Design Validation of Service Delivery Platform Using Modeling and Simulation

Tony Ingham, BT

Sandeep Rajhans, Dhiraj Kumar Sinha, Kalyani Sastry, and Shankar Kumar, Tech Mahindra Ltd.

ABSTRACT

The value of models as tools to design and plan networks, aid the decision-making process, and as methods to conceptualize abstract ideas is well known. Models play a key role in evaluating varied design options and assist in discovering design issues that likely will impact performance. In the service delivery platform, spreadsheet models can be applied to learn about strategic traffic volume, estimate end-to-end latencies and box capacities, whereas more complex simulation models can help to clarify an understanding of the dynamic aspects and trade-offs that influence the end-user experience. This article discusses how modeling and simulation effectively help validate the design of various components constituting the service delivery platform.

INTRODUCTION

Design validation of a system refers to the act of justifying the rationale for a particular choice of design for that system. In complex systems, where the number of entities interacting with one another is high and the variables influencing system behavior is higher, the impact of various design options on the end-user experience is difficult to predict. Large telecommunication applications running in real-time environments must have high, predictable, and scalable performance.

Service delivery platform (SDP) is an architecture used in the telecom domain for *rapid development and deployment* of new converged media services from the plain old telephone system (POTS) to the latest Internet Protocol television (IPTV) or triple-play services.

A post-implementation, dissatisfied user experience will result in an adverse impact on the business of a service provider. Also, after the design is implemented, performance problems seldom can be fixed by adding or modifying functionality (although caches are a counter-example); and generally, the solution lies in redesign, thereby delaying the service launch. Therefore, design validation is particularly important to ensure that performance problems are detected early in the design stage of an SDP.

The focus of the work on performance is the establishment of suitable performance metrics and building confidence that system design meets the targets expressed in terms of these metrics. In the context of an SDP, this means very early in the design cycle justifying that:

- The expectations for end-user response times, throughputs, concurrency, and availability are defined and realistic.
- The optimal design choice for the restructured and transformed framework is made.
- The design will result in acceptable latencies, transaction accept-reject ratios, queue lengths, wait times, server utilizations, and so on.
- The design is scalable, available, and reliable.

SDP: DEFINITION AND DESIGN ISSUES

SDP solves the age-old problem of delivering telecom services in silos. SDP enables rapid development and delivery of new services from basic phone services to complex audio/video conferencing for multiplayer games, through a reusable, common capabilities-based architecture. An SDP typically consists of a service-control environment, a service-creation environment, a service-orchestration and execution environment, and abstractions for media control, presence/location, integration, and other low-level communications capabilities.

Some example services offered by an SDP are as follows:

- Wireline or wireless users can view incoming phone calls and online instant-messaging buddies or locate their friends from global positioning system (GPS)-enabled devices on their television screens.

0163-6804/09/$25.00 © 2009 IEEE

Sensitivity analysis is used to determine how sensitive a model is to changes in the value of the parameters of the model and to changes in the structure of the model. Sensitivity analysis helps build confidence in the model by studying the uncertainties associated with model parameters.

- Users can order video-on-demand services from their mobile phones or watch streaming video at home or on mobile phones.
- Airline customers can receive a text message from an automated system regarding a flight cancellation and then, opt to use a voice self-service interface to reschedule.

The key SDP design issues are outlined below.

END-TO-END RESPONSE TIME

The examples described above illustrate how the responsiveness of the SDP architecture is important for the value of the service to be realized. A text message regarding a flight cancellation must be delivered in real time and cannot get delayed beyond stipulated time. Similarly, video cannot have long intermittent delays. Considering that SDP components work together to provide a variety of distinct services; the impact of random synchronous and asynchronous component interactions on end-to-end response time must be ascertained and evaluated at the design phase. Setting realistic response-time requirements is as important as achieving them.

ARCHITECTURE CAPACITY AND SCALABILITY

The system design must ensure it meets the forecasted business transaction volume. Design must be scalable to gracefully cope with the load arising from a sudden increase in concurrent user requests. The ability of SDP components to carry the required load and scale to transient load conditions must be understood during platform design.

END-TO-END PERFORMANCE ASSURANCE

The SDP uses a common architecture to deliver different services but must ensure that each of these services meets its respective service level agreements (SLAs). For example, an IPTV service and a voice service have distinct SLAs. Engineering the shared components of the SDP to meet a variety of disparate SLAs is a challenge.

INTEROPERABILITY BETWEEN DIFFERENT COMPONENTS

The SDP typically consists of disparate components from different third-party vendors. The solution architect must be concerned with the interoperability of different interfaces and technologies, as well as the performance issues.

DEPLOYMENT DESIGN CONFORMANCE

A deployment design describes the hardware platform and interconnecting underlying network. A deployment designer must address questions about the number of servers and CPUs; types of CPUs; memory requirements; server architecture, for example, for clustering or virtualization; load balancing; and resilience before a service goes live. Because the components are shared and reused across dissimilar services, sizing the SDP platform optimally is a challenge.

All these issues indicate the requirement to employ sound design-validation techniques that aid informed, design decision making by highlighting upfront the effect of the proposed design on end-user experience during the design phase itself — prior to system implementation — thereby, increasing the degree of confidence in the proposed design. The next section discusses different approaches to address design validation and then elaborates on the modeling and simulation techniques.

DESIGN VALIDATION TECHNIQUES

There are a number of design validation techniques. Some of them are discussed below.

PROTOTYPING

When there is uncertainty as to whether a new design actually does what it is designed to do, a prototype is built to test the performance of the new design before starting industrial-strength production. Sometimes, a variant called [[rapid-prototyping]] is used for the initial prototypes that implement part, but not all of the complete design. This enables rapid and inexpensive testing of the parts of the new design that are most likely to have problems, solving those problems, and then building the full design. Prototyping results in a quick working model of the end system, provides an early view of the performance of the final system, and points to areas of improvement in the design.

SENSITIVITY ANALYSIS

Sensitivity analysis is used to determine how sensitive a model is to changes in the value of the parameters of the model and to changes in the structure of the model. Sensitivity analysis helps build confidence in the model by studying the uncertainties associated with model parameters (e.g., server processing time, queue lengths, wait time, etc). Many parameters in sensitivity-analysis models represent quantities that are very difficult, or even impossible, to measure with a great deal of accuracy in the real world. Also, some parameter values change in the real world. Therefore, when building a sensitivity-analysis model, a designer is usually uncertain about the parameter values he or she selects, and he or she must use best estimates.

Sensitivity analysis enables the designer to determine the level of accuracy that is required for a parameter value to make the model appropriately useful and valid. If the tests reveal that the model is insensitive, then it may be possible to use an estimate rather than a value with greater precision. Sensitivity analysis also can indicate which parameter values are reasonable to use in the model. If the model behaves as expected from real-world observations, it gives some indication that the parameter values reflect, at least in part, the real world.

FAILURE MODE EVALUATION AND CRITICALITY ANALYSIS

Failure mode and effects analysis (FMEA) and failure modes, effects, and criticality analysis (FMECA) are techniques used to identify potential failure modes for a component or process, to assess the risk associated with those failure modes, to rank the issues in terms of importance, and to identify and perform corrective actions to address the most serious risks.

FMECA is a technique used to identify, prioritize, and eliminate potential failures from the system design before they reach the customer. It helps in selecting design alternatives with high reliability and availability during the early design phase. It also helps develop early criteria for test planning and requirements for test equipment.

FMECA is a structured and reliable method for evaluating hardware and systems, and the approach makes evaluating complex systems easy. However, it is a tedious and time consuming process, unsuitable for multiple failures, and prone to human errors.

SIMULATION MODELING

A model is a simplified representation of a system at some particular point in time or space intended to promote understanding of the real system. A simulation model operates on time or space to compress it and enables one to perceive interactions that would not otherwise be apparent because of their separation in time or space.

Simulation modeling is a process for developing a level of understanding about the interaction between parts of a system and the behavior of the system as a whole. Dynamic interaction between reactive system components can be validated through simulation models. The objective of models built for design validation is to explore hidden facets of how the design under study impacts performance.

For results from a well validated and calibrated model to be useful, they must be timely and relevant to the design phase of the system. Only a true representation of the system can be relied upon to provide indications with respect to the real system and to help discover bottlenecks within the design that must be understood, and options to correct these bottlenecks can be evaluated again through the model.

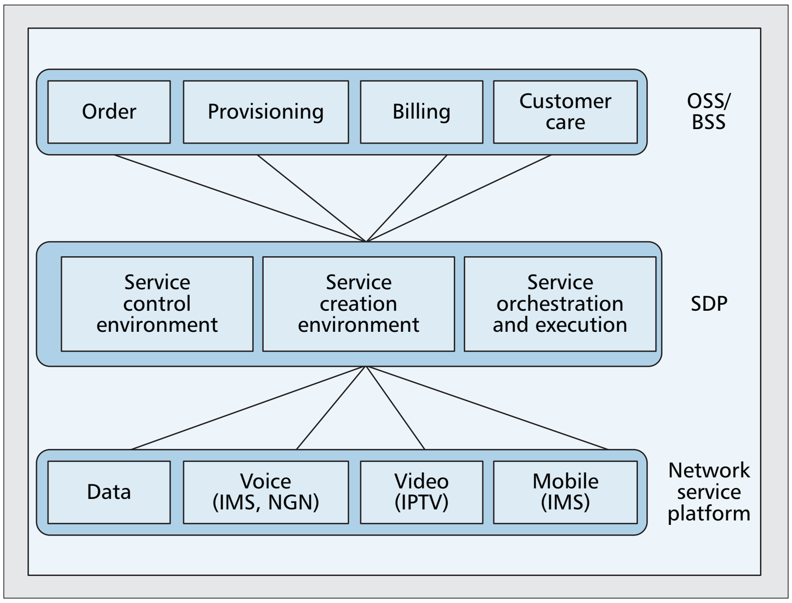


Figure 1. *Service delivery platform reference architecture.*

MODELS FOR SDP DESIGN VALIDATION

The formal approach to design validation involves both specification and verification of the system design. In SDP, static models like latency models, volumetric models, capacity models, and availability models are used to validate SDP design requirements as described below.

Latency models help in validating end-to-end response-time expectations for various transactions across platform interfaces. Sequence diagrams and network deployment designs serve as input into latency models. The impact of component latencies on end-to-end latencies for different transaction flows and co-location options can be understood using these models. These models help designers and system developers obtain a view of the upper bound and lower bound for the latency budget of each component in order to meet the end-to-end response-time requirements.

Volumetric models use product forecasts from business and convert them into transactions-per-unit time that the design must support at each interface and component, even before design commences. These models are prototypes that provide a quick view about capacity and scalability requirements in terms of throughput for each component in the SDP. This helps deployment designers, for example, to use high-speed servers for highly loaded components and to incorporate resilience into their designs appropriately. These models also provide direction to performance test design.

Capacity models document resource utilization for various activities at different servers, based on industry benchmarks and vendor performance test results, and aid in identifying how many servers of a particular type are required for a mix of transactions as specified by the volumetric model. The capacity model can help answer questions about which component gets overwhelmed first and at what load. This is useful insight for deployment designers.

Availability models provide a view of the hardware availability of a design option providing validation of resilient design, fall-back policies and failover, and recovery SLAs.

Dynamic simulation models are applied to discover, quantify, and explain the impact of random transaction arrivals, queuing disciplines and deployment options, and overload-control mechanisms on end-user experience. Queue length versus response-time trade-off, selective co-location versus wide area network (WAN) latency impacts, transaction rejects versus transaction leak rates, and leaky bucket depths and server-thread pool limits can be evaluated and quantified to aid scientific decision making in the design phase.

Network deployment designs, end-to-end transaction-sequence diagrams, and forecast transaction volumes feed as input into the simulation models. Most models that are built early during the design phase use industry benchmarks, design assumptions, and/or educated guesses for service times to provide an early view of the performance bottlenecks.

When these models are calibrated with inputs from performance testing and/or in-life system monitoring, the accuracy of results improves,

Deployment options colour coding in graph below	Percentage of requests meeting SLA of 1 s	Percentage of requests meeting SLA of 2 s
Scenario A1	0%	68.38%
Scenario A2	0%	0%
Scenario B	0%	99.21%
Scenario E	0%	95.61%
Scenario E2	0%	0%

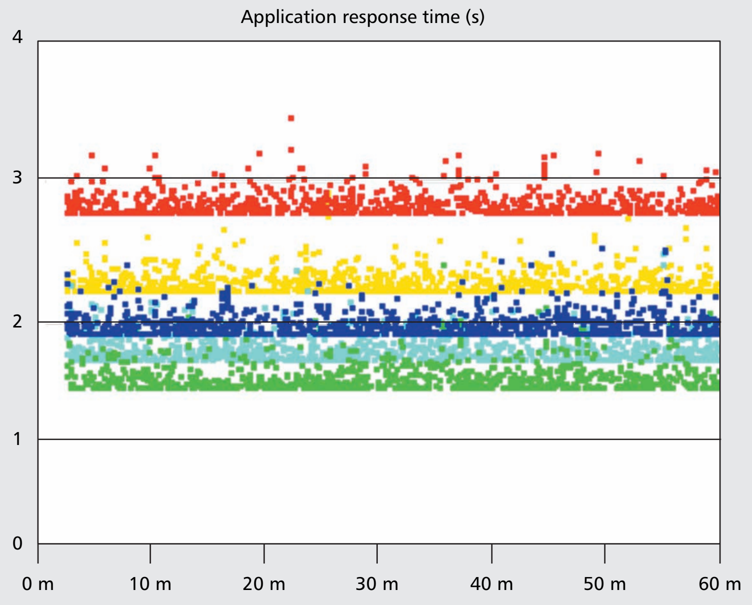


Figure 2. *Quantifying differences in end-to-end response time using five different deployment scenarios to aid in deployment option decision making.*

and the model can be used to validate full deployment scalability of the design and can aid in-life capacity planning.

Real Life Example: Results and Analysis

In this section we discuss the real-life case study of a design validation of the SDP of one of biggest telecom service providers in Europe. The service provider is in the process of implementing SDP to deliver its twenty-first-century next-generation network and application services. All the services are provisioned, delivered, and assured through a common capability stack, built using commercial off-the-shelf products to guarantee equal customer experience and faster and simpler service delivery.

The SDP simulation model started at a high level with two capabilities and one service and grew to evaluate scalability of more than six capabilities and seven services. The full-scale and enhanced model consists of more than 200 servers and network elements. The model simulates application interfaces such as remote method invocation (RMI)/remote procedure calls (RPCs) to understand concurrent invocations, priority-queuing mechanisms, and overload-control mechanisms at application interfaces, namely the leaky bucket and the token bank. The model is based on use cases and business processes reflecting provisioning and service execution flows on the SDP. Transaction rates are derived from normal and busy hour forecasts for various business transactions. The model has been used to evaluate scalability up to 200 transactions per second (tps).

Validating Deployment Design

Deployment designers faced the requirement to host a set of common components across a set of geographically separated data centers. They were keen to understand which set of components should be co-located at the same site to optimize response time from the end user perspective.

There were five different component co-location options with the designers. The network design for all five options was fed into the simulation model, and the transaction flows were run through the model at forecasted transaction volumes for year one after product launch. Component latencies were not required to be accurate for this model; these could be set to anything, so long as component latency per incoming request remained the same in all scenarios. In this model run, however, best possible values for component latencies were configured so that SLA quantifications could be made.

The number of network hops to and from data centers that the end-to-end transaction journeys require for each design option has an influence on the end user response time in each scenario.

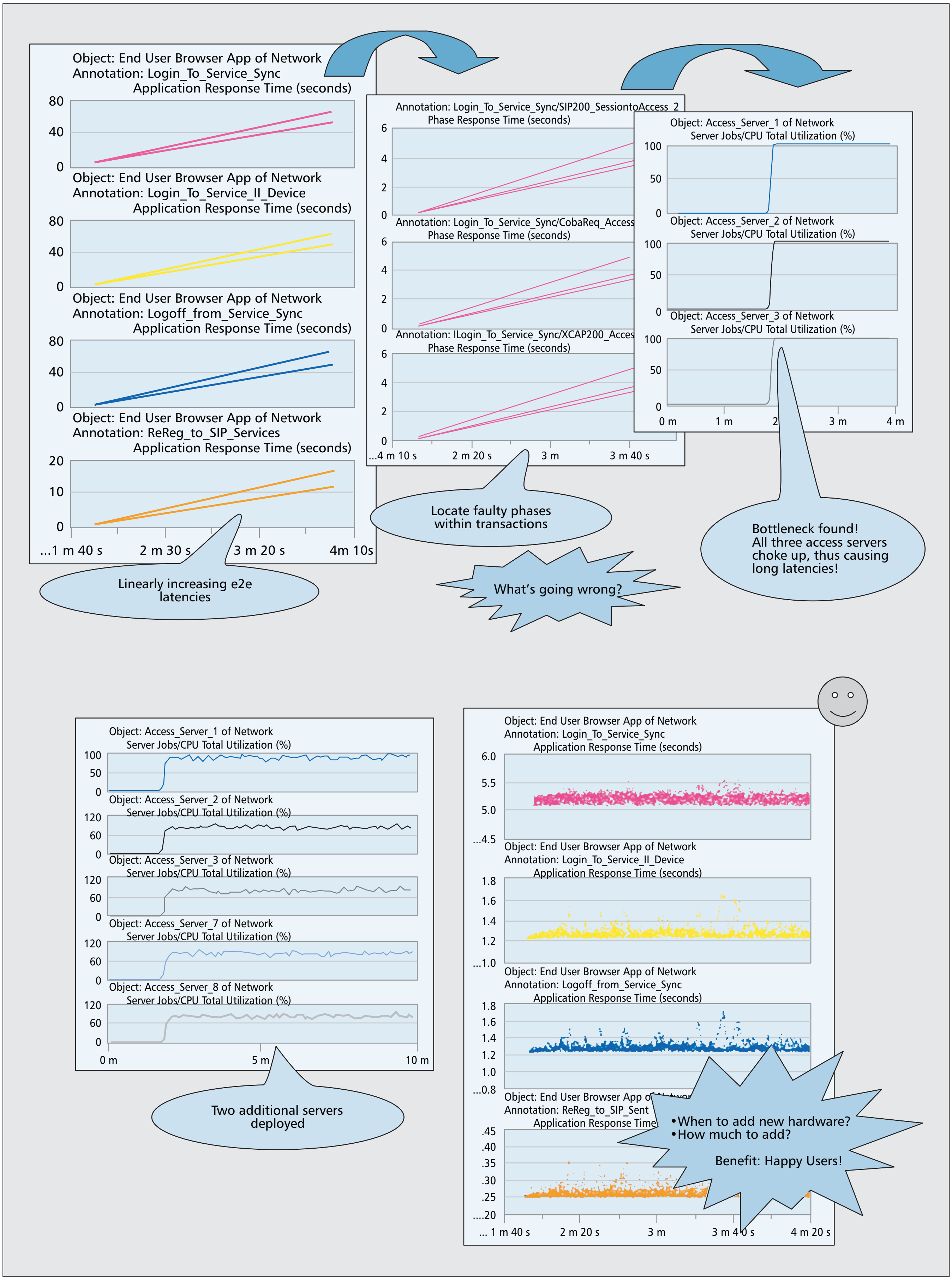
The result of these five scenarios is summarized in Fig 2. We can see that scenario A2 is the worst option, and scenario B is the best. None of the scenarios meet the SLA of 1 s (if component latencies were as per those configured in this model run), whereas in scenarios A2 and E2, all transactions fail to achieve an SLA of 2 s as well. Further, in all scenarios, there are at least a small percentage of requests outside the 2-s SLA.

Validating Platform Capacity

The capacity planner was keen to understand how long the current platform can provide acceptable end-user experience and what proactive steps should be taken to prevent unacceptable SLAs.

Volumetric forecasts from the volumetric model feed into the simulation model. All major transaction flows are modeled. The network design remained constant in all scenarios in this case. Each scenario simulated a forecast load from a different year. In this case, reasonably accurate component service times must be configured in the model to ascertain a realistic breakpoint. Therefore, information in the high-level capacity models also feed into the simulation model. It is interesting to note the synergy of three models working hand-in-hand here.

The results of the simulation model run for the year 10/11 forecast load is shown in Fig 3. All transaction response times are seen increasing linearly as simulation progressed at constant



■ **Figure 3.** *Identifying that latency will begin to increase in the future due to high traffic resulting in overutilized servers, aiding capacity planning.*

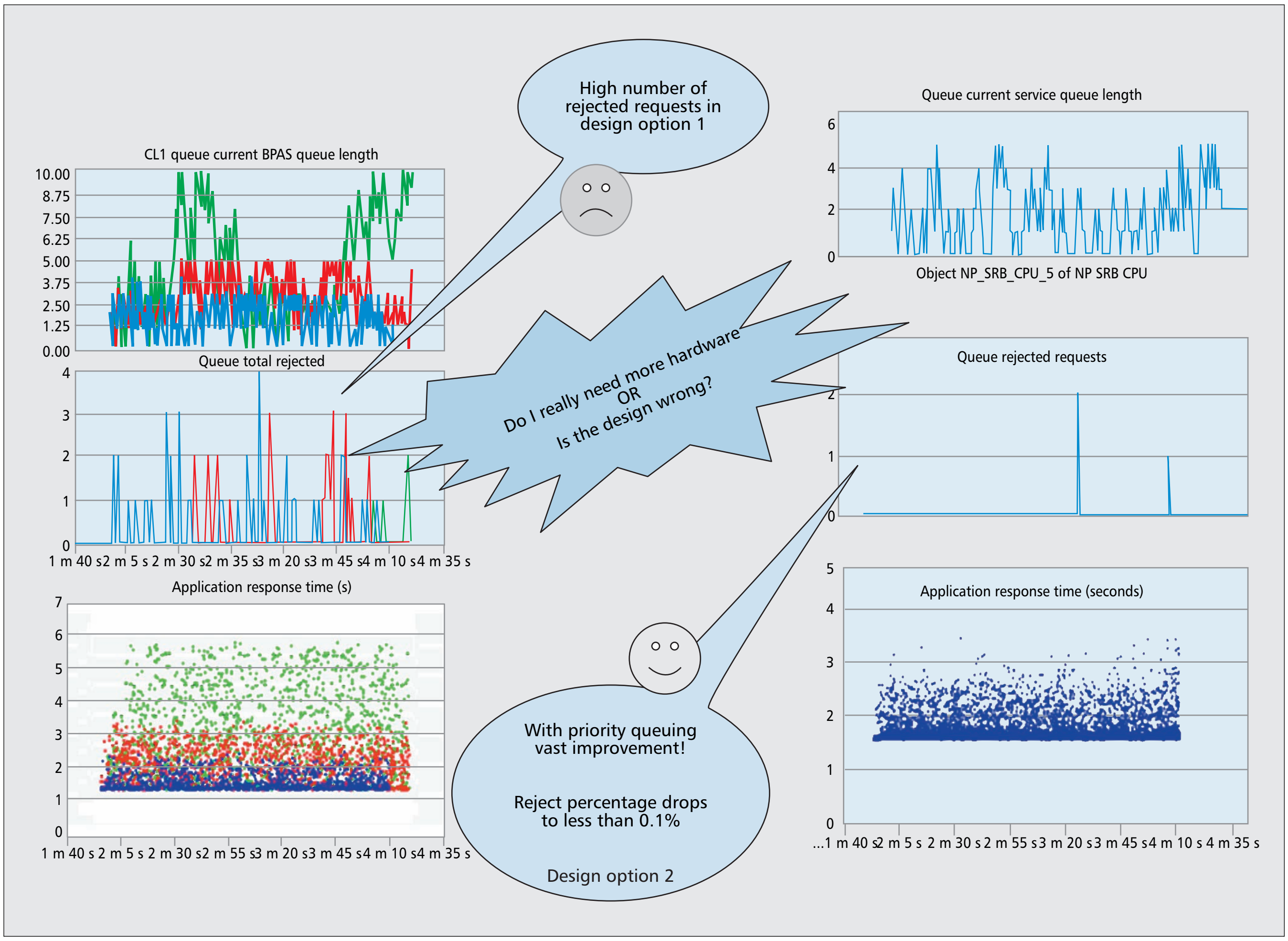


Figure 4. *Evaluating the impact of two different queuing disciplines on the percentage of rejects, aiding informed decision making for queue design implementation.*

load in the first graph above. Drilling down revealed that all subtransactions interacting with Access Server 1, 2, and 3 were facing queuing delays at the server. Further analysis revealed that these three access servers were congested. Thus, an estimate is gained about the transaction rates at which capacity breakpoints are reached. Further, the simulation model pinpoints the exact system components that are congested.

A few additional simulation runs helped determine that the addition of a few servers solves the latency problem at least temporarily, by removing the CPU bottleneck.

Based on the level of information provided to a system model as input, the models provided insights about resource utilization of that system, amount of data received and sent (throughput), and protocol-specific statistics.

However, note that the underlying application itself must be scalable for these model results to hold good. The next example shows how an application-layer interface could become a bottleneck.

Validating Interface Design

Finite state machines were used in the SDP model to evaluate the impact of varied queuing disciplines, to aid design decisions about optimal settings for session inactivity timeouts, and to justify the synchronized effectiveness of overload-control schemes of various systems when the systems are subjected to overload. In one particular case, solution designers were modeling two distinct design options at a particular interface of the solution. Their aim was to understand the impact of queue lengths on downstream applications, transaction reject percentages, and end-to-end response times in both scenarios. The simulation model assisted in evaluating two different queuing disciplines and the impact on the percentage of requests that were rejected due to a full queue.

The result of the simulation run is shown in Fig. 4. The design option on the right showed a remarkable improvement, in that it shows that less than 0.01 percent of the requests that were rejected in option 1 were rejected in option 2, and end-to-end response times also were reduced.

Thus an interface design validation using simulation helps evaluate the best option for a given traffic pattern. An incorrect design option at the application interfaces, if selected without validation, could result in a poorly scalable implementation and increase hardware and maintenance costs for the business in the medium to long term.

ACHIEVEMENT AND BUSINESS BENEFITS

ACHIEVEMENTS

The achievements are related to the real-life SDP case study discussed in the Results and Analysis section above. Results foreseen through the model have been used to direct and focus performance testing onto the critical components.

Despite using design assumptions, the model pointed out issues that later also were identified in performance testing — as a result, modeling has been used to validate designs before commencing development for future releases. This model has over 400 systems in it. Modeling is considered as the only way that so many instances of all the systems can be studied, and various what-if scenarios can be analyzed prior to testing and deployment of the system. Simulation has become part of business practice and is relied upon for providing pointers to impending performance bottlenecks and design changes.

BUSINESS BENEFITS

The long-term business benefits that the service provider deploying the SDP realized are as follows:

- Reduced risk for deploying applications that failed to meet the performance requirements.
- Reduced post deployment re-work efforts by 15 percent after using the model to proactively identify issues.

CONCLUSION

The modeling and simulation technique helps to validate proposed system design changes even before the development and deployment of services, thus providing long-term, cost-saving benefits. All the re-work costs can be avoided by performing design validation. The models help to concentrate on the areas of concern or potential risks by determining the level of system abstraction and/or depth. The what-if scenarios help provide parameter tuning for real systems. To conclude, using the simulation and modeling technique for design validation can proactively identify and minimize the non-functional risk of the platform or solution.

The modeling and simulation technique helps to validate proposed system design changes even before the development and deployment of services, thus providing long-term, cost-saving benefits. All the re-work costs can be avoided by performing design validation.

ACKNOWLEDGMENT

We would like to thank Jas Gill, 21CN CC Service Champion — Communications, BT Wholesale, and Ranjit Surae, CC Service Management Consultant — BT Wholesale, for providing us with the funding and the opportunity to apply design validation using modeling and simulation on the service delivery platform. Our thanks to Dr. Gowri Krishna, VP, NDE, Tech Mahindra Ltd. for encouraging us to come together, write this article, and share our experience with the wider industry. We would fail our duty if we do not acknowledge Dr. Mandaar Pande, Principal Performance Consultant Tech, Mahindra Ltd. We thank him for the coordination and support we received throughout the writing of this article.

BIBLIOGRAPHY

[1] T. Ingham, J. Graham, and P. Hendy, "Engineering Performance for the Future of Intelligence," *BT Tech. J.*, vol. 23, no. 1, Jan. 2005

[2] R. E. Gantenbein and S. Y. Shin, "A Practical Method for Design Validation of Critical Software Systems," *CrossTalk, J. Defense Software Eng.*, vol. 8, no. 8, Aug. 1995, pp. 23–26.

[3] R. Aberg and S. Gage, "Strategy for Successful Enterprise-Wide Modeling and Simulation with COTS Software," *AIAA Modeling Simulation Tech. Conf.*, Aug. 16, 2004.

[4] W. Damm and M. Cohen, "Advanced Validation Techniques Meet Complexity Challenge in Embedded Software Development," *Embedded Sys. J.*, 2001.

[5] N. W. Macfadyen, "Performance Modeling," *BT Tech. J.*, vol. 21, no. 2, Apr. 2003.

BIOGRAPHIES

TONY INGHAM graduated from Cambridge University, United Kingdom, with an M.A. in mathematics in 1987 with statistics, probability, and stochastic processes as major fields of study. He joined the BT Performance Engineering Department in 1987, and has worked on a number of major projects involving network routing and dimensioning, signaling, call control, intelligent networks, and service execution, with a focus on performance design. He is currently head of Shared Services for the Service Execution Platform in BT Design. He has co-authored a number of papers, including "Network Intelligence — Performance by Design" (*IEICE Transactions on Communications*, Feb. 1997) and "Engineering Performance for the Future of Intelligence" (*BT Technology Journal*, Jan. 2005).

SANDEEP RAJHANS acquired a Bachelor's degree in electronic engineering from the University of Pune, India, in 1994. His major field of study included communications and microprocessors. He is currently based at Hoffman Estates, Illinois, working on a project with a U.S.-based telecom service provider as principal consultant. His current job involves research and consultation in application performance engineering and performance modeling using simulation modeling techniques. He started his career as a network engineer in 1994 and handled various techno-commercial responsibilities. He joined Tech Mahindra Ltd., Pune, in 2004.

KALYANI SASTRY graduated from the University of Pune with an M.Sc. in computer science in 2002 and joined Tech Mahindra Ltd. in 2003. A Microsoft Certified Professional in VC++, she started her career in software development using C and C++. At Tech Mahindra, she has contributed to and led simulation modeling studies using OPNET Modeler, which include enhancing the GPRS model followed by numerous modeling studies for the BT Service Execution Platform. She is currently a performance design consultant to BT. She co-authored the paper "GPRS Model Enhancement" published at OPNETWORK 2004.

SHANKAR KUMAR earned a Bachelor's degree in telecommunication engineering from B.I.T, Bangalore, India in 2004. He is currently working on the BT Service Creation Framework as a performance consultant. He started his career as a network engineer in 2005 with Tech Mahindra Ltd., Pune. He has experience in network and application performance engineering and management, capacity planning, and performance modeling using simulation modeling techniques.

DHIRAJ KUMAR SINHA earned a Bachelor's degree in computer science and engineering and a Master's degree in information technology with specialization in computer networks from Symbiosis (SCIT), Pune, in 2004. He is currently based in San Diego, California, and is working as a network design consultant with a U.S.-based telecom service provider. He started his career as a network engineer in 2003 with Tech Mahindra Ltd., Pune. He has experience in network planning, design, and engineering, and network and application performance management.