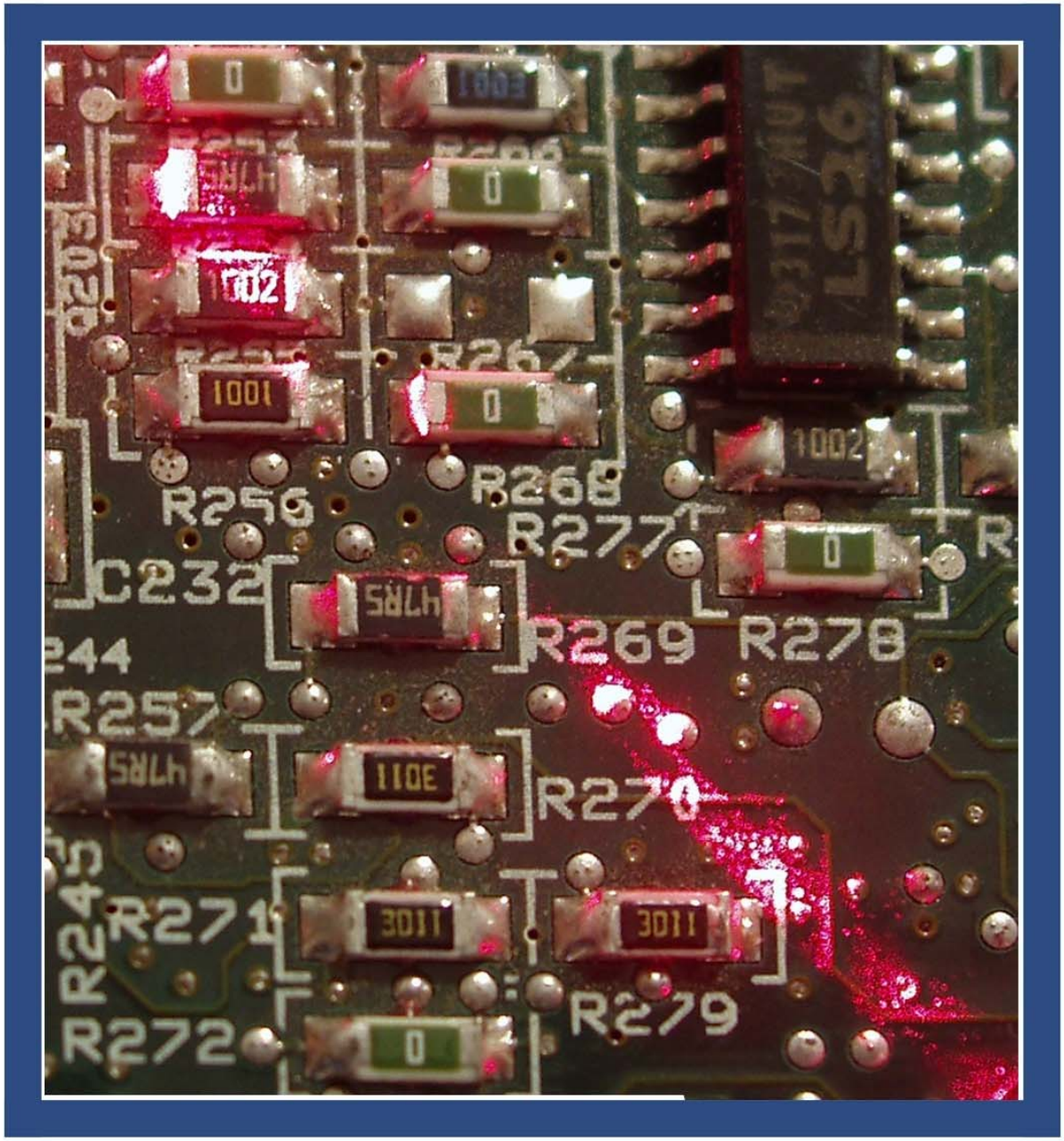
Issue
One Hundred Thirty
Autumn 2011

Journal of Computer Resource Management

A Publication of
The Computer Measurement Group, Inc.



The Computer Measurement Group, Inc. (CMG) is the most influential organization worldwide for the exchange of information among Computer Performance Evaluation (CPE) professionals. To keep CPE professionals on top of their field, CMG is dedicated to providing its members the education, networking, and leadership opportunities vital to success in today's competitive industry.

PUBLICATIONS & MEMBERSHIP BENEFITS:

- The ***Proceedings*** contain all the information presented at the annual CMG conference. With more than one hundred papers on current hot topics, the *Proceedings* is your best source of the latest information in the expanding field of CPE. Some Late Breaking papers may not be in the *Proceedings;* they appear in a *Journal* issue.
- The ***Journal*** provides an additional source of information with emphasis on special interest topics and/or focus on specific CPE industry issues. Members receive the *Journal* at least three times a year.
- The ***Bulletin*** is a newsletter written by members for members. This publication is available in electronic format and includes vital information on CMG's national, regional, and international groups.
- **Various Internet Services.** CMG provides various online membership only services including a directory of all CMG members.
- CMG's electronic newsletter, **MeasureIT,** sent to all those who subscribe.

INFORMATION FOR AUTHORS

The *Journal of Computer Resource Management (CMG Journal)* is distributed only to members of the Computer Measurement Group, Inc. (CMG). CMG membership consists primarily of practicing professionals in the area of computer performance, analysis, and capacity planning. The objective of the *CMG Journal* is to bring the membership information of current importance involving:

- performance characteristics of computer systems of all sizes and architectures
- techniques to measure, analyze, compare, predict, and report performance
- techniques for managing computer performance, including capacity management, computer cost accounting, and computer performance reporting
- integration of disparate components into hardware and software systems to meet organization needs.

The intent of CMG is to publish relevant papers as rapidly as possible, preferably within six months of completion. Authors can assist CMG in achieving this objective by submitting papers in a uniform format.

To submit your paper for consideration in the *CMG Journal,* it must follow a format consistent with CMG guidelines. For complete instructions on paper submittal you should download two documents from http://www.cmg.org/national/journal.html located on the CMG Website. These documents are 'Paper Guidelines and Instructions' and 'CMG License to Publish.' The first document will provide guidelines on format and text; the second document is a required form that must be signed/returned to CMG that grants your permission for CMG to publish your paper.

Here are some highlights on what is expected from authors:

- Camera-Ready Papers, in Microsoft Word, should be sent via email to journal@cmg.org
- Electronic copies of the paper are preferred; send two hardcopies of the paper in the required format (see 'Instructions' document for details) if an electronic copy is not possible. Hardcopies must be camera ready Papers should be submitted three months prior to the Journal publication date
- Please indicate if you wish to review the paper after editing, prior to publication
- Refer to all tables and figures by number

Authors must read and follow the 'Instructions for Preparing Papers' document. The above highlights do not include all the guidelines required for submitting your paper. Papers not complying with the instructions may be rejected for inclusion in the *CMG Journal.*

Journal of Computer Resource Management

A Publication of the Computer Measurement Group

Issue 130

Table of Contents

CMG Journal #130: Letter from the Editor

Welcome to CMG Journal number 130, our third and final issue for 2011. Fall is upon us in the northern hemisphere, all the leaves are brown and the sky is gray. However, once again we have a great issue for you, guaranteed to brighten up your day and carry you through to Conference. This issue features six outstanding papers on a wide variety of topics. This will be our last issue for 2011.

Leading this issue off is Tim Sweetz with *Capacity Planning Concepts for Telecom Systems.* Telecom systems are the primary means of communication between customers and companies throughout the world. As these systems have evolved over time, the computerized infrastructure behind modern call centers provide companies with a high level of customization, but can also lead to complicated and expensive operational processes. One way to reduce costs associated with telecom technologies is through the discipline of capacity planning. Tim's paper will present some of the key concepts and basic methodologies required to provide capacity planning services for telecom systems.

Our second paper, *Why Models Fail-A Case Study: A Workload Analysis Model* is from frequent CMG contributor Tom Wilson. What does it mean for a model to fail? It means that the model failed to provide the insight that it was meant to provide. A model starts with an objective, has a design and implementation, and then is put to use. The case study in Tom's outstanding paper examines a workload analysis model, from which a performance test model is derived, and discusses why the model failed.

Our third paper, *An Effective Implementation of CMMI for Performance Testing Projects – a Case Study* was written by Nidhi Tiwari and Veena Rajendiran. Today performance testing is well recognized, widely practiced and sufficiently equipped with tools. However, little emphasis is given to process implementation, tracking and improvement of performance testing projects, resulting in exponentially high Cost of Quality (COQ). The authors share their experience implementing CMMI for performance testing projects to control their COQ. Subsequent benefits obtained by organization are also included in the paper.

Batting cleanup is David Lytle with his paper titled *Not Your Father's or Grandfather's Mainframe Any More.* David reviews the history of mainframe I/O and compares it with distributed systems I/O. He then highlights some of the more recent developments in mainframe I/O introduced the past three years and that are currently being implemented on the latest mainframe processors from IBM.

Our fifth paper, *Processor Selection for Optimum Middleware Price/Performance* was written by David Kra. Many middleware products can be deployed onto many combinations of processor architecture and operating system. Finding the most cost effective combination is complicated by software pricing based on vendor core weighting factors. David's paper explains how to combine core weights, core counts, and performance data to calculate and compare a "Performance Rate per Weighted Core." Results are provided for the Oracle data base server as used in published TPC-C and TPC-H benchmarks.

Last but not least is our sixth and final paper this issue. *Exploratory Study of Performance Evaluation Models for Distributed Software Architecture* was written by Boluwaji A. Akinnuwesi, Faith-Michael Uzoka, Hyacinthe Aboudja, Mathieu Kourouma, Victor W. Mbarika, S.O Olabiyisi, and E.O Omidiora. Several models have been developed to evaluate the performance of Distributed Software Architecture (DSA) in order to avoid problems that may arise during system implementation. This paper presents a review of DSA performance evaluation models with the view of identifying the common properties of the models. It was established in this study that the existing models evaluate DSA performance using machine parameters such as processor speed, buffer size, cache size, server response time, server execution time, bus and network bandwidth size and lots of others. The models are thus classified to be machine-centric. Moreover the involvement of end users in the evaluation process is not emphasized. Software is developed in order to satisfy specific requirements of the client organization (end-users) and therefore involving users in evaluating DSA performance should not be underestimated. This study suggests future works on establishing contextual organizational variables that can be used to evaluate DSA.

Thanks to everyone who contributed to this CMG Journal. CMG'11 is a month away and will be here before we know it. Hopefully you are planning on attending CMG'11, have registered, and if you have not already done so, please consider volunteering to help with the Conference. We are always looking for session chairs, and other on site volunteers to help at the Conference. Even if you are submitting a paper for the Conference, please consider writing a paper for the CMG Journal. You can submit your papers, as well as feedback to us at cmgjournal@cmg.org.

Thanks again for reading, and we hope you enjoy this issue.

Stephen R. Guendert, Ph.D

CAPACITY PLANNING CONCEPTS FOR TELECOM SYSTEMS

Tim Sweetz
Bank of America
tim.sweetz@bankofamerica.com

Telecom systems are the primary means of communication between customers and companies throughout the world. As these systems have evolved over time, the computerized infrastructure behind modern call centers provide companies with a high level of customization, but can also lead to complicated and expensive operational processes. One way to reduce costs associated with telecom technologies is through the discipline of capacity planning. This paper will present some of the key concepts and basic methodologies required to provide capacity planning services for telecom systems.

Introduction

Arguably one of the most significant inventions in our history, the invention of the telephone in the late 1800s truly revolutionized communication methods. Replacing the well established telegraph infrastructure was not easy, but the technology of the telephone was a key enabler in the realm of communication that allowed for the sophistication and complexities that exist today. As the technology has continued to evolve over the years, the telephone is still a key component in telecom systems throughout the world.

Telecom systems provide the primary means of communication between customers and companies. While there are many aspects and technologies involved in telecom systems, the focus of this paper will be on Integrated Voice Response (IVR) systems. There are many benefits in having an efficient IVR system; for example:

- Automation of customer-based business processes
- Decreased call center staffing requirements
- Decreased time required to answer the customer's call
- Ability to provide self-service options to customer's without having to speak with a human agent
- Identify and authenticate the caller prior to human agent taking call to reduce call times

However, an inefficient or unavailable IVR system can have a very negative impact on a company's reputation. Like it or not, many customers associate the person on the phone with the company, and if the company will not answer their call, it is going to hurt the company's reputation. With this in mind, many companies rely on vendor recommendations and overprovision resources to ensure their IVR system will always be readily available. This is where the discipline of capacity planning, when applied appropriately, can be utilized to dramatically reduce costs and instill a sense of confidence that there will be enough capacity to accommodate peak periods in the call center.

Telecom Capacity Planning

Capacity Planning for IVR systems is not dramatically different than planning for many other types of systems; however, while many of the principles and methodologies involved in capacity planning for other systems remain the same, there are some significant differences that are typically only found when capacity planning for a call center environment. Specifically, the use of the Erlang Distribution is typically only used when planning for telecom technologies, although it is sometimes found in other cases as well.

Erlang Distribution

Capacity planning for telecom technologies utilizes a queuing theory based on the Erlang Distribution. The Erlang Distribution was developed in the early 1900s by Agner Krarup Erlang. While working for the Copenhagen Telephone Company in Denmark, Erlang developed mathematical formulae to evaluate the trade-off between low cost/poor service & high cost/excellent service. His work has been extremely influential across the globe and remains the basis for telecom engineering. In 1946, the International Consultative Committee on Telephones and Telegraphs (CCITT) adopted the name "erlang" as the basic unit of telephone traffic.

There are two main formulae for the Erlang Distribution: Erlang B and Erlang C. Erlang B, sometimes referred to as the Erlang Loss Formula, is the most commonly used formula and is designed to evaluate how many lines, or ports, are required for a specified amount of traffic. The other formula is the Erlang C, which allows one to calculate the probability that a customer will have to wait for a resource. In addition to these two main formulae, there is also the Extended Erlang B (sometimes referred to as Erlang B+), which is very similar to Erlang B but it assumes that a percentage of calls are immediately represented to the system if they encounter blocking.

Erlang

The basis for all Erlang calculations is the erlang. An erlang is a dimensionless unit of telecommunications traffic measurement describing the total traffic volume for a specified timeframe (typically one hour). Traffic in erlangs (E) is defined by the following formula:

$$E = \frac{\lambda h}{\mu}$$

- $E =$ Total amount of traffic offered in erlangs
- $\lambda =$ Call arrival rate
- $h =$ Average call handling time
- $\mu =$ Service Rate, or total time

For example, if you received 30 calls in one hour and each had an average duration of 5 minutes, the traffic figure would be (30 * 5) / 60 = 2.5 erlang. Finding the erlang number for your situation is rather straightforward and is the first step in using the Erlang formulas.

Erlang B

The Erlang B queuing model is denoted in Kendall notation as M/M/n/n. In this model, arriving customers have zero waiting positions. It assumes Poisson arrivals and exponentially distributed service times. Since the model does not provide for any waiting positions, if a customer finds no servers available, it is assumed the customer goes away and is lost. For this reason, the model is called a loss system. The lost customers are also said to experience blockage or to be blocked.

The Erlang B formula assumes an infinite population of sources, which jointly offer traffic to *N* servers. The rate of arrival of new calls is constant and does not depend on the number of active sources, because the total number of sources is assumed to be infinite. The rate of call departure is equal to the number of calls in progress divided by the mean call holding time. The formula was designed to calculate blocking probability in a loss system and provides the grade of service (GoS).

$$P_b = B(E, m) = \frac{\frac{E^m}{m!}}{\sum_{i=0}^{m} \frac{E^i}{i!}}$$

- $P_b =$ Probability of blocking
- $m =$ number of resources (servers or circuits)
- $E =$ Total amount of traffic offered in erlangs

Extended Erlang B

The Extended Erlang B, also referred to as Erlang B+, is an iterative calculation rather than a formula. It adds an extra parameter to the Erlang B formula, the recall factor, which defines the recall attempts. It assumes a specified number of calls are immediately represented to the system if they encounter blocking (a busy signal).

Erlang C

The Erlang C queuing model is denoted in Kendall notation as M/M/n/∞, or simply M/M/n. As is the case with Erlang B, it assumes Poisson arrivals and exponentially distributed service times. However,

where the Erlang B model has zero waiting positions, the Erlang C model assumes an infinite number of waiting positions. Therefore, every arriving customer will eventually be served, even if they have to wait a long time. In direct contrast to the Erlang B model, no customer ever experiences blockage. However, if the number of erlangs exceeds the number of servers, then the system becomes unstable in the sense that the number of customers grows without limit. Part of the Erlang C queue definition specifies how waiting customers are serviced. It assumes a FIFO queuing discipline; the longest waiting customer will always be the next customer to begin service.

$$P_W = \frac{\frac{A^N}{N!}\frac{N}{N-A}}{\sum_{i=0}^{N-1}\frac{A^i}{i!} + \frac{A^N}{N!}\frac{N}{N-A}}$$

- $A =$ Total traffic offerend in units of erlangs
- $N =$ Number of servers
- $P_W =$ Probability that a customer has to wait for service

While one can certainly perform the calculations necessary to obtain the necessary output using the Erlang formulae, the process can become tedious. Because of this, the textbooks and tables that were previously required have been replaced by software with many Erlang calculators that are now widely available. Some are free to use and download, while others charge a nominal fee. There are also Excel plug-ins and code for C and JavaScript depending on your preferences. Any of these options make using the Erlang formulae quick and easy and are rather easy to find using your search engine of choice.

Other Key Components for Telecom Capacity Planning

Now that we have detailed some of the basic concepts related to erlangs and the Erlang formulas, we need to briefly touch on the other key components involved in Telecom Capacity Planning: call volume and average handle time (AHT).

When dealing with call volumes, it is important to remember that calls have a tendency to bunch up. Because of this, all capacity planning efforts related to telecom traffic need to focus on peak periods. The industry norm is to deal with the peak hour. There are two primary options to choose from when attempting to determine the peak busy hour traffic. One method is to take the busiest hour from the past 13 months (if that amount of data is available) and use that as the peak. The other would be to take the busiest hour of each day for five or ten days during the busiest time of the year, and then calculate the average of those hours' traffic load to derive the average busy hour. This decision can be driven by business requirements, but in my experience it is best to go with the actual peak hour. Averaging peak values just has the potential to water down the peak to a point that can leave you susceptible to unavailability during a true peak period.

The other key input that we need for telecom capacity planning is the AHT. Simply stated, AHT is a telecom metric for the average duration of one transaction. AHT is typically measured from the caller's initiation of the call and including any hold time, talk time, and related tasks. AHT is required to determine the number of erlangs which is a key input into the Erlang formulae.

Putting it all together

Now that we have an understanding of these key concepts, we will bring it all together with some hypothetical examples. Using a typical telecom environment, we have IVR servers located in three data centers. Each data center has 14 servers, and each server has 192 physical ports available. Accounting for disaster recovery, our SLA states we must be able to handle the peak workload across any two data centers at any time (N-1). We have accumulated call volume data for the previous 6 months, and we have obtained the peak hourly call volume and the AHT. In addition, we have compiled the busy hour concurrent port (BHCP) usage at the server level. In the absence of specific call volume projections, the business is providing a BAU growth projection of 3% per month. This information is summarized below and can be used to build a simple linear capacity model to visually represent the capacity forecast for the next 12 months.

Month	BHCP	Peak Call Volume	AHT	Erlang
July 2010	2434	62,450 / hour	132 seconds	2290
August 2010	2048	52,231 / hour	132 seconds	1915
September 2010	2282	58,876 / hour	131 seconds	2142
October 2010	2420	61,175 / hour	134 seconds	2277
November 2010	2461	65,225 / hour	134 seconds	2316
December 2010	2384	60,673 / hour	133 seconds	2242
AVERAGE	**2338**	**59,605 / hour**	**133 seconds**	**2197**
PEAK	**2461**	**65,225 / hour**	**134 seconds**	**2316**

Table 1 – Historical Call Data

Site	# of Servers	Total # of Ports
Data Center 1	14	2688
Data Center 2	14	2688
Data Center 3	14	2688
TOTAL	**42**	**8064**
TOTAL (N-1)	**28**	**5376**

Table 2 – Infrastructure Data Used in Example

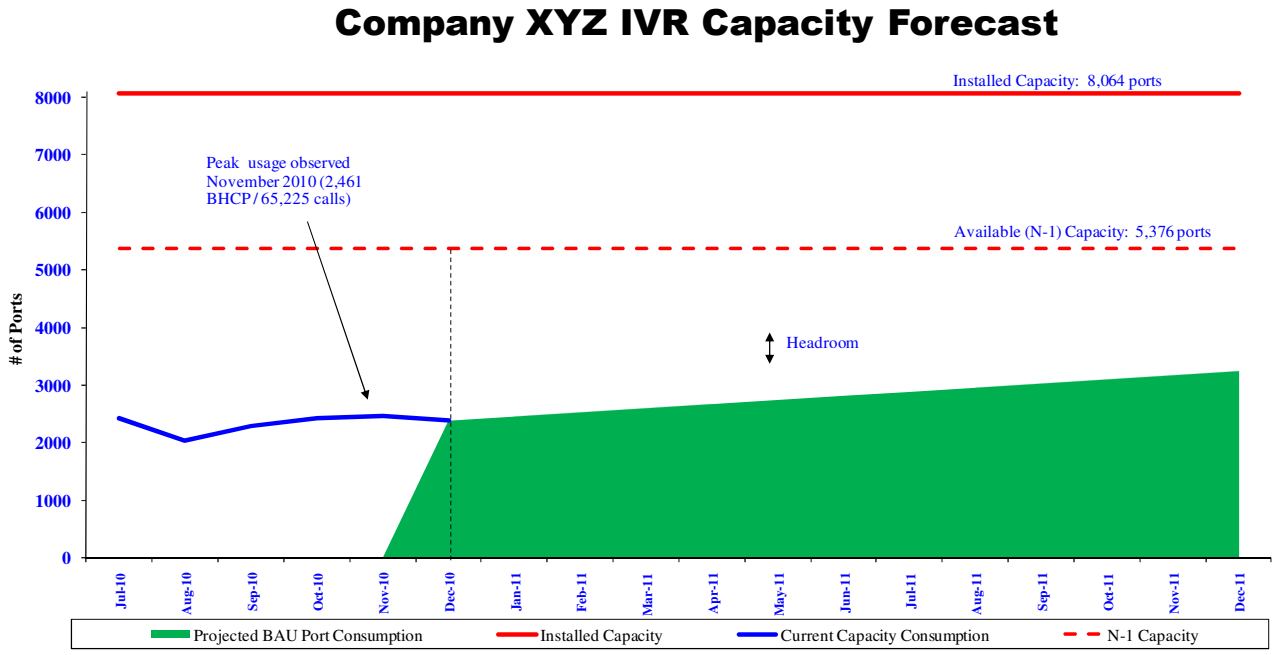


Figure 1 – Example Capacity Forecast Chart

As seen in Figure 1, the company has been underutilizing the available hardware and ports. Over the past six months, the peak port usage is 46% of their N-1 capacity threshold of 5,376 ports. However, given the aggressive growth projection of 3% per month, the forecast by the end of the year is 3,242 ports, or 60% of their N-1 capacity. Given this level of utilization, there would be an opportunity to cut costs by reducing the number of servers and ports in their environment. The following graph shows what the company's IVR system's capacity would look like if we released two servers from each data center (6 total servers):

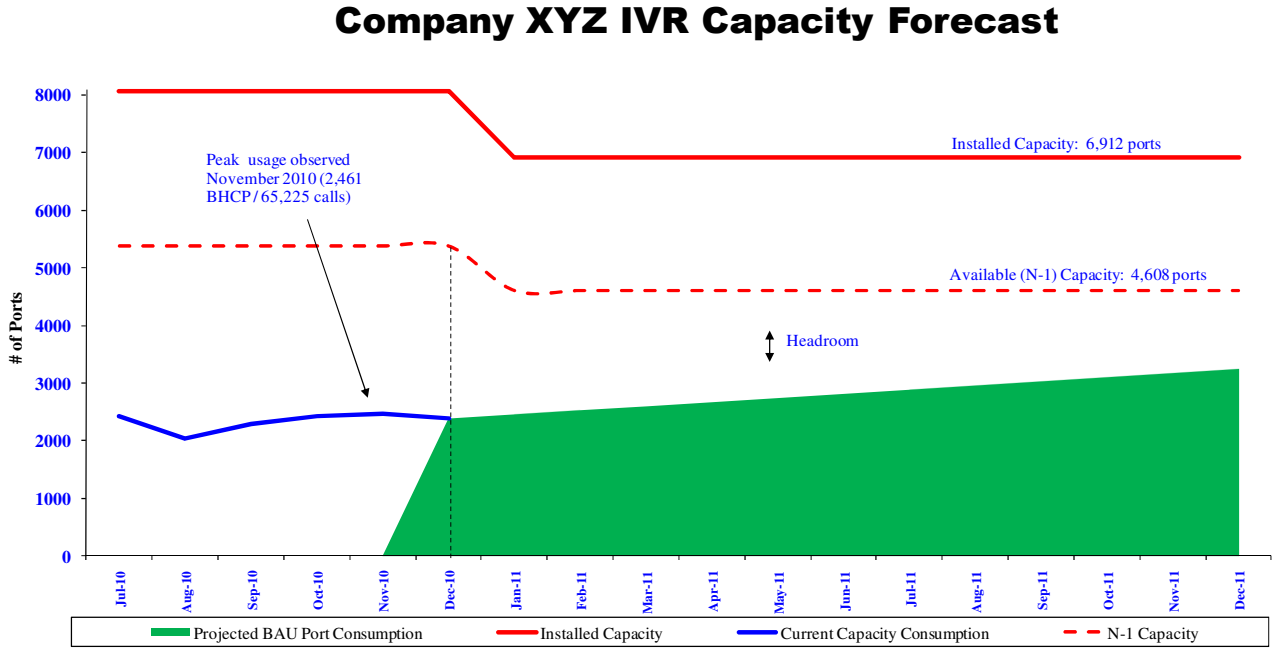


Figure 2 – Capacity Forecast Chart Showing Reduction in Capacity

As seen in Figure 2, even after removing these 6 servers from the environment, the company would maintain sufficient capacity through the end of the year. Given the forecast of 3,242 ports in December 2011, the company would be using 70% of their N-1 capacity while still maintaining a nice buffer of 30%. Releasing 6 servers and 1,152 ports would present a significant cost savings while still maintaining a high level of service and confidence in the capacity of the environment.

Practical Examples

Another key application of these telecom capacity concepts involves analyzing changes in either call volume or AHT, typically associated with a project or marketing campaign. Several scenarios are presented below:

- A new marketing campaign slated for May 2011 in which the business anticipates a 20% increase in call volume and no change in AHT
- A significant change in product slated for July 2011 in which the business anticipates a 10% increase in call volume and an increase in AHT by 10 seconds
- A code release expected to improve efficiency of the IVR system slated for September 2011 in which the business anticipates no change in call volume but a decrease in AHT by 15 seconds

Scenario 1

For the first scenario, in which we anticipate a 20% increase in call volume and no change in AHT, we would utilize the Erlang B formula to determine the number of ports required for this increased workload. The steps are detailed next:

1) Use the peak hour call volume (65,225) to calculate the new peak with the additional 20%: 65,225 * 1.2 = 78,270
2) Find the number of Erlangs for the increased call volume (using same AHT of 134): (78270 * 134)/3600 = 2913 erlangs
3) Next, find the number of ports required to support this increased workload (using an Erlang B calculator): 3,073 ports required
4) Find the difference between the new projected peak and the current actual peak: 3073 – 2461 = 612 ports

Given the projections in this scenario, the IVR system would need an additional 612 ports to support the marketing campaign. We can plug this number back into our capacity forecast model to determine if this will push us over our thresholds or if we will have sufficient capacity to accommodate this additional volume. As seen in Figure 3, the increase in call volume would obviously increase our usage, but we would still be well within our capacity thresholds.

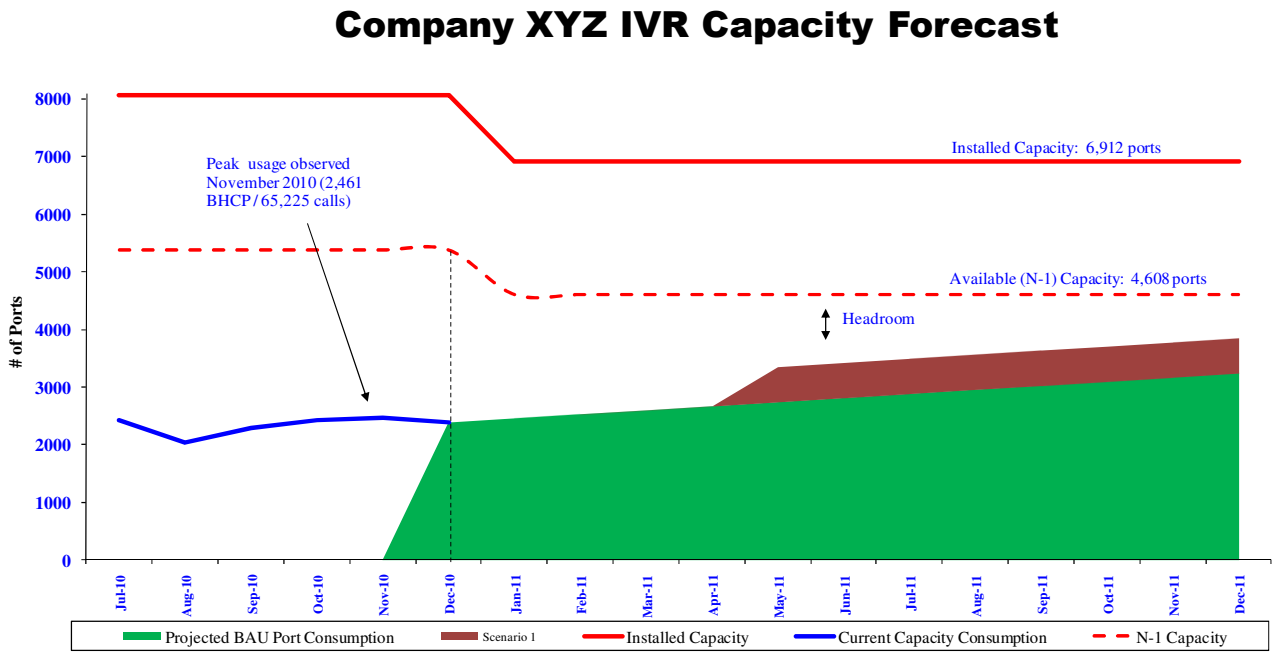


Figure 3 – Capacity Forecast Chart Showing Impact from Scenario 1

Scenario 2

For the second scenario, in which we anticipate a 10% increase in call volume and a 10 second increase in AHT, we would also utilize the Erlang B formula to determine the number of ports required for this increased workload. The steps are detailed next:

1) Use the peak hour call volume (65,225) to calculate the new peak with the additional 10%: 65,225 * 1.1 = 71,748
2) Find the number of Erlangs given the increase in call volume and AHT: (71748 * 144)/3600 = 2870 erlangs
3) Next, find the number of ports required to support this increased workload (using an Erlang B calculator): 3,029 ports required
4) Find the difference between the new projected peak and the current actual peak: 3029 – 2461 = 568 ports

Given the projections in this scenario, the IVR system would need an additional 568 ports to support the expected workload changes due to the change in product. We can then plug this number back into our capacity forecast model to determine if this will push us over our thresholds or if we will have sufficient capacity to accommodate this additional volume. As seen in Figure 4, the increase in call volume would increase our usage, but we would still be well within our capacity thresholds.

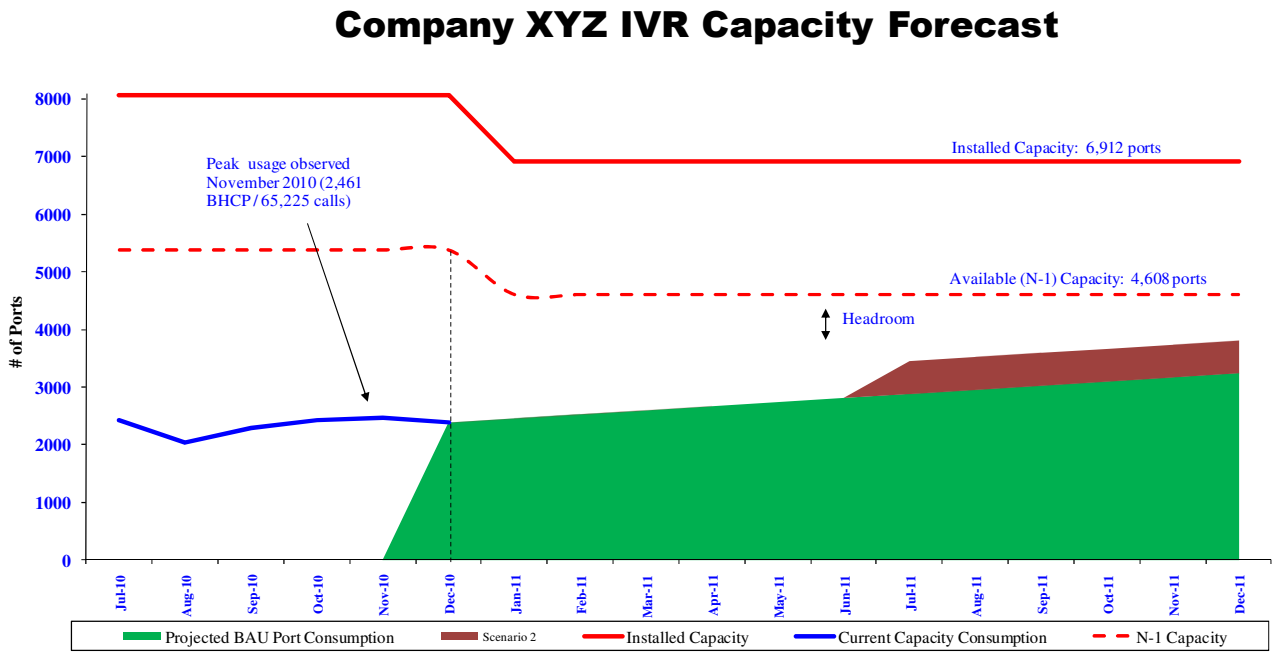


Figure 4 – Capacity Forecast Chart Showing Impact from Scenario 2

Finally, for the final scenario in which we anticipate no change in call volume but a 15 second decrease in AHT, we would also utilize the Erlang B formula to determine the number of ports required for this decreased workload. The steps are detailed next:

1) Use the existing peak hour call volume: 65,225
2) Find the # of Erlangs given the decrease in AHT: (65225 * 119)/3600 = 2156 erlangs
3) Next, find the number of ports based on this decreased workload (using an Erlang B calculator): 2,296 ports required
4) Find the difference between the new projected peak and the current actual peak: 2296 – 2461 = -165 ports

Given the projections in this scenario, the IVR system would see a decrease of 165 ports as a result of the expected decrease in AHT due to the efficiencies gained from the code changes. We can then plug this number back into our capacity forecast model to adjust our forecast accordingly. As seen in Figure 5, the impact is not as significant as the first two scenarios, but the decrease is observed.

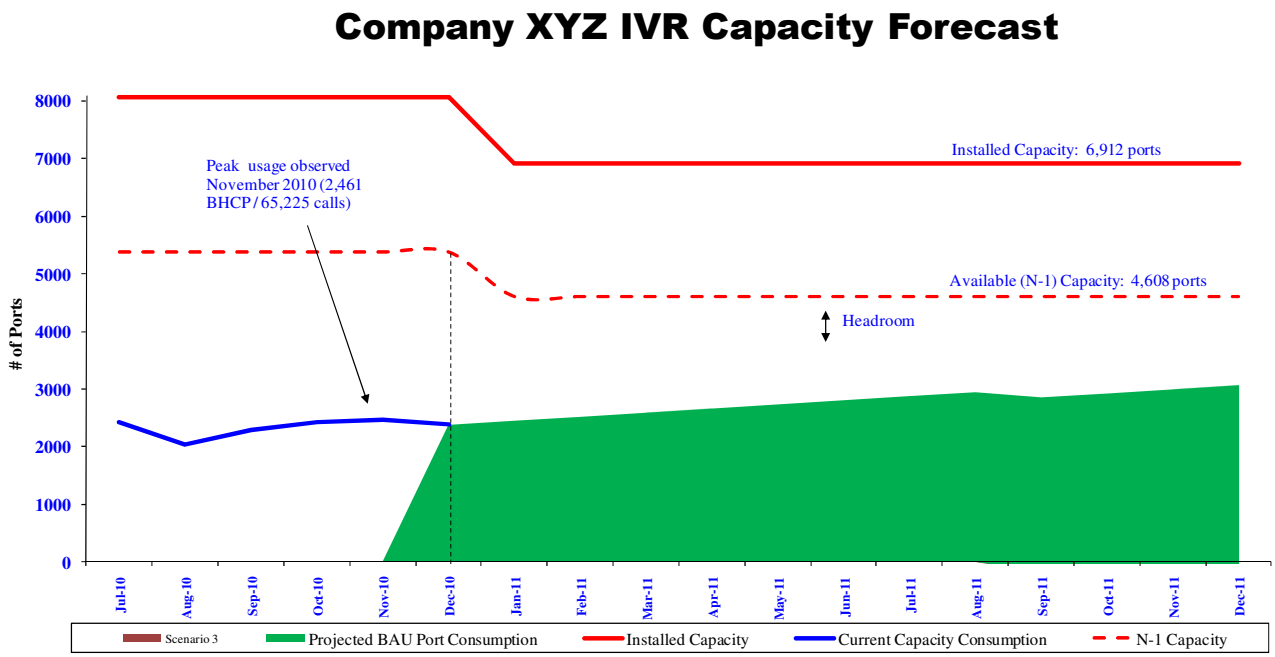


Figure 5 – Capacity Forecast Chart Showing Impact from Scenario 3

Combining the Scenarios

The real benefit to modeling impact from various scenarios is when multiple changes are planned and you need to determine what your capacity will look like when they are all combined. This holistic view is critical to ensuring you provide a telecom environment that has adequate capacity. The three scenarios are now shown together in Figure 6.

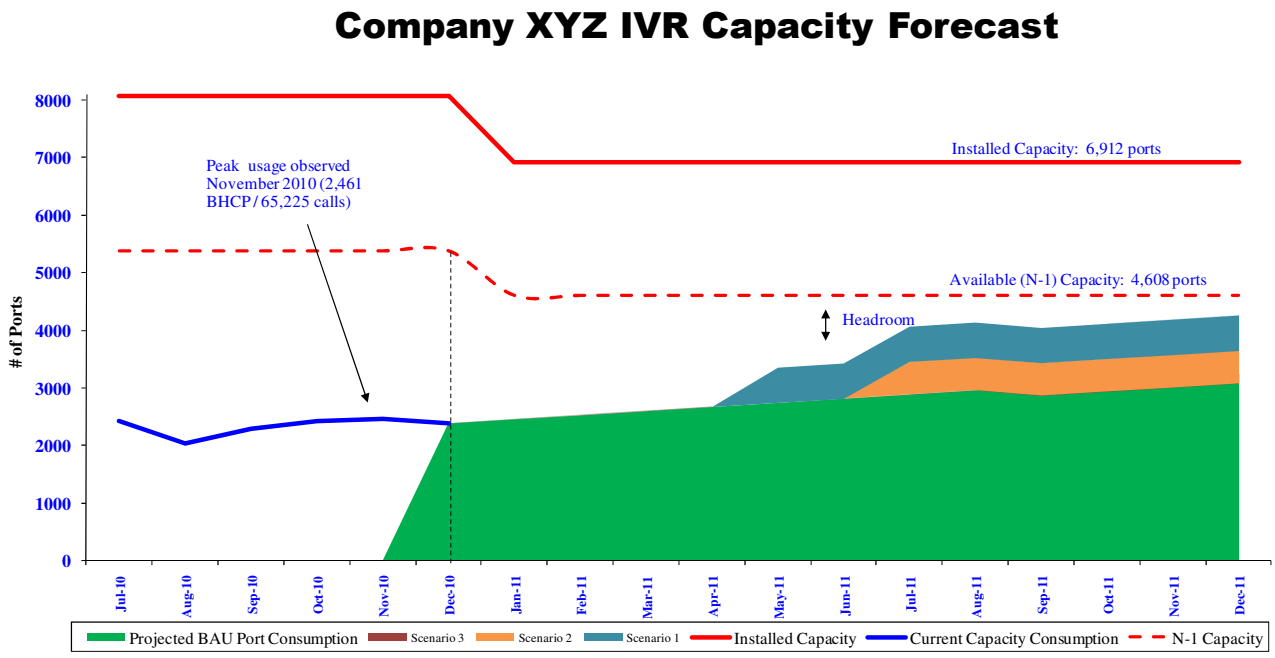


Figure 6 – Capacity Forecast Chart Showing Impact from all Three Scenarios

As seen in Figure 6, the impacts from all three scenarios combined bring us closer to our N-1 threshold, but we still forecast sufficient capacity. The third scenario, while not looking as significant by itself, really helps bring the forecast down when taking all three scenarios into account. Without the decrease gained from scenario 3, we would be right at our N-1 threshold and most likely looking at adding capacity to provide more breathing room.

Conclusion

Capacity planning for telecom systems involves some unique concepts not found in other disciplines of capacity planning. In order to provide capacity planning services for telecom systems, one should work to develop an understanding of these concepts. This paper has provided an overview of erlangs and the Erlang formulas, as well as discussing some other key inputs involved in telecom capacity such as call volume and average handle time. With an understanding of these concepts, the remaining steps to develop a telecom capacity plan follow standard capacity planning methodologies. Collecting good data in terms of your call volumes and average handle times will be critical in developing a quality capacity plan for your telecom systems.

References

[ABST08] "About Queuing Models" Abstract Micro June 2, 2011: <http://abstractmicro.com/erlang/helppages/mod-about.htm#kendall>.

[ABST08] "Erlang B Queuing Model" Abstract Micro June 2, 2011: <http://abstractmicro.com/erlang/helppages/mod-b.htm>.

[ABST08] "Erlang C Queuing Model" Abstract Micro June 2, 2011: <http://abstractmicro.com/erlang/helppages/mod-c.htm>.

[ANGU01] Angus, Ian, "An Introduction to Erlang B and Erlang C," Telemanagement #187 (July-August 2001).

[EVEN11] "Resource Dimensioning Using Erlang-B and Erlang-C," Event Helix June 6, 2011: http://www.eventhelix.com/RealtimeMantra/CongestionControl/resource_dimensioning_erlang_b_c.htm>.

[SHAM09] "Using the Erlang Equation" Shamrock Software June 6, 2011: <http://www.shamrock-software.eu/erlang.htm>.

[WEST11] "What is an Erlang," Westbay Engineers Limited June 3, 2011: <http://www.erlang.com/whatis.html>.

Why Models Fail—A Case Study:
A Workload Analysis Model

Tom Wilson

What does it mean for a model to fail? It means that the model failed to provide the insight that it was meant to provide. A model starts with an objective, has a design and implementation, and then is put to use. This case study examines a workload analysis model, from which a performance test model is derived, and discusses why the model failed.

1 Introduction

Models come in various forms, but their common purpose is to increase our understanding of the things being modeled. Models give us the ability to be proactive concerning problems. Models are not necessary since we can move forward in ignorance and react to problems as they are encountered. As silly as this perspective sounds, it is too common. However, what may be worse than doing nothing is creating an ineffective model.

So, what does it mean for a model to fail? It means that the model failed to provide the insight that it was meant to provide. This probably means that there was something wrong with the design, implementation, or use of the model (we will assume that at least we got the objective right). A model's design incorporates its goal or intent. This is sometimes where the design will go wrong. The implementation concerns the details of how the model accomplishes its goal. Errors in the model can render it useless. A model's use includes the data put into the model. Bad inputs result in bad outputs. This is common point of failure for a complex model.

This paper will describe a real model and then discuss where it fell short of its objective. The sections containing the model's details may be skimmed (Sections 2 and 5) or skipped (Sections 3 and 4) and later referenced after reading the conclusions (Section 6).

2 Model Objective

An existing proprietary transaction system supports logistics and maintenance of military equipment. A new system is being designed to provide more functionality and storage and serve a larger user base. A *Service Level Agreement* (SLA) will be used during operations to determine payments to the contractor by the customer. The existing system also has an SLA governing operations, although it differs from the new system's SLA in a few unimportant aspects.

A performance test model will be used to evaluate the new system against the SLA since the SLA is the only guidance for developing the system. The performance testing will help management assess the performance risks associated with the new system before it enters into production. The performance test model requires a workload upon which to base its assessment. So, it is important that the performance test model have an accurate workload. A performance test model should account for:

- the frequency of the functionality in the workload
- the timing of the workload
- the data being operated on by the workload

The frequency of the functionality specifies which functionality is executed and how often. Note that not all functionality needs to be executed in a performance test. The timing of the workload directly impacts the load on the system. In an interactive system, think times separate the activities performed by the users. Think times dictate throughput and are not accounted for in this workload analysis. [Wil10a] describes think times and other user analyses.

Determining the data being operated on is a complex analysis. That analysis is influenced by roles that the user can take on as well as privileges that he has. Some users have access to more data than other users, but that does not mean that such users access all of the data that they can. This aspect of the workload model was not addressed by the WAM since there was no simple way to mine such data from the existing system. [Wil11a] describes some of the issues concerning workload data.

Functionality is described by means of a *Business Process* (BP). A BP details the actions that the user takes when using the system. Here, a BP is synonymous with an engineering use case. A BP can have many paths of execution based on choices and/or parameter settings, such as permissions. The *Workload Analysis Model* (WAM) defines the BP workload for the performance test model. What the WAM needs to produce is a list of frequencies for the BPs.

3 Model Design

Conceptually, the WAM design is fairly simple. The new BPs are divided into groups of new and existing functionality. Existing functionality is determined by an association with a BP in the existing system (termed "old BPs"). Frequencies for old BPs are determined by analysis of production data. Frequencies for new BPs with new functionality are estimated based on comparison with new BPs with existing functionality. The two groups are then combined to produce the workload specification. Figure 1 illustrates the WAM.

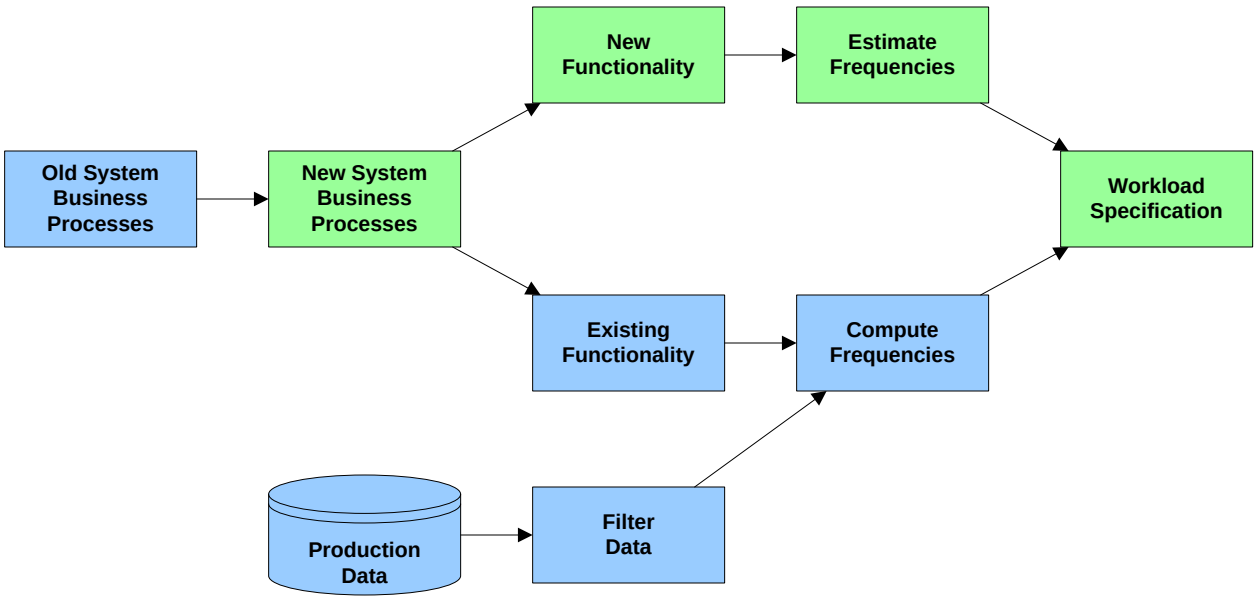


Figure 1: This figure illustrates how the WAM is constructed. The green boxes indicate tasks involving new functionality, while blue boxes indicate tasks involving existing functionality.

In practice, several of these steps are difficult. We want BP frequencies for the new system but only have transaction data from the existing system. Transactions are associated with a *resource*[1] within the software. This association turns out to be many-to-many. The resources are associated with *screens*. This association is also many-to-many. Finally, screens are associated with the BPs; again, this is a many-to-many association. Figure 2 illustrates the mapping concept. This mapping allows us to produce frequencies for the new system BPs as long as it has corresponding old system BPs.

Since the old system is complete, one would think that all of the desired information would be available. Unfortunately, this is not true. Most of the relationships are not documented and were derived by design and development *Subject Matter Experts* (SMEs).

4 Model Implementation

The WAM is implemented as an Excel spreadsheet, which is a collection of worksheets. The implementation is presented by describing the worksheets. One worksheet contains the few user-specified parameters that there are to the WAM. One specifies the number of users in the performance model. This parameter determines the number of each script that should

[1] Unfortunately, I cannot find an adequate description for this term.

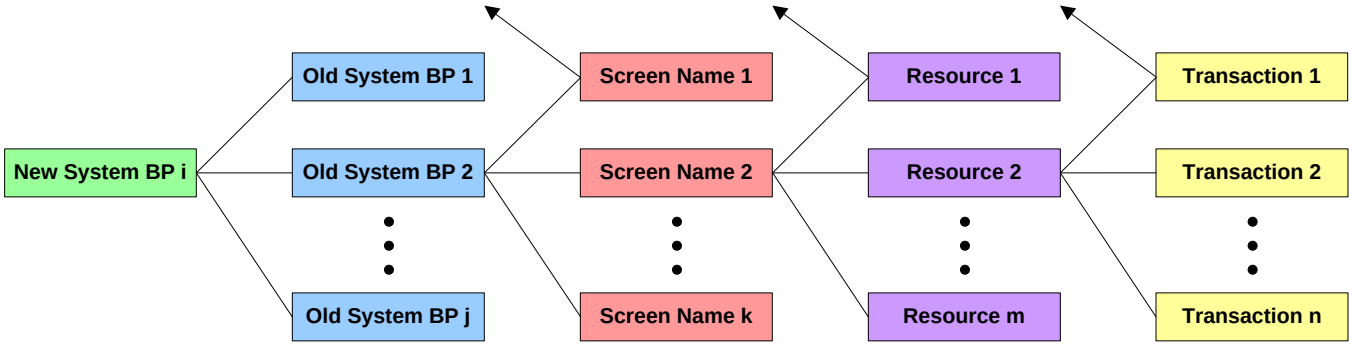


Figure 2: This figure illustrates the difficult task of mapping transactions to the business processes of each system.

be run. Another parameter determines the percentage of old functionality in the new system. This is used on the BP summary worksheet (described in Section 4.4). The remaining worksheets are described in the subsequent subsections.

4.1 Production Data Worksheet

The production data worksheet contains pairs of resources and transactions and their counts obtained by mining production data. A filtering process is applied to the data so as to eliminate transactions that do not contribute to the computation of BP frequencies. Figure 3 shows the filtering process.

Figure 3: This figure illustrates the transaction filtering process.

The first filtering operation removes all data that do not fall within the interval of interest. The interval of interest is defined as the time periods where high user-counts (called "peak-transaction-hour data")exist. We will not bother detailing the periods any further because other analysis shows that the frequency of functionality is roughly the same across all time periods (refer to [Wil10b]). Nonetheless, transactions outside the high-user-count time periods were discarded.

Two filtering operations are applied to the peak transaction hour data in order to eliminate some anomalies that inflate the activity frequencies. The first concerns overlapping or nearly overlapping transactions. If an activity outputs more than one transaction, we do not want to count all the transactions toward the frequency of the activity. There are other times that multiple transactions can be output. In the case of a slow response, the user may submit the transaction again. This is not normal behavior and should not influence the frequency of the activity. There can also be occasions where two transactions do not overlap, but are so close together that they are not likely to be different actions by the user. A one-second threshold was used to define transactions that are nearly overlapping.

For the remaining data, other transactions are removed if they are considered not to be indicative of the functionality. Examples include navigational transactions and error-related transactions. In the former case, a user may move from one screen to another. In the production data, the transaction generated is associated with the screen being left. This is not a true usage of the screen's functionality. In the latter case, transactions can be generated that give error messages back to the user. This is not representative of what the user is trying to accomplish.

A list of transactions is the output of the mining and filtering process. Over 1 million transactions remained after the filtering process. Associated with each transaction is a resource. The production data worksheet has pairs of resources and transactions with associated counts.

4.2 Resource Frequency Worksheet

The resource frequency worksheet allows a resource remapping to occur before the final frequency is computed from the production data. In a small number of cases, a resource and its group of transactions were not indicative of the activity on a screen. A mapping step allows a resource to be given an alternate name and assigned a subset of its transactions. Only a small number of resources required this mapping. All others were mapped to the same resource name. Table 1 shows an excerpt of the worksheet.

Table 1: Resource/Transaction Counts and Frequencies (Excerpt)

Resource				
Alt.	**Orig.**	**Transactions**	**Count**	**Freq.**
2	2	105	873	0.08%
3	3	105, 110	54,023	4.88%
4	4	105	331	0.03%
6	6	105	10	0.00%
7	7	3	499	0.05%
8	8	5, 92, 94-96, 105	1,059	0.10%
9a	9	132	0	0.00%
9b	9	142	0	0.00%
9c	9	105, 120, 140	1,402	0.13%

Resource 9 has 5 transactions associated with it. Two of these transactions indicated different functionality than the other three. In this particular case, the count for each of those transactions is 0. This is because the transactions occurred in production, but during time periods which were filtered out. So, why bother splitting the resource? Because the transactions occurred with other resources which were also split. So, some transactions caused resources to be split regardless of the counts for the transactions.

The list of alternate resources has a list of transactions associated with it. From these transactions, a frequency can be computed for the resource. These resources and frequencies are then used on the BP mapping worksheet (discussed in Section 4.3). There were originally 118 named resources and 165 named transactions. After splitting resources, there were 131.

4.3 BP Mapping Worksheet

The BP mapping worksheet consists of a list of new system BPs with zero or more old system BPs mapped to it. Each old system BP has a list of zero or more screen names associated with it. The reason why there could be zero screen names is because these lists were reverse engineered from the production data. The complete list of old BPs was created from engineering documentation; the lists of screen names were derived from the reverse mappings of transactions and resources (which occurred during operations) to screens. So, if a screen was never visited, it does not appear.

Table 2 shows an excerpt of the worksheet that captures these mappings. Because of the many-to-many mappings, a resource, a screen, or old BP can occur several times. Examples are resource 49, screen 45, and BP 06.15. This makes numerous rows in the worksheet. The resource frequencies, which are computed on the resource frequency worksheet, and are evenly distributed across all occurrences (introducing error since the distribution is probably not uniform). For example, resource 49 occurs two times in the actual spreadsheet. Its frequency of 2.32% is distributed evenly across the associated screens. The frequency of any new BP is simply the sum of all of the relevant rows.

Table 2: Business Process Frequencies Mapping (Excerpt)

New System		Old System			
BP	**Freq.**	**BP**	**Screen**	**Res.**	**Freq.**
001.1.a	3.00%	06.01	45	49	1.16%
		06.02	45	49	1.16%
		06.07	45	-	0.00%
		06.15	46	50	0.33%
		06.16	46	50	0.33%
			74	88	0.02%
001.1.b	0.78%	06.03	07	08	0.03%
		06.04	07	08	0.03%
		06.05	07	08	0.03%
			68	85	0.01%
		06.08	07	-	0.00%
		06.15	46	50	0.33%
		06.16	46	50	0.33%
			74	88	0.02%

4.4 BP Summary Worksheet

The BP summary worksheet lists all BPs and computes their final frequencies. Table 3 shows an excerpt of the worksheet. Each BP is defined as *new*, *old*, or *both* (only one BP is marked as *both*). For those marked as *old*, frequencies come from the BP mapping worksheet. New frequencies are estimated and are manually entered on this worksheet.

Table 3: Final New System BP Frequencies (Excerpt)

BP	Func. Type	Comb. Freq.	New Freq.	Old Freq.
001.1.a	Old	1.20%	0.00%	3.00%
001.1.b	Old	0.31%	0.00%	0.78%
001.1.c	Old	0.00%	0.00%	0.00%
001.1.d	Old	0.32%	0.00%	0.79%
001.1.e	New	0.06%	0.10%	0.00%
001.1.f	Old	0.27%	0.00%	0.68%
001.2.a	New	0.94%	1.56%	0.00%
001.2.b	Old	2.78%	0.00%	6.95%
001.2.c	New	0.44%	0.73%	0.00%
001.2.d	Old	0.05%	0.00%	0.11%
001.2.e	New	0.44%	0.73%	0.00%

A manual process was used to estimate the frequencies of the new functionality. First, all of the BPs with existing functionality were assigned to groups with the following frequency ranges: 0%, 0-1%, 1-5%, 5-10%, 10-15%, and $\geq$15%. Then a SME that was knowledgeable in both systems assigned BPs with new functionality to the groups based upon similarity to the BPs already present in the range.

Initially, the frequency for each BP with new functionality was assigned the midpoint value of its range (no BP was assigned to the "$\geq$15%" range). The resulting values were then normalized. These final values are manually entered on this worksheet.

Finally, the new and existing functionality are combined by scaling their associated frequencies. The same SME provided a guess as to the frequencies of new and old functionality. That guess was 60% and 40%. So, the combined frequencies are the result of scaling the new and old frequencies by the appropriate value.

5 Model Usage

Using the model requires it to be populated with data. Most of the data came from mining production data. Relationships between transaction, screens, and BPs came from source code and SME knowledge. These take the form of data in the spreadsheet. Other data came from SME estimates. Finally, the number of users is entered, and the spreadsheet computes the number of each BP to have in the workload.

When scripts were being recorded, it was realized that many BPs (e.g., searching for equipment) were necessary to execute other BPs. The WAM was computing the frequency of BPs but not the frequency of the scripts (the fact that scripts were named after BPs was a point of confusion).

In an attempt to correct this, a new worksheet was added that captured what BPs occurred in each script so that the frequencies of the scripts could be adjusted to improve the frequencies of the BPs. The population of the worksheet was not supported due to schedule pressures and the WAM's BP frequencies were used for the script frequencies. The result was an inaccurate workload.

6 Conclusions

The problem with the WAM was that it was outputting BP frequencies rather than script frequencies. In some cases, a script and BP were equivalent. However, in many cases, a script contained two or more BPs. Some BPs occurred in several scripts. In the end, frequencies for the BPs in the collection of scripts did not match the WAM frequencies.

The WAM could have been less complicated. First, some concept of scripts needed to be defined in the old system. This would mean identifying sequences of BPs that would make up those scripts. Such scripts would need a counterpart in the new system. Then frequencies for the old system scripts would be derived from transaction information. Additional

scripts would be defined for new functionality with associated frequencies. The result would be a more accurate workload. The biggest challenge would be computing the frequencies for the old system scripts based upon sequences of BPs.

Also, not all functionality needed to be considered for a performance test. This would not only remove some of the data that was mined, but also reduce the number of scripts that had to be developed and maintained (because the software was still be developed).

This modeled failed because of misalignment of the types of workload. Transaction workload was gathered from the existing system, yet BPs were modeled in the performance test. The model also attempted to cover too much functionality and had too low a level of detail.

7 Series Summary

This series provided examples of why models can fail to achieve their goals. In [Wil11c], the model lacked data. No input means no output. No output means that the model has no use in spite of its detail and potential. The source of most of the needed data was the customer. Gathering these data was not a priority in the project.

In [Wil11b], lack of data was also an issue. However, in this case data were needed from both the customer and the contractor. For the contractor, most of the missing data was related to incompleteness of the design or the absence of tests to gather measurements. Again, gathering these data was not a priority in the project.

In this paper, the design was flawed. Most aspects of the model's design and implementation are correct, but the misalignment of the types of workload results in bad data in the model. Unfortunately, no analysis was performed to compare the performance test's workload with the actual production workload after the system was deployed.

Finally, we will say that we should probably not judge the models using a pass-fail assessment. No model is perfect, so we must allow some amount of error. However, even that perspective is very subjective. In all cases, the various phases of the model development allowed things to be learned about the system under development. But, if that alone were the goal, the effort could be better focused. Nonetheless, the models could have been much more useful.

References

[Wil10a] Tom Wilson. "Data Mining User Behavior". *CMG MeasureIT*, September 2010.

[Wil10b] Tom Wilson. "Workload Correlation and Visualization". *Proceedings of the CMG 2010 International Conference*, December 2010. Reprinted in *CMG MeasureIT*, Issue 5, 2011.

[Wil11a] Tom Wilson. "Developing Toward an SLA: Understanding Data Complexity". *CMG MeasureIT*, Issue 3, 2011.

[Wil11b] Tom Wilson. "Why Models Fail–A Case Study: A Multi-year OLTP and OLAP Database Storage Capacity Model". *Journal of Computer Resource Management*, Issue 129:13–19, 2011.

[Wil11c] Tom Wilson. "Why Models Fail–A Case Study: A Transaction Synchronization Model for Computing Removable Media Capacity". *Journal of Computer Resource Management*, Issue 128:18–29, Winter 2011.

An Effective Implementation of CMMI for Performance Testing Projects – a Case Study

Nidhi Tiwari, Infosys Technologies Ltd., nidhi_tiwari@infosys.com
Veena Rajendiran , veena.rajendiran@gmail.com

Today performance testing is well recognized, widely practiced and sufficiently equipped with tools. However, little emphasis is given to process implementation, tracking and improvement of performance testing projects, resulting in exponentially high Cost of Quality (COQ). Here we share our experience of implementing CMMI for performance testing projects to control their COQ. Subsequent benefits obtained by organizations are also included.

Introduction

Today performance of software systems in terms of throughput, responsiveness etc is becoming critical for various businesses. Its awareness has in turn boosted the performance testing technology [1]. Consequently, a plethora of integrated suites of products like Mercury's LoadRunner, Radview WebLoad etc are available to verify the performance of business-critical applications. However, it is observed that performance testing projects run over schedule and budget, with much chaos and ambiguity. A key reason for this is a lack of identification and conformation to well-defined processes for performance testing.

Ad-hoc execution of projects results in schedule and budget slippages. Their people centric execution makes their success dependent on individual's skill level and experience. Defining a work process for performance testing would allow streamlining the activities and make them person independent to some extent. Further implementation of the Capability Maturity Model Integration (CMMI) framework would instill continuous process improvement and help sustain the quality of products i.e. system performance [2].

In this paper we use a case study to highlight some of the practical issues experienced during performance testing projects in the absence of any formal process. We illustrate a process and CMMI framework implementation for this organization and show the COQ improvement through various levels.

The paper is structured as follows: Section 1 sets context of the case study used in this paper, section 2 provides a brief overview of CMMI framework, section 3 describes implementation of CMMI for performance testing projects, and section 4 lists the benefits achieved by using CMMI and section 5 presents summary and conclusion for the paper.

1. Case Study

A manufacturing client was facing numerous issues in performance testing projects such as frequent schedule slippages, errors slippages to production etc. Due to these issues, their downtimes and rework costs of code and testing were increasing, so they wanted to minimize performance related issues and risks in production while meeting tight delivery timelines. The client engaged us to provide high quality and efficient performance testing.

To understand the client performance testing methodology, brain storming sessions were done with the client team, available documents / reports were studied and joint tests were conducted. Subsequently, the following root causes were identified for their aforementioned problems:

- Lack of common definition and/or understanding of performance testing terminologies in the organization.
- Lack of standard process, templates and checklists available for performance testing.
- Unavailability of estimation models for performance testing
- Lack of a knowledge sharing process in place.
- Lack of a validity process for COTS acquisition
- Dependency of performance tests on individual's experiences.

After analysis and discussion, it was decided that the client's ad-hoc performance testing mechanisms need to be streamlined using a comprehensive and progressing framework. For doing so a work group of subject matter experts (SMEs) was formalized consisting of people from the performance engineering research group, performance testing practitioners group and process consultants. This group decided on using the CMMI framework based on dynamics involved in performance testing. Here we describe the progress of client processes through various CMMI maturity levels.

2. CMMI overview

CMM Integration (CMMI) model provides a set of guidelines for evaluating and improving an organization's

software development processes [3]. As it has successfully helped a large number of IT organizations in streamlining their software development activities, CMMI guidelines were adopted to improve the performance testing process.

As shown in Figure 1, the CMMI® model is a method for organizing evolutionary steps into five levels of maturity that lay successive foundations to support process improvement. Processes at level 1 are very unpredictable and have great variability, while those at level 5 are highly predictable and continuous.

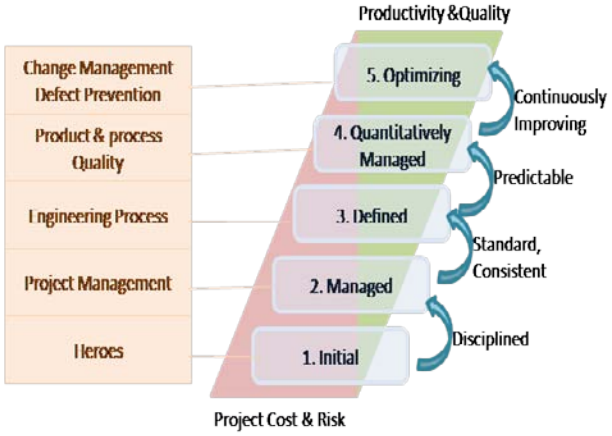


Figure 1: The SEI CMMI framework

As CMMI is a non-prescriptive model, we have extended it suitably for performance testing projects. We describe our experience of mapping CMMI practices to performance testing.

3. CMMI implementation for Performance Testing projects

For performance testing projects following Key Process Areas (KPAs) at various levels were identified for CMMI implementation [4]:

Figure 2: KPAs for CMMI model

4.1. Level 1

Initially the cost and schedule were a high priority for the project manager. So the performance testing team's job was completely directed to report the test results quickly. There was no clarity about objectives, success criteria, process etc for test execution. Test scripts were recorded and executed in an ad-hoc manner as requests came in from the client for different applications/transactions. There was a lot of dependency on COTS (Commercial Off-Shelf Tools, i.e. third party tools like Webload, LoadRunner etc) for reporting, without any focus on building expertise in the field. These COTS were often used without much diligent background work, based on their cost and availability.

Though cost was given highest priority, no measures other than these quick fixes were adopted. Estimation was not based on any formal process. Project activities did not include any specific performance measurement and tracking related efforts, quality attributes were getting ignored for cost and schedule. Subsequently high appraisal, prevention and failure costs were occurring which contributed to increasing COQ for project. The COQ was touching 44% as shown in COQ graph in figure 5, which ideally must not be higher than 20%. To control this as a first step CMMI level 2 was introduced.

4.2. Level 2

CMMI level 2 maturity is about inculcating discipline in project execution. This level initiates the shift from individual dependence to leadership of a manager. For realizing level 2 KPAs a performance testing process was required. Multiple rounds of brainstorming sessions of the work group along with client managers were done to define the performance testing process.

Performance Testing Process

The Performance Testing Life Cycle (PTLC) process was defined based on a classic waterfall model to guide, monitor and control performance testing projects [5]. It consisted of multiple phases with detailed activities, entry/exit criteria, templates and guidelines to streamline requirements management, measurement and analysis tasks for performance testing. The PTLC process is described in figure 3.

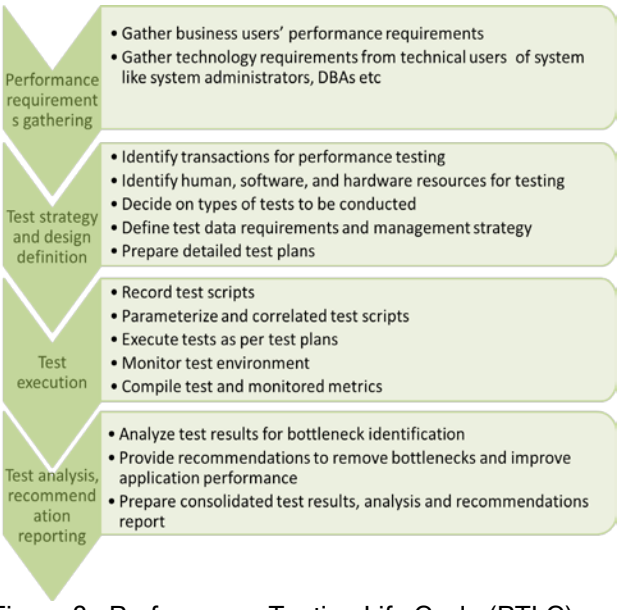


Figure 3: Performance Testing Life Cycle (PTLC)

A project management plan was prepared by the project manager based on the performance testing process and group reviewed by the work group. A PTLC phase wide effort estimation model was prepared based on the experience of the manager as similar project data was initially not available. A detailed project execution schedule was prepared and tracked by the project manager. These steps enforced a shift in project management priorities. Emphasis on size increased as compared to level 1 resulting in usage of better estimation techniques, which led to better planning and tracking activities.

For ensuring performance tests quality and tracking met stated requirements, relevant performance metrics were monitored and reported from the system performance tests. Some of the metrics monitored were:

1. Utilization (CPU, Disk, Memory, Network)
2. Throughput
3. Response Time
4. User Load

For instance the reporting of response time versus user load graph helped in confirming the load level till which the SLA requirements were made. A sample graph of monitored response time versus the user load for various transactions from a COTS tool is shown below in figure 4.

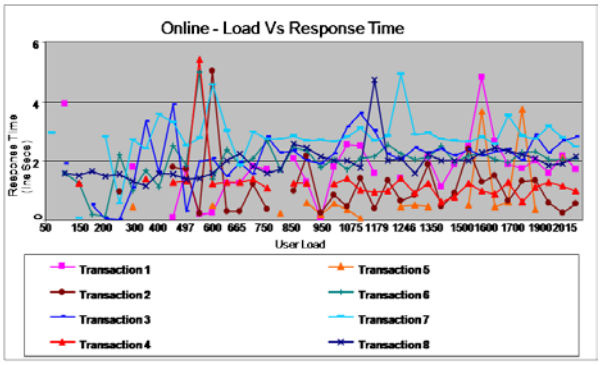


Figure 4: Response Time vs User Load graph

The process helped to clearly define purpose, activities and outputs expected from COTs which made COTs selection more streamlined. Cost of COTs tailoring like parameters initialized, complexity of script writing, security/access requirements, were also factored in for their acquisition.

At this level along with test metrics, a process metrics culture was also inculcated. Metrics were defined to measure the operational goals, quality of deliverables and productivity of the team. Some of the metrics that excited upper management are listed below:

- Schedule Adherence
- Effort Variation
- Cost Of Quality

Quality check points were defined at the end of each project phase to monitor the project performance. Defect prevention activities based on the Pareto principle were started and their effectiveness was monitored by means of sub process goals such as:

- Defect Injection Rate in test data, scripts, scenarios etc.
- Review Effectiveness of test plans, scripts etc.

At this juncture there was visibility into the progress of achieving product and process goals as multiple metrics were captured. After one full cycle, some process capability baseline values were available for other projects in the client pipeline. For the next cycle of projects, goals were laid down based on this data. The quality and predictability were still not considered to be crucial for project success.

4.3. Level 3

CMMI level 3 emphasizes on standardization of processes across the organization. To achieve it, the performance testing process and standards were institutionalized across client's testing teams. Training sessions were conducted to evangelize and enable people org wide with process implementation. This helped in effective adoption of processes by testing teams. In conjunction with different client project managers, working group established guidelines for the extent to which customization to the standard process were allowed for particular applications/projects.

Further emphasis on process engineering activities was drawn by making the related verification and validation activities an essential part of the process. The following verification activities were made mandatory: verifying test data for script parameterization, verification of test scenarios against the test strategy for evaluating the performance requirements correctly, checking that the load generated is as decided in the strategy, and the test report was reviewed for the monitored test results and graphs and so on. Compulsory validation activities included: validating the test environment by executing a sample test script using the load testing tool, executing the smoke tests for validating the test scripts etc.

The performance test results were analyzed to find the performance gaps and suggest the best technical solutions. Based on cost, risks, effectiveness and time analysis, best solutions were agreed upon for implementation with development team. Knowledge sharing activities like preparation of a book of knowledge (BOK) were kick started for reuse.

At this level, meeting customer requirements of quality and speedy delivery took precedence over cost and size for the project manager.

4.4. Level 4

At maturity level 4 the projects become more predictable as previous projects execution metrics provide a baseline and processes are in place to track them for adherence. In this case, use of standard reporting templates for capturing performance metrics during a few testing cycles provided an opportunity to provide performance's predictions in subsequent cycles. For doing so some additional performance engineering activities including scalability analysis, performance modeling and what-if analysis were conducted [6]. CPU utilization versus user load graph analysis was used to find if the application can scale up with the increasing user load. If an application is found to be scalable then its more predictable as throwing more hardware would allow it handle increase in load. Additionally performance models were created using test results and what-if analysis was done to predict the application performance for future user loads. SLA's were monitored and stored for future reference, which helped to individually predict the response time of the various applications under test and also to predict the response time for any new application for similar kinds of application tests in pipeline.

The project management focus was shifted to schedule and quality. From a process perspective, metrics data from individual quality check points defined at regular frequencies from various performance testing projects was used to build upon predictability. Using this data process capability baseline was established and referred for projects spanning across the organization.

Statistical process control (SPC) techniques [7] were used to ensure that the process is within the control limits and variation is minimal at each quality check point. The SPC graph in figure 4 for effort deviation shows that effort deviation for initial applications/cycles was outside the control limit and later brought within the control limits. Tracking of this metric made effort deviation more predictable for next phases.

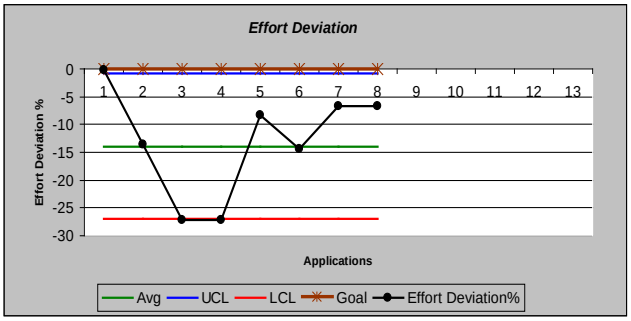


Figure 4: SPC for effort deviation

Process improvements were measured at every milestone. Analyzed optimization strategies were applied to performance testing projects across the organization. As a result costs were observed to be minimized across wide segments of the organization.

4.5. Level 5

In this optimizing phase, innovation and defect prevention (DP) activities were continuously applied and improved. Some of the verifiable benefits seen include improved performance testing knowledge and skill level of the resources, which resulted in reduced time to market. The productivity and quality improvement strategies clearly lead to reduced cost of testing. There was more emphasis on quality and schedule with cost and size being given the back seat.

At this level there was focus on improving processes for dealing with business changes like technology changes, customer positioning, time to market, test execution methods etc. Decision analysis and resolution techniques were used to identify and implement new strategies. Process work flow was continuously updated and kept current. Implementation aspects of the key strategies and risk management planning were carried out as part of the detailed project plan.

Test management monitored and contributed towards improving following operational goals: COQ, productivity, quality, schedule adherence, effort variation and defect removal efficiency. These goals monitored at project level were mapped to organizational strategies due to which cross disciplinary cooperation and top management involvement in decision making became prominent.

At this level, the client realized the fact that "Prevention is always better than cure". So an upward trend was seen in prevention cost for quality, which automatically led to reduction in appraisal and failure costs. The end result was an overall reduction in cost of quality to 4%. Overall Cost of Quality [8] and its components observed at each level for a project are shown below.

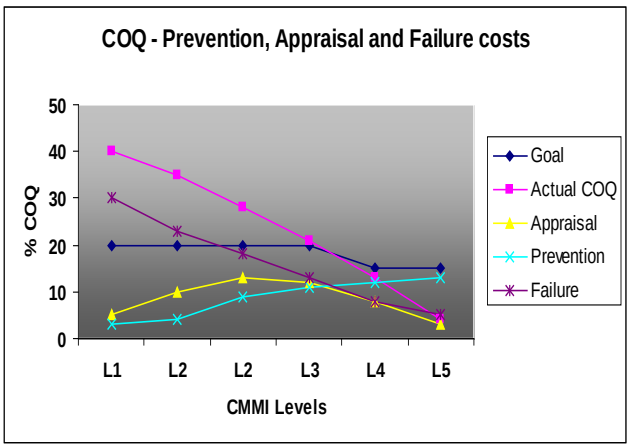


Figure 5: COQ, appraisal, prevention and failure cost

The trend here shows that as the project matured usage of appropriate preventive measure like defect prevention, training and review activities brought down the failure cost where by improving the COQ.

4. Benefits from CMMI implementation

Some of the important benefits achieved by CMMI guidelines implementation in this project were:

- Reduced testing cost – Implementation and streamlining of performance testing processes helped increase productivity and reduced the failure cost. Thus an overall reduction in testing cost was achieved, which also increased profitability.
- Customer delight - Addition of performance engineering activities helped reducing post production defects. Along with proper process this helped improved quality and reliability which made customers happy. This also improved repeat business.
- Reduced cycle time – Continuous performance testing process improvements led to reduced cycle times for future releases. This positively impacted time to market, delivery time and increased bonuses for early delivery.
- Elevated employees' motivation – Clarity of goals, activities to be performed and setup of tracking mechanisms increased employee morale and service provider's confidence. This decreased employee turnover and increased employee retention, reduced retraining costs and improved overall competitive advantage.

5. Summary and Conclusion

In absence of processes, there is neither any guarantee of repeating the project success nor any basis for improving the productivity and/or quality. This paper summarized our experiences of using the CMMI framework for taking performance testing projects from initial ad-hoc executions to continuously optimizing levels. The benefits obtained at each maturity level in the presented case study accentuate the importance of model usage. So it is suggested to implement performance testing processes evolving using CMMI maturity model for delivering cost, schedule, size and quality effective projects and products.

6. References

[1] http://en.wikipedia.org/wiki/Software_performance_testing
[2] http://www.sei.cmu.edu/cmmi/general/
[3] http://www.sei.cmu.edu/solutions/softwaredev/
[4] "Capability Maturity Model® Integration (CMMISM), Version 1.1", CMMI Product Team, March 2002
[5] J.D. Meier, Carlos Farre, Prashant Bansode, Scott Barber, "How To: Manage the Performance Test Cycle in a Regulated (CMMI) Environment", CodePlex website.
[6] Pete Utton, Gino Martin, "Further Experiences with Software Performance Modelling", WOSP98, Santa Fe, NM.
[7] http://csqa.info/statistical_process_control_spc
[8] http://thequalityportal.com/q_CoQ.htm

Not Your Father's or Grandfather's Mainframe Any More

David J. Lytle, Brocade Communications
dlytle@brocade.com

So, the truth is, the old mainframe way of doing things that I grew up with back in the 1960's and 1970's is gone. The mainframe back then was a monolithic approach to computing. Within a few decades it became a dinosaur – a large and efficient dinosaur – but a dinosaur nonetheless. It became slow moving in a faster and faster paced world. Users wanted and needed new applications and application updates at a rapid pace while the mainframe programming staffs seemed to move at glacial speed. As a direct result of this disconnect between expanding user's needs and the inability of the centralized computer system to respond appropriately, the users looked for new ways to fulfill their needs. And off they went to decentralized processing and direct control of their computing resources. Over time, the terminal result was that computer platform competition was reducing the mainframe's feeding ground to the point that by the latter part of the twentieth century it was headed for extinction. But then, rather than succumbing to obsolescence it reinvented itself and evolved into the magnificent machine that it has become today. In the end the twentieth century "computer wars" made the mainframe supremely powerful, more agile and incredibly more capable.

That evolution from the "my way or the highway" kind of early day mainframe computer processing allowed the mainframe to morph into a platform for the multitudes where legacy mainframe applications run happily alongside more distributed workloads such as Linux and Unix – and all on the same re-centralized and easier to manage computer platform. IBM has once again made the mainframe relevant across the spectrum of computer users that inhabit the globe today. In fact, IBM often refers to its "mainframes" as large servers and emphasizes that they can be used to serve distributed users and smaller servers in a computing network.

Cloud computing is a great example. In recent years the notion of "Cloud Computing" has emerged and many customers have a desire to move toward a cloud based structure. This alone has re-energized interest in the mainframe since it is the only platform really capable of providing private cloud computing services.

For the distributed world, the primary stumbling block for cloud computing has been the hypervisor. All of the entities in the cloud must coordinate their capabilities, availability and resources which require a common hypervisor to collect and communicate this information before acting on it. That has proven to be more than difficult when using a distributed server farm.

But today's goals and requirements for cloud computing can be met with the industry leading reliability, serviceability, and availability (RAS) features that are built into the mainframe along with its additional ability to distribute resources as the demand ebbs and flows. One of the embedded mainframe services, the Resource Manager, already provides the necessary coordination functions and the system resources are managed in a homogeneous manner. Consequently, the System z® mainframe is really the closest implementation of the heralded cloud computing complex available today. There is not a single distributed solution can match the cloud capabilities of the mainframe – and System z® can do it right now!

Beyond the high level notion of cloud computing, high-performance software solutions have also evolved that can leverage the performance, security and availability of the mainframe in this

world of internet time, Web interfaces and Service-Oriented Architectures (SOA). An example of mainframe agility for distributed processing workloads would be the IBM WebSphere Portal for System z®. Today's mainframes are designed to excel at business computing, which typically involves hundreds or thousands of transactions per second.

The mainframe has always had it strengths: Its robust RAS that provides for zero or almost zero downtime over a year or many years; scalability which is the ability of the hardware, software, or a system to continue to function well as it is changed in size or volume; security which provides protection against unauthorized access, transfer, modification, or destruction, whether accidental or intentional; and virtualization which builds on physical partitioning and offers the ability to simulate availability of hardware – CPU, memory and I/O – and operating system (OS) resources.

Of course, a distributed system has its strengths as well: Speed of deployment; inherent distribution; decent (or "good enough") reliability; perceived cost savings and incremental scalability for growth. Overall, I believe that the traditional benefit of distributed computing has been that it enables a customer to optimize their computing resources for both responsiveness and economy. But neither of these technologies work in a vacuum so some really good minds have looked at these various technologies and incorporated bits and pieces that would help their own systems work better.

As some of the mainframe technologies trickle down to distributed systems, those systems are getting better at hosting mainframe-class applications and they are slowly beginning to achieve some of the traditional mainframe benefits like high availability, scalability on demand and improved overall utilization. But, at the same time, the mainframe is becoming more like distributed systems with an ability to locally execute UNIX and Linux applications and also to link with IBM blade servers and manage AIX, Linux and Windows applications using the unique mainframe-based Resource Manager application. So in our world today, all of this makes it perplexing for a customer to decide which is the best platform to meet their unique computing needs.

When customers are trying to understand the difference between distributed platforms and a mainframe platform one of the significant differences is how their I/O subsystems work. I am sure that customers sometimes puzzle over the benefits and costs of running DB2, WebSphere, Unix and Linux on the mainframe versus running them on an open systems platform. I also suspect that they often calculate total cost of ownership without understanding both the benefits of collapsing the different tiers into one, much more easily managed system as well as the cost and performance benefits that can accrue by using a mainframe I/O subsystem. So I think that it is important for a user to understand how significantly different I/O is accomplished on distributed processor systems compared to the mainframe.

Mainframe I/O is Main Stream Functionality and It Is Very Robust:

On a mainframe, I/O is arguably just as important a task to be performed for applications as the computing that is done for those applications. In order to make certain that I/O is treated as a mainstream task the mainframe I/O subsystem has several unique and very powerful design features that create a major differentiator between distributed computing systems and a mainframe computer environment.

First of all, virtualization is everywhere in the mainframe and has been everywhere for decades allowing it to mature into a very stable infrastructure. For example, through the use of

virtualization, a DASD storage array can have as many as 256 Logical Control Units (LCUs) each with 256 devices so a mainframe can address up to 65,536 total volumes within just one storage array. All of the information needed to get to any LCU and volume is contained within each frame of a mainframe Fiber Connection (FICON) I/O. The mainframe does not need any special services to make this happen. What do I mean by a special service? An example of a special service for distributed processing would be Single Root I/O Virtualization (SR-IOV) that allows a PCIe device to appear to be multiple separate physical PCIe devices. In effect, this provides a form of virtualization. An example of a special service on the mainframe would be Node_Port ID Virtualization (NPIV), developed in the mid-1990s, which is an FCP SCSI I/O service that allows Linux on the Mainframe to capitalize on a similar type of channel virtualization that the mainframe has been providing for its legacy applications for 30+ years.

One of the most important differences in how I/O is carried out by the mainframe and how it is carried out by distributed processors is how server to storage connectivity is initially created. In my experience, mainframe people tend to be type A, control-oriented personalities so they have historically always desired to directly manage everything that happens on their mainframes. A mainframe systems programmer will use a tool called Hardware Configuration Definitions (HCD) to describe his mainframe environment including exactly the path(s) that every I/O will take from the CHPID out to a storage device. If it is not described in HCD then the I/O just will not happen. Performance and predictability are king in the mainframe world so mainframe technicians rely upon their own tools to create robust I/O delivery.

Distributed systems administrators, on the other hand, seem to rely more on the Plug-n-Play model and are more casual about how I/O gets accomplished. In their world it seems that ease-of-use and simplified management is king. Therefore they utilize all of the protocol stack capabilities of the Fibre Channel Protocol including the name server service to identify I/O routes and connectivity. The Systems Administrators then leave it up to the FC protocol to find the I/O path(s) that provides them with server to storage connectivity. This Plug-n-Play methodology will generally be successful but sometimes at the expense of poor performance and less robust I/O frame delivery since the I/O path connectivity is completely left up to the FC protocol.

Mainframe Systems Programmers have other tools that aid in providing robust I/O delivery. The System z® operating system has a built in capability known as “Path Group”. On the mainframe a user can group up to 8 of their physical connections between the Channel Path IDs (CHPIDs), which are the mainframe I/O ports, out to connected storage ports. It is the mainframe channel subsystem that decides which path in the path group will be used by deciding which path is least busy and which paths are operational, etc. Path Groups allow I/O to be automatically spread evenly and fairly across a number of physical channel paths without over-subscribing any given I/O path.

Mainframe Path Group functionality is not a capability that is provided for distributed processors and their data paths. Once again special services must be provided to balance I/O across multipath configurations between servers and storage (and this software is often left out of TCO calculations). Examples of this are EMC® PowerPath® Multipathing and IBM® System Storage® Multipath Subsystem Device Driver (SDD). Both are special purpose software applications installed on a distributed server to control and balance multipath I/O operations. Other vendors also have their own multipathing solutions.

Addressing, and particularly device addressing, is very robust on mainframe platforms. If a customer needs to do a great deal of I/O to online or tape files, the mainframe is a customer’s

best choice. Mainframes utilize hexadecimal addressing (base 16) and a single mainframe channel can access device addresses from x"0000" to x"FFFF" (e.g. 0000 to 65,535 in decimal).

A logical partition (LPAR) on a mainframe is allowed by the z/OS operating system to have up to 256 channels to access data. That is potentially 256 channel paths, each running at 800Gbps, and all together they can be connected to 256 devices concurrently from just one LPAR. A z9, z10, z196 or z114 mainframe can have up to 60 LPARs running concurrently. And each of them can be using up to 256 channels (often channels are shared between LPARs) to access data. Mainframes can have as many as 1,024 physical channels that can be parceled out to as many LPARs as are running on the mainframe, but no LPAR can utilize more than 256 I/O channels. While a System z® processor complex may be capable of running thousands of applications simultaneously across as many as 60 logical partitions (LPARs), since each System z® has only 256 channels (paths) that it can supply to any given LPAR, channel addresses are a precious commodity and must be used wisely. Even with this channel limitation per LPAR, mainframes can fairly easily access and make use of many thousands of the potential 65,535 addresses (data volumes) that are available to it per channel and per storage array.

Now consider a Parallel Sysplex (multiple mainframes working together) where, for very large enterprises maybe as many as 20 or 30 mainframes are participating together in this clustered kind of environment! The scale obviously ramps up until it is just incredible.

Many customers agree that mainframes provide the most robust and secure I/O connectivity available. And today's most modern mainframe channel can run at 800Gbps (an aggregate 1600MBps full duplex) which is the same speed that is possible when using HBAs on distributed servers. But even with similar performance characteristics, the difference is night and day between mainframe I/O and distributed I/O.

Fibre channel protocol-oriented, distributed, online I/O (FCP) will map Small Computer Systems Interface (SCSI) into the payload of a frame from the FC-4 protocol layer that is then sent over fiber cables to disk devices. At its core SCSI provides an agreed upon set of standards for physically connecting and transporting data between distributed server initiators (computers) and targets (peripheral devices). The target port is always responsible for making sure frames are received in order sequentially as well as making sure that all frames meet high requirements for data integrity. I/O is accomplished through I/O "exchanges". Timing of I/O in SCSI environments is rather tolerant. Disk is parceled out in Logical Units (LUNs) that can be size formatted in a variety of ways.

Fibre channel protocol-oriented, mainframe, online I/O (FC FICON) will be to "Count, Key, Data" formatted Direct Access Storage Devices (DASD) volumes. Input/Output (I/O) is accomplished through I/O "exchanges". The major difference here is that mainframe FICON will have a standards-based FC-SB2, FC-SB3 or FC-SB4 payload in the frame from the FC-4 protocol layer while FCP always has a standards-based SCSI payload which is incompatible with FICON payloads. The FICON receiving port is always responsible for making sure frames are received in order sequentially as well as making sure that all frames meet high requirements for data integrity. Timing of I/O in mainframe environments is very strict (2 second channel timer). DASD is parceled out in Volumes that can be size formatted in a variety of ways. Volumes are further sub-divided into data sets and it is data sets that are used by mainframe applications.

On the mainframe there are two modes of FICON I/O operation: Command Mode FICON (standard FICON); and Transport Mode aka High Performance FICON (zHPF). zHPF does a more effective job of building frames to meet the requirements of the I/O exchange that it is transporting than does standard FICON. This results in a dramatic increase in the number of start I/Os and MBps of data transferred for zHPF compared to Command Mode FICON. But both deliver I/O frames more successfully and robustly than any SCSI-based distributed server. A mainframe channel running zHPF has the capability to deliver as many as 92,000 start I/Os per second. And if a customer is looking at throughput as a metric, 8Gbps mainframe zHPF channels can deliver as much as 1,600MBps of throughput each. In the ultra-extreme and highly unlikely case that all of the 8Gbps mainframe channels were running full speed, then the 320 FICON Express8S channels being used by zHPF would be providing 512,000 Megabytes per second of data movement – (320 x 1600MBps = 512,000) – a phenomenal .5 Terabytes per second of data throughput rate.

From the mainframe outbound, FICON allows many different commands to be done in one I/O stream which is not the case for FCP SCSI I/O. In one operation a mainframe can execute lots of different I/O operations. And when using Command Mode FICON a channel path can disconnect pretty easily from its destination port. zHPF, however, is more strict. When using zHPF a path disconnect can only be done on the last command of a string of commands. Channel End/Device End (CE/DE) status will signal the end of a FICON I/O operation.

From the storage array outbound, it is the storage control unit that really chooses a return channel path after an I/O disconnect. This allows each storage vendor to have a different algorithm to accomplish sending I/O back to the mainframe. Although an I/O often uses the same path from CHPID to Storage and then again from Storage to CHPID, a storage adapter busy or other condition might have the storage Control Unit (CU) pick a different channel path than the original path for the I/O path reconnect.

With all of the complexity inherent in a data processing complex today, it would seem appropriate that some form of coordination take place across the common resources like CPU, I/O and Storage so that all of the applications running on computer systems benefit accordingly. Not too surprising, the distributed world is just beginning to develop common, cohesive, dedicated functions to provide this detailed level of resource coordination. Thankfully, the mainframe has had it for decades.

One of the strengths of the System z® mainframe and its z/OS operating system has been its ability to run multiple workloads (legacy and distributed) at the same time within one operating system image or across multiple images. Such workloads have different, often competing completion and resource requirements. These requirements must be balanced in order to make the best use of the resources of an installation, maintain the highest possible I/O throughput and achieve the best possible system responsiveness. The unique mainframe function that makes this possible is its dynamic workload management which is deployed via two synergistic functions – the Workload Management component of the z/OS operating system and the Unified Resource Manager (URM – sometimes called zManager).

With z/OS Workload Management (WLM), a customer defines performance goals and assigns a business importance to each goal. The customer defines the goals for legacy work in business terms, and the System z® decides how much resource, such as Channel, CPU or Storage, should be given to it to meet the goal. Workload Management will constantly monitor the system and adapt processing to meet the goals. The Unified Resource Manager, introduced with System z196, enables a customer to install, monitor, manage, optimize, diagnose, and service

resources and workloads from a single point of control while extending System z® qualities of service across the entire infrastructure including its distributed processing. It's important to recognize that the URM provides value to heterogeneous workloads running only on the Computer Electronics Complex (CEC), meaning a z/OS and z/Linux workload. The use of Linux on System z® is growing rapidly and the URM makes deploying a workload on a Linux server running on System z® much easier than ever before. And although it is beyond the scope of this paper to discuss, when the System z196 or System z114 are connected to an IBM zEnterprise BladeCenter (zBX) the URM can not only manage all of the System z® z/OS and z/Linux workloads, it can also manage Linux, AIX and Windows applications that are running on the zBX – a level of distributed processor blade center management never possible before.

All of these System z® capabilities have lead to easier but very robust storage management for the mainframe system administrators. At the same time, storage management has been a rocky road for distributed processor administrators. Some analysts have projected that a non-mainframe storage administrator should be able to manage an average of 30 terabytes of disk storage. In comparison, the typical mainframe storage administrator, using powerful tools, effectively manages well over 100 terabytes of DASD storage. Mainframe environments simply require less manpower resources for their management.

I/O Is Taken So Seriously On The Mainframe That It Is A Specialized Function:

As was already pointed out, many applications running on many LPARs may simultaneously traverse a relatively small number of I/O paths on a mainframe. This means that channel path bandwidth utilization is almost always significantly higher (i.e. more efficiently utilized) on System z® than on a typical distributed systems FCP path. The mainframe therefore must take care to feed the I/O appetite of its applications very carefully – and it does.

So another tremendous differentiator is that the mainframe, unlike its distributed server cousins, DOES NOT use its own compute processors to do application I/O!

The mainframe can make use of specialized processors that are designed to enhance performance and hold down mainframe software costs. These special processors are: the IFL or Integrated Facility for Linux which is dedicated to Linux OS processing (and optionally used under z/VM); zAAP or System z® Application Assist Processor which is currently limited to running only Java and XML processing; and zIIPs or System z® Integrated Information Processors which are dedicated to running specific workloads including DB2, XML, and IPSec.

The final specialized processor type are the I/O channel processors (System Assist Processors – SAP) which are dedicated to handling I/O. Basically, when a mainframe wants to do an I/O it writes that request to memory, then alerts the SAP that the I/O is waiting for it in memory, and then the mainframe itself moves on to other compute-oriented work. The special I/O channel processors then take care of getting the I/O processed and sent down the appropriate channel path and they do all the waiting for devices to respond to commands, the data, etc. Once an I/O is complete the SAP communicates that status back to the mainframe who can then return to processing the application that issued the I/O in the first place.

All of these specialized processors combined are why mainframes can get such a tremendous amount of compute work done. All that the mainframe engines do is "compute" work while the lengthy and time consuming I/O interactions are handled by the specialized I/O processors. This unique division of work and I/O, with each process doing what it does best, is unique to the mainframe!

What follows then is that compute-centric, transaction related workloads, such as data warehouses, all run better on the mainframe than anywhere else. As long as the workload is I/O intensive and not CPU intensive, the mainframe is hands down the best platform for a customer to utilize. On the other hand, that is also why not all workloads will run as well on a mainframe as on a distributed server. A customer must choose these platforms wisely. There are many forums that will distinguish which workloads run best on mainframes and which run best on distributed servers.

On a distributed server, when a customer does I/O, they are doing it by utilizing the main CPU on that chassis. Of course, distributed CPUs can be very speedy but it is important to note that with the advent of the System z196, the mainframe now has the fastest-in-the-industry microprocessor and clock speed. All of that notwithstanding, I/O takes time and distributed processors must therefore have their CPU wait until the required I/O is complete before more computations can occur on that chassis. Since all processors wait at the same speed, when dealing with I/O intensive applications the customer using distributed servers can end up with marginal processor utilization (often distributed servers average only 20-30% busy). These customers will also receive much less than the full value of their fast server processor particularly during I/O operations. Obviously, per the discussion above, this is not true on mainframes.

There is an outstanding white paper titled, “Why Your Organization Should Use Workload Optimized Servers“. The paper was written by Clabby Analytics (www.clabbyanalytics.com). It is available for download at http://www.workloadoptimization.com/uploads/WhyWrkloadOptimization.pdf,

According to this paper, “Clabby Analytics has obtained benchmark information on System z® (mainframes), Power Systems® (POWER-based servers), and System x® (x86-based servers) from IBM’s software group project office located in Poughkeepsie, New York. This data compares how each environment handles workloads that involve heavy I/O, heavy data-intensive processing, and light workload processing. In each case, System z®, Power Systems®, and System x® servers were asked to handle the same workload and were given identical service level requirements.”

In the paper’s example, the cost of processing an online Linux banking application that completes 22 transactions per second while processing 1 MB of I/O per transaction varied significantly when comparing Mainframes to Power Systems® and to x86 servers.

It can be extrapolated from the study that a single System z196 32-way mainframe can run 240 of these Linux workloads when using 32 Integrated Facility for Linux (IFL) features.

Compare that to a distributed systems Intel® Xeon 8® core blade which was capable of running only 10 virtual machines handling the same application at the same service level per core blade. So it would require 24 Xeon 8® core blades (and associated enclosures, networking components, and software) to handle the same application at the same service level as a mainframe.

And for the final comparison, an 8-way Power System® blade can run about 15 virtual machines running the same application at the same service level. This means that it would take 16 Power System 8® core blades (and associated enclosures, networking components, and software) to handle the same workload as a mainframe. And keep in mind that the mainframe would be

maximizing the value of its processors; would be able to utilize NPIV in a switched-FICON environment to drive high bandwidth utilization per channel path; would be would be much easier to manage; and would require less power, less cooling, and less floor space.

Let's Discuss Why Switched-FICON I/O Provides The Best Value For Data Traffic:

Brocade Communications Inc. has a whitepaper that describes the benefits of doing mainframe I/O through Fibre Channel switching devices. It can be found at the following website: http://www.brocade.com/downloads/documents/white_papers/why-ficon-wp.pdf

What I want to do here is simply provide a few of the many, many reasons why it is prudent to deploy a FICON I/O infrastructure by utilizing switching devices rather than by direct attaching mainframe channels to storage ports.

Both Storage Area Networking (SAN with FC SCSI) and FICON Fabrics (FC FC-SB2/3/4) can and should make use of FC switching devices. But with the mainframe there are several capabilities that do not apply to SAN implementations and these are the capabilities that I want to mention here.

Since the delivery of the System z9 years ago, IBM has been modifying the mainframe I/O subsystem to provide users with additional functionality. Some of this functionality can only be utilized when switched-FICON fabrics are deployed. In fact, IBM has announced a series of technology enhancements that require the use of switched FICON infrastructures. These include: NPIV support for z Linux SCSI I/O; Dynamic Channel Path Management (DCM) for FICON; and z/OS FICON Discovery and Auto-Configuration (zDAC).

Node_Port ID Virtualization (NPIV), as discussed above, is an excellent special process available for Linux on the Mainframe. NPIV allows many FCP I/O users to interleave their I/O across a single physical but virtualized channel path which minimizes the number of total channel paths. For example, if a System z® is running 300 Linux guests then maybe 20 channels can be virtualized with NPIV such that each set of 15 Linux guests makes use of one of the 20 virtualized channel paths driving each of those individual physical channels's utilization towards peak performance. NPIV is only available when using switched-FICON environments.

FICON Dynamic Channel Management (DCM) provides an ability to dynamically add or remove channel resources at the Workload Manager application's discretion. This allows Workload Manager's Goal Mode to effectively utilize mainframe channels to make sure that application's are completed on time. Use of DCM is available only in switched-FICON environments.

z/OS Discovery and Configuration (zDAC) provides a simplified and discovery-oriented method for configuring new and/or changed FICON connected DASD and tape configurations. zDAC is only available when using switched-FICON environments.

As IBM continues to introduce innovation onto the mainframe, customers will very likely see more and more I/O capabilities that are tied to the use of switched-FICON infrastructures. And the only way that customers will reap the benefits of these new functions is to deploy switched-FICON fabrics.

The Latest Generation of Mainframe Is Simply The Swiss Army Knife of Computers:

The latest System z® Business-class mainframe (z114) is single frame design about the size of a refrigerator; uses less energy than an American clothes dryer; heterogeneously runs legacy as well as UNIX and Linux distributed system applications; effectively manages legacy and distributed systems on its own CEC as well as AIX, Linux and Windows on attached zBXs; can handle about 30 Linux servers per z114 core and about 300 Linux servers in total; can deploy a new virtual Linux server in just a few minutes and provision each of its Linux servers at a cost of about US$500 per year; yet surprisingly, an entry level z114 sells at an economical cost of around US$75,000. It is also important to note that z114 introduced the PCIe I/O (Peripheral Component Interconnect Express protocols) infrastructure as a new feature to Mainframe systems (common in UNIX and other distributed systems).

As for the latest Enterprise-class mainframe (z196), well it can do everything the z114 can do and much more. The z196 is a two frame design that uses the world's fastest microprocessor which clocks in at a blazing 5.2 GHz. According to IBM, a z196 can replace up to 1,500 x86 servers while requiring an 85% smaller footprint; it is capable of executing more than 50 billion instructions per second; each of its processors uses less energy than a 40 watt light bulb so it provides up to 85% lower energy costs (when considering both power and cooling) than distributed systems; it can support up to 47 distributed servers (like Linux) on a single core and up to 1000's on a single system; and it also can make use of the new PCIe I/O infrastructure.

It is obvious to me that IBM is laser-focused on improving the mainframe's momentum for providing computing to both large and medium market segments. All of the factors mentioned above will help almost any customer achieve better systems management, faster deployment and quicker response time. I also suspect that many industry observers, who once saw the mainframe as a fading dinosaur, now must concede that this new "Big Iron" is going to stick around and that IBM is working hard to keep it relevant.

In Summary:

IBM's System z® mainframe draws on decades of innovation and collaboration with advanced customers in all segments of computing – customers who run the most complex computer operations on the planet. Executives all over the world are finding out that the System z® is simply the most powerful tool available to them to reduce cost and complexity and improve security and reliability in their enterprises. A telling point to that argument is the mainframe's upsurge in adoption, over the past several decades, for solving the world's most complex business, governmental and academic challenges around the world.

When you go trolling around the internet you can do some searches on the amount of data hosted on mainframes. Dozens of entries will proclaim that, "more than 70% of the world's business-critical data resides on mainframes." Since that data has to be processed, maybe some of the points in this paper have made it clear to you why so much of the world's business-critical data IS hosted and processed on mainframes. I hope so.

References:

[1] Governor, James, *Its the I/O Stupid: Linux on z*, The Mainframe Blog, March 2006.
[2] Madden, Ned, *The Mainframe Lives*, www.technewsworld.com, March 2008.
[3] Madden, Ned, *Big Iron Keeps on Trucking, Part 2*, www.technewsworld.com, March 2008.

[4] Ng, Dennis, *zEnterprise FICON Channels Review*, July 2011.
[5] Clabby Analytics, *Why Your Organization Should Use Workload Optimized Servers*, February 2011.
[6] IBM Corporation, *z/OS Workload Management*, http://www-03.ibm.com/systems/z/os/zos/features/wlm/
[7] Brocade Communications, Inc., *Why Switched FICON? (Switched FICON vs. Direct-Attached FICON)*, http://www.brocade.com/downloads/documents/white_papers/why-ficon-wp.pdf, September 2011
[8] Guendert, Steve Ph. D., *To Switch or Not to Switch? That's the Question!,* The Mainframe Zone, http://www.mainframezone.com/it-management/to-switch-or-not-to-switch-thats-the-question/P4, October 2011
[9] Wexler, Steve, *IBM Breathes New Life Into Mainframe With zEnterprise 196*, http://www.networkcomputing.com/servers-storage/229501123, September 2010
[10] IBM Corporation, *zEnterprise 114 (z114)*, http://www-03.ibm.com/systems/z/hardware/zenterprise/z114.html, July 2011

Processor Selection for Optimum Middleware Price / Performance

David A. Kra, Principal Architect / Account CTO, Infocrossing℠, Inc.

Many middleware products can be deployed onto many combinations of processor architecture and operating system. Finding the most cost effective combination is complicated by software pricing based on vendor core weighting factors. This paper explains how to combine core weights, core counts, and performance data to calculate and compare a "Performance Rate per Weighted Core." Results are provided for the Oracle data base server as used in published TPC-C and TPC-H benchmarks.

Introduction

Question: What platform would provide the best price / performance for your usage of a middleware product, such as Oracle's Database Server?

It may be a straightforward question, but there are complications that make this analysis not-so-easy:

> Oracle comes in different versions, such as Standard, Enterprise, and RAC.
>
> It is expensive to run benchmarks and there are contractual constraints on publishing the results.
>
> If we subtract out items which are relatively independent of platform, such as disk storage and networking, then the biggest cost item will probably be the Oracle licenses and maintenance.
>
> Oracle Database license and maintenance pricing is often negotiated to be less than the published list price. However, even at a substantial discount, the DBMS Software usually costs more than the computers it runs on.
>
> Oracle DBMS pricing is based on the quantity of "weighted cores" it will run on. Oracle places different core weighting factors on different processors depending on the architectures, speeds, implementations (chip models), the servers in which they are installed and when they were sold. For example, according to the Oracle Processor Core Factor Table, (current as of August, 2011) a SPARC family processor core may be weighted by a factor of 0.25, 0.5, or 0.75 depending on several other attributes of the chip and how it is used.

The software cost for four cores with a core weighting factor of .25 is the same as for 1 core with a weighting factor of 1. If all these cores performed the same, there would be a 4x software price

performance advantage for the system with the cores weighted 0.25, as indicated in the following table.

Table 1
Sample Oracle Core Weighting Factors

Core Weighting Factor	Ratings Advantage	Example Processor
0.25	4x	Oracle SPARC T3
0.5	2x	Intel XEON 75xx
0.75	1.33x	HP PA-RISC
1	1x	IBM POWER6

Question: So which platform should you put your Oracle Database Servers on to get the best price / performance?

> Answer: The one that can do the most work per dollar, which is to say, the platform which can do the most work per weighted processor count, because that drives the software costs. Only if comparisons come very close do we need to consider other cost items, such as hardware and operating system.

Question: What is the source of the weighted performance advantage? Is it processor speed, cache size, cache per core, hyper-threading, or the software vendor's choice of weighting?

> Answer: That question is asking how and why, which may be interesting, but is not really pertinent to the issue of determining the best performance-for-the- price platform based on experience. However, it does become pertinent when we need to select among new products with which we do not have experience.

The majority of this paper uses:

- Oracle's Database Management System (DBMS) as the example middleware product
- The Transaction Processing Council's TPC-C and TPC-H applications as the workload
- Published TPC-C and TPC-H benchmark results for performance data

The method used here could be applied to your benchmarks of your own workload. It also could be applied to any other middleware product from any vendor who charges based on weighted cores. If your DBMS is Oracle and your usage is similar to the TPC-C or TPC-H usage pattern, then these results may be directly applicable to your situation.

Methodology

To circumvent benchmark effort and publication restrictions, I analyzed already published Transaction Processing Council (TPC) TPC-C v5 and TPC-H benchmark results where the database server was any form of Oracle.

Rather than calculating performance per core, I calculated performance per weighted core (P/WC),

using Oracle's weightings. In the TPC-C and TPC-H technical architectures, “back end” servers run the DBMS.

For each back end server, I looked up its Core Factor in the “Oracle Processor Core Factor Table.” I then multiplied that factor by the total number of server cores as given in the benchmark results spreadsheet. This gives the weighted DBMS server cores in the solution.

While the TPC-C benchmark results table also lists “front end” processors, since these do not run the DBMS server software, they do not count in this analysis.

I derived how many TPC-C transactions per second were achieved per weighted server core by dividing the TPC-C TPS by the solution's weighted server core count.

I did the same analysis for published TPC-H benchmarks.

One TPC-C benchmark was omitted from the top performer analysis. It was one of two Power based benchmarks that appeared to be identical except for the fact that one was submitted by IBM and the other by Bull.

To answer the question about the source of the weighted performance advantage, an additional TPC-C analysis factored out processor clock frequency. The analysis determined the weighted performance per core per Gigahertz. I also analyzed cache size, cache per core, and hyper-threading attributes for the processors,

Results

Oracle TPC-C Results

Table 2, shows the Top 20 TPC-C Oracle Performance per Weighted Core Results.

Table 2’s results are summarized in Table 3, the Top 20 TPC-C Performance per Weighted Core Summary Results Table. It shows the chip architectures and, for the Intel XEONs, the operating system they ran. All the IBM Power processors ran AIX, the Oracle SPARC processors ran Solaris, while the on the Itaniums, two ran HPUX while one ran Red Hat Enterprise Linux.

Hyper-threaded Intel Xeon processors came out on top, followed by IBM Power6, Power5+, Oracle SPARC T3, IBM Power5, Intel Itanium2 and non-Hyper-threaded Xeon.

TPC-C Relative Architecture and Design Rating, Table 4, shows the results of factoring processor speed out of the analysis. Generally, the hyper-threaded Intel Xeon processors come out on top except that the Oracle SPARC T3 comes out ahead of the Intel Xeon X5650. Itanium2’s are both ahead and behind the Power5+ and Power5. The Power6 is below both the Itanium2 and the Xeon Quad-Core X5460.

These are the raw facts which are to be made sense of in the discussion and conclusions sections.

Table 2
Top 20 TPC-C Oracle Performance per Weighted Core Results

Processor	Cores / Processor	Cache Per Processor	Clock	Cache Per Core	**Perf / Weighted Core**	TpmC	Cores	Weighted Core Factor
Intel Xeon Processor X5570 2.93GHz	**4**	8	2.93	2	157942	631766	8	0.50
Intel Quad-Core Xeon E5520 2.26GHz	**4**	8	2.26	2	119696	239392	4	0.50
Intel Xeon E5520 2.27 GHz	**4**	8	2.26	2	116001	232002	4	0.50
IBM POWER6 - 4.7 GHz	**2**	8	4.7	4	101116	404462	4	1.00
Intel Xeon X5650 6-core 2.66GHz	**6**	12	2.66	2	96680	290040	6	0.50
IBM POWER5+ - 2.2 GHz					78757	236271	4	0.75
SPARC T3 1.65GHz					70022	30249688	1728	0.25
Intel Xeon Quad-Core X5460 - 3.16 GHz	4	12	3.16	3	68417	273666	8	0.50
IBM POWER5 - 1.9 GHz					67813	203440	4	0.75
IBM POWER5 - 1.9 GHz					66741	1601785	32	0.75
IBM POWER5 - 1.9 GHz					64797	194391	4	0.75
IBM POWER5 - 1.9 GHz					61841	371044	8	0.75
Intel Itanium2 Dual-Core - 1.6 GHz					57642	230569	4	1.00
Intel Xeon X7460 - 2.67 GHz	6	16	2.67	2.67	53271	639253	24	0.50
Intel Xeon QC 5440 - 2.83 GHz	4	12	2.83	3	52246	104492	4	0.50
Intel Xeon X5355 - 2.66 Ghz	4	8	2.66	2	51227	102454	4	0.50
Intel Xeon X5355 - 2.66 GHz	4	8	2.66	2	50463	100926	4	0.50
Intel Itanium2 Dual-Core - 1.6 GHz					50207	200829	4	1.00
Intel Xeon QC 5440 - 2.83 GHz	4	12	2.83	3	48542	97083	4	0.50
Intel Itanium2 Dual-Core - 1.6 GHz					44930	359440	8	1.00

Table 3
Summary Top 20 TPC-C Performance per Weighted Core Results Table

Results Summary Table:	XEON	*Oracle Enterprise Linux x86-64 on Xeon*	*MS Windows Server x86-64 on Xeon*	POWER	SPARC	Itanium
Top 5 TPC-C TPS per weighted core	4 =	*(3 +*	*1)*	1		
Top 10 TPC-C TPS per weighted core	5 =	*(4 +*	*1)*	4	1	
Top 20 TPC-C TPS per weighted core	10 =	*(6 +*	*4)*	6	1	3

Table 4
TPC-C Relative Architecture and Design Rating

Processor	Relative Weighted Architecture & Design Rating	Perf / Weighted Core / GHz	Clock (GHz)	Perf / Weighted Core	TpmC
Intel Xeon Processor X5570	1.00	53905	2.93	157942	631766
Intel Quad-Core Xeon E5520	0.98	52963	2.26	119696	239392
Intel Xeon E5520 2.27 GHz	0.95	51328	2.26	116001	232002
SPARC T3 1.65GHz	0.79	42438	1.65	70022	30249688
Intel Xeon X5650 6-core 2.66GHz	0.67	36346	2.66	96680	290040
Intel Itanium2 Dual-Core - 1.6 GHz	0.67	36026	1.6	57642	230569
IBM POWER5+ - 2.2 GHz	0.66	35799	2.2	78757	236271
IBM POWER5 - 1.9 GHz	0.66	35691	1.9	67813	203440
IBM POWER5 - 1.9 GHz	0.65	35127	1.9	66741	1601785
IBM POWER5 - 1.9 GHz	0.63	34104	1.9	64797	194391
IBM POWER5 - 1.9 GHz	0.60	32548	1.9	61841	371044
Intel Itanium2 Dual-Core - 1.6 GHz	0.58	31380	1.6	50207	200829
Intel Itanium2 Dual-Core - 1.6 GHz	0.52	28081	1.6	44930	359440
Intel Xeon Quad-Core X5460 - 3.16	0.40	21651	3.16	68417	273666
IBM POWER6 - 4.7 GHz	0.40	21514	4.7	101116	404462
IBM POWER6 - 4.7 GHz	0.40	21514	4.7	101116	404462
Intel Xeon X7460 - 2.67 GHz	0.37	19952	2.67	53271	639253
Intel Xeon X5355 - 2.66 Ghz	0.36	19258	2.66	51227	102454
Intel Xeon X5355 - 2.66 GHz	0.35	18971	2.66	50463	100926
Intel Xeon QC 5440 - 2.83 GHz	0.34	18461	2.83	52246	104492
Intel Xeon QC 5440 - 2.83 GHz	0.32	17152	2.83	48542	97083

Oracle TPC-H Results

Table 5 provides the Summary of Analysis of TPC-H Performance per Weighted Core. The results do not show the consistency of the TPC-C analysis. They vary greatly with the scale of the benchmark. At the highest scale, Itanium comes out on top. In the other two, Itanium comes out on the bottom. SPARC has the highest rating for the midscale benchmark, but comes in second at the high and low end benchmarks.

At benchmark Scale Factor 10000, the Itanium 9x40 benchmarks came out at 34% and 63% ahead of the UltraSPARC IV+.

At benchmark Scale Factor 3000, the Power5 and SPARC 64 VII were 20% ahead of the Opteron dual core 285, which was 46% ahead of the Itanium2 9050.

At benchmark Scale factor 1000, the XEON x5450 is 84% ahead of the SPARC 64 VI, which is 13% ahead of the best result from the Itanium family.

This shows that what is best depends not only on the benchmark workload, but also its scale.

Table 5
Summary of Analysis of TPC-H Performance per Weighted Core

TPC-H Benchmark Scale 10000	**TPC-H Benchmark Scale 3000**	**TPC-H Benchmark Scale 1000**
1 Itanium (9x40)	1 SPARC (64 VII) & POWER5	1 XEON (x5450)
2 SPARC (UltraSPARC IV+)	2 Opteron (285)	2 SPARC 64 VI
	3 Itanium (Itanium2 9050)	3 Itanium

Discussion

Oracle TPC-C Analysis

Hyper-threaded XEON's provided 80% of the top 5 results, while non Hyper-threaded XEON's were half of both the top 10 and the top 20.

Do not, however, place very much value on the quantity of benchmarks in the top 5, 10, or 20. The quantity of entries represents the willingness of vendors to perform benchmarks and publish results. The value of several similar results lies primarily in their consistency even as other elements may vary, such as chipset, SAN attachment, storage subsystem, etc.

With regard to the operating system used in the XEON based benchmarks within the top 5, 10, and 20, the operating system was Oracle Enterprise Linux for 75% in the top 5, 80% in the top 10, and 60% in the top 20. The remainder of the XEONs in the top 20 ran an Oracle DBMS on Microsoft Windows Server.

It is important to consider, however, that the operating system choice reflects the preferences of the people who performed the benchmarks more than anything else, at least to some extent.

It is much more important to consider the groupings in Table 2's Performance per Weighted Core column. The first entry, 157,942 P/WC for the Hyper-threaded Xeon X5570 2.93GHz clearly stands out. The next group, with ratings from 119,696 to 96,680 P/WC includes Hyper-threaded Xeon's, and Power6 with its Simultaneous Multi-Threading (SMT). The third group spans from 78,757 down to 44,930 P/WC.

Oracle TPC-H Analysis

There are not many benchmark results reported for TPC-H on Oracle, especially at scale 3000 and 10000.

The differences between 1st and 2nd place and 3rd place are much bigger at the high and low ends than in the middle. At scale 3000 the differences between 1st and 2nd place is 20% while it is 63% and 84% for scales 10000 and 1000 respectively.

Conclusions

The Methodology

The methodology is not difficult, certainly as compared to the effort to perform the published benchmarks of middleware on several processor architectures. Given hardware and software pricing trends, it is valid to assume that software costs completely overwhelm hardware costs to the point that it can be ignored in the comparison.

- Oracle DBMS for TPC-C Analysis Related Conclusions

Architecture Assessment Summary

The Hyper-threaded XEONs come out on top because of performance and advantageous core factor weighting.

The very top TPC-C Xeon performers had 4 Hyper-threaded cores. Other Xeon based solutions in the top 20 had 4, 6, 8 or 24 non-Hyper-threaded cores, not necessarily all on one chip. With only one XEON example in the top 20 with more than 8 cores, there are too few data points to be confident that this analysis scales up much beyond 8 XEON cores.

The Power 6's are next, due to sheer speed, in spite of heaviest weighting.

The SPARC T3 would still be in the top 20 chart, ahead of the some of the XEONs, even if it had the same weighting factor as the XEONs.

The best Itanium result compares closely to the Power5 results. While the best XEON rating is 158K, the Itanium results vary from 58K down to 45K rating units.

Source of High Ratings

What made the top performers do so well? Was it clock speed, cache size, cache size per core, multithreading or Oracle's core weighting factor for them?

Cache: It wasn't cache per core. The top three outperformed those with larger cache per core ratings and larger total cache.

Speed: It wasn't clock speed. The top three outperformed others with higher clock rates.

It was a combination of Oracle's weightings and the lack of a weighting differentiation among processors with and without multithreading.

The top Xeon processor models had Hyper-threading which the lower performing models with faster clocks, more cache, and more cache per core all lacked.

- Oracle DBMS for TPC-H Analysis Related Conclusions

Platform choice seems to be much more important at the high end, 10000 scale, and at the low end, 1000 scale, than for the middle, 3000 scale. It seems to imply that high-end hardware, e.g. Itanium, is most appropriate for the highest scale, while high performing low–end hardware is most appropriate for the lowest scale.

- Processor Cache Conclusions

I was very surprised. I had expected that gains from having a large processor cache or cache per core would outperform both higher speed and Hyper-threading. I was wrong. It appears that 2MB of cache per core is sufficient for Oracle in the TPC-C benchmark environment.

However, in a real deployment, you might run multiple middleware instances on one server or multiple server instances on a hypervisor. These would benefit from more cache per core.

That is why, whenever more cache per core is available, I would recommend it, if the extra cost is modest. Every 1 percent of improvement in cache hits makes a very large percent difference in the percent of cache misses. For example, going from 90% cache hits to 91% is only a 1.1% improvement in hits, but is a 10% improvement in misses, since they drop from 10% to 9%.

Cache misses are idle processor busy time. How can processor time be busy and idle at the same time? In your processor utilization reporting, that idle time due to cache misses is counted as CPU busy time, since the processor is in the midst of fetching or executing an instruction. Unless another thread can successfully run during the other thread's cache miss, the processor core is "busy" doing nothing

during the cache miss's 50-100 CPU clock cycle duration.

In summary, your current or future situation might really benefit from the additional cache.

Futures

We cannot foresee with certainty what architectural & clock speed improvements will be, nor how software vendors will weight them, nor how the software will perform on them. Forecasting is complicated by the fact that some of the top results came from older generation processors. We can, however speculate on the impact of processor speed, as long as the software vendor does not change the weightings.

The Table 4 shows the results after factoring out the clock rate, but leaving the Oracle weighting, and then comparing the results to the top rating, the XEON x5570. The results show that the Hyper-threaded XEONs are still on top. The SPARC T3 comes next. One XEON, the Itaniums and POWER5's form another group, while the POWER6's and other XEONs come in at the bottom.

As long as Oracle does not change the core factor weightings as processor speeds improve, which do appear to be improving after a few year lag, the Hyper-threaded XEONs continue to lead. An Oracle SPARC T3 (or its follow on) with a jump in speed might move it into competitive position in the Performance per Weighted Core Rating. Unfortunately, without current generation Itanium 9300 and Power7 in the published TPC-C results it is hard to forecast how they will fit.

If Oracle's Core Weighting Factors change, it is easy enough to perform these calculations again, applying the new factors to the old benchmarks, to see what the results would be. For example, Table 6 shows the changes that would result in the TPC-C P/WC ratings if Oracle were to replace its Core Weightings with IBM's Processor Value Units. It shows that:

- Hyper-threaded Xeon's would stay on top;
- SPARC would drop 24 steps, completely out of the top 20;
- Power family members would rise 1 and drop 1, 3, 5, & 6 steps;
- Itanium would rise 4 & 5 steps.

Table 6
TPC-C on Oracle P/WC Ranking Shifts Due to Alternative Weightings
(similar rows have been omitted)

Server CPU Type	Order based on Oracle Core Weighting Factors	Order based on IBM Processor Value Units	Order Difference
Intel Xeon Processor X5570 2.93GHz	1	1	**0**
Intel Xeon E5520 2.27 GHz	3	4	**-1**
IBM POWER6 - 4.7 GHz	4	3	**1**
Intel Xeon X5650 6-core 2.66GHz	5	5	**0**
IBM POWER5+ - 2.2 GHz	6	7	**-1**
SPARC T3 1.65GHz	7	31	**-24**
Intel Xeon QC X5460 - 3.16 GHz	8	6	**2**
IBM POWER5 - 1.9 GHz	9	12	**-3**
IBM POWER5 - 1.9 GHz	10	15	**-5**
IBM POWER5 - 1.9 GHz	12	18	**-6**
Intel Itanium2 Dual-Core - 1.6 GHz	13	8	**5**
Intel Xeon X7460 - 2.67 GHz	14	9	**5**
Intel Xeon X5355 - 2.66 GHz	17	13	**4**
Intel Itanium2 Dual-Core - 1.6 GHz	18	14	**4**
Intel Xeon QC 5440 - 2.83 GHz	19	17	**2**

Summary

It is possible to determine that the Hyper-threaded Intel Xeon, by a large margin, is the optimal price performing processor to host Oracle DBMS workload today as long as that workload is comparable to the TPC-C benchmark.

Middleware, application, scale, processor and weighting all really do matter. The methodology produced clear results for Oracle running TPC-C workloads. For Oracle running TPC-H workloads, the results very much depended on the scale of the benchmark. The observed best to worst ratio exceeds 7x in one case.

The main limitation on this methodology is finding or performing benchmarks that match your middleware workload.

Acknowledgements

I wish to thank:

- The Transaction Processing Council for making it so easy to acquire benchmark results in readily usable form;
- Oracle Corp. for making its Core Weighting Factor Table readily available; and especially,
- All the staff at all the companies who implemented, tuned, measured, and reported the benchmarks.

Disclaimer

The views and opinions expressed in this article are those of its author, David A. Kra, and not necessarily those of his employer, Infocrossing, Inc., and/or any affiliates of Infocrossing, Inc.

Trademarks

References

Oracle Processor Core Factor Table (As updated 06/03/2011)
http://www.oracle.com/us/corporate/contracts/processor-core-factor-table-070634.pdf

TPC-C results as a spreadsheet:
http://tpc.org/downloaded_result_files/TPC-C_results.xls

TPC-H results as a spreadsheet:
http://tpc.org/downloaded_result_files/TPC-H_results.xls

For an opposing opinion on the platform selection topic, see:
What Do Oracle Core Factors Really Mean?
http://oracleoptimization.com/2010/04/21/what-do-oracle-core-factors-really-mean/

Exploratory Study of Performance Evaluation Models for Distributed Software Architecture

S.O. Olabiyisi; E.O. Omidiora
Department of Computer Science & Engineering
Ladoke Akintola University of Technology, Ogbomoso
Oyo State, Nigeria

Faith-Michael Uzoka
Department of Computer Science & Information Systems
Mount Royal University
Calgary, Canada

Boluwaji A. Akinnuwesi
Department of Information Technology
Bells University of Technology, Ota, Ogun State
Nigeria

Victor W. Mbarika
International Centre for Information Technology and Development
Southern University, Baton Rouge, Louisiana, USA

Mathieu Kourouma
Department of Computer Science
College of Sciences
Southern University, Baton Rouge, Louisiana, USA

Hyacinthe Aboudja
Department of Computer Science
School of Business
Oklahoma City University

Several models have been developed to evaluate the performance of Distributed Software Architecture (DSA) in order to avoid problems that may arise during system implementation. This paper presents a review of DSA performance evaluation models with the view of identifying the common properties of the models. It was established in this study that the existing models evaluate DSA performance using machine parameters such as processor speed, buffer size, cache size, server response time, server execution time, bus and network bandwidth size and lots of others. The models are thus classified to be machine-centric. Moreover the involvement of end users in the evaluation process is not emphasized. Software is developed in order to satisfy specific requirements of the client organization (end-users); therefore, involving users in evaluating DSA performance should not be underestimated. This study suggests future works on establishing contextual organizational variables that can be used to evaluate DSA. Also to complement the existing models, works should be done on development of user-centric performance evaluation model which will directly involve the end-users in the evaluation of DSA using the identified contextual organizational variables as parameters for evaluation.

Keywords: *Distributed software, Performance, Performance evaluation model, Software system architecture, Client organization, machine-centric, user-centric*

INTRODUCTION

Today, distributed computing applications are used by many people in real time operations such as electronic commerce, electronic banking, online payment, et cetera [22]. Distributed computing is used as enabling technology for modern enterprise applications; thus in the face of globalization and ever increasing competition, Quality of Service (QoS) attributes like performance, security, reliability, scalability, and robustness are of crucial importance [29]. Companies must ensure that the distributed software (DS) they operate does not only provide all relevant functional services, but also meet the performance expectation of their customers. Therefore it becomes imperative to analyze and predict the expected performance of distributed software systems at the level of the architectural design in order to avoid the pitfalls of poor QoS during system implementation.

Software architecture (SA) is a phase of software design which describes how a system is decomposed into components, how these components are interconnected, and how they communicate and interact with each other. This phase of software design is a major source of errors if the organizational structure of the different components is not carefully defined and designed. There are two parts to SA [6, 33]. The first part is the micro-architecture which covers the internal structure of the software system such as conceptual architecture, module interconnection architecture, execution architecture, and code architecture. The second part of SA is the macro-architecture that focuses on external factors that could influence the design and implementation of the software system. Examples of the external factors are: culture and belief of people (users), government policies and regulations, and disposition of people towards the use of computer.

SA is an important phase in the software life cycle as it is the earliest point and highest level of abstraction at which useful analysis of a software system is possible [35]. Hence, performance analysis at this level can be useful to establish whether a proposed architecture satisfies the end users' requirements and also meets the desired performance specifications. It also helps to identify eventual errors and verify that the quality requirements have been addressed in the design and thus saving major potential modifications later in the software development life cycle or tuning the system after deployment.. SA is considered the first product in an architecture-based development process and evaluation at this level should reveal requirement conflicts and incomplete design descriptions from stakeholders' perspective [6].

Performance of software is a quality attribute that is measured in any of the following metrics: system throughput, responsiveness, resource utilization, turnaround time, latency, failure rate, and fault tolerance. Thus, assessing and optimizing system performance is essential for the smooth and efficient operation of the software system. There are several approaches for evaluating the performance of system architecture. One of the earliest approaches is the fix-it-later approach [3] which advocates software correctness and deferring performance considerations to the integration testing phase. If performance problems are detected, then, additional hardware may be needed; otherwise, the software will be tuned to correct the problems. This approach has some limitations, such as,: it takes time to acquire and install new hardware; also tuning the software takes time and could be costly. Tuning may distort the original software design and testing must be repeated after code changes. This gives a negative impression to users after it is corrected. The rational for the fix-it-later approach is to save development time and cost. This however will not be realized, if initial performance is unsatisfactory because of additional time and cost of tuning and maintenance. Also, Connie [3] proposed a Design-Based Evaluation and Prediction Technique (ADEPT), an analysis technique used in conjunction with the performance engineering discipline. ADEPT was the strategy used to combat the fix-it-later principle and supported the performance engineering process. ADEPT evaluates the performance of information system early in the life cycle using specifications for both expected resources requirement and upper bounds. The system design is likely to be stable if the performance goal is satisfied for the upper bound. ADEPT had the following challenges: lack of automatic feedback component, not robust enough to evaluate large and complex systems, inability to eliminate unwanted argument in the course of evaluation, and inability to work in concurrent processing environments.

In recent years, several models have been developed to constantly evaluate the performance of DSA. The survey done in this paper provides the developments over about a decade (1999 – 2010) with the aim of identifying the parameters used by each model for evaluating DSA performance and also deduces the properties that are common to the models. Further research direction is proposed as a consequence.

RELATED WORKS

Many studies have been carried out on the survey of system performance evaluation models with the ultimate goal of providing recommendations for future research activities. Those activities could significantly improve the performance evaluation and prediction of software system architecture. A survey of the approaches to evaluate software performance from 1960 to 1986 was done in [4]. The study pointed out the breakthroughs leading to the software performance engineering approach (SPE) and a comprehensive methodology for constructing software to meet performance goals. The concepts, methods, tools, and use of SPE were summarized and future trends in each area were suggested.

In [6] eight architecture analysis methods were reviewed with the view of discovering similarities and differences between these methods by making classifications, comparisons, and appropriateness studies. The eight methods considered are: SAAM (Scenario-Based Architecture Analysis Method), SAAMCS (SAAM Founded on Complex Scenarios), ESAAMI (Extended SAAM by Integration in the Domain), SAAMER (Software Architecture Analysis Method for Evolution and Reusability), ATAM (Architecture Trade-Off Analysis Method), SBAR (Scenario-Based Architecture Reengineering), ALPSM (Architecture Level Prediction of Software Maintenance), and SAEM (Software Architecture Evaluation Model). The authors discovered at that time that SAAM was used for different quality attributes like modifiability, performance, availability, and security. In addition SAAM was applied in

several domains unlike the other methods that were undergoing refinement and improvement as at that time. As a result, some future works were proposed to evaluate the effects of their various usages and create a repeatable method based on repositories of scenarios, screening and elicitation questions.

Three indications that concern software design specifications, performance models, and analysis of processes were highlighted in [31]. The following recommendations were made in the paper: the use of standard software artifacts like Unified Modeling Language (UML) diagrams for software design specifications; the existence of strong semantic mapping between software artifacts and the performance models as strategy to reduce the performance model complexity and still maintaining a meaningful semantic correspondence; use of simulation in addition to analytical simulations to address performance model complexity and provision of feedback which is a key success factor for a widespread use of performance analysis models.

In [1] a review of performance prediction techniques for component-based software systems was carried out and the following recommendations were made: (1) integration of quantitative prediction techniques in software development process; (2) design of component models allowing quality prediction and building of component technologies supporting quality prediction; (3) inclusion of quality attributes such as reliability, safety or security in the software development process; and (4) study of interdependencies among the different quality attributes to determine, for example, how the introduction of performance predictability can affect other attributes such as reliability or maintainability.

In [7], three foundational formal software analyses were described. The authors reviewed emerging trends in software model and identified future directions that promise to significantly improve the cost-effectiveness.

CLASSIFICATION OF DSA PERFORMANCE EVALUATION MODELS

This paper classifies existing performance models based on the technique used to develop the models. The techniques are: (1) Factor Analysis; (2) Queuing Network; (3) Petri net; (4) Pattern-Based; (5) Hierarchical Modelling; (6) Performance Analysis and Characterization Environment (PACE) Based; (7) Component-Based Modelling; (8) Scenario-Based; (9) Soft computing approach; (10) Relational Approach; (11) Software Architecture Analysis Methods (SAAM); (12) Aspectual Software Architecture Analysis Methods (ASAAM); (13) Hybrid Approaches such as UML-Petri net, UML-Stochastic Petri net, Queue Petri Nets Approach and Soft Computing Approach. The models are reviewed in order to establish the kind of parameters used in them to evaluate DSA.

Factor Analysis (FA) Based Approach

FA approach was used in [2] to develop a model for analysing Information Technology (IT) software projects with the aim of establishing the success or failure of the project before it takes off. FA as contained in SPSS and Statview software was used. Fifty performance indices of IT projects planning, execution, management, and control were formulated. Eleven factors were extracted and subjected to further analysis with a view to estimating and ranking their contribution to the success of IT projects. The model was tested using sample life data gotten using questionnaires that were administered to the principal actors of the popular IT software projects in Nigeria. The significant contribution of the research is the provision of a working model that utilized both quantitative and qualitative decision variables in assessing the success or failure of IT projects. This serves as template for evaluating IT projects prior to its implementation. This model was not used to evaluate performance of software system architecture.

Queuing Network Based Models

This is a conventional modelling paradigm which consists of a set of interconnected queues [28]. The models based on Queuing Networks are categorized in Table 1.

Table 1 Queuing Network Based Performance Models

Description of Model	Parameters Considered	Class of Parameter
[30] designed and implemented object-oriented queuing network model – a reusable performance models for software artifacts.	Buffer size, processor speed of server, queue size, number of incoming request, request arrival time, request departure time.	Machine centric parameter

[31] integrated performance and specification model to provide a tool for quantitative evaluation of software architecture at the design phase.	Number of service centers, service rate of service center, arrival rate of requests at service centre, number of servers in service centers, routing procedure of requests, Number of request circulating in the system, physical resources available system workloads, network topology.	Software process centric and machine centric parameters
[35] modeled layered software system as a closed Product Form Queuing Network (PFQN) and solve it for finding performance attributes of the system	Range of number of clients accessing the system, average think time of each client, number of layers in the software system, relationship between the machines and software components, number of CPUs and disks on each of the machine and thread limitation (if any), uplink and downlink capacities of the connectors connecting machines running adjacent layers of the system, size of packets of the links, service time required to service one request by a software layer, forward transition probability, rating factors of the CPU and the disks of each machines in the system	Software and Machine centric parameters
[31] proposed an approach based on queuing networks models for performance prediction of software systems at the software architecture level, specified by UML.	Same as in [35]	Software and Machine centric parameters
[12] developed Software Architecture and Model Extraction (SAME) technique that extract communication patterns from executable designs or prototype that use message passing, to develop a Layered Queuing Network Performance Model in an automated fashion.	Same as in [35]	Software and Machine centric parameter

Petri Net Based Approach

Petri nets were introduced in 1962 by Dr. Carl Adam Petri [27]. A Petri net is a graphical and mathematical modelling tool [26]. It is a directed bipartite graph with an initial state called the initial marking. Petri Nets consist of four basic elements: places, transitions, tokens, and arcs. System performance models based on Petri net approach are categorized in Table 2.

Table 2 Petri Net Based Performance Models

Description of model	Parameters Considered	Class of Parameter
[18] developed performance evaluation model for Agent-based system using petri net approach	System load, system delays, system routing rate, latency of process, CPU time.	Machine centric parameters
[20] did performance analysis of Internet based software retrieval systems using petri nets	Network time.	Machine centric parameters
[13] developed stochastic petri nets model from UML activity diagrams	Routing rate, action duration, system response time.	Machine centric parameters

[14] translated UML activity diagram into stochastic Petri net model that allows to compute performance indices.	Routing rate, action duration, system response time.	Machine centric parameters
[23] derived performance parameters from Generalized Stochastic Petri Net (GSPN) using Markov chain theory.	Routing rate, action duration, system response time.	Machine centric parameters

Queuing Petri Net (QPN) Based Models

The hybrid of Petri Net and Queuing Networks is Queuing Petri Nets (QPNs) which facilitates the integration of hardware and software aspects of system behaviour into the same model. In addition to hardware contention and scheduling strategies, using QPNs, one can easily model simultaneous resource possession, synchronization, blocking, and contention for software resources. Thus QPNs combines Queuing Networks and Petri Nets into a single formalism in order to eliminating their disadvantages. QPNs allow queues to be integrated into places of Petri Nets and this enables the modeller to easily represent scheduling strategies and to bring the benefits of Queuing Networks into the world of Petri Nets [28]. System performance models based on Queuing Petri net approach are categorized in Table 3.

Table 3 Queuing Petri Net Based Performance Models

Description of Model	Parameters Considered	Class of Parameters
[28] applied QPN formalism to analyse the performance of distributed e-business system.	Service demand of queue, service rate of queue, token population of queue, queue size, buffer size, processor speed of server, routing rate.	Machine centric parameters
[29] presented a novel case study of a realistic state-of-the-art distributed component-based system, showing how the QPN modelling formalism can be exploited as software system performance prediction tool.	Same as in [28].	Machine centric parameters

Performance Analysis and Characterization Environment Based Approach

The motivation to develop Performance Analysis and Characterization Environment (PACE) based approach in [15] was to provide quantitative data concerning the performance of sophisticated applications running on high performance systems. The framework of PACE is a methodology based on a layered approach that separates out the software and hardware system components through the use of a parallelization template. This is a modular approach that leads to readily reusable models, which can be interchanged for experimental analysis. Each of the modules in PACE can be described at multiple levels of details thus providing a range of result accuracies, but at varying costs in terms of prediction evaluation time. PACE is aimed to be used for pre-implementation analysis, such as design or code porting activities, as well as, for on-the-fly use in scheduling systems. The core component of PACE is a performance specification language, CHIP^{3}S (Characterization Instrumentation for Performance Prediction of Parallel Systems). CHIP^{3}S provides a syntax that allows the description of the performance aspects of an application and its parallelization to be expressed. This includes control flow information, resource usage information (for example number of operations), communication structures, and mapping information for a parallel or distributed system. The software object in the PACE system were created using the Application Characterization Tool (ACT). ACT aids the conversion of sequential or parallel source code into the CHIP^{3}S language via the Stanford Intermediate Format (SUIF). ACT performs a static analysis of the code to produce the control flow of the application, count the number of operations in terms of high-level language implemented, and also the communication structure. The hardware objects of the model are created using a Hardware Model Configuration Language (HMCL) by specifying system-dependent parameters. On evaluation, the relevant sets of parameters are used and supplied to the evaluation methods for each of the component models.

Hierarchical Performance Modeling Approach

In [32] a Hierarchical Performance Modelling (HPM) technique for distributed systems, which incorporated different level of modelling abstraction, was presented. HPM is a technique to model performance for different layers of abstraction. It includes several layers of organization from primitive operation to software architecture, therefore, providing a degree of accuracy that cannot be achieved with single layer models. The application is developed in a top-down fashion from general to more specific, but performance information is generated in bottom-up method, thus linking the different levels of analytic models into a composite model. This approach support specification and performance model generation that incorporates computation and communication delays along with hardware profile characteristics to assist in the evaluation of performance alternatives. HPM models provide a quantitative performance assessment of an entire system comprising of hardware, software, and communication. The HPM provided a well-defined methodology to allow system designers to evaluate the application based on the system requirements of their application and fine tune the values of performance parameters.

Pattern Based Approach

Design patterns are defined as description of communicating objects and classes that are customized to solve a general design problem in a particular context. The components of design pattern are: Pattern name, Intent, Motivation, Applicability, Structure, Participants, Collaborations, Consequences, Implementation, Sample code, Known uses and Related pattern. Performance models based on pattern based approach are presented in Table 4.

Table 4 Pattern Based Performance Models

Description of Model	Parameters Considered	Class of Parameter
[19] presented an approach based on patterns to develop performance models for object oriented software system in the early stages of the software development process. This complement the approach given in [18]	Event load, time to perform an action, request arrival time, request service time, number of concurrent users	Software process centric parameters
[21] presented a pattern-based approach to model the performance of software system and used it to evaluate the performance of mobile agent system	Same as in [19]	Software process centric parameters
[9] presented a pattern-based performance completion for message-oriented middleware	System configuration (hardware & network components), message size (incoming & outgoing), delivery time for message, number of message sent, size of message sent, number of message delivered, size of message delivered, transaction/request size, buffer/pool size	Software process centric parameters and machine centric parameters

Soft Computing Approach

Soft computing is an approach to computing which parallels the remarkable ability of the human mind to reason and learn in an environment of uncertainty and imprecision [8]. It is a consortium of methodologies centering in fuzzy logic (FL), artificial neural networks (ANN) and evolutionary computation (EC). These methodologies are complementary and synergistic, rather than competitive. They provide in one form or another flexible information processing capability for handling real life ambiguous situations. Soft computing aims to exploit the tolerance for imprecision, uncertainty, approximate reasoning, and partial truth in order to achieve tractability, robustness, and low-cost solutions. The attributes of these models are often measured in terms linguistic values, such as very low, low, high, and very high. The imprecise nature of the attributes constitutes uncertainty and vagueness in their (subsequent) interpretation. Performance models based on soft computing approach are presented in Table 5. The advantage of Soft computing models particularly fuzzy logic and ANN are [10]: they are more general and they mimic the way in which humans interpret linguistic values and the transition from one linguistic value to a contiguous linguistic value is gradual rather than abrupt.

Table 5 Performance Models Based Soft Computing Approach

Description of Models	Parameters Considered	Class of Parameter
[10] applied fuzzy logic to measure similarity of software projects when their attributes are described by categorical values (linguistic values in fuzzy logic)	Seventeen parameters: software size, project mode plus 15 cost drivers.	Software process centric and machine centric parameters
[11] presented a new technique based on fuzzy logic, linguistic quantifiers and analogy-based reasoning to estimate the cost of or effort of software projects when they are described by either numerical data or linguistic values.	Same as in [10]	Software process centric and machine centric parameters
[17] showed how fuzzy logic can be applied to computer performance work to simplify and speed analysis and reporting.	CPU Queue length, memory (RAM) available, pages input per second, read time, write time, I/Os per second.	Machine centric parameters
[25] Developed a fuzzy model for evaluating information system projects based on their present value using fuzzy modelling technique.	Three parameters representing three possible values of project costs, benefits, evaluation periods, and discount rate.	Software process centric parameters

Other Performance Models

In [5], multivariate Adaptive Regression Splines (MARS) was used for software performance analysis. A resource function was designed and automated, having the following parameters - size of data objects, number of disk blocks to be read, size of messages to be processed, memory and cache size, processor speed, bus and network bandwidth.

In [16], PASA, a method for performance assessment of software architectures, was developed and it was scenario-based. It identifies potential areas of risk within the architecture with respect to performance and other quality objectives. It identifies strategies for reducing or eliminating the risks if a problem is found. Scenario for important workloads are identified and documented. The scenarios provide means of reasoning about the performance of the software as well as other qualities and they serve as starting point for constructing performance models of the architecture.

ASAAM (Aspectual Software Architecture Analysis Method) is scenario-based proposed in [34]. It introduces a set of heuristic rules that help to derive architectural aspects and the corresponding tangled architectural components from scenarios. It takes as input the architecture design and measures the impact of predefined scenarios on it in order to identify the potential risks and the sensitive points of the architecture. This helps to predict the quality of the system before it is built and therefore reducing unnecessary maintenance costs.

In [36], performance analysis based on requirements traceability was presented. Requirement traceability is critical to providing a complete approach which will lead to an executable model for performance evaluation. The paper investigated the software architectures that are extended based on the performance requirements traceability to represent performance property. The extended architectures are then transformed into a simulation model colored Generalized Stochastic Petri Nets (GSPN) and the simulation results are used to validate performance requirements and evaluate system design. The parameters considered are queue length, number of requests to be serviced, server response time, server execution time, and processor speed.

GENERAL PROPERTIES OF THE EXISTING DSA PERFORMANCE EVALUATION MODELS

From survey of the existing DSA performance evaluation models, the following common attributes are identified:

i. The models are algorithmic using hard computing principles.
ii. Parameters for evaluation are machine centered and they are objective. For example, processor speed, bus and network bandwidth size, RAM size, cache size, server response time, server execution time, number of disk to be read and message size. Therefore the models are machine-centric.

iii. The models are implemented at the architectural stage of the software life cycle.
iv. Though in the existing models, the contributions of the client organization (end users) during software development process were acknowledged but none of the models draws parameters for evaluation from the contextual organizational decision variables.
v. The models are re-useable and scalable.
vi. Performance metrics considered are mostly the following: throughput, response time, and resource utilization.
viii. The models are limited by their inability to cope with uncertainties and imprecision of data or information surrounding software projects in the early stage of the development life cycle.
ix. The conceptual structures of some model (for example, probabilistic models) that can represent vague information are inadequate for dealing with problems in which information is perception-based and is expressed in linguistic form.
x. The models are computationally intensive and are intolerant of noise. They cannot handle categorical data other than binary valued variables.

CONCLUSION AND FUTURE WORK

Conclusion

In this paper, a review of research works on performance evaluation models from 1999 to 2010 is presented in order to establish the properties common to these models. It was deduced that most models for evaluating DSA performance are machine-centric. The following are some of the evaluation parameters identified: *buffer size, processor speed, cache size, server response time, server execution time, number of disk block to be read, queue size, request arrival time, request departure time, bus size, network bandwidth size (uplink and down link), number of Central Processing Unit (CPU), number of request circulating in the system, system routing rate, latency of system, network time, system RAM (Random Access Memory) size, size of data object, size of message to be processed.* The performance evaluation models are, therefore, classified as machine-centric models. They are established and used to evaluate DSA performance with respect to satisfying the machine and system process requirements. However subjective decision variables of users are not considered in the machine-centric models; also the models cannot cope with uncertainties and imprecision of data or information surrounding software projects in the development life cycle. Users are involved in DSA development in order to feed the software developers with the necessary organizational information. This helps the software developers to develop software system that will be accepted by end users and satisfies the organization's requirements using available machine infrastructure. The question is "how do we measure the performance of the DSA from users' perspectives in order to establish the extent of responsiveness of the DSA to the requirements of the client organization". It is hoped that future research works will address this question.

Future Work

Management of the client organization and the end users are key players in software development process. Therefore, contextual organizational decision variables (for example: *Organizational goals and task; Level of users competence/experience in Information Technology; Information requirements of users and the format; Internal service of the organization, and their relationships; The organization's defined functions required in the user interface; Organization's policies, rules or procedures for transaction process flow etcetera*), should not be underestimated while establishing the variables to evaluate performance of software architecture. We therefore propose, as a result, that future works should identify and verify with some empirical analysis, both objective and subjective contextual organizational decision variables that could influence the choice of architectural style and design pattern made by the software developer. We are of the view that if some organizational variables can be established as parameters to evaluate DSA performance, it will be possible to have some DSA performance evaluation models that will be user-centric or a hybrid model having both organizational decision variables and machine/system variables as parameters for evaluation.

REFERENCES

[1] Bailey, H.D., Snavely, A. "Performance Modeling: Understanding the Present and Predicting the Future", Proceedings of Euro-Par, Lisbon, Portugal: 2005

[2] Chiemeke, S.C. "Computer Aided System for Evaluating Information Technology Projects", PhD thesis submitted to the School of Postgraduate Studies, Federal University of Technology, Akure, Ondo State, Nigeria: 2003.

[3] Connie, U.S. "Increasing Information System Productivity", Proceedings of the Computer Measurement Group's International Conference,The Computer Measurement Group Inc: 1981.

[4] Connie, U.S. "The Evolution of Software Performance Engineering: A Survey", Proceedings of ACM Fall Joint Computer Conference: 1986, pp 778 – 783.

[5] Courtois, M., Woodside, M. "Using Regression Splines for Software Performance Analysis", Proceedings of WOSP, Ontario, Canada. 2000.

[6] Dobrica, L., Niemela, E. "A Survey on Software Architecture Analysis Methods", *IEEE Transactions on Software Engineering*, (28:7), 2002,

[7] Dwyer, B.M., Hatcliff, J., Pasareanu, S.C., Visser, W. "Formal Software Analysis: Emerging Trends in Software Model Checking", Proceedings of Future of Software Engineering (FOSE'07): 2007.

[8] Gary, R.G., Frank, C., 1999. "Application of Neuro-Fuzzy Systems to Behavioral Representation in Computer Generated Forces", Proceedings of 8th Conference on Computer Generated Forces and Behavioural Representation, Orlando FL: 1999.

[9] Happe, J., Friedrich, H., Becker, S., Reussner, H.R. "A Pattern-Based Performance Completion for Message-Oriented Middleware", Proceedings of WOSP'08, Princeton, New Jersey: 2008.

[10] Idris, A., Abran, A. "A Fuzzy Based Set of Measures for Software Project Similarity: Validation and Possible Improvements", Proceedings of METRICS 2001, London, England: 2001, pp 85 – 96.

[11] Idris A., Alain A. and Khoshgoftaar. "Fuzzy Case-Based Reasoning Models for Software Cost Estimation". 2004. Available @ http://www.gelog.etsmtl.ca/publications/pdf/803.pdf

[12] Israr, A., Tauseef, L.H.D., Franks, G., Woodside, M. "Automatic Generation of Layered Queuing Software Performance Models from Commonly Available Traces", Proceedings of WOSP'05, Palma de Mallorca, Spain: 2005

[13] Juan, P.L., Jose M., Javier, C. "From UML Activity Diagrams to Stochastic Petri Nets: Application to Software Performance Engineering", Proceedings of WOSP'04, Redwood City, California: 2004.

[14] Juan, P.L., Jose, M., Javier, C. "On the use of Formal Models in Software Performance Evaluation", News in the Petri Nets World, Dec. 27, 2008. Available @ <http://webdiis.univzar.es.crpetri/paper/jcampos/02_LGMC_JJCC.pdf>

[15] Junwei, C., Darren, J.K., Efstathios, P., Graham, R.N. "Performance Modeling of Parallel and Distributed Computing Using PACE", Proceedings of IEEE International Performance Computing and Communications Conference, IPCCC-2000, Phoenix: 2000, pp 485 – 492.

[16] Lloyd, G.W., Connie, U.S. "PASA[SM]: An Architectural Approach to Fixing Software Performance Problems", Software Engineering Research and Performance Engineering Services: 2002.

[17] Maddox, M. "Using Fuzzy Logic to Automate Performance Analyses", Proceedings of the Computer Measurement Group's 2005 International Conference, The Computer Measurement Group inc: 2005.

[18] Merseguer, J., Javier, C., Eduardo, M. "Performance Evaluation for the Design of Agent-Based Systems: A Petri Net Approach", Proceedings of the workshop on Software Engineering and Petri Nets within the 21st International Conference on Application and Theory of Petri Nets, University of Aarhus: 2000a. pp 1 – 20.

[19] Merseguer, J., Javier, C., Eduardo, M. "A Pattern-Based Approach to Model Software Performance", Proceedings of the 2nd International Workshop on Software and Performance, Ottawa, Ontario: 2000b, pp 137-142.

[20] Merseguer, J., Campos, J., Mena, E. "Performance Analysis of Internet Based Software Retrieval Systems Using Petri Nets", Proceedings of 4th ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile System, Rome Italy: 2001.

[21] Merseguer, J., Javier, C., Eduardo, M. "A Pattern-based Approach to Model Software Performance Using UML and Petri Nets: Application to Agent-based Systems", Proceedings of 7th World Multiconference on Systemic Cybernetics and Informatics, Orlando, Florida: 2003, (9), pp 307 – 313.

[22] Merseguer, J., Javier, C. "Software Performance Modeling Using UML and Petri Nets", LNCS2965, Springer Verlag: 2004, pp 265-289.

[23] Motameni, H., Movaghar, A., Siasifar, M., Montazeri, H., Rezaei, A. "Analytic Evaluation on Petri Net by Using Markov Chain Theory to Achieve Optimal Models", *World Applied Sciences Journal* (3:3), 2008, pp 504 – 513.

[24] Olabiyisi S.O, Omidiora E.O, Uzoka F.M.E, Victor Mbarika, Akinnuwesi B.A**.** "A Survey of Performance Evaluation Models for Distributed Software System Architecture". Proceedings of International Conference on Computer Science and Application, World Congress on Engineering and Computer Science (WCECS 2010), San Francisco: 2010, Vol. 1, pp 35 – 43.

[25] Omitaomu, A.O., Adedeji, B. "Fuzzy Present Value Analysis Model for Evaluating Information System Projects", *Engineering Economist* (52:2), 2007, pp 157 – 178.

[26] Peterson, J.L. *Petri Net Theory and the Modeling of Systems*, Prentice Hall, 1981.

[27] Petri, C.A. "Communication with Automata". Technical Report RADC-TR-65-377, Rome Air Dev. Centre, New York: 1962

[28] Samuel, K., Alejandro, B. "Performance Modeling of Distributed E-Business Applications Using Queuing Petri Nets", Proceedings of IEEE International Symposium on Performance Analysis of Systems and Software: 2003, pp 145 – 153.

[29] Samuel, K. "Performance Modeling and Evaluation of Distributed Component-Based System Using Queuing Petri Nets", *IEEE Transactions on Software Engineering*. (32:7), 2006, pp 487 – 502.

[30] Savino-Vazquez, N., Puigjaner, R. "A Component Model for Object-Oriented Queuing Networks and its Integration in a Design

Technique for Performance Models", Proceedings of the 2001 Symposium on Performance Evaluation of Computer and Telecommunication System (SPECTS 2001), Orlando, Florida: 2001.
[31] Simonetta, B., Roberto, M., Moreno, M. "Performance Evaluation of Software Architecture with Queuing Networking Model", Proceedings of ESMc'04, Paris, France: 2004.
[32] Smarkusky, D., Ammar, I.A., Sholi, H. "Hierarchical Performance Modeling for Distributed System Architecture". Available @ <http://www.cs.sfu.ca/~mhefeeda/papers/ISC2000-HPM.pdf>, 2000.
[33] Soni, D., Nord, R., Hofmeister, C. "Software Architecture in Industial Applications", Proceedings 17th International Conference on Software Engineering (ICSE17): 1995, pp 196-207.
[34] Tekinerdogan B. "ASAAM: Aspectual Software Architecture Analysis Method", Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design Workshop, Boston, USA: 2003.
[35] Vibhu, S.S., Pankaj J., Kishor S.T. "Evaluating Performance Attribute of Layered Software Architecture", CBSE 2005: Vol. 3489 of LNCS, pp 66-81.
[36] Wise, J.C., Chang, C.K., Xia, J., Cleland-Huang, J. "Performance Analysis Based on Requirements Traceability", Technical Report, Dept of Computer Science, Iowa State University, Iowa: 2005.

Note: *This work is a revised version. The first version is [24] that was presented in the International Conference on Computer Science and Application, World Congress on Engineering and Computer Science (WCECS 2010), San Francisco, USA, October 20 – 22, 2010. It is one of the preliminary results of an ongoing research that focuses on developing User-Centric Model to evaluate the performance of Distributed Software Architecture.*

Acknowledgement: *This research is partly sponsored by the National Science Foundation (NSF) under Grant Nos. 1036324 and 0811453 and UNCFSP NSTI under the supervision of Dr Victor Mbarika in International Centre of Information Technology and Development, Southern University and A & M College, Baton Rouge, Louisiana, USA. Also Bells University of Technology, Ota, Ogun State, Nigeria is acknowledged for providing a partial support.*

About the Authors

S.O. Olabiyisi, Ph.D. is a Senior Lecturer in the Department of Computer Science and Engineering, LAUTECH, Ogbomosho, Nigeria. His Research interests are Software Performance Evaluation, Computational Mathematics, Discrete Structures and Softcomputing. (e-mail:tundeolabiyisi@hotmail.com)

E.O Omidiora, Ph.D. is a Senior Lecturer in the Department of Computer Science and Engineering, LAUTECH, Ogbomosho, Nigeria. His research interests are Computer Architecture, Softcomputing and e-Learning system. (e-mail: omidiorasayo@yahoo.co.uk)

F.M.E. Uzoka, Ph.D. is a Faculty member in the Department of Computer Science and Information System, Mount Royal University, Calgary, Canada. He was a Senior Lecturer in Information Systems, University of Botswana. He conducted a two year postdoctoral research at the University of Calgary (2004-2005). His research interests are Organizational Computing, Decision Support Systems, Technology Adoption and Innovation and Medical Informatics. He serves as a member of editorial/review board of a number of Information Systems journals/conferences (e-mail: uzokafm@yahoo.com).

Boluwaji A. Akinnuwesi, Ph.D.**,** is a Lecturer with Department of Information Technology, Bells University of Technology, Ota, Ogun State, Nigeria. He is also the Director of the Computer Centre in Bells University of Technology. He was a Research Scholar in International Centre of Information Technology and Development in Southern University, Baton Rouge, Louisiana, USA. His research area is Software Performance Engineering. His other research interest areas are Medical Informatics, Soft-computing, Expert System, and Software Engineering. He is a professional member of ACM, CPN (Computer Professional Registration Council of Nigeria) and NCS (Nigeria Computer Society). e-mail: akinboluade@yahoo.com.

Victor Wacham A. Mbarika, Ph.D. is the Executive Director, International Center for IT and Development (ICITD, Southern University, T. T Allain #321, Baton Rouge, LA 70813, USA. He is Editor-in-Chief of The African Journal of Information Systems (AJIS, Phone: +1 225 715 4621 or +1 225 572 1042; Fax: +1 225 208 1046. (Email: victor@mbarika.com)

Mathieu Kokoly Kourouma, Ph.D., is a professor in the Department of Computer Science, College of Sciences, at Southern University and A&M College. He has a Bachelor in Electrical and Computer Engineering from the Polytechnic Institute of the University of Conakry, Guinea, a Master and Ph.D. in Telecommunications and Computer Engineering, respectively, from the University of Louisiana at Lafayette - U.S.A. His research areas of interest are wireless communications, Sensor Networks, Cognitive Radio Networks, Telecommunications, Network Performance Analysis, Software Engineering and Development, and Database Design. He is a professional member of ACM, NSTA,

and AAC&U. Emails: mkkourouma@cmps.subr.edu and mkourouma@gmail.com. Web site: www.cmps.subr.edu. Office number: (225)771-3652.

Hyacinthe Aboudja, Ph.D. is currently visiting Assistant Professor in the Computer Science Department of the School of Business at Oklahoma City University. His research interest ranges from computer architecture, Real-time Systems Design, Theory of computing, System Performance Analysis, Software Engineering, and Computer Simulation of Biological Systems. He is a professional member of ACM and IEEE. (email: haboudja@okcu.edu)

2011 BOARD OF DIRECTORS AND CMG STAFF

President
Donna S. Folkerts
ACS, Inc.
Bus: 630-468-2262
cmgpres@cmg.org

Vice President
Frank M. Bereznay
IBM
Bus: 626-564-7530
Fax: 626-564-3595
cmgvp@cmg.org

Treasurer
Charles Savage
Bank of America
Bus: 404-607-3060
Fax: 706-387-0395
treasurer@cmg.org

Secretary
Kathy J. Steffens

Bus: 614-262-7952
secretary@cmg.org

Director
Glenn R. Anderson
IBM
Bus: 312-635-1319
Fax: 312-635-1319
grander@us.ibm.com

Director
Rick Lebsack
IBM
Bus: 303-773-7985
ralebsa@us.ibm.com

Director
Adam Grummitt
Bus: +44 (0) 1823 259231
Fax:
adam@grummitt.com

Director
Xianneng Shen
RMS Inc.
Bus: 510-608-3394
shennonshen@gmail.com

Director
Alexander Podelko
Oracle Corporation
Bus: 203-703-4355
Fax: 203-595-8516
alex.podelko@oracle.com

Director
Dave Thorn
Bus: 856-216-7550
dthorn55@gmail.com

General Chair
Charles T. McGavin, Jr.
CHE Consulting Inc.
Bus: 510-523-1184
cmgpc@cmg.org

Office
CMG Headquarters
Computer Measurement Group, Inc.
Bus: 856-401-1700
Fax: 856-401-1708
cmghq@cmg.org

Office Manager
Barbara Flemming
Computer Measurement Group, Inc.
Bus: 856-401-1700
Fax: 856-401-1708
barbara@cmg.org

Program Coordinator
David Troxel
Computer Measurment Group, Inc.
Bus: 856-401-1700
Fax: 856-401-1708
david@cmg.org

Conference Coordinator I
Michelle C. Cervantes
Computer Measurement Group, Inc.
Bus: 856-401-1700
Fax: 856-401-1708
michelle@cmg.org

Administrative Assistant
Kathleen M. Kinnarney
Computer Measurement Group, Inc.
Bus: 856-401-1700
Fax: 856-401-1708
kathy@cmg.org

Corporate Bookkeeper
Linda G. Stermer
LGS Bookkeeping Services
Bus: 805-238-2410
Fax: 805-238-2458
lgstermer@aol.com

Food & Beverage
Hugh Hunt
Hunt Conference Group, Inc.
Bus: 817-410-4660
Fax: 817-410-4661
hugh@huntconferencegroup.com

The Computer Measurement Group, Inc.

151 Fries Mill Road
Executive Campus, Suite 104
Turnersville, NJ 08012
www.cmg.org