Performance Analysis of a Website in Production Using Modeling and Simulation – Case Study

Nidhi Tiwari
Infosys Technologies Ltd
nidhi_tiwari@infosys.com

Harish Kashyap T S
Infosys Technologies Ltd
HarishKashyap_TS@infosys.com

ABSTRACT
For business and resource planning, foresight of application's performance in production is essential. This can apparently be obtained by replicating production environment and scenarios, incurring heavy expenditures. Alternately, time and cost efficient performance modeling and simulation methodology can be used. However complicated real applications and environments deter implementation of this scientific approach. Here we present case study of effectively using model simulation for analyzing a microsite's performance in production. Some of the key challenges, learning's and benefits observed during this exercise are also cited.

Categories and Subject Descriptors
I.6.4 [**Simulation and Modeling**]: Model Validation and Analysis

General Terms
Measurement, Performance, Experimentation, Verification.

Keywords
Modeling, Simulation, Performance Analysis, LQN, DES.

1. INTRODUCTION
System performance being a key factor for customer satisfaction, business stakeholders don't want to leave any loose ends on it. They seek for performance thresholds beforehand and strategize accordingly to avoid any last minute surprises from customers. Likewise, it is vital for operations team to quickly manage performance hindering situations like increase in workload etc, in production. A preview of system's performance in changing production scenarios can facilitate such tasks.

Performance testing is one of such means which assesses application's performance under various load scenarios when hosted in a given environment [5]. Commonly these tests are conducted in scaled down replicas of production environment with minimal workloads due to time and cost restrictions.

Industry track 2011, March 22-February 22

DOI 10.4108/icst.simutools.2011.245595

As workload and underlying hardware are some of the key drivers of a system's performance, any change in either of them impacts the resulting system's performance considerably [14].

Consequently performance tests' findings from different test environment do not hold true completely for production environment and load scenarios. Then, generally rule of thumb is used to establish production performance from the test results which may not be accurate and reliable for every situation. Besides, the test environment may not be available always to study any expected changes in production scenarios for a specific system, as it would be shared across organization. Under such circumstances sometime guesstimates are also used to determine the performance consequences of any changes. With use of such methods there is lot of uncertainty on system's performance in production.

Hence theoretical mechanisms of performance modeling and simulation [4] are recommended. A performance model is an abstract quantitative representation of system architecture, calibrated using the measurements from existing system [1]. Its analysis using simulation provides view of application's performance for multiple scenarios like workload changes, hardware scaling etc [10]. As system model is based on well established theories and actual system, its simulation provides more realistic results than guesstimates and rule-of-thumb. Performance modeling and simulation can thus help comprehending production performance and capacity with certain reliability in optimal cost. They can be used to predict performance of multi-tier, multi-threaded, complex architecture systems as illustrated in [11] [15] [13]. Still these methods are not popular with various industry stakeholders as they require time, effort, expertise and certain accuracy tolerance.

Now a model is as accurate as its inputs. The inputs for model creation can be broadly divided into 2 categories – component architecture for model creation and parameters for its calibration [14] [2]. There are 2 categories of parameters – workload intensity parameters and service demands [3] which are derived from various system performance metrics. Therefore, capturing adequate architecture details and valid performance metrics are significant for model creation. Obtaining these for real-life applications is challenging due to complex architectures, execution environments, distributed hybrid shared deployments etc. Moreover lack of time, resources and end-to-end technical expertise negatively impact the modeling activities. So key to get more precise insights on system performance in production using simulation lies in warily resolving such issues during modeling.

Here we present case study of using LQN (Layered Queuing Network) (see Note 1) performance model and discrete-event

simulation to find stress points for a real-life application in production. This paper also highlights some of the challenges faced during various stages and summarizes the key learning's obtained from the exercise. Further the benefits realized by client as a result of using model simulation are provided.

The remainder of this paper is organized as follows. Section 2 describes the case study of using performance modeling and simulation - methodology used, challenges faced and resolutions applied. It also includes analyses done using simulations for production and consequent benefits accrued by client. Next the key learning's from this exercise are given in section 3. Section 4 provides the summary and conclusion derived from this case study.

Note 1: Layered Queuing Networks (LQNs), an extension of queuing models, are used to model the concurrent and distributed systems. They use concept of layers to naturally model the software servers in multi-tier architectures with their nested services and calls [7].

2. CASE STUDY

A leading drinks manufacturer wanted to run online campaigns for its brands through corresponding microsites and websites. These campaigns were of utmost importance to client for endorsing the brands. Websites being meant for marketing were heavily loaded with images and flash animations. And they were required to handle high user loads. Thus their performance during the campaigns was of major concern to various stakeholders. As client had strict timelines for brands launches, existing scaled down replica of production environment was provided for performance testing of all the applications. For this case study one of these brands' microsite, performance tested under such constraints is considered.

Though the test results from scaled down environment were made available, client had further questions on the site's performance in production environment. They wanted to know the load levels that can be handled well within given response time limits by microsites. They wanted to know the stress point for current production hardware, so that capacity road map could be defined well in advance. Now for providing these required details to client, multiple what-if analyses were required to be conducted in production like environment.

So predictive performance modeling was suggested to determine the site performance in production environment. An in-house predictive simulation tool [8] was used for creating and analyzing model to get the prediction results. This tool can predict the performance of computer systems for varying load conditions, different hardware and changing deployments. It is based on concepts of Layered Queuing Networks (LQNs) and Discrete Event Simulation (DEVS). It uses xml representation for LQN components and home grown Java library for simulation to provide plug-n-play and resources addition ability [12].

Next we provide detailed process used for model and simulation based performance analysis of system.

Methodology

The steps used for doing performance metrics predictions of microsite in production environment were as follows:

- Performance testing was conducted in scaled down, production-like test environment.
- LQN performance model was created based on application architecture, deployment, transaction flow and load tests results.
- Performance model was validated by comparing its simulation results against the actual load test results.
- Performance model for production environment was derived by scaling up the test environment model. What-if analysis for various user loads was conducted by simulating the model to determine threshold number of concurrent users.

2.1 Performance Testing

The microsite was deployed in test environment as two tier application, similar to production deployment (see Figure 1). Four business critical transactions were identified for load testing and modeling based on non-functional requirements (NFRs) and client's inputs.

- Isolated load tests were conducted for each of the identified transactions with different user loads (e.g.: 30, 40, 50 virtual users)
- Transaction mix tests were conducted using the percentage mix similar to production workload patterns.
- For all load tests, individual transactions' "Response Time" and "Throughput" were obtained from load test tool. Also various servers' performance metrics like processor, memory, disk utilizations etc. were gathered from performance monitoring tool [6].

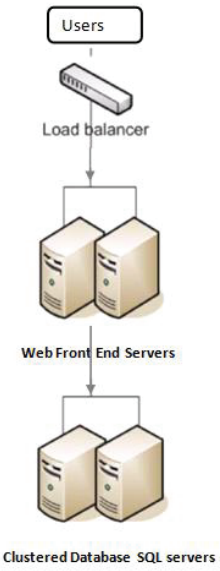

Figure 1. System Deployment Architecture

Challenges and Resolutions in Performance Testing

Like most of the real system scenarios, present environment and resources caused multiple hindrances during performance testing. Some of the constraints faced and resolution used during this exercise were as follows:

- The servers' processor utilizations were observed to be behaving inconsistently during load tests due to other websites running in same environment. Thus 'processes' related to the microsite were monitored during performance tests.
- Transactions' throughputs were coming low at times, causing inconsistencies in test results. Because the same environment was being loaded by several teams from multiple time zones. Subsequently, performance testing was conducted by identifying relatively idle time slots in environment.
- Additionally to avoid incorrect data capture, test results were thoroughly analyzed using Little's Law [14]. Tests were repeated until valid results were obtained.
- During load tests, response time of transactions was increasing exponentially though server utilizations were below 60%. Detailed analysis of performance metrics established network as a bottleneck. The pages were heavy due to images and flash files, and available network bandwidth was limited. So response was taking long time to cross network, increasing the overall response time.

2.2 LQN Performance Model Creation

For LQN model creation, understanding of the application's deployment architecture, transaction flows and service time values are required [11] [9]. The microsite's LQN model was created as follows:

- Based on deployment architecture, 2 processors (one for each tier) and 2 tasks (one for each software layer) were created in model.
- To incorporate the effect of exponential network delay found during testing, a task running on processor was added in the model.
- All 4 transactions were flowing through network, web front-end server and database server, so 4 entries (one for each transaction) were created in the corresponding tasks.
- Service demands for entries were calculated using utilization law [4] [11] from the corresponding transaction's isolated load test results.
- During performance tests, disk utilizations were observed to be very low on web front-end and database servers. Thus, service demand on both servers' disk was assumed to be negligible for all transactions and disks on the servers were not included in model.

The resulting LQN model with entries for transaction 2 is shown in Figure 2.

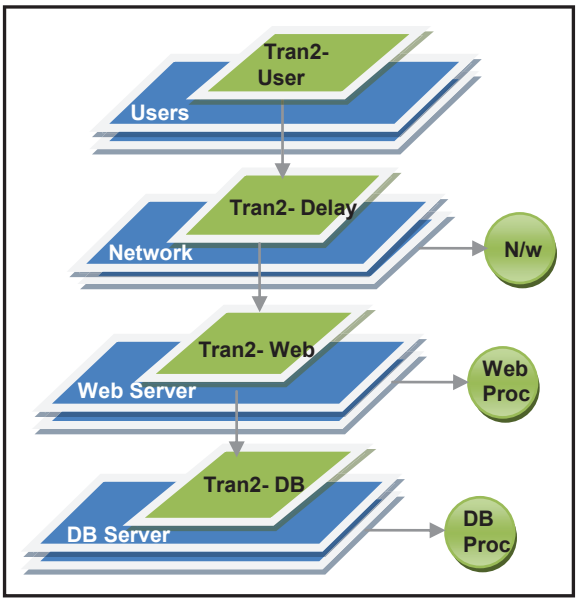

Figure 2. Application LQN model

2.3 Performance Model Validation and Refinement

Once the LQN model was created, it was required to be validated for accuracy of predictions. For this model simulation and actual testing results of isolated and mix transaction loads were compared. Key performance parameters like processor utilizations, transactions' response times and throughputs were considered.

The individual transactions' model parameters were tuned to minimize the differences in results for various isolated load levels. A validation done for 'Transaction 2' by comparing results from tuned model and test results at various load levels is shown in Figure 3a.

The comparison of 4 transactions' response time values obtained from actual mix tests and model simulations for various transaction mix loads is shown in Figure 3b.

The graph in Figure 4 shows comparison of monitored and predicted server utilizations at the two servers for various mix loads.

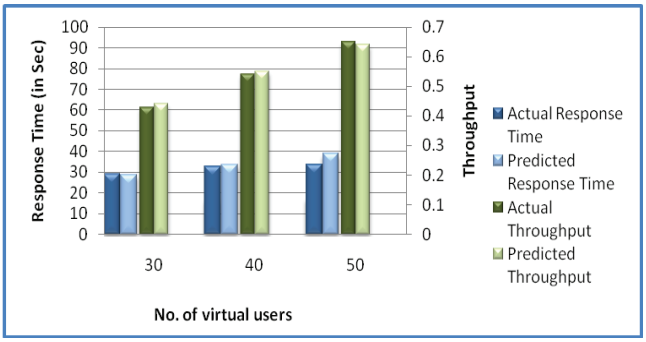

Figure 3a. Isolated 'Txn 2' response time and throughput validation

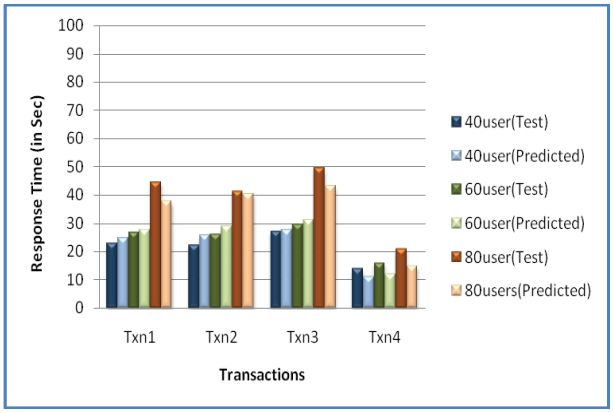

Figure 3b. Transactions response time validation

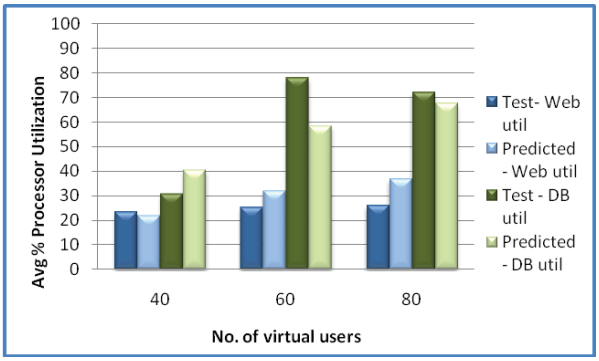

Figure 4. Web front-end server and DB server utilizations validation

Challenges and Resolutions during performance model validation

Some of the key challenges faced during model validation and resolutions used were as follows:

- As the test environment was shared, process level utilization of the web front-end and database servers were compared with the model simulation results.
- The task of measuring microsite's metrics in isolation was more difficult as single database server instance was used by multiple websites' databases. Even after multiple rounds of testing microsite's database server's process utilization could not be captured precisely. This attributed to relatively higher deviations for DB server utilization values from model simulation and actual tool.
- To reduce the impact of inconsistent test results in model calibration, the model was tuned and finalized by comparing the model simulation results against multiple data points from isolated and mix test runs.

Once the deviations of model simulation results were observed to be within given limits for all performance metrics, model was designated ready for production predictions.

2.4 Performance Predictions for Production Environment

The initial LQN model was created using results from scaled down test environment, so it was required to be scaled up for production predictions.

2.4.1. Scaling up model from Staging to Production

The production and staging environments had similar machines; however production had more number of machines with higher number of processors in each machine as shown in Table 1.

Table 1. Production and Test environment details

	Web-front End tier		**Database tier**	
	Number of servers	Number of CPUs in each server	Number of servers	Number of CPUs in each server
Test Environment	1	2	1	2
Production Environment	4	4	1	4

As similar machines were used in test and production, scale factor at machine level was determined by taking the ratio of number of processors. Using the scale factor various parameters values were updated in the model. More hardware nodes and tasks were added to depict more machines and software instances in production's LQN model. Thus based on details given in Table 1 the production LQN model had:

- 4 Web front-end server nodes and tasks, with scaled attributes from staging model.
- 1 DB server nodes and tasks, with scaled attributes from staging model.

Now the performance model was ready for doing production performance predictions.

2.4.2. What-if analyses using performance model simulation

Client's requirement was to understand microsite's performance on the given environment during campaign. They wanted to know the servers' break points, in terms of workload for capacity roadmap definition. To get these, what-if analysis i.e. study of system performance at various load levels in the production was required. So multiple simulations of production's model were conducted for increasing user loads and resulting performance metrics were studied.

2.4.2.1. Production server stress point prediction

To determine the servers' stress points in production, server utilizations predicted by model for changing workload scenarios were studied. Figure 5 depicts the processor utilizations obtained at web front-end and database server respectively for various workloads from the model simulation.

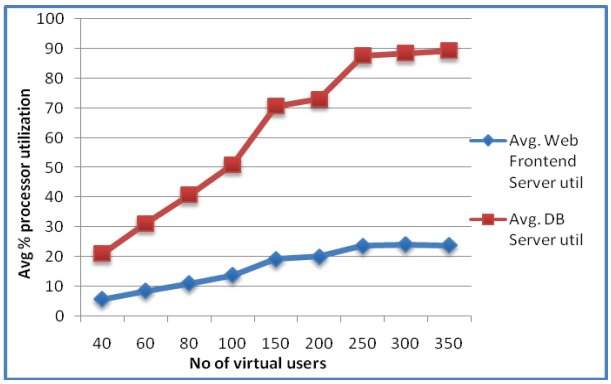

Figure 5. Predicted Web front-end server and DB server utilizations in production

Observations and Recommendations for servers' utilizations:

Some observations and recommendations provided to client based on study of servers' utilizations predictions were as follows:

- Processor utilization on web front-end server due to application's transactions would gradually increase with increase in user load.
- After 250 users, average web front-end server utilization would remain constant around 23% even with increase in load. This could be due to database server saturating at 89%.
- Database server would reach the stated threshold utilization of 40% at 80 concurrent users.
- In prediction results it was observed that database server would become bottleneck at user load of 250, so optimization was suggested at database layer.

2.4.2.2. Transactions Response time thresholds prediction in production

Performance modeling was done at transaction level where each transaction consisted of multiple pages. Total threshold response time for end-to-end transaction was based on pages accessed in the transaction. These threshold values were used for identifying workload thresholds in production environment. The graph in Figure 6 shows the transaction response time predictions from model simulation for various mix workloads.

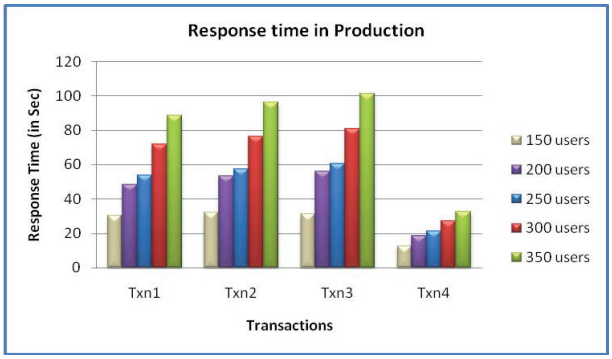

Figure 6. Transactions' response times graph for various user loads in production environment.

Observations and Recommendations for transaction response times:

Few observations and recommendations provided to client based on study of predicted transaction response times were as follows:

- Response time for transactions in production would be better than that in staging environment.
- Response times for all transactions were observed to be increasing exponentially with increase in user load.
- As the 4 transactions had different number of pages, their threshold response times were different. It was observed that 'Transaction 1', 'Transaction 2' and 'Transaction 3' would reach their threshold response times at 250 user load. A possible reason for this could be high contention at DB server at this load.
- Beyond 250 user load, sudden increase in predicted response times was observed. This could be attributed to database servers' saturation (89% processor utilization) predicted at 250 user load.

2.4.2.3. Performance predictions for system with scaled-up DB server

In production environment predictions, it was observed that DB server would be limiting the number of concurrent users in system. Such finding raised some curiosity on possible ways to push up the threshold user limits. Say if DB is becoming the bottleneck in supporting more users, will increasing the capacity at DB tier help? Will it improve response times, how will response time values get changed and so on. To address these, DB servers' capacity in model was doubled by increasing its number of CPUs to 8. Thereafter multiple simulations were conducted with increasing workloads to find servers' stress points and response time thresholds in upgraded deployment. Figure 7a depicts the processor utilizations obtained at web front-end and database server respectively for various workloads from the DB scaled-up model simulations. Figure 7b depicts the 4 transactions' response times obtained for various workloads from the DB scaled-up model simulations.

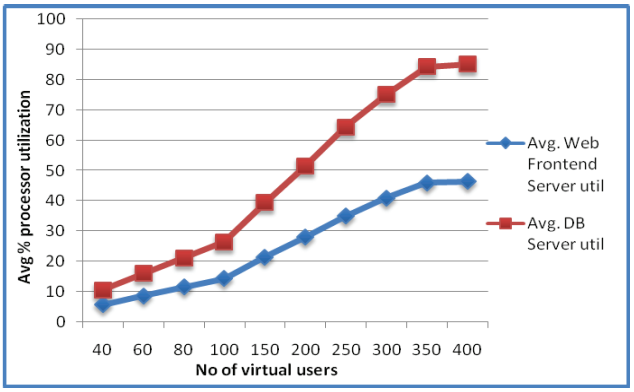

Figure 7a. Predicted Web front-end server and DB server utilizations from DB scaled-up production model simulations

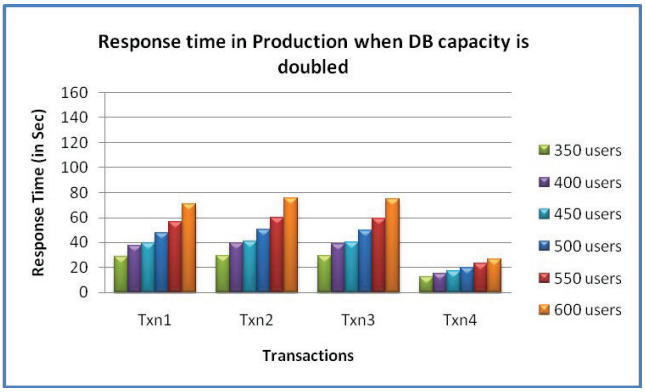

Figure 7b. Predicted transaction response times with DB scaled-up production model simulations

Observations and Recommendations for servers' utilizations and transaction response times:

Some observations and recommendations provided to client based on study of servers' utilization predictions from performance model were as follows:

- Database server would reach the threshold utilization of 40% at around 150 concurrent users. This stress point load was almost two times of 1 DB server threshold load.
- In prediction results it was observed that double capacity database server would become bottleneck at increased user load of around 350.
- At around 350 users, average web front-end server utilization would become stagnant around 46% due to database server saturating at 84%.
- Simulation results indicated that increasing the DB server capacity can remarkably improve response times' of all transactions. At a given load, response times for various transactions were almost half of their value in earlier setup as shown in Figure 7b.

Benefits obtained by performance predictions

Successful completion of performance modeling and simulation for production environment provided the required system predictability and visibility for future planning along with the time and cost benefits. Some of the key benefits obtained by client from this exercise were as follows:

- The performance prediction findings of various websites for production scenarios helped client in strategizing the campaigns roll out across different regions. As a result, brands' microsites performed as expected for high loads during the staggered campaign launches.
- Client was able to get preview of microsites' performance in production at a lesser cost. As performance testing was conducted in the scaled down production-like environment.
- High user load scenarios were analyzed using the performance model, resulting in saving virtual user license cost of the testing tool.
- Proactive bottleneck analysis done by model simulation helped to define a future capacity roadmap. The possible bottleneck scenarios in the application at higher user loads were determined and server upgrade for the application was suggested.
- Additional what-if analysis done using model simulation provided view of likely improvements that can be obtained in system performance and stress point by scaling up DB server.
- Modeling and simulation provided time efficient alternative to actual testing for understanding the application performance characteristics in production scenarios.

3. KEY LEARNING'S

Performance modeling theory has evolved over time [11] to incorporate developments in system architectures. Still hardly gets used in projects due to doubts on accuracy of its results. In addition, like any other theory, practical implementation of this also poses multiple hindrances. The accuracy of model simulation results can be improved by using holistically [3] and precisely measured application metrics as inputs. Now as performance testing is done on final stable code, its results would help to create the most accurate model. However real-life performance testing exercises have multiple challenges like shared test environment, unavailability of transaction designs, inconsistent performance test results etc. These need to be understood and resolved for using their outputs in performance model creation. Some of the learning's from our experience of implementing performance modeling from performance test results are as follows:

- To mitigate the problem of inconsistent utilization values in shared test environment, processes related to the application should be monitored during performance tests. These metrics can then be used for deriving performance model inputs.
- For consistency in performance test results, relatively idle time of the shared environment should be identified and load tests should be conducted during these idle time slots.
- Transaction sequence diagrams, application design and deployment architecture diagram should be used to understand the transaction flow through various layers in test deployment. These can then be used to create the model.
- Detailed monitoring and analysis should be used to identify, understand and model the obscure architecture elements like memory, network, disks etc., that impact systems' performance.
- For ensuring precise inputs for performance model and consequently increasing the accuracy of the simulation results, unambiguous performance test results are required. Therefore, all performance test results should be analyzed in detail for consistency and validated against basic laws of performance engineering. Performance tests should be re-executed where ever inconsistency or high deviations are observed.

4. SUMMARY AND CONCLUSION

This paper presents a case study of production performance analysis for a real-life application using model simulation. It demonstrates LQN performance model creation from performance

test results. The challenges faced and resolutions used in the course of testing and model creation are also highlighted. Resulting models' validation is shown by comparing its simulation results with actual test results. Finally the recommendations from what-if analyses of model and benefits accrued by client are given to provide a view on usefulness of performance modeling and simulation.

This case study reveals several nuances involved in model creation for a real-life application. Yet the validation results in this paper show that deviations for performance parameters obtained from model simulation can be within the acceptable limits. Further prediction results usage by client shows how foresight of production performance can be helpful in defining the deployment strategy and future capacity road map. Based on these observations we conclude that, performance modeling and simulation provides a cost and time reasonable way for production performance prediction. It also provides the edge of being proactive to maintain optimal system performance.

5. REFERENCES

[1] Daniel A. Menascé, Virgilio A. F. Almeida, L. W. Dowdy. 2004. "Performance by design: Computer Capacity Planning by Example", Prentice Hall, 2004

[2] Daniel A. Menascé, Virgilio A. F. Almeida. 1998. "Capacity Planning for Web Performance: metrics, models, and methods", Prentice Hall, 1998

[3] Daniel A. Menasce. 2008. "Computing missing service demand parameters for performance models", CMG conference 2008.

[4] Edward D. Lazowska, John Zahorjan, G. Scott Graham, Kenneth C. Sevcik. 1984. "Quantitative System Performance - Computer System Analysis Using Queuing Network Models", Prentice-Hall, Inc., 1984

[5] http://en.wikipedia.org/wiki/Software_performance_testing

[6] http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/nt_command_perfmon.mspx?mfr=true

[7] http://www.sce.carleton.ca/rads/

[8] Infosys Prediction Tool & Casper. 2010. http://www.infosys.com/newsroom/features/Documents/Infosys-RADIEN-CASPER.pdf, 2010.

[9] Murray Woodside. 2003. "Layered Resources, Layered Queues and Software Bottlenecks: A tutorial", Performance Tools 2003 conference, Sept 2, 2003.

[10] Nidhi Tiwari, Kiran C Nair. 2010. "Performance extrapolation that uses industry benchmarks with performance models", Performance Evaluation of Computer and Telecommunications Systems (SPECTS), 2010 International Symposium on, July 2010.

[11] Nidhi Tiwari, Prabhakar Mynampati. 2007. "Experiences of using LQN and QPN tools for performance modeling of a J2EE Application", CMG conference 2007

[12] Nidhi Tiwari, Pratik Kumar, Tapaswini Patra. 2010. "Extending LQN model representation for plug-n-play of architectural components", CMG Journal of Computer Resource Management, Issue 126, Spring 2010.

[13] Nikolaus Huber, Steffen Becker, Christoph Rathfelder, Jochen Schweflinghaus, Ralf H. Reussner. 2010. "Performance Modeling in Industry: A case study on Storage Virtualization", SIMUTools conference 2010.

[14] Raj Jain, 1991. "The Art of Computer Systems Performance Analysis", John Wiley & Sons, Inc, 1991

[15] Tuli Nivas. 2007. "Using simulation to forecasting performance: A case study", CMG conference 2007.