The Modeling Pyramid: From Design to Production

Dr. Tim R. Norton
Simalytic Solutions, LLC
CMG2000 Session 4404
December 13, 2000
Orlando, Florida

The Modeling Pyramid looks at modeling from the application point-of-view starting with the design phase through development, implementation and production support (performance management and capacity planning) to replacement (evaluation of a new design). All of these techniques are well known. However, they are seldom connected to allow the modeling effort to flow seamlessly with the application development process. This paper discusses how a model used for each step in the process can relate to the model in both the prior and the next steps, as well as the overall business process model.

1. Introduction

1.1 Background

Modeling. System modeling. Server modeling. Network modeling. Data modeling. Application modeling. There doesn't seem to be any shortage of things to model or ways to model them. The issue isn't what or how to model, but what to do with the results. In other words, we need to understand WHY to do the modeling before we address HOW to do it. The first step for any modeling effort is to define the objective. However, there is often an aspect of mutual exclusion between objectives because of different points-of-view, depending on the requester and the phase of application development. For example, the application design architect may want to know the responsiveness of a new function, while the capacity planner may want to know the impact of the new function on server utilization.

This paper discusses the progression of modeling an application from birth to death (replacement) to get objective information for sound business decisions at every step. Not only does there need to be a progression from one phase to another as the application is developed, but the development loop needs to be closed by connecting the death of an application to the birth of its replacement.

Before we can look at how these different models are connected, we need to understand the different types of models and how each of them is used. The next section, *Aspects of Modeling*, presents a brief overview of modeling in the various application life-cycle stages. This section is not intended to explain how to actually do the modeling, but just to give the reader a general idea of what can be accomplished and the types of techniques available. (The reader should also note that the application life-cycle stages used throughout this paper do not necessarily conform to the formal definition of any application life-cycle development methodology. The stages presented are meant to convey the total process from an overall business point-of-view.) Then section *3, Role of Modeling Techniques*, presents a brief overview of modeling techniques and discusses how each applies to the application life-cycle stages in section *2*. Section *4, The Modeling Pyramid*, presents the idea of a hierarchical relationship between the models used at each application life-cycle stage and how that relationship can be exploited.

2. Aspects of Modeling

2.1.1 Value to Management

Although not actually one of the stages in the application life-cycle, the value of modeling to management is important enough to discuss by itself. Modeling within an organization is often a binary situation, where it is either completely supported or totally discounted. It is unusual for management to be partially supportive. Unfortunately, management's value of modeling activities seldom depends on the technical aspects, such as the accuracy of the results or the level of effort to create a model. Instead, it is a political situation related to management's objectives, both the publicly proclaimed (such as responsiveness to the end customer) and the 'hidden agendas' (such as 'empire building'). The point here is to express the need to understand those motivations, not explain how to deal with them (CMG and other organizations have published many papers on techniques to persuade management, far too many to even reference in an unbiased way, and the reader is encouraged to investigate those that apply to their particular situation). If management does not value modeling within one of the stages they will certainly not find it worthwhile to invest time and effort to understand

and exploit the relationship between modeling activities. The remainder of this paper assumes that there is management support for at least two adjacent application life-cycle stages. Once management understands the value of modeling for any of the application life-cycle stages, that support can be expanded to the adjacent stages using The Modeling Pyramid.

2.1.2 Application Design

Modeling during the application design stage can be examined from different perspectives. One uses a modeling technique such as UML (Unified Modeling Language) to model the ***function*** of the application. Another uses a modeling technique such as SPE (Software Performance Engineering) to model the ***performance*** of the application (both are discussed in section *3.2, Specialized Tools for Modeling*). The objective of modeling at this stage of the application life-cycle is to improve the design from either or both perspectives. This type of modeling predicts how the parts of an application (performance, functional relationships, data flow, user interface, etc.) will work while building the components and sizing the required hardware environment prior to implementation.

2.1.3 Implementation

Once the application components have been designed and written, the pieces are put together, converting the application design into a working system. In addition to the techniques used during the design stage (SPE and UML), other techniques that rely on measurement data from a running application can be used, such as end-to-end response time measurement and node models (see 3.2*, Specialized Tools for Modeling*). The objective at this stage is to determine how an overall application works on the intended hardware.

2.1.4 Deployment

The Deployment stage connects the new application into the production environment. Seldom is an application an island to itself; it must receive inputs from, and provide outputs to, other systems. Therefore, how the new (or modified) application interacts with the other existing applications is often a very large factor in the overall performance. Modeling at this stage is complex because of the involvement with these other systems. However, when done, it can avoid many unpleasant surprises caused when these inputs or outputs are not delivered as quickly by one system as the other system expects. An additional benefit to modeling at this stage is the additional information that can be used to reevaluate the sizing of the existing systems because of the identification of any increases to their workloads. Often at this stage the model of the new application is not as useful as the understanding of its impact on the environment where it will be deployed.

2.1.5 Production

The Production stage addresses the day-to-day running of both the application and the environment. This is often the longest stage of an application's life-cycle and includes measuring the running application and identifying system level performance problems. Modeling at this stage is varied and depends on the dynamics of both the application and the overall environment. The most common use is predicting hardware capacity requirements for growing applications. If everything is relatively stable (i.e., little growth and infrequent hardware or operating system changes), then there is much less value to modeling. Assuming that application changes are addressed in other stages of the application life-cycle, then the next greatest use of modeling in the Production stage is to determine the impact of other environmental changes. For example, an increasingly popular topic in the Open Systems arena is server consolidation. The question is "What happens to performance?" when the hardware (and maybe the operating system) is changed and several applications are allowed to compete for the new, and hopefully larger, resource. A model may be used to determine which applications can and cannot be consolidated on the same hardware.

2.1.6 Planning

Capacity Planning is considered here as a stage of the application life-cycle because of the cyclic nature of the budgeting process in most organizations. The ideal situation would be to have on ongoing planning process that continually reassessed the capacity requirements of each application to provide "just in time" capacity increases. However, annual budgeting processes in most organizations force capacity planners to predict application performance at least a year into the future. They must also determine the impact of that projected performance on the overall business to develop the justification for specific budget items. A model at this stage can use the application historical data along with the business growth projections to develop a convincing, or even compelling, justification for capacity increases a year or two in the future.

2.1.7 Enhancement

The final stage of the application life-cycle is Enhancement, where the application is either decommissioned or it is transformed into a new application by the addition of new features and functions. Although it may seem odd to talk about the act of turning off an application as an "enhancement" it can have just as profound an impact on the production environment as major application changes. Seldom does an application just disappear. The business functions that were provided by a decommissioned application are usually incorporated as enhancements in some other application. This relationship may be difficult to see when the applications are in radically different parts of the com-

pany. For example, a mainframe batch application with 'key punch' data entry might be replaced by a mid-range application with a customer accessed web interface. The development and support organization of the old application may very well not be involved in the design and implementation of the new one, even though they both address the same end customer need. Modeling in this stage provides understanding into the performance impact of new application components or redesigns on both the application and the overall system environment. Although extremely important, modeling in this stage is often omitted because it is much more complex, combining aspects of modeling at both the Application Design stage and the Production stage.

3. Role of Modeling Techniques

A discussion of modeling techniques and tools can approach the subject from any of several different directions and points-of-view. The objective of this section is to provide a reader unfamiliar with one of the techniques enough information to understand how it fits into the Modeling Pyramid. It does not attempt to specifically address all of the different techniques, but rather attempts to give the reader a general overview of what is available and how to determine which technique is better suited for a given problem or situation. It is divided into two sections, *3.1, General Modeling Tools*, which provides a very brief overview of the two main techniques used for modeling computer applications and systems and *3.2, Specialized Tools for Modeling*, which provides an introduction to several specializations of these techniques to address specific application issues.

3.1 General Modeling Tools

For most of the modeling activities associated with application performance analysis, there are two types of models, simulation and queuing theory. Each is discussed briefly with some of the major reasons for using that technique.

3.1.1 Simulation

Simulation modeling tools are computer programs that emulate the behavior and structure of a system, either as discrete events (DES or discrete event simulation) or as continuous flows. (Continuous models use complex formulae to describe the physical nature of processes, such as the expansion of a gas, and are seldom used for modeling computer systems. Unless otherwise noted, for the remainder of the paper 'simulation' refers to DES models. Although continuous models are not discussed in detail, it is worth noting that some business process modeling tools use this technique; see section *3.2.5, Business Models*).

> The simulation model describes the operation of the system in terms of individual events of the individual elements in the system. The interrelationships among the elements are also built into the model. Then the model allows the computing device to capture the effect of the elements' actions on each other as a dynamic process.
> (Kobayashi 1981, p. 221)

There are many types of simulation models and techniques. Trace-driven simulations are of more use in capacity planning and performance modeling because they remove a major issue in model construction; transaction arrival distribution. Self-driven simulations generally assume some standard distribution pattern of inter-arrival times, such as a normal or exponential distribution, which may or may not represent the actual distribution of application transactions (Kobayashi 1981). Regardless of the underlying technique used in the simulation tool, there are some important characteristics of these tools. They have the ability to maintain the identity of each transaction, and its associated attributes, throughout the entire model. The model can react to these attributes to control the flow of transactions, such as data dependent routing or fork-join transactions. They also preserve the specifics of the inter-arrival times between individual transactions over time and can show the overall dynamics of the system, such as how transaction response time increases approaching the peak of the day and when the system catches up again. These characteristics make simulation models extremely useful for modeling complex systems, application designs and dynamic environments.

3.1.2 Analytic

Analytic modeling tools are mathematical representations that describe the behavior of a system.

> Analytical models capture key aspects of a computer system and relate them to each other by mathematical formulas and/or computational algorithms.
> (Menascé, Almeida, and Dowdy 1994, p. 45)

Analytic models can be implemented many different ways from paper-and-pencil to spreadsheets to advanced commercial products. One analytic technique, queuing theory, plays the most dominant role in the area of capacity planning because capacity and performance problems are most often related to queuing delays caused by contention for resources within a system (Kobayashi 1981). Queuing network models can be solved using several techniques, such as convolution (Menascé, Almeida, and Dowdy 1994, p. 155) or MVA (mean value analysis) (Menascé, Almeida, and Dowdy 1994, p. 159). The greatest advantage to analytic models is that they can be solved very quickly using simple tools, such as a spreadsheet. Because they rely on the assumption of homogeneity (that there is little difference between individuals described by a given value), the model does not have to be run repeatedly or for long simulation times to achieve convergence. However, this same characteristic causes problems when attempting to identify the

proper workload characterization classes or interval of sample data to use. The accuracy of these tools depends on the homogeneity of both the modeled workload and the sample interval. As the variability of either increases, the accuracy of the model decreases.

3.2 Specialized Tools for Modeling

The techniques described in this section are either implemented using one or both of the general techniques discussed in section *3.1, General Modeling Tools*, or provide additional information to improve the use of the techniques.

3.2.1 UML

UML (Unified Modeling Language) is a technique to describe, specify, and document application functions during the design phase (UML). The value of UML to modeling is that it has become a very popular technique to design applications and it can be used to define application workloads and tie the development effort to what to measure. UML Sequence Diagrams, a variation of Message Sequence Charts (MSCs) describe the interactions between application components. Some work has been done to incorporate additional features into MSCs that are needed to establish the correspondence between software design scenarios and performance scenarios (Smith and Williams 1998). The OMG (Object Management Group) has recently published a draft of UML extensions to incorporate timing into an application model (http://cgi.omg.org/cgi-bin/doc?ad/00-08-04, *Response to the OMG RFP for Schedulability, Performance, and Time*). Although these extensions are specifically to address issues related to the design of real-time systems, they may make it easier to generate a performance model from a UML model. The advantages of this type of formalism will be discussed in Section *4, The Modeling Pyramid*.

3.2.2 ETE Response Time

ETE (End-to-end) Response Time is a broad category of many data collection techniques to measure response time from the end user's point-of-view. The objective of ETE response time is to gain a better understanding of the overall performance of an application. ETE response time measurement is more complex than it appears due to issues such as accounting for user think time, identifying business transactions and mapping them to components of the application, application component reuse (i.e., CORBA and DCOM), and the level of parallelism in the application (Norton 1999). The main value of ETE response time measurement to any modeling effort, and especially to client/server models, is the additional information for model verification and validation. In addition, ETE response time measurement allows the results of an application model to be expressed in the same terms as the application's SLO (Service Level Objective). Without some type of ETE response time measurement, the overall accuracy of a client/server model, and thus its underlying assumptions, cannot be verified.

3.2.3 SPE

SPE (Software Performance Engineering) is a methodology to focus the design effort on the importance of designing performance into the application rather than adding it after implementation. SPE is more than a modeling technique to predict the performance of an application during the design phase; it addresses data collection, quantitative analysis techniques, prediction strategies, management of uncertainties, data presentation, verification, validation, critical success factors, and performance design principles (Smith 1988; Smith 1990). Even if no actual application model is constructed, SPE is still a very valuable activity because it focuses attention on many of the problem areas between application development and performance. When SPE models are implemented, they tend to provide an overall business focus beyond the system level view of many of the other modeling approaches.

3.2.4 Simalytic

The Simalytic™ [*] Modeling Technique uses a simulation model framework to connect models of the components of a client/server environment that have been implemented with the modeling techniques most appropriate for each component. A Simalytic Model allows the application workload to be characterized at each node in the environment with the best tool or technique available within the parameters of the current situation. When the revised situation requires increased accuracy or additional details for an application component, more time and effort can be invested into an improved model of just that component, possible using a more sophisticated tool, that replaces the initial component model without requiring significant revisions in the framework. Thus Simalytic Modeling allows the rapid development of application models at the business level that also include whatever lower level details are appropriate (Norton 1997a; Norton 1997b).

3.2.5 Business Models

Business Process Modeling, also referred to as System Dynamics, is a methodology for modeling business processes that includes functions to address all aspects of most types of businesses. Most Business Process Modeling tools are simulation based. Some of the tools use continuous simulation techniques to model the wok in a business process as flows (work moving from one part of the process to

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

another) and levels (the amount of work at a given stage in the process). These tools do not specifically address the requirements of modeling computer applications. Their focus is on the overall business process, including the manual or human interaction aspects. These models are used to investigate and project the impact of all activities on the overall business. They can be used in conjunction with application and system models to determine the ROI (Return On Investment) for various alternatives suggested by modeling application modifications and environmental changes (Norton 1998a).

3.2.6 Node Models

The term 'Node Models' refers to server and component level models within a larger context. They are detailed models of individual hardware components, such as servers, disk subsystems and networks, that are used as building block to construct an overall application model (Norton 1996). The advantage to using node models is that they can be developed somewhat independently of each other (provided they all use a common set of basic assumptions and workload definitions) to improve reusability and reduce model development time.

3.2.7 Platform-Centric Models

The term 'Platform-Centric Models' was developed by the author to describe complex models, generally analytic tools, that have a high degree of detailed information about the environment they are intended to model (Norton 1996). The advantage to using platform-centric models is that they simplify the overall model development process and generally increase the accuracy of the model results at the same time. Most of the platform-centric modeling tools are highly specialized for a single operating system environment and include substantial behavioral information that is only available because of the vendor's significant research and development efforts. Most performance analysts and capacity planners do not have either the time or the resources to develop models at this level of detail. Therefore, commercial platform-centric tools provide a significant productivity advantage over more general-purpose tools.

4. The Modeling Pyramid

The Modeling Pyramid is both a hierarchical view of modeling computer environments and a technique to connect the different layers together. It is presented as a pyramid to show the progression from strategic objectives to detailed implementation. Each level of the pyramid builds on the levels below, either by using reasonable assumptions or collected measurement data about the behavior of components. The Modeling Pyramid doesn't map directly to the application lifecycle stages but presents an overview of modeling within an established IT (Information Technology) organization.

4.1 Modeling Pyramid Levels

The levels of the Modeling Pyramid are shown in *Figure 1 The Modeling Pyramid*. Each level addresses a different business requirement.

4.1.1 Strategic

The Strategic level is really the overall view of how the organization plans to move forward and is not what most capacity planners or performance analysts think of as modeling. It provides the long-term objectives that all of the other objectives and activities throughout the organization should support. A model at this level

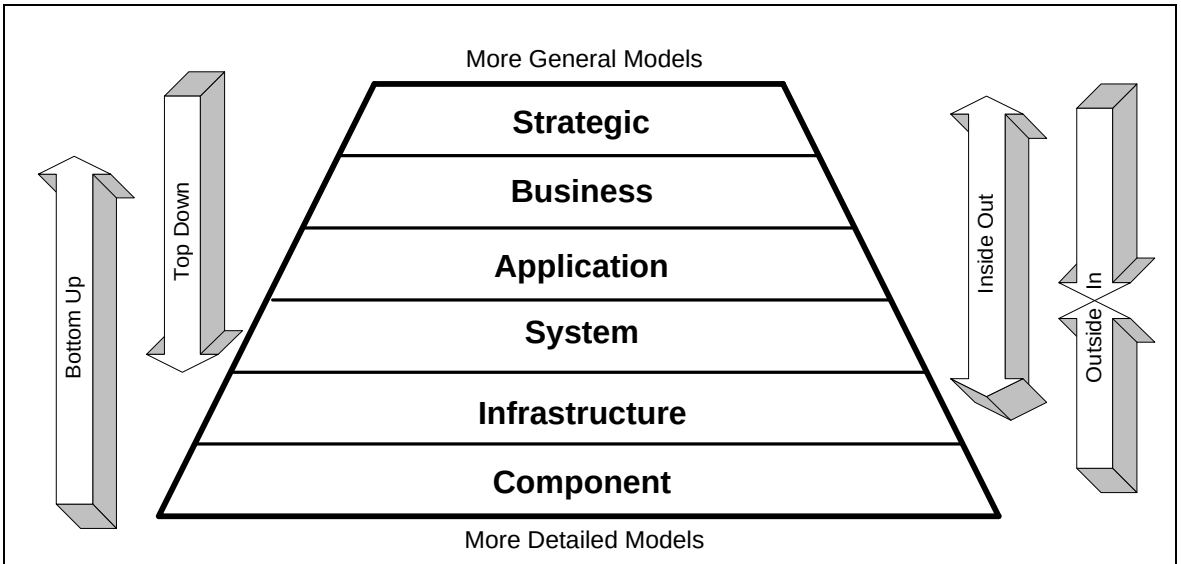

Figure 1 The Modeling Pyramid

provides the vision of where the business is going and what the different organizations, including IT, must provide in the way of support. These models are sometimes called 'business plans' or 'marketing plans' and provide information such as the expected workload, in business unit terms, over the next several months to years, depending on how dynamic the business is. The Strategic level provides the overall consistency across the application life-cycle stages by providing high-level objectives to focus on at each stage.

The outputs from the Strategic level are projections of growth, both for existing lines of business and for new business areas. Often this information is presented to shareholders and other investors to show the long term value of the company.

An example of a Strategic level for a mail-order catalog company would be the marketing plan (i.e., adding new items to the catalog and increasing the number of customers) and the business growth plan (i.e., expanding the business from printed catalogue and telephone orders to include on-line web services and partnerships to extend customer reach and fulfillment). Reasonable Modeling Pyramid metrics for each line of business might be the number and types of customers, the number of catalogue items, inquiry volume and percentage that result in sales, frequency and effectiveness of marketing activities (direct mailings, print ads, media ads, etc.) and new business projections. This example will also be used in the following sections to illustrate the relationship between the Modeling Pyramid levels.

4.1.2 Business

The Business level is a more detailed view of how the organization plans to move forward with respect to individual business processes. Business Process Modeling, or System Dynamics, provides insight into all aspects of a given business process, not just the IT functions familiar to capacity planners and performance analysts. A model at this level predicts how that business process will react to changes in either the inputs to the process or how the process is implemented. The Business level provides an objective way to evaluate alternatives at each of the application life-cycle stages by showing the impact of changes in business terms.

The outputs from the Business level are projections of how the line of business will react to changes in terms such as staffing requirements, work-flow adjustments and process bottle-necks.

Following on with the above example, a business process model for the catalogue order entry function would use the marketing plan for increasing the number of customers to see if the existing staff and infrastructure are adequate, and if not, what needs to be changed. To do this, the business model would include the activities necessary for one of the "operators that are standing by" to answer the customers telephone call and take the information, check availability and shipping status, do any required paperwork, take breaks, discuss problems with his or her supervisor, and finalize the order. Reasonable Modeling Pyramid metrics might be the required number of operators, the required number of telephone lines, the computer application transactions required in the order entry process, the required response times for each transaction, and percentage of call time spent in each major function (on the telephone, using the transactions, paperwork, away for work area, etc.).

4.1.3 Application

The Application level is a more detailed view of how the computer application supports the business process. It is the lowest level focused on the IT functions normally thought of as modeling by capacity planners and performance analysts. A model at this level predicts how the application (transactions, batch processes, interactive sessions, etc.) will react to changes in load (transaction arrivals, amount of data, or number of users) or implementation (changes or additions to the application code). The Application level provides an objective way to evaluate alternatives at each of the application life-cycle stages by providing the impact of changes in business terms using the workload definitions from the Business level.

The outputs from the Application level are projections of application performance (i.e., response time and through-put), maximum acceptable load (i.e., transactions per second or the number of supportable users) and application bottle-necks (i.e., database lock contention). At this level the modeling objectives can take two different directions. The first case is enhancements to an existing application and the second is an entirely new application. In the first case, the application model would use the workload definitions (application transactions required in the order entry process) and required responsiveness (response times for each transaction) to project the required capacity for the application. This corresponds to the Deployment and Productions stages of the application life-cycle. In the second case, an SPE (Software Performance Engineering) model (Smith 1990) is developed for the new application based on the requirements defined at both the Strategic and the Business level. If it is a totally new application to replace non-computer portions of the process, then the changes in operator time to complete the function are fed back into the Business level model. If it replaces an existing application, such as moving to a distributed application design, the SPE model for the new application should be compared against the last 'enhancement' model of the application it replaces. This corresponds to the Design and Implementation stages of the application life-cycle. Reasonable Modeling Pyramid metrics might be the

transaction rate where the response time exceeds the business requirement or the required capacity for the projected load.

In our example, an application level model could be created, using the number of calls and the number of transactions per call, to understand how many additional transactions per second could be handled by the existing application. For example, the impact on response time can be determined when a new component is introduced into the application, such as replacing shipping paperwork with an online transaction. The projected response time is a function of the additional resources consumed by the new function and the increased transaction arrival rate of the existing transactions due to reduced user think time (the time it used to take the operator to fill out the shipping paperwork).

4.1.4 System

The System level is not necessarily a more detailed view, but it is a much broader view because it includes all of the applications using a given system. The term 'system' is used here in a very broad sense to mean the hardware, operating systems, databases, and everything else that makes up the computing environment of the application. It also includes every other application using that same environment. It is the intersection between the application view and the operational view of supporting the business and is the traditional area normally thought of as modeling by capacity planners and performance analysts. A model at this level predicts not only how the systems will perform when applications are added, changed, or removed, but also how such changes in one application will impact another application. The System level provides an objective way to evaluate alternatives at each of the application life-cycle stages by providing the impact of changes to both the supporting hardware and other applications sharing the same system environment. The Systems level model must account for growth to existing workloads, enhancements to existing applications, and the addition of new applications and workloads as well as changes in the supporting hardware such as server and operating system upgrades.

The outputs from the System level are projections of system performance (i.e., utilization and queue lengths), maximum acceptable load (i.e., transactions per second or the number of supportable users) and system bottle-necks (i.e., I/O path contention or memory shortages). The objectives are really the same for either of the cases from the Application level; the only difference is how the model is constructed. In the case of enhancements to an existing application, the model starts with actual measurement data of that application and modifies things like arrival rates and service times to account for the changes. In the case of a new application, the SPE model must be merged into an existing system model. In the first case, the system model would use the workload definitions (application transactions required in the order entry process) and required responsiveness (response times for each transaction) for all of the applications using the system to project the required overall capacity. This corresponds to the Deployment, Productions, and Planning stages of the application life-cycle. In the second case, the model would use the workloads created in the SPE (Software Performance Engineering) model developed for the new application. This corresponds to the Design, Implementation, Enhancement, and Planning stages of the application life-cycle. Reasonable Modeling Pyramid metrics might be the transaction rate where the response time exceeds the business requirement or the required capacity for the projected load.

Here our example changes to focus on the relationship between the order entry process and the other applications using the same system. This could include the impact of systems services (i.e., backups and security) or other aspects of the same application (i.e., frequent reports showing the best selling products for marketing). By using the peak transaction arrival rate for the order entry workload, the system analyst can quickly see the impact of these other workloads and explain that impact in terms of a reduction of calls (and thus orders) per hour. The business advantage of the frequent reports can then be assessed against the cost (loss of revenue) for running them.

4.1.5 Infrastructure

The Infrastructure level is a more detailed view in the sense that it focuses on the interconnections used by many systems. These interconnections can be related to either networks or external devices. It also includes all applications using that same environment, but the definition of environment is greatly expanded from the System level. On the one hand, it is the intersection between the system view and the network view (which can now include the entire world for web applications) and the details of configuration options, such as SCSI versus Fibre Channel connections for disk. It is also an area normally thought of as modeling by capacity planners and performance analysts. A model at this level predicts how sensitive the systems are to a wide variety of connection related issues, such as when applications are added to, changed or removed from a system providing inputs to the environment being modeled. The Infrastructure level provides an objective way to evaluate alternatives at each of the application life-cycle stages by providing the impact of changes to the connections between applications sharing the same overall environment. While not a major issue to many legacy applications, the Infrastructure level is becoming critically important to newer applications that utilize multi-tiered client/server designs. The Infrastructure level model must account for

growth to existing workloads (both arrival rates and data volumes), enhancements to existing interconnection techniques (i.e., Fibre Channel, FICON, ATM, Gigabit Ethernet, etc.) and the addition of new techniques (i.e., wireless). The emergence of new techniques, such as Storage Area Networks and Network Attached Storage, has blurred many of the boundaries and it is harder to tell where storage issues end and network issues begin (both IP over SCSI and SCSI over IP are being pursued by several vendors).

The outputs from the Infrastructure level are projections of connection performance (i.e., utilization and queue lengths), maximum acceptable load (i.e., I/Os per second or through-put in bits or bytes) and connection bottle-necks (i.e., router delays or fibre switch latency). Although the objectives are really the same as at the System level, they are focused on the interrelationships between the systems, and the results are much more usable as modifiers to Application level and System level models. Seldom does an Infrastructure level model provide useful information in and of itself (that is, outside of the context of the environment that it is connecting). Infrastructure level models correspond to the Deployment, Production and Planning stages of the application life-cycle. Reasonable Modeling Pyramid metrics might be network or disk response times at the expected loads, or the rate for either that causes the transactions to exceed the response time requirement.

Now we have to make some additional assumption about our simple example. Let's assume that the new client/server application to replace the shipping paperwork has been implemented and that it is running on a separate system. When the operator uses the order enter transaction to place the order, it invokes the shipping function to get backlog and pricing information before returning to the operator so the call can be completed. The response time the operator sees (and thus what governs the number of calls per hour he or she can answer) is now dependent on not only the responsiveness of both the order entry system and the shipping system servers, but also on the responsiveness of the connection between them. An Infrastructure model would be used to determine the responsiveness parameters for each, and that information would then be used to update the Application level model to determine the business impact.

4.1.6 Component

The Component level is the most detailed view because it focuses on the underlying components. These components can be related to anything in the environment because almost anything can be broken down into smaller components and modeled to see the impact of various changes. For example, chip vendors can model how changes to the Level-1 and Level-2 cache affect CPU performance, and system vendors can model how changes in the internal buses affect memory and disk accesses. Neither of these models is of much interest to application designers or system administrators because they cannot make changes at those levels; they can only choose which systems to use. However, some component level models are very useful. If the performance analyst had an accurate model of the cache algorithms for each type of disk subsystem, then he or she could assess the performance impact of each subsystem on individual applications. This is not normally an area of concern for capacity planners and performance analysts because there are few modeling tools and very little measurement data to use with the tools. A model at this level predicts how sensitive the systems are to a wide variety of issues, such as component level limits to application scaleability. Even though it is seldom of much use by itself, the Component level provides additional information for the other levels of the Modeling Pyramid. The objective is to understand the workload dependent characteristics of different component architectures and to evaluate alternatives at each of the application life-cycle stages based on how the application of interest takes advantage of each architecture.

The outputs from the Component level are varied depending on the individual components. For network components they could be through-put and latency, while for disk subsystems they could be I/O response times and the number of concurrent I/Os. This corresponds to the Deployment, Production and Planning stages of the application life-cycle. Modeling Pyramid metrics will depend on the components and the level where the model will be used.

As vague and general as the above description is, when faced with a specific example it is relatively easy to see the value of a Component level model. Let's assume that the new shipping system is to be connected to the order entry system over one of the two existing LAN (Local Area Network) segments the organization already has in place. One LAN is a 10 megabit Ethernet and the other is a 16 megabit Token Ring. Remember, the response time the operator sees (and thus what governs the number of calls per hour he or she can answer) is dependent on the responsiveness of the connection between these servers. A Component model would be used to determine how the different LAN architectures effected the responsiveness of the transactions given the specific data and timing requirements of the application. In addition, the model could then be used to show the improvements of using a 100 megabit Ethernet segment, even though the organization does not have one installed. That information would then be used to update the Application level model to determine the business impact.

4.2 Modeling Pyramid Implementation

The Modeling Pyramid can be used in a number of different ways depending on the needs of the or-

ganization. The objective is to relate the activities at each level with the level above and below. Although the techniques used at each level are independent of this relationship, the outputs from some tools may match better with the required inputs for tools at other levels. Therefore, it is advantageous for the analysts at each level to understand the uses of modeling information at all levels. This understanding promotes synergies between activities at the different levels.

Once the decision has been made to connect the activities between levels in the Modeling Pyramid, the next question is how should the overall flow of control progress? The most obvious answers are top down and bottom up, similar to the traditional application code development techniques. In addition, two other approaches, inside out and outside in, can be used to improve the parallelism of the overall effort. There are advantages to each approach and any one of them is better than not making the connection. However, there are also disadvantages to each approach. Just as the Spiral Modeling Methodology (Norton 1998b) addresses the shortcomings with each level, the Spiral Approach to the Modeling Pyramid provides better integration and communication between the Modeling Pyramid level.

4.2.1 Top Down

Starting at the top of the pyramid means to begin with the overall strategic objectives and move to the next level when all of the implications of the current one have been fully explored. This approach provides the best long term strategic results because activities at each level are developed using the objectives from the higher one. Although an ideal approach, there are practical limits to it because of the time required. Few organizations have the luxury to delay change until everything is well understood. Deadlines are often established by outside forces, and hardware vendors require some lead-time to build the equipment.

In the call center example, the projected call volume from the marketing plan (Strategic level) would be used in the Business model to identify the required call duration time. That in turn would be used in the Application model to determine the required transaction response time, which would be used in the System model and the Infrastructure model to validate the ability of the environment to deliver the necessary response time. Any specific problems would be modeled at the Component level to resolve issues or select better alternatives.

4.2.2 Bottom Up

Starting at the bottom of the pyramid means to begin with the bits and pieces and to fit them together, hoping for some overall strategic direction to emerge along the way. This approach provides the fastest reaction when a need has been identified, but often leads down the wrong road. Having to backtrack because of missing a single turn can easily consume any advantage of the quick start. However, if the organization is relatively stable and the overall strategic objectives are generally well known, this approach has the advantage of providing demonstrable results very quickly.

In the call center example, proposed components would be modeled to determine their performance characteristics. The System model and Infrastructure models would use that information to design the best performing configuration. The Application model would identify the best of the design alternatives for the proposed configuration and provide the Business model with expected call capacity. This information would feed the Strategic model to allow marketing to project the possible business growth.

4.2.3 Inside Out

Starting in the middle of the pyramid means to begin with the Application level and work outward in both directions. Although attractive on the surface because it is more application focused, this approach has the disadvantages of both the top down and bottom up approaches because it lacks the overall initial strategic direction and doesn't get to the component level as quickly.

In the call center example, an Application model would determine the best overall design and provide the required transaction response time information to the System model and the Infrastructure model, which in turn would validate the ability of the environment to deliver the necessary response time. Any specific problems would be modeled at the Component level to resolve issues or select better alternatives. At the same time, the Application model would provide the Business model with expected call capacity. This information would feed the Strategic model to allow marketing to project the possible business growth.

4.2.4 Outside In

Starting in the edges of the pyramid means to begin with the both the Strategic and the Component levels to establish the overall initial strategic direction and also get the hardware identified quickly. This can be a good approach when it works, but the chances of moving from both extremes toward the center and meeting are low. There are too many alternatives to be considered along the way and any one of them can invalidate all of the work that has been done moving from the other direction.

In the call center example, the projected call volume from the marketing plan (Strategic level) would be used in the Business model to identify the required call duration time. That in turn would be used in the Application model to determine the required transaction response time. At the same time, the proposed components would be modeled to determine their performance characteristics. The System model and Infrastructure models would use that information to

design the best performing configuration. The Application model would identify the best of the design alternatives for the proposed configuration.

4.2.5 The Spiral

If all of the above approaches to using the Modeling Pyramid have limitations, then what was the point? The Modeling Pyramid is really about the relationships between the levels and not the levels themselves. The best way to use it is to start everywhere at the same time by making assumptions about the other levels. As information becomes available from each level, it is used to either validate or replace the assumptions at the other levels.

In the call center example, the speculated call volume from the marketing (Strategic level) would be used in the Business model to identify the required call duration time while the Application model investigated design alternatives and determined the required transaction response time. The System and Infrastructure models would investigate different environmental configurations while any specific problems would be modeled at the Component level to resolve issues or select better alternatives. Once the model from a level produces improved results, that information is made available to the models at the other levels so they can be refined. The process continues until all levels are satisfied with the overall projections. Changes at any level initiate a new cycle through the Spiral process to insure that the impact of the change is addressed at all levels.

5. Conclusion

The Modeling Pyramid provides a technique to see the progression of modeling an application from birth to death (replacement) to get objective information for sound business decisions at every step. The technique is built on the foundation of understanding the value of a model at each of stage of the application life-cycle: Design, Implementation, Deployment, Production, Planning, and Enhancement. The technique allows the use of either of the general modeling tools, simulation or analytic models, at any of the levels because the interfaces between the levels are defined in application and business terms, not some internal model construct. As required, the models at any stage or level can be enhanced with other specialized techniques, such an UML, ETE response time measurements, SPE, Simalytic Models, business models, node modes, or platform-centric models. Each of these specialized techniques improves the accuracy of the model for that stage of the application life-cycle and those improvements are passed along to the other models using the Modeling Pyramid.

The Modeling Pyramid defines a methodology for communication across different organizations with different objectives. It helps identify when these objectives are mutually exclusive and when they support each other. The levels in the Modeling Pyramid (Strategic, Business, Application, System, Infrastructure, and Component) provide the needed perspective, in terms of objectives and granularity of detail, to keep all of the activities focused on making the best possible decisions for the overall business.

6. References

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

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.

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

Norton, Tim R. 1997a. Simalytic Hybrid Modeling: Planning the Capacity of Client/Server Applications. In Proceedings Volume 6. 15th IMACS World Congress,: 583-588. Berlin, Germany: International Association for Mathematics and Computers in Simulation.

Norton, Tim R. 1997b. Simalytic Modeling: A Hybrid Technique for Client/Server Capacity Planning. In Proceedings. Summer Computer Simulation Conference,: 172-177. Arlington, Virginia: Society for Computer Simulation.

Norton, Tim R. 1998a. Don't Predict Applications When You Should Model the Business. In Proceedings. Computer Measurement Group,: 922-933. Anaheim, CA: CMG, Inc.

Norton, Tim R. 1998b. A Practical Approach to Capacity Modeling. In Workshop Proceedings. Computer Measurement Group,: 1-123. Anaheim, CA: CMG, Inc.

Norton, Tim R. 1999. End-To-End Response Time: Where to Measure? In Proceedings. Computer Measurement Group,: 322-330. Reno, NV: CMG, Inc.

Smith, Connie U. 1988. How to Obtain Data for SPE Studies. In Proceedings. Computer Measurement Group.

Smith, Connie U. 1990. Performance Engineering of Software Systems. The SEI Series in Software Engineering. Reading, MA: Addison-Wesley Publishing Co.

Smith, Connie U. and Lloyd G. Williams. 1998. Performance Evaluation of a Distributed Software Architecture. In Proceedings. Computer Measurement Group.

UML. Unified Modeling Language. http://www.omg.org/: Object Management Group, Inc.