS, Unix, OS/2, Windows, etc.) connected by a variety of communication network techniques (RPC, DCE, NFS, FTP, SNA, APPN, etc.) (Hatheson 1995; Wilson 1994).

As applications move into this new client/server world, how do we select the right systems at each level and, once selected, how do we insure those systems are the right size? If any one of them is too small the whole application will fail. If any are too big, the cost of running the application may exceed the revenue it generates. Neither is a very attractive situation.

The objective of capacity planning is to find the successful middle ground. Today, planning the capacity of large computer installations with multiple systems requires an understanding of not only the operating systems, the platforms, the clients, the servers, the networks, the transaction systems, etc., but more importantly, the applications and the relationships between them. Once those relationships are defined and understood, the application's performance can be assessed against the business objectives and goals. Projected business volumes are then modeled to predict the capacity required to meet those goals at future volumes.

There are many modeling tools and techniques that address both performance and capacity for each of the systems in today's multi-platform environment (Pooley 1995; Smith 1995). The Simalytic™ Modeling technique provides a bridge across these existing tools to allow the construction of an enterprise level application model that takes advantage of models and tools already in place for planning the capacity of each system. More detailed background information, including examples and business motivations, is provided in (Norton 1996).

1.1 Capacity Planning

The capacity of a system can be measured many different ways, depending on the business the system supports. Generally, the way a system is measured centers around the performance of one or more of the applications. The system "has enough capacity" if everything is getting done when it is needed. Capacity planning is making decisions about the resource requirements of a given computer system based on the forecasting of future application performance using the goals and expectations of the business. What do we have to buy and when do we have to buy it to make sure that the applications that run the business perform at the level required to insure the business succeeds?

1.2 Transaction Based Applications

Although there are still many important batch applications, this discussion will center around transaction based applications. Transaction processing systems, often referred to as OLTP (On-Line Transaction Processing), allow the end-user to enter a relative small independent unit of work into the system

* Simalytic™, Simalytic Modeling™, Simalytic Modeling Technique™ and Simalytic Enterprise Modeling™ are trademarked by Tim R. Norton All other trademarked names and terms are the property of their respective owners.

and receive some information as a response in near real-time. Transactions include entering an order at a terminal (business transaction), an SQL command (database transaction), some keystrokes followed by a carriage-return (interactive transaction)—whatever is meaningful from the end-user's point-of-view. Transactions can be counted to establish load (e.g. arrival rate) and measured to establish performance (e.g. response time). The responsiveness of the transactions associated with an application determine if that application meets the needs of the business. Projected business volumes are then modeled to predict the capacity required to meet the business goals at future volumes.

1.2.1 Client/Server Environment

Figure 1 A Sample "Client/Server" System shows a hypothetical client/server environment to illustrate the problem of modeling application performance. Which techniques and products are chosen doesn't matter, but what is important is understanding that any of the client applications on any of the PWS's can, and will, send transactions to several of the legacy applications to provide the end-user a screen of complete and interrelated information. For example, the Order Entry user may type in the name of an existing customer and get not only their address but any pending or past orders and the status of their account. This may provide better service, but it also causes transactions to be sent to each of the other systems.

Capacity planning in a client/server environment is much harder than in a single computer environment. In the example above, if the Shipping workload outgrows the Unix system, it can impact the responsiveness of the Order Entry transactions. In addition, growth in the Order Entry workload will now impact the Unix system, but only if the orders are from existing customers.

Modeling in this environment is a challenge because each of the systems requires a different knowledge base and expertise (Gunther 1995; Hatheson 1995). None of the systems can be modeled independently because the transaction arrival rate for one system may be dependent on the response times of the others. The client software on the PWS may issue transactions to several servers (send everything about customer #123) or it may have to wait for one response before sending the next (what is

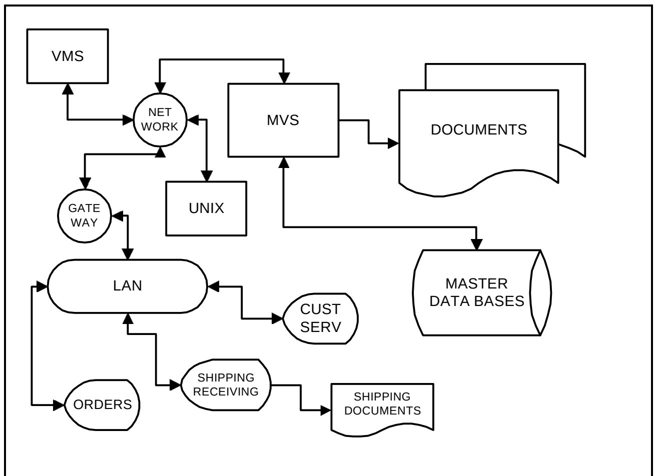
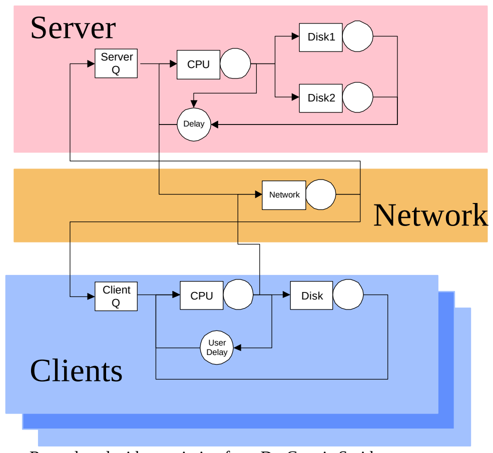


Figure 1 A Sample "Client/Server" System

Jones' customer number; then send everything about that number). While the former situation will cause the momentary peaks to synchronize on all of the servers; the latter will slow everything down as one of the servers becomes overloaded and its response times increase. "While it is important to be able to model specific UNIX or NT hardware, the problem we face is modeling the environment that has a diverse collection of hardware, operating system, database management system, and network hardware." (Domanski 1995).

Figure 2 An Enterprise Model shows a very simplistic model for each of the major areas of a client/server application



Reproduced with permission from Dr. Connie Smith, Performance Engineering Services.

Figure 2 An Enterprise Model

and, although it only shows a single server, the interdependence is evident. Reduced responsiveness of one part of the model (server, client or network) will have an impact on the other two.

1.3 Modeling Capacity Projections

The use of models to assess and predict the performance of computer systems is not new. Capacity planning has always relied to some extent on modeling because of the need to predict future requirements. A capacity planner can analyze the workloads and make predictions based on experience or simple trending. Models can be constructed to understand how an application functions without any intention to predict future performance. The area of interest here is the intersection of the two fields; models used to predict capacity requirements based on performance expectations.

1.3.1 Approaches to Capacity Planning Modeling

The approaches to capacity planning range from the application of rules-of-thumb to full scale benchmarks of the application or system (Brunetto 1984; Gilmore 1980; Hanna 1988; Mills 1991). *Figure 3 Capacity Planning Approaches* shows the relationship between these approaches. (This figure shows only the relative relationships.) Business Analysis (Rules-Of-Thumb) and Trends (Linear Projections) rely on historical analysis and the assumption that future performances is a direct extension of past performance. Application benchmarks can be the most accurate because they actually implement the applications, but at the greatest cost. Modeling is the middle ground

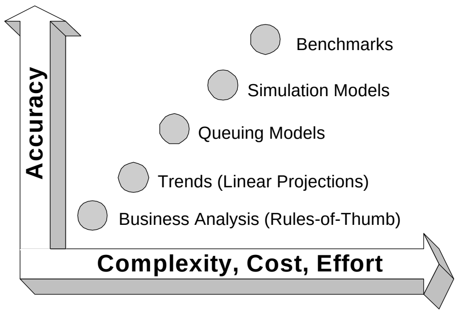


Figure 3 Capacity Planning Approaches

between the high cost and effort of benchmarks and the low prediction ability of trends.

1.3.2 Response Time Modeling

The key to the capacity planning methodology discussed so far is the ability to predict the performance of a future workload, given a desired system configuration. As applications move towards being transaction based, the definition of application performance becomes centered around transaction response time and modeling the response time becomes crucial to the ability to predict the future performance of that application.

There are two basic modeling techniques used for computer performance modeling: simulation and analytic queuing theory (Kobayashi 1981; Menascé, Almeida, and Dowdy 1994). Either of these techniques will build a model that represents the major components of the computer system to be modeled. A third technique, hybrid modeling, is the combination of both simulation and analytic techniques in a single model (Kobayashi 1981).

1.4 Modeling Tools

In addition to the choice between analytic and simulation tools, the capacity planner or performance analyst has the choice between platform-centric and general purpose tools. The basic difference between these two groups is the problem set the tools were designed to address.

1.4.1 Platform-Centric

Platform-centric means the tool contains detailed information about the platform, but does not allow more than one platform to be modeled at a time. For example, they would include information about the number and type of processors for each system in the model. Platform-centric models are generally easier to build because they are made of "building blocks" already defined to the tools, and the relationships between them are fully understood by the model. However, these tools cannot be used to model an environment not built into the tool. Although many platform-centric tools allow the user to define new servers with new performance characteristics, they generally do not provide large libraries of device and system definitions dramatically different from the supported platform. Platform-centric tools are *generally* implemented using analytic, or queuing theory, modeling techniques and process performance and configuration data collected from existing running systems.

1.4.2 General Purpose

General purpose means the tool contains the features to allow the user to model almost anything, but with little or no "built-in" understanding of any given computer platform. These tools are used to model more than just the hardware, including application design, traffic flow and communications protocols. System components are modeled using either a sub-model to implement the underlying architecture or a pre-determined delay value. Although many general purpose tools provide libraries of sub-models for a variety of systems and devices, they generally do not provide the required level of granularity, being either too general or too detailed for the situation. Building the relationships between the submodels is part of the overall model construction and may require an in-depth understanding of all of the submodels used, some of which are provided by the tool vendor in executable–only formats. General purpose tools are *generally* implemented using simulation modeling techniques.

2. SIMALYTIC MODELING METHODOLOGY

Simalytic™ Modeling (from **Sim**ulation/An**alytic**) is a hybrid modeling technique that uses a general purpose simulation modeling tool as a underlying framework and the results of an analytic modeling tool to represent the individual nodes or systems. The problem addressed by Simalytic Modeling is at the intersection of several areas: capacity planning, modeling (both simulation and queuing theory), client/server transaction processing systems, and commercial tools (both general purpose and platform-centric). The goal of a Simalytic Model is to predict the capacity requirements of an application executing on heterogeneous computer systems by creating an enterprise level application model.

There are two key differences between the existing modeling tools and the Simalytic Modeling methodology. The first is the ability to use the results from not only a different tool, but a different modeling technique altogether, as a submodel within an enterprise model. The second is the ability to use the results from tools or techniques already being used to model individual nodes in the system. These differences reduce the time and effort to build an enterprise level model by using the results from commercially available platform-centric tools or existing detailed application models.

2.1 Methodology

Simalytic Modeling brings together existing performance models and application information. Queuing theory models rely on averages, such as average response time, average service time and average arrival rate. These models are generally more efficient to execute than simulation models, but, because of the use of averages, their accuracy generally decreases as the data collection interval increases due to variability in the data. Simalytic Modeling allows the application to be modeled over longer periods of time to understand the application dynamics without increasing the error due to greater variation in the data items used for the above averages.

When using commercial queuing theory tools, it is generally understood that shorter intervals (the time period for which measurement data was collected to use in building a model) usually produce better model results because there is less variation in the measurement data. Trace-driven models are the most common in capacity planning and performance modeling because of the focus on existing systems. As an additional benefit, the trace data provides the transaction arrival distributions, which is often a major issue in model construction.

Simalytic Modeling is based on a hybrid technique that allows the models to use the best features of each tool. Sub-models allow some part of the model to be replaced with a different model, using a different technique, as long as it provides appropriate functionality and results; similar to the FESC (flow-equivalent service center) decomposition technique discussed in (Menascé, Almeida, and Dowdy 1994). A valid model (proven to produce accurate predictions) must exist for each system or node to be included in the application enterprise model. The application details must be understood, and consistently defined, at the enterprise level.

2.1.1 Methodology Process

An enterprise level model is constructed by starting with a very high level simulation model of the application, where each system is a single server. Then, instead of using a pre-defined service time, each server uses a transform function that maps the transaction arrival rates to service times. In the enterprise model, the service time and the response time for each server will be the same because the queue time is accounted for in the response time data for the server. Each node in the simulation model must allow enough parallelism to avoid queuing to enter the node.

Continuing with the example from section *1.2.1*, some number of the Order Entry transactions would be routed to the Shipping server. Assume it has been determined that Shipping can provide a response time of one second when arrivals are less than three per minute and a response time of two seconds when arrivals are more than three per minute. When the Order Entry transaction rate increases such that more than three per minute are sent to Shipping, the response time will jump from one to two seconds. This is an overly simple example, but it illustrates the point. The increased service time at Shipping will cause the overall response time for those transactions to increase, which will be seen as a longer average response time or reduced through-put for the application.

Figure 4 Simple Enterprise Model shows a diagram of this model. The response time is measured from Arrivals to Departures, either through the Shipping node or around it. This example shows how the Simalytic Model connects what is happening in the application on the different servers. If the Order Entry system is modeled by itself, the workload representing the long (Shipping) transactions would not reflect the increased response time due to the load at Shipping. Because of the additional application information in the Simalytic Model, it could adjust the service time in the Shipping server based on the current load, which will then be reflected in the Order Entry transactions that visit the Shipping server.

The next step is to analyze the model using the business objectives. Assume that the manager of the Order Entry department has requested a model to determine when the Order Entry system will need to be upgraded in order to maintain the required response time of less than 1.7 seconds. The arrival rate is assumed to have a constant increase over the next 18 months (the scope of the analysis) and the percent of the Order Entry transactions that also query the Shipping system is assumed to be 30%. The response time goal for the Shipping system is less than 10 seconds (because these transactions generally do not involve a waiting customer). The objectives of the analysis are to answer two questions: "When does the Order Entry system fail to meet the business response time goal?" and "What will fix the problem?".

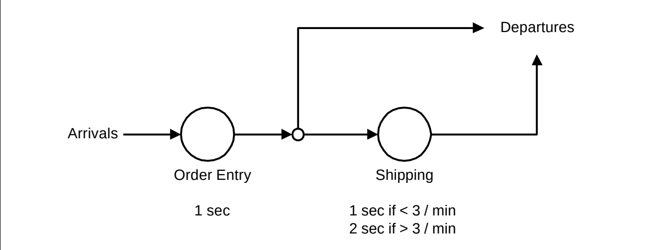


Figure 4 Simple Enterprise Model

Figure 5 Model Results Analysis shows the hypothetical results of this example model. When each of the systems are analyzed independently, neither of the response times ever approach the business goal of 10 seconds for the Shipping transactions and 1.7 seconds for the Order Entry transactions. However, when the relationship between Shipping and Order Entry is added to the chart in the form of results from a Simalytic Model, the revised Order Entry response times show that system will need to be upgraded by year end, well within the scope of the analysis. In addition, the 'fix' to the problem is to upgrade the Shipping system, which never exceeds its response time goal. The Simalytic Model allows the analyst to see the impact of relationships that, although known, may not be fully appreciated.

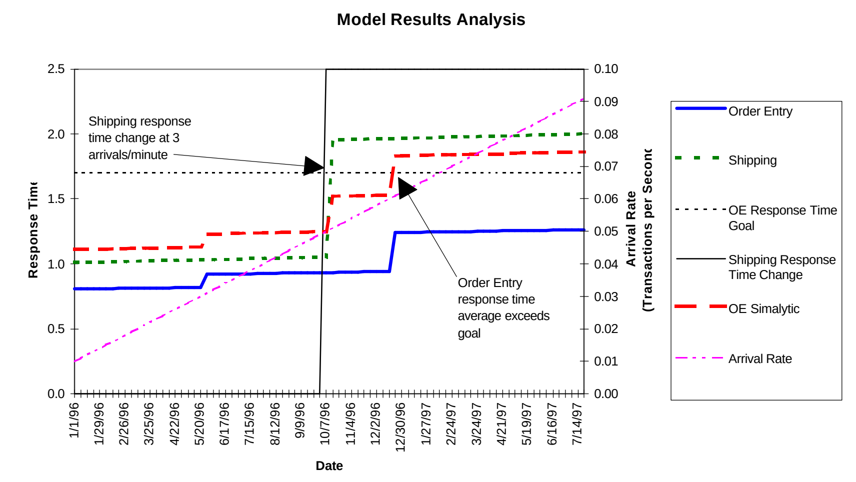


Figure 5 Model Results Analysis

2.2 Methodology Implementation

The formulae presented in this section are overly simplified to illustrate the relationships between them. It is assumed that the reader is familiar with the mathematics in these areas and, once the concepts are understood, will extend those relationships using more complex formulae appropriate to their problem domain.

The queuing theory mathematical formula for the average response time (T) of transactions in a simple single-server analytic model is shown in *Equation 1 Analytic Response Time Formula* from (Menascé, Almeida, and Dowdy 1994, p. 108) where S is the average time spent at the server, and l is the average transaction arrival rate. A detailed description of this formula and its application can also be found in (Buzen 1984).

Equation 1 Analytic Response Time Formula

$$T = \frac{S}{1 - l\,S}$$

The mathematical formula to estimate the average response time (T) of transactions in a simple single-server simulation model is shown in *Equation 2 Simulation Response Time Formula* from (Menascé, Almeida, and Dowdy 1994, p. 108) where T_i is the response time of the i^{th} transaction and n_t is the total number of transactions that visited the server during the simulation.

Equation 2 Simulation Response Time Formula

$$T = \frac{\sum_{i=1}^{n_t} T_i}{n_t}$$

The Simalytic methodology combines these two mathematical formulae into the Simalytic Modeling formula shown in *Equation 3 Simalytic Response Time Formula*. As the framework for a Simalytic model is a simulation model, we start with the simulation response time formula shown in *Equation 2*. The server time from each iteration (T_i) is replaced with function $f(l_i)$, where i is the iteration index and l is the arrival rate calculated from the interarrival time.

Equation 3 Simalytic Response Time Formula

$$T = \frac{\sum_{i=1}^{n_t} f(l_i)}{n_t}$$

The function f represents whatever is required to return the correct response time for each given arrival rate derived from proprietary algorithms implemented in a commercial tool. The simulation model invokes a submodel that implements the Simalytic function which returns the response time for the given l. Details of the foundation of the methodology are presented in other published works (Norton 1996).

Even assuming a best case situation where the interarrival times of the actual transactions are uniform (no variation), there would still be variation in the response times due to variation in the workload (different transactions, different calculations, different logical path, etc.) and due to variations in the device service times (CPU cache and pipeline, disk cache and rotation, interference form higher priority workloads, etc.). When using any queuing theory model, two assumptions must be accepted. First, that the average of each of the parameters over the data collection interval represents the actual system. Second, that the predicted results represent an acceptable average response time. As with the FESC (flow-equivalent service center) decomposition technique (Menascé, Almeida, and Dowdy 1994), the system to be modeled is divided into parts that can each be analyzed and solved independently. Each part is then replaced with a single server that is representative of the flow through that part of the system. The FESC must be solved independently and must maintain the 'flow' of the overall model (i.e. the behavior of each FESC must be almost indistinguishable from the subsystem it replaces). The subsystem model is solved to obtain the subsystem throughputs as a function of multiprogramming level (for closed models) or the subsystem response times as a function of arrival rate (for open models). The FESC is then substituted for the subsystem in the overall model using the function to describe the subsystem's behavior. (Menascé, Almeida, and Dowdy 1994, p. 236) cites (Cortois 1975) that little error is introduced if the transition rate within the submodel is much greater than the interaction rate between the submodel and the overall model (which will necessarily be the case when the submodel represents an entire independent system or node).

2.3 Preliminary Results

The preliminary results were produced using MathCAD (MathSoft 1995) to implement the three formulae. Both the service times and the interarrival times were exponentially distributed around the average shown because this matches the M/M/1 queuing model, used in most commercial platform-

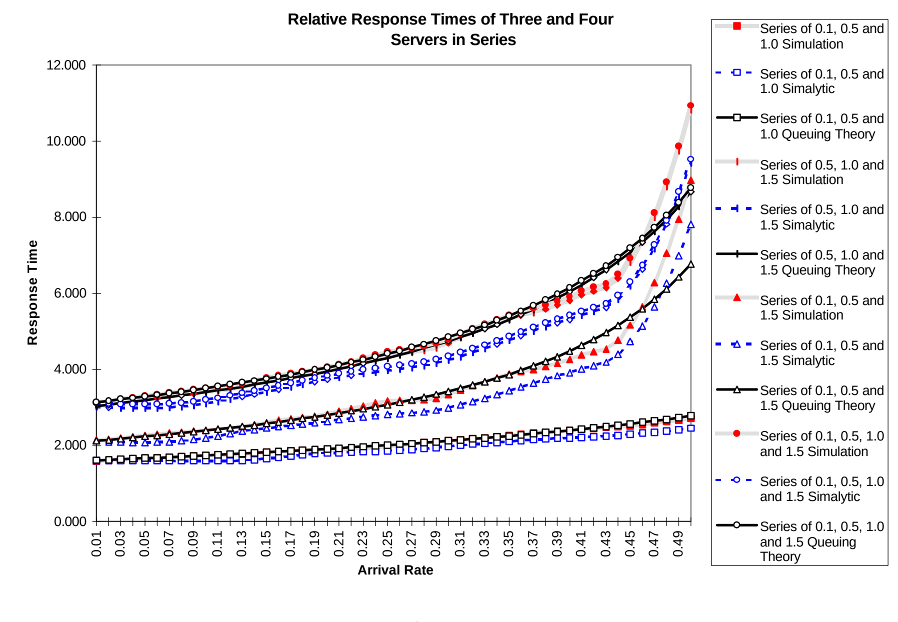


Figure 6 Response Time Comparison for Three and Four Servers

centric tools. Graphs were produced for single server scenarios with service times of 0.1, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0, 25, 50, 75 and 99 seconds, all showing a high degree of similarity for the three formulae. Graphs showing similar results were also produced for two, three and four server scenarios at a variety of service times using both series routing and probability routing. *Figure 6 Response Time Comparison for Three and Four* Servers is one of these graphs and shows the correlation between the three techniques for three and four servers in series at service times from 0.1 seconds to 1.5 seconds.

These results are acceptable within the capacity planning domain even with the simplistic Simalytic function that was used. The limited facilities available within MathCAD only allowed the function to use the interarrival times to calculate the response time. More sophisticated commercial simulation tools will allow the function to use other available information, such as the current queue length and server utilization.

3. CONCLUSION

Capacity planning for single-platform computer systems has developed over the years into a well disciplined field, but predicting the resource requirements in a client/server environment is still a challenge. Applications designed to exploit a client/server architecture greatly increase the complexity of both the computer system configurations and the applications themselves. Predicting the responsiveness of complex applications, at a reasonable effort and cost, requires a complex modeling methodology. Modeling an application at the enterprise level requires an understanding of the applications and measurements of the transaction response times. Different modeling techniques (simulation, analytic queuing theory or hybrid) and different modeling tools (platform-centric or general purpose) can be used to predict transaction response times for individual systems or nodes.

Simalytic Modeling can be used to take advantage of existing application and system models to reduce the time and effort required to produce detailed enterprise level application models. Using this technique will both improve the understanding of the application and identify which systems require more detailed analysis to determine if they will continue to meet the business needs without additional equipment. The implementation of the technique, using any of the many existing tools, not only protects the investment an organization has made in tool acquisition and training, but it also reduces the time and effort to produce a model that will predict the impact of business growth on the entire enterprise. The preliminary results show that a transform function can be implemented such that an entire computer system can be represented as a single node in a simulation model. Although better results are expected with future implementations, these current results, using a simplistic Simalytic function, are within the generally accepted error range for computer system capacity planning models.

3.1.1 Future Research

The author is currently implementing Simalytic Modeling using a number of commercial simulation modeling tools for the framework. This methodology will allow someone interested in predicting the performance of an application based on the anticipated growth in business transactions to construct a simulation model of the application using the results of commercial platform-centric tools to express the service times for each node.

4. ACKNOWLEDGMENTS

The author would like to thank Dr. Jeff Buzen and Mr. Rick Lebsack for their interest and in-depth critiques of early versions of this research. A special thanks is also expressed to Dr. John Zingg, Dissertation Committee Chair, for his insight and assistance.

AUTHOR BIOGRAPHY

Tim R. Norton is a Doctor of Computer Science (DCS) Candidate at Colorado Technical University. He holds an MMS in Computer Science from the University of Texas at Dallas. He is employed as a Senior Staff Member in Technology Planning at MCI Telecommunications. He has 20+ years computer systems experience, including applications design, support, systems programming, capacity planning and modeling. His current area of research is hybrid modeling of client/server computer systems.

BIBLIOGRAPHY

Brunetto, Anthony F. 1984. Benchmarking Decisions - A Management Tutorial. In Proceedings. Computer Measurement Group,: 755-761: CMG, Inc.

Buzen, Jeffrey P. 1984. A Simple Model of Transaction Processing. In Proceedings. Computer Measurement Group,: 835-839: CMG, Inc.

Cortois, P. 1975. Decomposability, instabilities and saturation in multiprogramming systems. Communications of the ACM 18 (7).

Domanski, Bernard. 1995. Capacity Management for Client-Server Architectures. Enterprise Systems Journal 10 (11): p78(6).

Gilmore, Martha R. 1980. Capacity Planning Methodologies. In Proceedings. Computer Measurement Group: CMG, Inc.

Gunther, Neil J. 1995. Performance Analysis and Capacity Planning for Datacenter Parallelism. In Proceedings. Computer Measurement Group: CMG, Inc.

Hanna, Carolyn. 1988. A Production Control Model Of On-Line Systems A Capacity Planning Overview. In Proceedings. Computer Measurement Group,: 592-598: CMG, Inc.

Hatheson, Amanda. 1995. Two Unix Client/Server Capacity Planning Case Studies. In British Proceedings. Computer Measurement Group: CMG, Inc.

Kobayashi, Hisashi. 1981. Modeling and Analysis: An Introduction to System Performance Evaluation Methodology. The Systems Programming Series. Reading, MA: Addison-Wesley Publishing Company.

MathSoft. Mathcad 6.0. MathSoft Inc., Cambridge MA.

Menascé, D., V. Almeida, and L. Dowdy. 1994. Capacity Planning and Performance Modeling: from mainframes to client-server systems. Englewood Cliffs, New Jersey: Prentice Hall.

Mills, Terry L. 1991. Capacity Planning For Managers: an Implementation Architecture. In Transactions. Computer Measurement Group: CMG, Inc.

Norton, Tim R. 1996. Simalytic Enterprise Modeling: The Best of Both Worlds. In Proceedings. Computer Measurement Group,: 1-12. San Diego, CA: CMG, Inc.

Pooley, Rob. 1995. Performance Analysis Tools in Europe. Informationstechnik und Technische Informatik 37: 10-16.

Smith, Connie U. 1995. The Evolution of Performance Analysis Tools. Informationstechnik und Technische Informatik 37: 17-20.

Wilson, Gregory L. 1994. Capacity planning in a high-growth organization. In Proceedings. Computer Measurement Group. Orlando, FL: CMG, Inc.