4. Simalytic Model Development

Up to this point, the Simalytic Modeling Technique has been implemented at a theoretical level using the mathematical analysis tool MathCAD (MathSoft 1995). In order for the technique to be practical and usable for modeling real systems, it must also be implemented with commercial modeling tools and be shown to produce acceptable results. This section presents the results of a series of models to show that a commercial tool implementation is sufficiently functionally similar to the MathCAD implementation to allow the commercial tool results to be accepted as the baseline. This section also presents the implementation process with the commercial modeling tools using the earlier Order Entry/Shipping application as an example. Section *5 Investigations into Simalytic Modeling* provides additional results for other scenarios based on that implementation process.

4.1 Simulation Tool Implementation

Simul8 (Visual) was used as the general purpose simulation tool to generate the baseline simulation results. The primary reason it was selected was because of its suitability as the framework in Simalytic Modeling (as discussed in section *4.2.1 The Simulation Framework* on page 101), but it is also an adequate tool to use to generate the baseline results. In addition, using a single tool provides consistency across the reported results and reduces the implementation time and effort.

Simul8 was used to construct a number of models based on the results as described in section *3.5.3 Validation of the Mathematical Foundation* on page 86. These models were built for each of the cases shown in *Table 1 Mathematical Formulae Results* on page 93. All of the Simul8 models are reproduced in section *7.1 Appendix A: Simulation Models* on page 154. *Figure 23 Example Simul8 Model* shows one of these models (slightly reduced). A separate model was built for each of eight scenarios (a series of one, two, three and four systems and one, two, three and four systems using routing). The percentage of transactions routed to each system in the routing scenarios is controlled by parameters set within the model. The results of each of these models are shown in a series of comparison charts reproduced in section *7.5 Appendix E: MathCAD Formulae Results Charts* on page *180*.

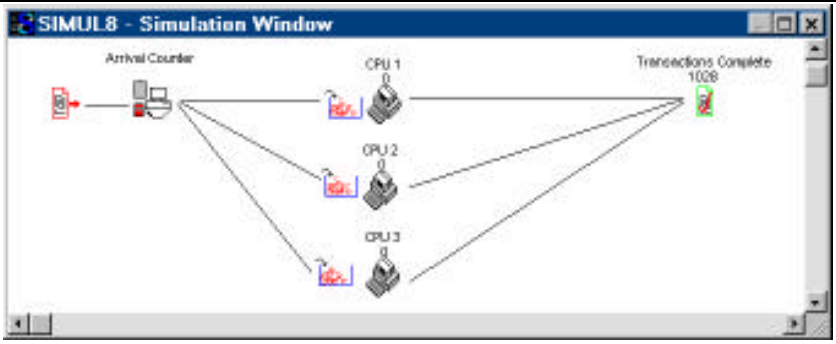

Figure 23 Example Simul8 Model

Figure 24 Example MathCAD/Simul8 Results Comparison (also slightly reduced) provides an example of one of

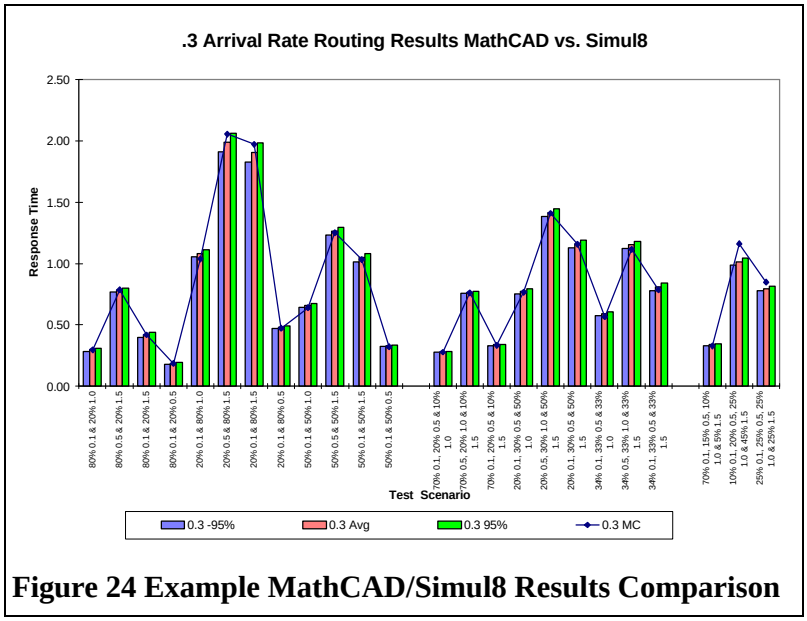

Figure 24 Example MathCAD/Simul8 Results Comparison

these charts for discussion here. Although there is a more detailed discussion of these charts in section *7.6 Appendix F: MathCAD/Simul8 Comparison Results* on page 204, this discussion of the chart in *Figure 24* will center on the relationship between the Simul8 results and the MathCAD results, rather than the absolute values of either set of results. *Figure 24* is typical of the results comparisons between the two modeling implementations. The chart shows results for two, three and four server routing scenarios (separated by a blank space) for an arrival rate of 0.3. Each group of three vertical bars represents the results of one modeling scenario implemented with Simul8. There are three bars because Simul8 produces response time results for the ±95% confidence limits in addition to an average. The line on the chart represents the results of the MathCAD implementation from *Table 1 Mathematical Formulae Results*. The format of the chart is somewhat non-conventional in that the line does not represent a continuous series of related values. A line was used to connect the independent values for several reasons: to allow the reader to easily find each data point, to visually distinguish the MathCAD results from the Simul8 results, to provide an overall impression of the shape of the MathCAD results and to allow easy comparison between the two different groups of data points. It is not necessary to read the detailed numbers and scenario descriptions in *Figure 24* to see the high degree of correlation between the two implementations (please refer to section *7.4* for the detailed information). In most cases, the MathCAD result is very close to the Simul8 average response time and between the ±95% confidence limits. The few cases where the MathCAD results are significantly different are assumed to be caused by distribution variations in the MathCAD results because they were generated by single run trials. This assumption was supported when multiple run trials of selected MathCAD models generated results con-

sistent with the Simul8 model results (please refer to section *7.4* for additional information). These results establish the Simul8 models as synonymous with the MathCAD models. Therefore, it can be assumed that the validity shown for the MathCAD models in section *3 Simalytic Modeling Methodology* also applies to the Simul8 models. In addition, only a representative subset of all possible scenarios are required to be modeled using Simul8 to establish a significant baseline for comparison with Simalytic Models.

4.2 Construction of a Simalytic Model

Simalytic Modeling is a technique for modeling applications running on client/server computer systems. Thus, there must be a formal process to implement the technique. This section defines that process and describes the phases of the development of a Simalytic Model using commercial tools. It includes the construction of an actual model using the Order Entry example presented initially in section *1.6 Hypothetical Company Example* on page 13.

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, the Simalytic Function™, that maps each transaction arrival rate to a service time. As the simulation model is run, the service time dynamically adjusts at each node depending on a combination of transaction arrival rates for the application and the other work at the node.

The Simalytic Modeling Technique has been designed to be independent of the actual modeling tools used. The tools that were selected for the example models in this, and the following, sections were chosen because they are simple, inexpensive, readily available and provide a reasonable development and reporting environment.

4.2.1 The Simulation Framework

The simulation tool selected for use in construction of the framework of the initial Simalytic Model is Simul8 (Visual), a general purpose simulation modeling tool developed primarily to model manufacturing situations. This tool was selected because it in an inexpensive simulation tool, with the required major features, which executes on a Microsoft Windows workstation. One of the most important of these features is the ability for Simul8 to interface to both Microsoft's Excel spreadsheet product and Microsoft's Visual Basic product, either of which can be used to implement the input distribution data, the analytic queuing submodels or look-up tables for results of such submodels. Excel can also be used to collect and compile the model results for more formal presentations. This flexibility allows the initial models to be implemented with simple transaction distribution data, which will be replaced with more interesting data as the model develops.

4.2.2 The Queuing Theory Nodes

The performance characteristics of the nodes are developed using the results of an analytic modeling tool specifically designed to model computer systems. The queuing theory tool selected is OpenQN because it is easy to use, fast and included with Dr. Menascé's book (Menascé, Almeida, and Dowdy 1994). It is a simple Pascal program that reads an input file of workload parameters and produces a report. This tool is easy to use and can generate a number of different reports very quickly.

4.2.3 The Simalytic Function

Finally, Microsoft's Visual Basic provides a rich programming environment and an interface to Simul8 to implement the Simalytic Function™. This interface was one of the main reasons Simul8 was selected for the research in Simalytic Modeling. Either Visual

Basic programs or Microsoft Excel spreadsheets can be used either for each server's service time or for the transaction distributions. In addition, either the programs or the spreadsheets can make subroutine calls to the model to get current state information or to control the model itself. Although not as sophisticated as many of the client/server tools available, Simul8 provides a simple to use GUI interface in addition to excellent extension capabilities to implement a Simalytic Function.

4.2.4 Steps to Build a Simalytic Model

As with any modeling effort, creating a Simalytic Model requires more than just putting the pieces together in some modeling tool. A substantial amount of information is required about the applications and systems involved. This section presents all of the steps to build a Simalytic Model. The major phases to creating a Simalytic Model are:

1. Workload Analysis.
2. Node Models.
3. Simulation Model.
4. Simalytic Model.
5. Model Analysis.

Each of these phases is discussed in detail in the following sections. This list is not meant to be comprehensive for all of the phases, but to document the process and to provide the modeler enough understanding to use that process. The most critical step, Workload Analysis, is a very complex and involved process, and only some of the issues involved, those that relate directly to the construction of a Simalytic Model, are discussed here. Other areas, such as calibration techniques for queuing theory tools and features of simulations tools are assumed to be covered in the training and documentation for the specific

tools. As with any modeling activity, once the Simalytic Model has been created, it must be used productively. Generating the speculations, planning the scenarios and analyzing the results in terms of application impact are all activities the experienced modeler should be very comfortable with within the context of the modeling tools being used. Simalytic Modeling requires the same level of analysis and presentation once the model has been completed and calibrated.

4.2.4.1 Workload Analysis Phase

In the Workload Analysis Phase the modeler collects information about the application to be modeled. This includes identifying, defining, documenting and measuring the application. This phase includes the same type of workload analysis done for system level modeling efforts, but it must be done for all of the systems supporting the application. It also includes collecting additional information about the application from the enterprise point-of-view.

Identify: Identify the workload. Identifying the workload to be modeled is often the most difficult step of any modeling activity. Because Simalytic Modeling takes an enterprise view of the application, the identification process is even more difficult. Not only does the application need to be identified for each system, but it must also be identified at a global level. How a workload is identified differs between platforms and depends on the database and middle-ware used by the application. A CICS transaction using DB2 provides different data collection than a Tuxedo transaction using Informix. A totally in-house developed application may provide better or worse data collection, but most certainly different. Although workload identification is done on each plat-

form, it cannot be done independently. How the workloads are correlated across the platforms must be considered during identification.

Start by identifying what the end-users think of as the business transaction. This is no longer a single CICS screen, but it is often a series of prompts and replies that, taken together, make a single activity such as entering an order. Regardless of the modeling technique used, workload analysis is very complex and requires substantial effort. The point of this step is to understand the objective, which is to define business activities, such as orders entered, in terms of measurable work elements, such as transactions A, B and C. The workload projections and response time measurements are at the business level and the models are built at the IT (Information Technology) transaction level.

This step must be done in conjunction with both the application developers and the end users. It is a series of trade-offs between what the end users would like to use and what is realistic considering how the application works. For example, if new orders and inquiries of existing orders are done from the same screen using the same transaction, it may not be possible to separate them into different workloads.

Document: Document the application topology. What transactions are routed where under what conditions? Are the IT transactions (i.e. CICS or Tuxedo) serialized or are some executed in parallel? Is the client/server architecture 2-tier, 3-tier, a combination or something altogether different? The documentation technique used should be whatever best supports the application and has the support of the users and developers, who must maintain the documentation.

The objective of this step is to produce a topology description of the application that can be easily and accurately translated into a simulation model. When this step is completed, the modeler should be able to track a business transaction from the originating workstation through the entire environment (including all splits, protocol translations, routing decisions, etc.) back to the same workstation. Every decision point must be identified and the criteria documented. If only some transactions are routed to a server, then how those transactions are selected (e.g. what value of what data field) must be documented.

Measure: Measure the workload. One of the key enablers for Simalytic Modeling is the ability to measure the application both from the business point of view as well as at the system level. The overall Simalytic Model can be calibrated only if the responsiveness of the business transactions can be measured. The node level model can be calibrated only if the IT transactions can be measured. The measurement of the IT transactions is generally already implemented for the node level models currently being used. However, in today's client/server environments, the measurement of the business transaction is very difficult.

In this step, the modeler must determine the ability to measure the application and workloads at each system and from the end-user point-of-view (sometimes referred to as end-to-end response time). It is assumed here that the node level measurement data are readily available. At the very least, the modeler must have additional information about the number, frequency and response times of the business transactions. If the application or the system does not collect the data, the modeler may need to observe the application users and collect the data manually. Although far from providing the

quality of data most modelers have come to expect from today's systems, manually collected data will allow the modeler to produce enough results to hopefully encourage the application designers to generate the required data on an on-going basis.

The objective of this step is to determine the feasibility of the modeling effort. If adequate measurement data cannot be collected then the value of continuing must be assessed to determine if sufficient interest will be generated by the effort to cause changes that will increase the quality of the measurement data. Another possible approach would be to use the Simalytic Model to establish reasonability bounds for each node based on overall application performance. However, how well such results will be accepted will vary by organization and company.

Correlate: Correlate the workload across systems. The final step of Workload Analysis is to determine the correlation between the workloads at each system. Does the definition for a workload at one system really mean the same thing as that workload at another system? There cannot be any additional or missing transactions. For example, if workload W1 is defined as three transactions, A, B and C, then the measurement of W1 at system S1 must include all of the transactions A, B, and C that are routed to S1. In addition, it cannot include any other transactions that run on S1 not included in the overall definition of W1, such as transaction D. This appears to be a straight-forward requirement, but it becomes very complex as the client/server environment grows and applications attempt to reuse functions. For example, an existing legacy application transaction might be invoked by more than one client/server business transaction to look up customer information. This transaction would need to be included in multiple workloads, which would cause error in the model. This situation cannot be resolved

without some additional application modification to enable collecting data about which workload invokes each transaction.

The objective of this step is to insure the consistency of the workloads across the entire enterprise model. Problems, such as the one mentioned above, need to be identified, documented and resolved with data collection, application changes, model changes or simplifying assumptions.

4.2.4.2 Node Models Phase

In the Node Models Phase, the modeler models all of the systems supporting the application. This phase includes the same type modeling done for system level modeling efforts, but coordinates the node level models to integrate with the additional information about the application from the enterprise point-of-view.

Build: Build a model of each node. Building a model of each node used by the application is not significantly different from existing system level modeling efforts. Whatever tool is currently used to model each system should be used for that node in the overall model. The major difference is that the workload definitions used in the node models are those developed in the prior Workload Analysis phase. From a realistic standpoint, only the application of interest should be modeled as an identifiable workload with response time predictions. All other activity at each node should be included only to model resource usage and correctly influence the workload of interest.

The objective of this step is to build a model of each system, but to also take advantage of any existing modeling efforts. Although the workload definition may change, the processes already in place to collect measurement data and calibrate the

models, and possibly some of the actual models, for any of the nodes can be effectively reused.

Calibrate: Calibrate the model of each node. The node level models must be valid (calibrated and verified) before continuing. This can be time consuming for a complex application if there are a large number of nodes involved. There must be a high degree of confidence in the predictive nature of each of the node models. Because the Simalytic Model will connect the nodes together using the workload definitions, an error or poor results from any one node model can impact the accuracy of the entire application level Simalytic Model.

The calibration techniques used are dependent on the modeling tools and are much too complex to discuss here (additional information about model calibration can be found in (Menascé, Almeida, and Dowdy 1994 304-339)). In addition, care must be taken to insure that steps taken to calibrate one node do not contradict assumptions made in a different node model.

The objective of this step is to have a solid predictive model for each node that presents a consistent view of the application across all nodes.

Run: Run the models. Develop a profile of the application by running each of the node models for a series of arrival rates from very low (i.e. 0.01 or 0.001) to very high (either the model saturates or the 'knee' of the response time curve is well established). The actual arrival rates used will depend on the application and should make sense to the actual users of the application. The arrival rate increment should be as fine as practical considering the time and resources required for each execution of the model. If the arrival rate range is 0.01 to 0.09 then the increment should be something

like 0.005 or 0.001. If the arrival rate range is 0.1 to 10.5, then the increment should be larger, like 0.05 or 0.1. The increment does not have to be uniform; use a larger increment when there is little change in the response times between arrival rates and use a smaller increment when there is a large change. The number of node models that must be run depends on the desired accuracy and characteristics of the application workload. Some experimentation may be required to determine if additional node model results are needed. If the Simalytic Function used is a step function then the number of node model results must be great enough to insure that the difference between steps is not greater than the desired margin of error.

The objective of this step is to establish a response time curve that can be used to extrapolate the response time when presented an arrival rate not modeled.

Create: Create a model results table. Create a table of arrival rates and response times for each system for the workload of interest. Other workloads on the system will not be modeled in the Simalytic Model but are still accounted for in the node level system models. A key assumption here is that the other workloads provide a consistent load on the system and thus a consistent level of interference to the application workload being modeled. If this is not true, then the table must be extended to include some external parameters, such as time of day, and the additional response time values based on the combination of those parameters and the arrival rate.

The objective of this step is to characterize the application performance and responsiveness. The information in this table will be used to create the Simalytic Function when the Simalytic Model is constructed.

4.2.4.3 Simulation Model Phase

In the Simulation Model Phase, the modeler builds an overall model of the application with each of the systems supporting it represented as a node or server. This phase uses the information from the Workload Analysis Phase to connect each of the systems together to provide the enterprise view of the application.

Build: Build an overall model. Using the simulation tool of choice, build an overall model of the application with a single server for each node in the system. This model is defined by the application topology documented in the Workload Analysis stage. It identifies what transactions are routed to which nodes under what circumstances. This overall model of the application can be built before any of the node level performance data has been collected. Make assumptions as to the expected or desired performance at each node and use the model to identify how sensitive the application is to changes in the performance of any given node. If large variations in service time at a node have only minimal system response time impacts then that node level model may be deferred. The prior phase determined how the individual nodes behave, where this phase determines the impact of that behavior on the system as a whole.

The objective of this step is to build a model of the application that represents the overall application behavior across the enterprise.

Set: Set the overall model parameters. Set the service time of each node to the lowest response time in the table created above. Set each node to have enough servers so that there is ***no*** queuing at any node. How this is done will differ with each of the simulation tools. Generally, it is some type of replication factor within the node. The value must be very high so that there is never any queuing to get a transaction through

the node. It is generally not a good idea to create multiple nodes because of the problems that creates with transaction routing. 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. (The service time of each node in the simulation model is the response time from the analytic model of that node, which is a combination of queue time and service time in the analytic model.)

The objective of this step is to set the simulation model such that the response time at any node can be controlled by the Simalytic Function when it replaces the static service time in a later phase. In addition, the simulation model at this stage can be used to verify the application topology and conduct sensitivity analyses of user expectations.

Calibrate: Calibrate the overall model. Calibrate the simulation model against the end-user response time for a very low arrival rate and verify that there is no queue time at any of the nodes. Because the response time from the queuing theory tool includes the queue time in the node, any queue time in the simulation model will, in effect, double count the queue time. The simulation tool is being used to control the flow and routing of transactions, not calculate the queue time. This step should insure that the topology and routing information is correct before too much effort is spent developing the Simalytic Model.

The objective of this step is to verify that the simulation model accurately reflects both the topology of the application and the response time seen by the users at very low arrival rates.

4.2.4.4 Simalytic Model Phase

In the Simalytic Model Phase, the modeler incorporates the results of the system models into the overall model of the application. This phase uses the information from the Workload Analysis Phase and the Node Models Phase to provide the predictive capabilities to the enterprise view of the application.

Create: Create the Simalytic Function. Using the table of response times and arrival rates created from the node models, create a Simalytic Function for each node. This can either be a look-up table or a formula derived from the curve established by fitting a line to the response time data. The details of the function and how it is implemented will depend on the simulation modeling tool used for the overall model framework.

The initial function is implemented by converting the interarrival time between each pair of transactions to an arrival rate and thereby to the associated response time (Norton 1996; Norton 1997a; Norton 1997b). It is most likely that the initial function will not provide accurate enough results due to issues such as arrival distributions. This is because of the difference between the individual transaction nature of the simulation model and the averaging effects of the analytic node models.

The function will need to be enhanced to include additional information about the state of the node for each transaction. The complex function is referred to as the Simalytic Function because it includes not only the results of the node models, but also the additional features to select the most appropriate result for each transaction visit. To do this, the arrival rate used will more than likely need to be modified by some technique. One approach is to maintain a rolling average over some number of transactions which, in effect, segments the modeling interval into smaller intervals where

the model can be more responsive to changes in the arrival rate distribution. The number of transactions included in the average needs to be small enough to maintain the responsiveness of the model to workload changes but large enough to minimize the influence of isolated instances of very small interarrival times (very large interarrival times are not an issue because they cause the response time to approach the service time, an acceptable response time, while very small interarrival times cause the response time to approach infinity, an unacceptable response time). Another approach would be to examine the node to determine the current number of transactions being serviced or the node utilization, then select an arrival rate more consistent with that node state. Although not formally explored here, increasing the number of node level model runs (i.e. the number of arrival rate/response time steps used in the Simalytic Function) and refining the number of arrivals over which the interarrival time moving average is calculated appear to improve the accuracy of the Simalytic Model, but at the expense of increased construction effort and execution time. The technique chosen is a trade-off between rapid development and model accuracy. Any combination of techniques can be used. Implement the simplest Simalytic Function possible and enhance it as required to achieve the desired level of accuracy based on the business requirements. If the objective of the modeling effort is to provide a high level understanding about how the application behaves, then a simple Simalytic Function, and hence a less accurate model, might be acceptable. On the other hand, if a detailed understanding of the application's behavior across some or all of the nodes is required, then a more complex Simalytic Function will be needed to accurately represent the application.

The objective of this step is to create a function for each node that accurately reflects the application's behavior. Each Simalytic Function must return a value for the service time of the node for each visit of a transaction based on transaction interarrival time and other node state information.

Replace: Replace the static service times. Replace the service time for each node with the function created in the prior step. Again, how this is done will differ with each simulation tool. For example, some simulation tools support load dependent service times and the response time values can be entered into the load dependent service time table. However, this technique may not be viable if a more complex function is required and the load dependent server cannot implement a Simalytic Function.

The objective of this step is to implement the Simalytic Function in each node of the overall simulation model. The service time used for each transaction visit is the value returned by the Simalytic Function.

Calibrate: Calibrate the Simalytic Model. First compare it against the prior simulation model for the very low arrival rate to insure the overall model structure is still correct. The overall response times should be very close to the simulation model assuming the transaction arrival distribution and routing is the same between the models. Next, compare it against known user response times for known arrival rates. If the model does not produce response times acceptably close to the known user response times, then either the model framework or the Simalytic Function or both will need to be adjusted. At this point, calibration of the Simalytic Model is very much like the calibration of a simulation model. All simplifying assumptions must be re-verified and the area of the model causing the inaccuracy isolated. If the inaccuracy is associated with

the performance of a node in the model, as opposed to a relationship between nodes, then the node model results and the Simalytic Function must be examined for correctness instead of the activities associated with the calibration of a simulation submodel. Enhance the Simalytic Function as required to get the necessary level of accuracy. The objectives of the modeling effort will determine what is an acceptable level of accuracy.

The objective of this step is to insure that the Simalytic Model provides the required level of application prediction within the required level of accuracy.

4.2.4.5 Model Analysis

The next phase uses the Simalytic Model to analyze the application. At this point, the Simalytic Model can be used just as any other type of model which has been calibrated. In addition, how a model is used to answer "what-if" questions is very dependent on the questions themselves. Therefore, the details of the phase will not be discussed here other than to note that all of the phases of constructing a Simalytic Model should be considered as a spiral development process. The completion of each phase may identify additional information or requirements for the prior phase. This provides the added benefit of allowing the modeler to implement a quick, simple Simalytic Model and then continue to refine it based on the business requirements and objectives.

4.2.5 Implementation Example

This implementation of a Simalytic Model uses a hypothetical client/server environment to illustrate the process. Assume the workload of interest is an Order Entry application on one server and there is a Shipping application on another server also used by the Order Entry application. The Order Entry user types in the name of an existing cus-

tomer and gets not only the address but any pending or past orders and the status of the account. This may provide better service, but it also causes transactions to be sent to another system. Defining the topology of the application identifies that some number of the Order Entry transactions are routed to the Shipping server and the measurement data provides the number and distribution. If the Shipping workload outgrows its system, it can impact the responsiveness of the Order Entry transactions. In addition, growth in the Order Entry workload will now impact the Shipping system, but only if the orders are from existing customers. The systems cannot be modeled independently because the service time for one system is dependent on the response time of the other. When the Order Entry transaction rate increases, more transactions are sent to the Shipping server. The increased response 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 throughput for the application.

Figure 25 Enterprise Model Example shows a diagram of this model. The response time is measured from **Arrivals** to **Departures**, either through the **Shipping** node or around it.

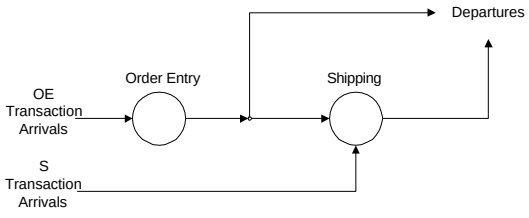

Figure 25 Enterprise Model Example

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 transactions (those also sent to Shipping) would not reflect the increased response time due to the load at Shipping. Because of the additional application information in the Simalytic Model, it can adjust the response 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.

As with any modeling effort, there must be business objectives to analyze using the model. 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 Order Entry system upgrade is needed to fix the problem?" as discussed on page 60. These questions have an assumed bias that the problem is with the Order Entry system. Therefore, the questions should be rephrased as: "When does the application fail to meet the business response time goal?" and "What must be upgraded to insure the application meets the goal over the required time period?" In other words, the focus of the modeling effort needs to be shifted from a system centric view to an application centric view.

4.2.5.1 Example Process Implementation

Although the implementation process for the example follows the steps presented in section 4.2.1, many assumptions have been made about the information collection process to simplify the example.

4.2.5.2 Workload Analysis Example

Identify: For this example, there is a single Order Entry transaction, OE, and a single Shipping transaction, S. The OE transactions are the workload of interest. The S transactions need to be included only to the extent they impact the OE transactions. However, some additional information about the S transaction response times is included to illustrate the pitfall of modeling the systems independently. The S transactions entering the system are kept constant at 0.1 arrivals per second. Only the OE transaction arrival rate is changed to represent growth in that workload.

Document: Refer to *Figure 25 Enterprise Model Example* . Assume that measurement data shows 30% of the OE transactions are also routed to the Shipping server. To keep this example simple, also assume that all of the transactions that execute on either server use the same resources. This means that there is no difference on the Order Entry server between the OE transactions that route to Shipping and those that don't. It also means there is no difference between the transactions that execute on the Shipping server (i.e. an OE transaction routed to Shipping consumes the same resources on the Shipping server as an S transaction).

Measure: Because this is a hypothetical client/server environment, there are no actual measurements. Therefore, the results of a pure simulation model of the environment will be used to represent these measurements.

Correlate: The workload correlation is assumed.

4.2.5.3 Node Models Example

Build: The service times for the OpenQN model of each node is shown in *Table 3 Example Device Service Times*.

Server	Order Entry	Shipping
Workload	OE	S
Device 1: CPU	0.02	0.10
Device 2: Disk1	0.06	0.70
Device 3: Disk2	0.02	0.50
Device 4: Disk3		0.40
Device 5: Disk4		0.30
Total Service Time	0.10	2.00

Table 3 Example Device Service Times

Calibrate: The model of each node is assumed to be calibrated for this example.

Run: An OpenQN model was run for each node.

Create: Partial results of the OpenQN model of each node are shown in *Table 4 OpenQN Example Results* (not all data points are shown in the table to save space). Notice that the model was not run for the Order Entry server for a number of arrival rates because there was no significant change in response time. Also notice that at 1.40 the arrival rate step was reduced from 0.05 to 0.02 to better define the knee of the response time curve for Shipping.

Server:	Order Entry	Shipping
OE Arrival Rates	Response Times	
0.01	0.10	2.01
0.50		2.70
1.00	0.10	4.54
1.10		5.43
1.20		6.98
1.25		8.33
1.30		10.65
1.35		15.76
1.40		38.21
1.42		119.96
2.00	0.11	
10.00	0.20	
15.00	0.66	
15.75	1.15	
16.00	1.56	
16.25	2.46	
16.50	6.06	

Table 4 OpenQN Example Results

4.2.5.4 Simulation Model Example

Build: The overall simulation model was built using Simul8 as in *Figure 26 Example Simalytic Model*.

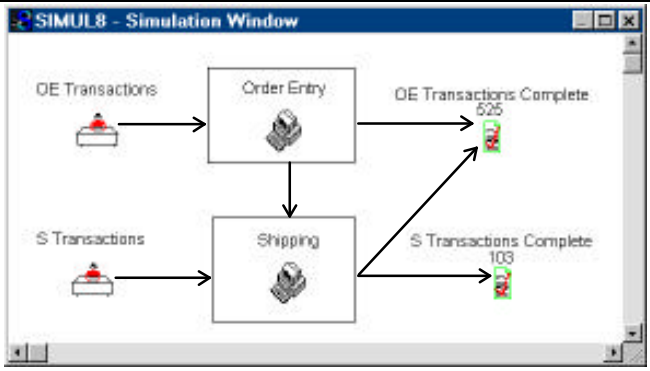

Figure 26 Example Simalytic Model

Set: The service times of both the Order Entry and the Shipping servers are set to the lowest response times from the table above. The replication factor for each server is set to 100 to avoid queuing. The average response times for that model run (ten trials) are 0.56 for the OE workload and 2.01 for the S workload. These response times are very close to the expected values of 0.7 and 2.00, respectively (the expected OE response time is 0.7 because of the routing to Shipping: 0.1+0.3*2.0=0.7) The modeled OE response time is slightly lower because the actual model routing was slightly lower than 30% in most of the trial runs due to the small number of transactions at the low arrival rate.

Calibrate: The results of the above model are compared to a pure simulation model, *Figure 27 Example Simulation Model*, also built in Simul8, to calibrate the model. The service times are the same as shown in the table in sec-

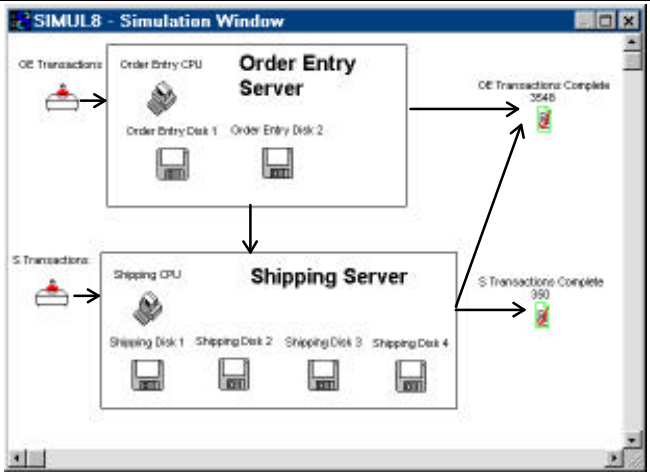

Figure 27 Example Simulation Model

tion 4.2.5.3.

Selected results from multiple runs of this model are shown in *Figure 28 Example Simulation Model Results*.

OE Arrival Rates	OE Response Times	S Response Times
0.01	0.677	2.093
0.10	0.730	2.119
0.20	0.749	2.151
0.50	0.787	2.273
1.00	0.857	2.498
2.00	1.069	3.155
3.33	1.777	5.495
3.70	2.332	7.487
4.00	3.317	10.674
5.00	59.415	215.970

Figure 28 Example Simulation Model Results

Each data point is the average of ten trials for each arrival rate to reduce the impact of arrival distributions. The arrival rate refers to the arrival rate for the OE transactions. The S transaction arrival rate is kept at a constant value because it is not the workload of interest. Each trial was for 3600 simulation seconds (one simulation hour) with a 100 second warm-up period.

4.2.5.5 Simalytic Model Example

Create: Create the Simalytic Function. The Simalytic Function was created using Microsoft's Visual Basic. For this example, it is a very simple function that calculates the rolling average of the interarrival times for each workload and looks up the corresponding response time in a table.

Replace: Replace the static service times. *Figure 29 Simalytic Example Results* shows the results of the model trials after the Simalytic Function was implemented.

OE Arrival Rate	OE Response Time	S Response Time
0.01	0.661	2.080
0.10	0.716	2.123
0.20	0.728	2.126
0.50	0.754	2.166
1.00	0.780	2.292
2.00	0.907	2.696
3.33	1.622	5.085
3.70	2.036	6.468
4.00	2.838	9.062
5.00	11.492	38.144

Figure 29 Simalytic Example Results

Calibrate: The results of the above model are compared to a pure simulation model to calibrate the model as shown in *Figure 30 Example Response Time Comparison*. The Simalytic results track the simulation results. The slight under predicting is consistent with the simple implementation of the Simalytic Function, which is the same algorithm as the one used in

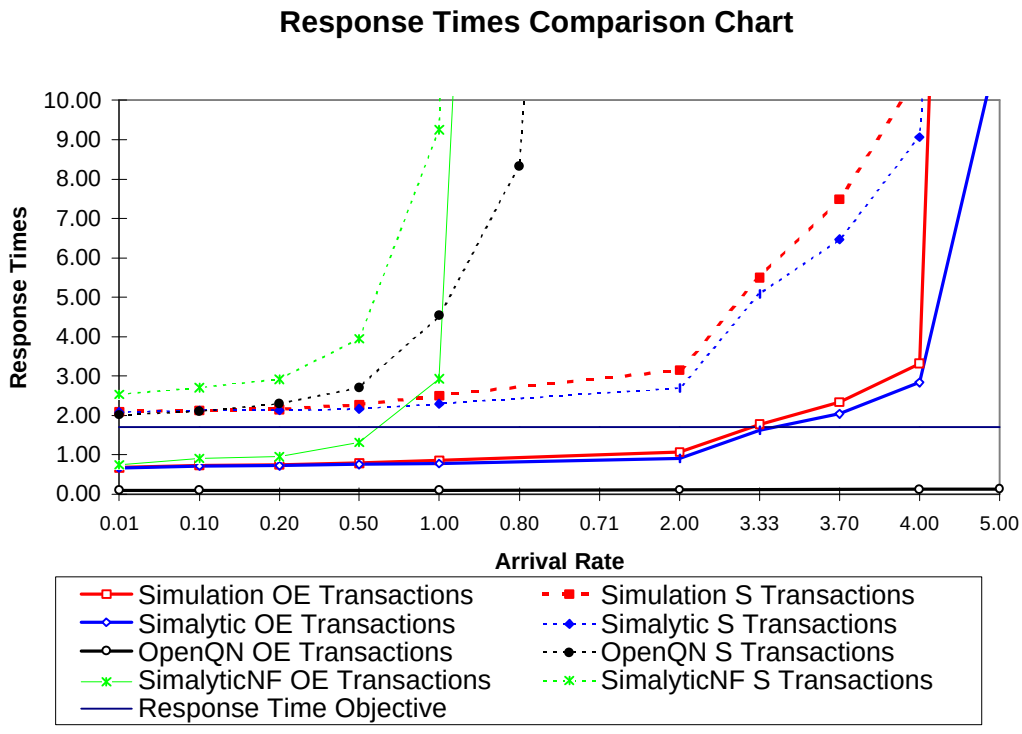

Figure 30 Example Response Time Comparison

the MathCAD implementation and is explained in section *7.5 Appendix E: MathCAD Formulae Results Charts* on page *180*. The under prediction is a result of the step function implementation of the Simalytic Function. The Simalytic Model was run with a modified version of the Simalytic Function that implemented only every third entry in the if-then-else statements for the **DistribOE** and **DistribS** sections of the Visual Basic code shown in *Figure 55 Visual Basic Code for Simalytic Function* on page 170. These results, graphed in *Figure 31*, show a consistent under prediction for both workloads of about 10% for most arrival rates. The lines **OE Standard SF** and **S Standard SF** are the same curves as the **Simalytic OE Transactions** and the **Simalytic S Transactions** curves in *Figure 30*. The lines **OE Special SF** and **S Special SF** show the same model rerun with the modified Simalytic Function.

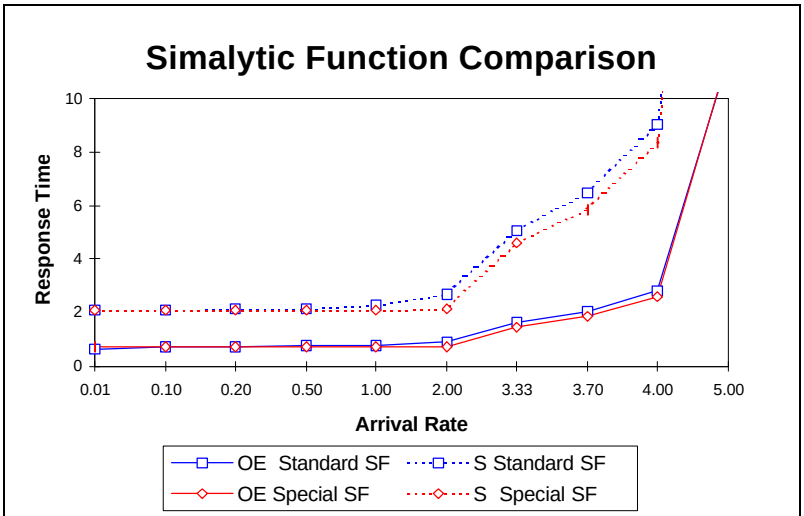

Figure 31 Simalytic Function Comparison

4.2.5.6 Analysis of Example Models

Figure 30 shows how these results relate to the questions asked in section *4.2.5*. When the business volume grows to 3.33 OE transactions per second the response time will exceed the goal of 1.7 seconds. Furthermore, the way to keep OE transactions under the goal is to upgrade the Shipping server rather than the Order Entry server because the OE response time is directly related to the longer Shipping transactions.

Figure 30 graphs the response times for a number of different models. The response times are shown in pairs: one response time for the OE transactions and one for the S transactions. The **OpenQN** lines represent only the workload at the respective servers. All of the others include OE transactions sent to Shipping in the OE workload. The **Simulation** lines represent the pure simulation model and the **Simalytic** lines represent the Simalytic Model results. The **SimalyticNF** lines represent the results of a model that is the same as the Simalytic Model but without the Simalytic Function (hence the NF, for *no function*, in the name). It also does not implement the increased replication at the servers. Each node is implemented as a single server where the service time of the node is set to the sum of the device service times for that node. The purpose of the **SimalyticNF** lines is to illustrate that the difference between a simplistic single server model and the simulation model is much larger than the difference between a Simalytic Model and the simulation model. Because the S transaction workload arrival rate was kept constant, the **Arrival Rate** axis is the OE arrival rate except for the **OpenQN S Transactions** line, which shows what happens at the Shipping server. In all other cases, the increase in S transaction arrival rate is due to the transactions sent to Shipping from Order Entry.

From *Figure 30* we can make the following observations: **OpenQN** shows the Order Entry System response time to be flat and does not predict a response time of 1.7 within the scale of the chart (almost 16 transactions per second). The **Simulation** and **Simalytic** result lines are very close. The **SimalyticNF** line predicts exceeding the goal at slightly more than half a transaction per second. The queuing theory (**OpenQN**) tool greatly underestimates the OE workload response time because it does not account for the impact from the Shipping system. The simple simulation model (**SimalyticNF**) greatly

overestimates the OE workload response time because of the rapid queue buildup at a single server. Only the **Simulation** and **Simalytic** models represent the actual workload behavior.

4.3 Simalytic Model Development Summary

Simalytic Models implemented with the tools Simul8 and OpenQN are consistent with the MathCAD implementations shown in section *3.5.3 Validation of the Mathematical Foundation* on page 86. Using these tools, the Simalytic Model is constructed from a simulation framework (using Simul8), queuing theory models of each node (using OpenQN) and a Simalytic Function for each node (using Visual Basic). Each of the five phases in the construction of a Simalytic Model is broken down into discrete steps. The Workload Analysis phase identifies the workload, documents the application topology, develops and implements the approach to measuring the application, and correlates the workload across all of the systems. The Node Models phase builds models of each node, calibrates those models, runs the models for an appropriate number of arrival rates for each node, and creates a workload profile table of the results for each node. The Simulation Model phase builds an overall simulation model of the application, sets the overall model parameters, and calibrates the overall model against measured response times at very low arrival rates to insure the correct topology has been implemented. The Simalytic Model phase creates the Simalytic Function for each node, replaces the static service time of each node with the Simalytic Function, and calibrates the overall model at all arrival rates for which measured response times exist. The Model Analysis phase is similar to the analysis of any calibrated model used to answer questions concerning the possible business ramifications related to various application growth scenarios. The two server example

used to illustrate these steps shows a consistent correlation between the simulation results used in place of actual measurement data and the Simalytic Model implementation.