Karan Singh

Web Application Performance Requirements Deriving Methodology

Helsinki Metropolia University of Applied Sciences

Master's Degree in Information Technology

Master's Thesis

17 June 2016

Author(s) Title Number of Pages	Karan Singh Web Application Performance Requirements Deriving Methodology 57 Pages
Degree	Master´s Degree
Degree Programme	Information Technology
Instructor(s)	Auvo Häkkinen, Principal Lecturer

This Master's Study evaluates the possibilities of improving the existing service model within a shared service of the case company. It proposes the new methodology and a series of supporting documentation to improve the existing process of performance requirement gathering.

The main goal of this study is to find the possible way(s) to improve the quality of the followed requirement gathering process in the case department of the case company. Secondly, to provide the supporting documentation for performance engineers of the department to help them in deriving the effective performance requirements well as design effective workload models.

As an outcome of the study, a new method to gather requirements and further derive the test types is proposed. The new methodology implemented during study in a case project, shows improvement in the quality of the gathered requirements and hence in quality of the performance test results.

The proposed methodology helps management to understand the benefits of the new method and a possible way to improve the area that was lagging earlier. The small improvement in the requirement gathering stage leads to major gains in the later stages of the performance testing and better prediction of the production performance.

Keywords	Performance Testing, Requirement Gathering, Performance Requirement, Log Analysis, Non-Functional Requirement Gathering

Contents

1 Introduction

Performance is one important attribute of a software system. Failing to provide the expected performance level might make the system unusable, which means a possible rejection by all users over time. With the rapid growth in IT application usage, expected to continue growing exponentially in near future, performance of any given software, system or network would key factor in its success or failure.

To develop and test a web application of acceptable performance it is crucial to define its precise performance requirements, which could lead to effective prediction of the future performance of the software system, but unfortunately often these requirements are not defined effectively.

Businesses are usually concerned about their ability to meet customers' performance requirements. Nevertheless, while it is widely recognized that it is necessary to have functionality-testing program with clear functional requirements in place, it is not unusual to find that a project has no explicit provision of performance testing or missing the application's non-functional requirements. At some stage, normally when they start facing the performance issues in their production environments, these projects start to investigate the possibilities to undergo performance testing and hence troubleshoot the bottleneck.

In the current scenario in IT organizations or IT department of various organizations, performance testing is commonly offered as a shared service across the IT department, it is run by a dedicated team of performance test engineers using market available or open source tools for example HP Performance Center and Jmeter etc. It is the same case in the IT department of the case company. The involvement of the performance test expert is restricted to a small duration of time as compared to the complete project development life cycle and in most cases, they possess minimal functional knowledge of the application.

Presently, performance testing requirements are gathered using the service request document and sometimes based on few generic checklists. There is no defined process and supporting documentation to assist the performance test engineers to collect the performance requirements and further develop an effective load model to effectively predict

the performance of the production system. The main challenges are with the projects with no previously defined or ineffective defined performance requirements of the application.

Due to other constraints such as restricted project timelines and budget, often the quality of the performance requirements is not good enough to predict the performance behaviour of the application or to identify the performance bottleneck. In most of cases, these defined requirements are not supported by sufficient data. To tackle this problem, a defined process to assist such projects is a missing link in overall performance engineering service offering by the concerned organization.

So clearly, there is a need of a process to be in place to define and collect precise performance requirements. This area could be further investigated in order to define an organizational process to define and collect the effective performance requirements of a software system.

Therefore, the objective of this research activity is to understand how effective performance requirements could be derived and most relevant test cases could be identified from the historical usage data and patterns to carry out effective performance testing to accurately predict the future performance of the IT web applications.

1.1 Scope of Study

This study presents a practical approach to identify the performance requirements, identifying the suitable test workflows and designing the workload model for performance testing for web applications and excludes any other type of IT application. An appropriate load model creates the backbone of an efficient performance test. There are many possibilities to design a workload, but the main challenge is to find the most efficient workload model, which accurately predicts the performance of the application in production.

1.2 Structure of Study

This section presents the highlights of the chapters and discusses the relation between their contents.

- Chapter 1 introduces the topic of the study by highlighting the problem statement. It also includes the scope and structure of the study.

- Chapter 2 provides the detail of the project and expected outcome details. It includes the overall research process for example steps involved in achieving the expected outcome apart from the details of the materials used for example data used and details of its collection method.

- Chapter 3 includes details of relevant theory in the area of performance testing as well as requirement gathering. It covers the most common used test types, their definition and the theory around workload modelling.

- Chapter 4 includes the initial state analysis of the case department i.e. a shared performance testing service as well as the initial state analysis of case project on which the new method of requirement gathering is implemented and results are captured in the later stages. This also include the requirements set for the study.

- Chapter 5 provides the details of the new methods proposed for requirement gathering as well as details of the related documents developed and used in the case project.

- Chapter 6 provides the details of the case project, which is a single sign on system. It includes the test tool and test setup details apart from the details of the requirements gathered and the tests derived for the case project. It includes the test results obtained during the case study.

- Chapter 7 provides the result summaries of the old and the new projects apart from the comparative analysis of the results to understand the effectiveness of the new methodology.

- Chapter 8 presents the summary of the study

2 Methods and Materials

This section includes the details of the methods being followed during the study as well as includes the various details of the materials used during the study by providing the reference to various stages of the overall research process.

2.1 Research Approach

This study presents a methodology to define a web application's performance requirements and deriving the test workflows in order to carry out an effective performance testing for the systems, which do not have their predefined non-functional requirements. It involves a practical approach to identify study requirements and the test workflows based applications on historical usage data by identifying and filtering the salient attributes from web application log.

The scope is restricted to IT web applications, which are predominantly used over any other type of IT applications.

2.2 Research Process

This section describes the overall research process and includes the various details of the activities performance during different stages of the process for example current state analysis stages primarily includes the details of the activities performed to understand the existing methods and processes being followed in the case company. Following is the details of various stages:

(I) Current State Analysis

The present state is analyzed initially to measure the effectiveness and the drawbacks of the existing system. In most of the cases, as mentioned in the introduction, the performance requirements are gathered using the questions based on the experience of the performance testing expert and in some cases generic checklists are also employed. In most of the cases these checklists are common for all types of IT applications and do

not lead to very effective questions. It also means that the quality of the gathered requirements is uncontrolled and also dependent on the experience level of the performance tester.

Current state analysis includes the conclusions based on the interviews of various performance testers, system designers/architects and system/business owners involved in performance testing of web application at some stage.

The existing best practices are identified based on the aforementioned interviews and also from the academic databases.

(II) New Methodology Development

This phase includes the identification of the Key Requirement Indicators for web applications in order to carry out its performance testing for example peak number of parallel sessions, average session length etc. and their identification methods from historical data It would include the implementation of various tools or possibly developing a new tool to achieve the end goal.

Defining the key KPIs of web applications performance testing workflows and their identification methods from historical data for example most access objects, business criticality etc.

(III) Implementation and Testing the Proposed Methodology

In this stage, the proposed approach to gather the requirement is implemented on the suitable project, new application without predefined performance requirements. The effectiveness of the new method to derive the workloads for performance testing and the quality of the results is compared with the old methodology. The comparison is based on the overall time taken in requirement gathering and the percentage of the objectives met in the previous project with similar complexity

2.3 Data Collection

The following data is used during the study. The details of the sources of the data are also specified below:

- Existing Knowledge (from Academic Databases)
- Existing Performance Testing Documentation (Company Data)
- Existing Performance Test Data (Company Data)
- Log Analyzer Tool(s) (Open Source) = Develop a new process/ tool
- Performance Testing Tool(s)
- Web, Application and Database Server Logs

Table 1 illustrates the further specific details of the data collected and used during the study.

Table 1: Details of data collection

S.No.	Type of data	Content	Input	Classification
1	Process Documents	- Service Documentation - Project Documentation for example Test Plans, Intermittent Test Reports and Test End Reports	~80 pages	Internal
2	Internal Wikis and intranet pages	Service Area Wikis	~10 pages	Internal
3	Discussion(s)	About Ways of Working and Challenges in Service with Service Manager Clive W, Testing practice head Tomasz Z, Test Engineer Bhakta V., Testing Unit Manager Jutta J.	~1 hour (per each)	Field Notes Internal
4	Log Analyzer	Tool Documentation for Implementation	~20 pages	Manuals Open Source
5	Single Sign on Project	Project Documentation for example System Requirements and Specifications, Design Documentation etc.	~60 pages	Internal
6	Performance Test Tool	Tool Manuals and Documentation	~20 pages	Manuals and Installation Guides
7	Review and Status Meetings	Meetings with Project Team and Service Management	Continuous	Field Notes, Project Status Reports and Presentations

The table above shows the specific details of the data used in various areas of the study apart from the volumes of the data used and the data security classifications.

2.4 Outcome

The outcome is an organisational process to define and collect the precise performance requirements and effective use cases in order to achieve effective outcomes of performance testing and better prediction of the future performance of the software system under testing.

The document includes the new method(s) to collect and analyze the historical web traffic data for web application and accordingly provides the recommendations for example the tool(s) used and how to technically filter the needed information.

It also includes the common key performance testing attributes of a web application and the strategy to identify the same in order to carry out an effective performance testing to meet the test objectives.

3 Fundamentals of Performance Testing

This section includes the theoretical background information on the performance testing subject. It covers the importance of conducting performance test apart from the types of performance tests and their definitions.

3.1 Why Performance Testing is Essential to Business Success

Performance failures are expensive [14] and website outage could cost a business beyond imagination, in 2012, Knight Capital’s computers started to fail in the worst possible way. Instead of shutting down gracefully, they began issuing commands to buy and sell securities. The orders were queued up to be executed over the coming weeks, but the errant computers dumped them on the market all at once, causing a buying and selling frenzy like a clerk who’d gone insane. It only took a half hour, but by the end, the losses

totaled an estimated $440 million. CNN asked whether it was the most expensive computer glitch ever, but somehow “glitch” does not seem adequate to describe an event that almost destroyed Knight Capital [14].

In a survey by YouGov [15], on behalf of HP Enterprise, which investigated the state of performance engineering and its business impacts by surveying 400 development and IT professionals from the organizations over 500-employee strength. Following are some of the conclusions presented by the survey.

Seventy percent agreed that the importance of performance engineering is increasing. The rise in importance of performance engineering is driven by the practical concerns. At least 50 percent of respondents admitted that slowdowns and outages were discouraging customers and frustrating employees.

The consequences are serious. The average firm that responded to the survey said that a major outage could cost between $100,000 and $500,000 in lost revenue per hour. Some of the larger companies with more than 10,000 employees said they could lose $5 million an hour from website or core system outages.

3.2 Concepts of Performance Testing

“Let us face the fact - performance testing is rocket science” - Dawn Haynes

Dawn Haynes is a Senior Trainer and Consultant for PerfTestPlus.com, and Secretary of the Association for Software Testing.

As a part of the project development cycle, performance testing is performed within the testing phase before production goes live. Within testing, performance testing comes at the end of after functional testing is completed. However, performance testing should start in early development stages when application architecture and capacity is being planned. During the design phase performance should be compared among the possible design solutions.

3.3 Performance Testing Process

Performance testing of an application is basically a process of evaluating how the web application would perform at various user loads levels. This is achieved using performance testing tools available commercially as well as open source to mimic the user behavior using minimum hardware resources.

Figure 1 below illustrates the various phases of a typical performance testing project.

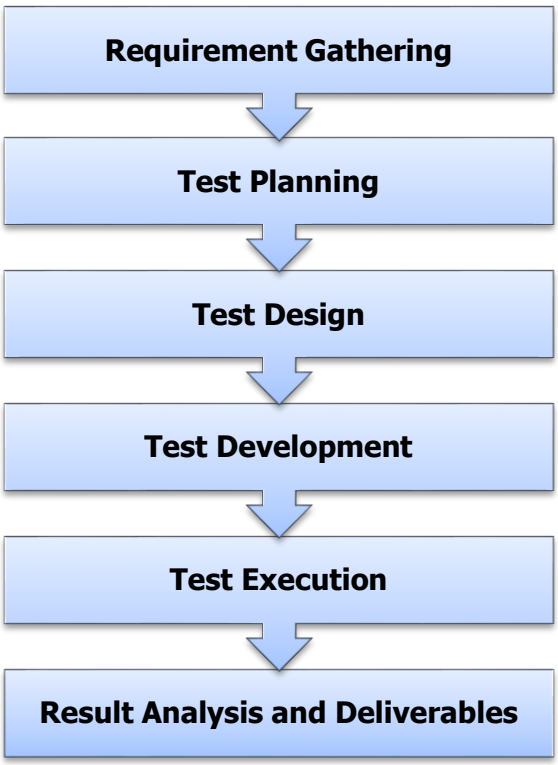


Figure 1: Performance Testing Stages

As illustrated in Figure 1, there are six main phases of a typical performance testing project. Each phase includes certain types of activities and typical activities are elaborated below:

1. Requirement Gathering: In this phase, business needs are identified and documented. The requirements typically defined in terms of desired level of performance in terms of response time, throughput, and resource utilization goals and constraints. Performance test objective are defined in order to meet the business long-term goals.

In some cases, the application usage is studied to identify the key test scenarios and test workflows apart from identification of Business critical transactions.

2. Test Planning: This activity includes identification of test environment, the tools to be used in performance testing and the people who can conduct the exercise. Identification and arranging test data. Test schedule is prepares apart from defining ‘Test Entry’ and ‘Exit Criteria’.

3. Test Design: The test workload model is prepared during this phase based on the gathered requirements or goals during requirement gathering phase. The counters needed to understand the performance and monitor the resource utilization are identified during this period for each test in scope.

4. Test Development: Test environment is validated, test setup and monitoring is configured during this phase. Test scripts are created and enhanced in order to use the test data in accordance with the test design. Test scenarios are setup in the test tool based on the workload load model of the respective test. Shakedown tests are executed in order to validate the readiness of the scripts, test data, test tool and application under test.

5. Test Execution: The tests are executed and test results are collated during this phase using the test tool.

6. Analyze Results and Deliverables: Test results data is analyzed and evaluated against the test targets. Result are cross-referenced and test report is created based on the test(s) outcome to be delivered or discussed with the stakeholders.

3.4 Types of Performance Testing

The following are the some of the common types of performance testing used for IT applications but not restricted to the below list. Please note that some organizations might be using these terms differently than described below.

3.4.1 Shakedown Test

Shakedown is designed to be a fully independent and is used to confirm environment readiness from a test tool perspective. This cycle provides an opportunity to test the Performance Center scripts and Performance Center parameterized data files with limited volumes. During this cycle, basic sets of functional scripts are executed to ensure the environment is stable prior actual performance testing. The main purpose of the shakedown test is to validate the readiness to start the full load testing and it is not meant to test any aspect of the target application.

3.4.2 Smoke Test

Smoke test is executed before main load test to verify that the designed load model achieves the desired rate of key transactions as well as intended rate of HTTP request to the server. This test does not meet any project specific target but actually is a supplementary test to ensure that the next or main test could be carried out smoothly.

3.4.3 Baseline Test

Baseline test will be carried out with 20% of the anticipated user load. The test is designed to be a fully independent cycle and is used to confirm environment is scalable for minimal load. This cycle provides an opportunity to test the environment behaviour with minimal usage of the work load. In case mixed baseline test found any bottlenecks, no further tests are conducted until bottlenecks are fixed. Baseline test is helpful in analysing the main load test results and to understand the system behaviour on minimal load.

3.4.4 Load Test

By default, load test actually refers to application full load or peak load. Load testing validates if the system/application can meet the requirements for volume / throughput, scalability and response times over a certain period. The actual load level being tested may be varied so that projected future load levels can also be verified. This is needed if, for example to, the application / system requirements include a statement that the system/application must be able to meet future performance needs.

3.4.5 Stress Test

Stress testing allows the measurement of the maximum throughput that the system/application can cope with. This test also indicates which component of the system/application first gives way under increasing load and tests the system's ability to recover when the load decreases again.

3.4.6 Endurance or Soak or Stability Test

The endurance test validates if the application is compliant to it availability requirements. This test could identify the abnormal resource utilisation during the test, for example, memory leaks. This test is performed over longer period and on the agreed load level.

3.4.7 Benchmarking or Reference Test

A reference test consists of a test with one (1) end user in an unloaded system. This gives the best possible end user response times that can be achieved in the system.

3.4.8 Failover Test

The primary purpose of failover tests is to validate the redundancy mechanism of the system. Typically once full load is applied to the system, the one component of the high availability layer is brought down and again up after a short interval of time systematically to understand the load balancing and recovery if such failover happens in real time.

3.4.9 Volume Test

Volume testing provides answers as to how the system handles a large amount of input data. This need not be done under full system load.

3.5 Workload Modelling

User load distribution across all the identified workflow scenarios is referred as workload modelling. Workload model defines how the application would be used during testing.

To collect the realistic test results and measurements the workload should be as realistic as possible to the real production load. It emphasises that fact that historical production data should be used in order to derive the workload model. Figure 2 shows the workload model of a shopping website.

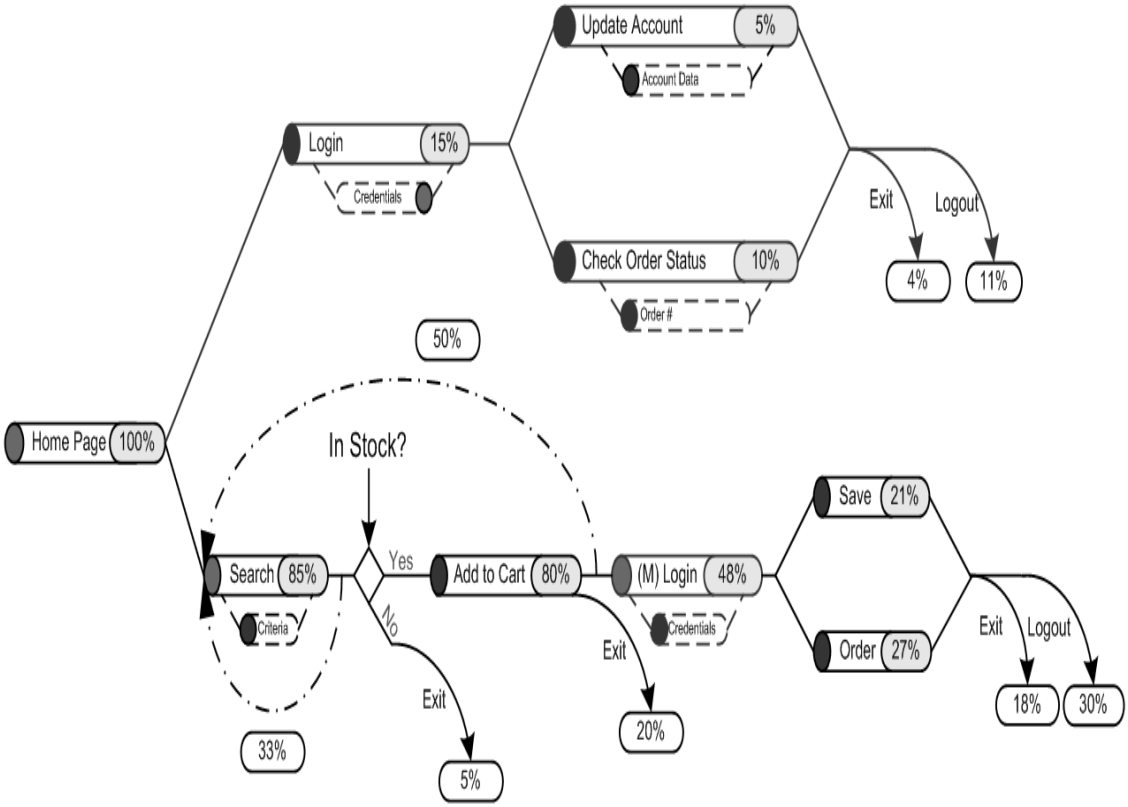


Figure 2: Workload Model of a Shopping Website – Example [13]

Figure 2 depicts the workload model of an example shopping web application. It represents the number of users with respect to user workflows. The production workloads are studied by collecting various statistics in order to identify the relevant user patterns, which could be used to create workload model for the test.

4 Existing Process and Initial State Analysis of Case Project

This section discusses the existing process involved in requirement gathering and further deriving the workload models, which is being followed in the shared service, called performance testing and monitoring service. The case company is a leading IT service provider based out in Finland. It is responsible for running the shared service for a world leading telecommunication equipment manufacturer and service provider. The shared service offers performance testing, performance monitoring and performance troubleshooting related services across the various active projects in its IT department. It allows active project owners to submit their applications and have them run through a structured, professional, tried-and-tested testing process - based upon their Performance Testing and/or Monitoring requirements.

4.1 Initial State Analysis of Performance Testing Shared Service

The objective of the Performance Testing Service is to simulate real-life processes and transactions upon the supplied application similar to the typical day-to-day business usage and to mimic a normal workload, as well as creating worse case scenarios if required also. Performance testing primarily uses Hewlett Packard's 'Performance Center' and 'LoadRunner' for performance testing. The tools also include Hewlett Packard 'SiteScope' solution for resource monitoring while executing the performance tests. Performance monitoring utilizes Hewlett Packard 'Business Availability Center' together with 'SiteScope' and the combinations of these tools are used for performance troubleshooting projects.

This section would also highlight the main drawbacks in the existing process of requirement gathering, which is a part of performance testing service in order to justify the requirement of new process as stated in the problem statement. Figure 3 shows the existing process diagram at use at the case company.

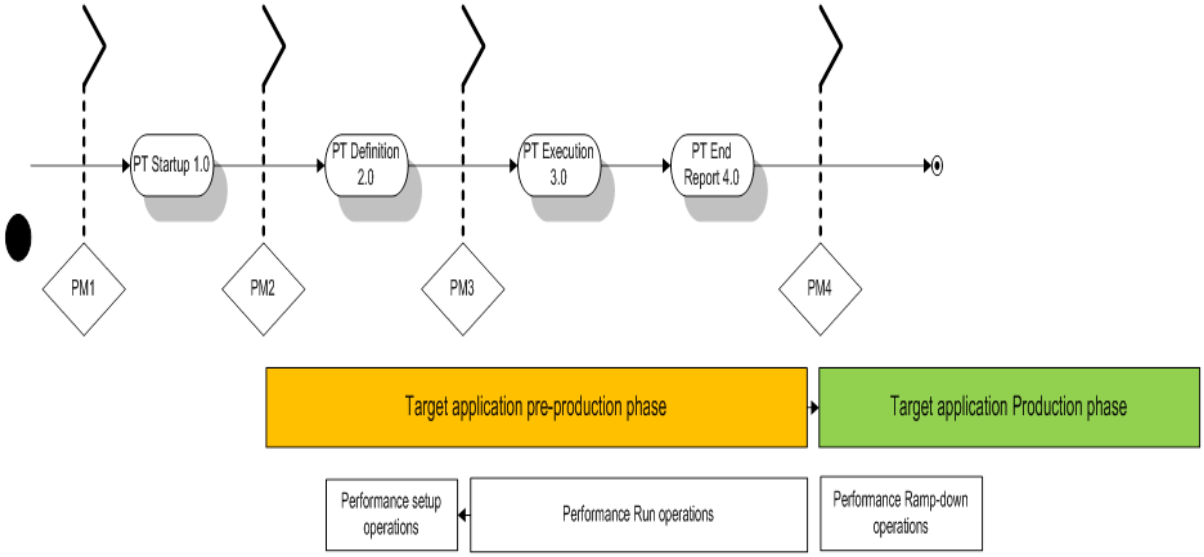


Figure 3: Existing Process Diagram

The process diagram above shows the existing Performance Testing Process being followed and its alignment with the overall project development process and project milestones (PM). In the diagram below PM1 represents requirement phase project milestone, PM2 marks the end of design phase, PM3 project is for software development phase, testing, and PM4 marks production go live.

The performance test activities are aligned for each phase, under PT start-up phase, Non-Functional requirements; use cases to be considered for performance testing are requested apart from the architecture diagram and software and hardware specifications of the servers. System usage statistics are also requested in this phase.

As a part of PT Definition, phase performance test engineer collect the non-functional requirements by asking random questions and document the same in test plan document which is a deliverable at the end of this phase. There are no supporting questionnaires or guidelines available in the service to assist the performance test engineer to derive the performance requirements in case those are not defined earlier.

The number and type of questions are dependent upon the competence level of the engineer. Often requirement gathering exercise is mixed with other topics like work scope definition, roles and responsibilities, project schedule and costing etc. which

leaves less room for the focused discussions and analysis around performance requirement gathering area.

In practice, projects do not raise the request to plan and implement performance testing well in advance. The service request in most of the cases is raised during testing phase only and the availability of limited time affects the quality and outcome of requirement gathering process as well as overall performance testing activity.

In addition, the main bottleneck at present is unavailability of process and proper documentation to support the performance test engineers in the initial phase of the performance test project that is how to collect and document the requirements.

In the service, there is high number of projects with either not defined NFRs (non-functional requirements) or the unclear or vaguely defined NFRs. The number of such projects is on increase after the department has started following ITIL process and performance testing has been made compulsory in order to evaluate and baseline the performance of the applications including legacy ones.

4.2 Initial State Analysis of Case Project

The application under test is an organization wide single sign on system called ‘WEB SSO’ for which the request is raised to access its performance through ‘Performance Test Service Request Document’. The service request includes main objective details apart from other commercial details not so relevant to the topic.

Apart from the service request document, Test plan and Test End Report are the main documentation available about the performance testing activity carried out in past for the particular release of the single sign on system. The test plan included scope of testing, the test approach, the testing phases, test types, the test strategy to be used. The test end report captures and summarises the results of the tests.

Based on the reference documentation, the following tests were planned be executed as part of the web single sign on system performance testing to meet the objective stated in the service request document. The test was aiming to access the performance of the application after its latest release at that time to understand if the system could handle

the anticipated load and does not degrade the performance during latest release compared to prior release. Following test types were executed:

1) Shakedown test- Supplementary test to ensure scripts and environment readiness.
2) Baseline Test – Supplementary test carried out with less than 20% of peak load to baseline the performance of the application.
3) Load Test – The Load test with 400 virtual users for the duration of 2-3 hours. The load was generated from Finland location only.

The load test was the main test planned to meet the target or customer request to evaluate the performance of the system at full load. There are no documentation available about how conclusion of 400 virtual user was considered as peak system load. Also there is no evidence about historical data analysis or how the load model was derived.

The single sign on system does not keep persistent sessions so actually the main definition of the load is being driven by the rate of authentication requests but no such calculations were involved in defining the work load model of the full load test.

The goal of performance evaluation is quite broad and specific measurable objective(s) was not agreed. In addition, no test was executed to baseline the performance of the old release in order to compare with the performance of new release.

Though the service had a defined process which includes the collection of nonfunctional requirements but the project do not have clearly defined requirements which lead to the situation and test engineer decided to execute a single test to cover the bare minimum of what was mentioned in the service request.

There is a clear requirement for further process development, which could ensure that the measurable performance requirements are agreed, and if the requirements are not available then those should be derived from the broad business goals.

4.3 Requirements

As mentioned in the initial state analysis, following are some pain areas, considered for possible improvements:

1. Study and propose the improvements for performance requirement gathering process being followed at service level
2. Study and propose the method to improve load modelling method being followed in the service
3. Study and implement the proposal to some specific project to evaluate the effectiveness

New methodology description detailed in section-5 could be mapped with requirement-1 and 2 whereas case study of a project, detailed in section-6 could be mapped with requirement-3.

5 Performance Requirements Deriving Methodology

This section defines the series of proposed changes and the documentation introduced in the shared service in order to identify the performance test requirements for the projects without or vaguely defined non-functional requirements.

5.1 Goal, Test Objectives and Targets Method

The proposed method is to assist the performance engineers to simplify the high level business goals and also to derive the high level requirements to testable requirement and further identifying the performance tests based on the test objectives apart from setting up SMART, which expands to Specific, Measurable, Achievable, Relevant and Time bounded, target. This method ensures that the performance requirements are identified and are testable.

Figure 3 shows the initial stages involved in performance testing process which includes performance requirement gathering and load modelling mainly. Please note that the diagram below does not shows the other stages of performance testing e.g. test executions etc.

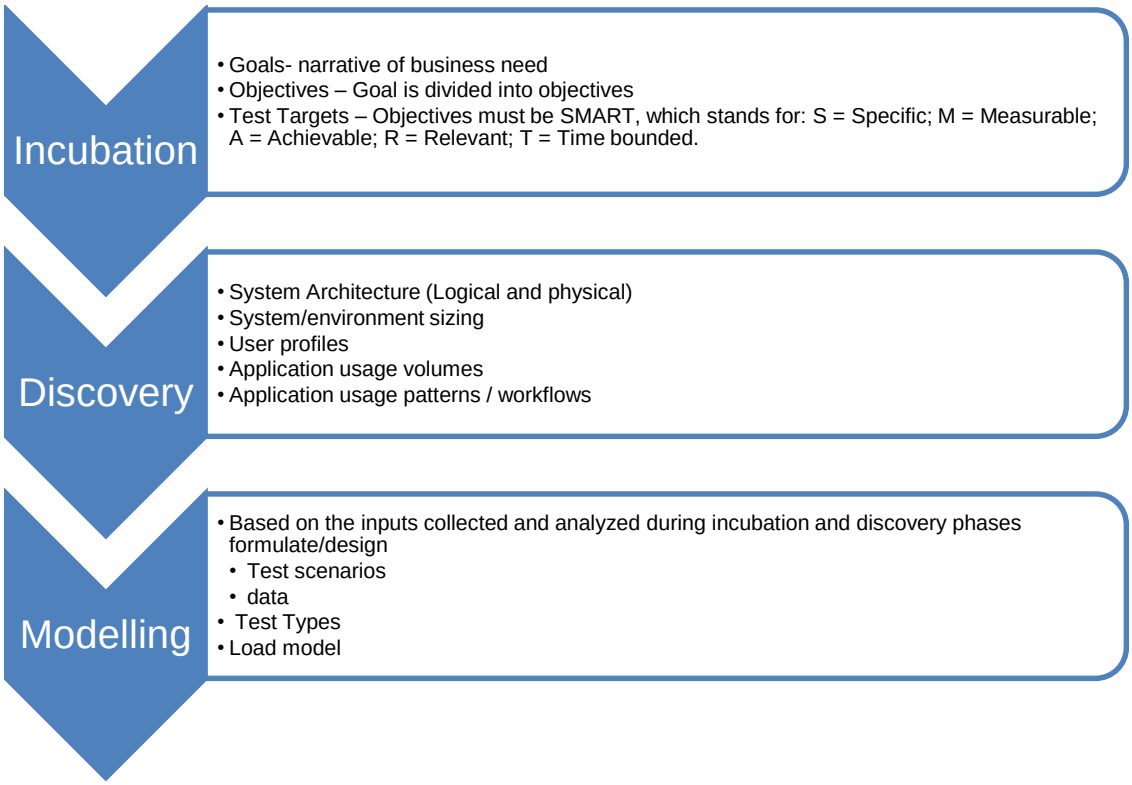


Figure 4: Initial Stages in Performance Testing

As shown in the diagram above, requirement gathering is done during incubation phase and there are three main sub-stages of incubation phase that is define performance testing GOAL, breaking the GOAL into various medium length achievement plans called Objectives and finally further breaking the Objectives into Targets, SMART short term achievement plans. The targets would lead act as an input to derive the test types required during performance testing of the application.

5.1.1 Performance Test Goal

To understand what one wants to accomplish with a performance test, it is where any discussion on performance test needs to start. Performance testing is expensive and needs significant upfront investment in terms of efforts, hardware and software tools. Performance Testing Goal understands the business drivers for the performance testing.

To assess the requirement to carry out performance testing, some of the example reasons to conduct performance testing are as follows:

- To evaluate existing performance
- For service level agreements assessment or to meet performance goals or non-functional requirements
- To collect baseline for future testing
- To estimate capacity or hardware sizing or configuration assessment
- To identify performance bottlenecks
- To conduct performance tuning for performance hardening

Hence, the needs of performance testing are varied and understanding the exact need is most important thing for all subsequent steps.

5.1.2 Performance Test Objectives

In this stage, a single goal is broken into various objectives with a medium term achievement plan. If all objectives are met, it should fully ensure the completeness of the performance testing goal. Example objectives based on an example goal could be ‘Baseline the performance of the application’, ‘identify the bottlenecks, bad performing workflows and transactions’, ‘Improve the performance of the application’, ‘Performance should not degrade over time’ etc.

5.1.3 Performance Test Targets

Once test objectives are derived, the next step is to deduce the tests needed to achieve each objective and setup the targets for each tests. The targets would help to agree on the success criterion of the individual test.

5.2 Identifying Testable Performance Requirements through Questionnaire

In the cases where clear nonfunctional requirements are not defined for the application, an interview based approach using a questionnaire or checklist is effective in order to understand the expectations from performance testing and more importantly identify the objectives of the test. Once the objectives are defined and agreed on, the next step would be to identify the tests to meet the objective and set the target for the test preferably

based on some historical data. Following are the main question areas and the sections of the questionnaires.

5.2.1 Business Transactions and Application Usage Related Questions

This section of the checklist developed for the service of the organization in case, covers all the questions focusing on the business importance as well as the project purpose. In a way it is important to know the project high-level details before presenting the questions, depending on the project purpose, the checklist could be further modified to include the specific questions if required.

Table 2 presents some Application Usage based example questions from the questionnaire:

Table 2: Performance Requirement Gathering Questionnaire - Application Usage Based

S. No.	Requirement Definition
1.	What is the total number of registered users in the system?
2.	What are the different types of users of the system? External Business to business, Customer self-serve, External - Third party call centres Internal Front office, Back office, Front and back office, Combination of external and internal How is the total number of users split by user type?
3.	What are the expected number of Concurrent, Sequential and Simultaneous users during Peak Times? Simultaneous Users - no of users in the system at a point in time i.e. Active users logged in Concurrent users - no of users doing an activity in the system at a point in time i.e. no of requests send by users at a point in time to the app. server Sequential Users - no. of users doing an activity one after other
4.	What is the growth pattern of users? All at once, front loaded, steady increase, back loaded , Others specify
5.	What is the growth volume of the users? All at once, front loaded, steady increase, back loaded, Others specify What is the growth volume of the users? All at once, front loaded, steady increase, back loaded , Others specify

The questions above primarily collect information about application usage in the production including the future growth expectations.

Table 3 presents some example questions from the questionnaire targeting Transactions information:

Table 3: Performance Requirement Gathering Questionnaire – Transactions

S. No.	Requirement Definition
1.	What are the key Business Transaction of the system?
2.	What are the required average and peak volume of transaction?
3.	What are the Users, TPS and Response Time SLAs for above Key Business Transactions? If possible, please provide additional information regarding the nature of these transactions, for example activate, cancel, modify, enquiry, update, polling, reporting, customer orders activation etc.
4.	What are the peak periods of usage for this system? Identify any known peak hour, peak day, peak month periods.
5.	What is the expected Interface Volume of Interface, Databases or Reports? Number of Records in Interfaces/Reports
6.	What is the expected Response Time of Interface, Databases or Reports? Response Time of Interfaces or Reports
7.	What transaction volume growth (volume, frequency) is expected in next 1 year?
8.	Any Downstream systems make use of this project? If yes can those systems manage the transaction volumes?

The questions above focus on the business workflows and transactions in order to identify their business importance and the volumes to effectively develop the load model.

Table 4 presents some sample questions to collect information about background processes e.g. batch jobs:

Table 4: Performance Requirement Gathering Questionnaire – Background Processes

S. No.	Requirement Definition
1.	Frequency of batch programs for example daily, weekly, monthly and their execution time and duration.
2.	Complexity of batch programs in terms of number of records fetched and processed
3.	The concurrency of batch processing with online transaction processing (overlapping periods of time)
4.	Any other background jobs emulating load on the system or part of the system?
5.	Schedule and duration of database backup(s)

The questions shown above collect information about the background process present in the production system. The background jobs affect the system performance significantly.

5.2.2 System Architecture and Test Types Related Questions

This section of the checklist includes the questions specific to identify the system's performance validation points from the architectural view point for example if the application has multiple web server nodes being utilized using load balancer then it could lead to identification of an objective to validate the failover and resiliency mechanism of the web server nodes. Following are some of the common architectural based questions and tests:

- What is the technology stack? (Programming Language, OS. Application Server, Database Server, Middleware, Load Balancer and Deployment Topology)
- Describe the technical architecture of the application. Provide the link to the same
- Describe the Integration Architecture of this application? How many Interfaces are in scope for this application testing? List them, if any.
- Does application use load balancer? Method of load balancing?
- Is the application high available at application server layer? How many applications server nodes are present?
- Is database layer clustered? How many DBs application have?
- What are the various types of tests in scope?
- Is there any specific ramp-up/ramp-down pattern to be followed?
- Validation of memory leaks in the application?
- Finding limits of the application or breaking point or the error which application registers before crashing

The questions listed above mainly help performance engineer to collect information about the application architecture. It would help to conclude the types of tests required to meet the requirements.

5.3 Simplified Guide for Performance Testing Process

Performance Testing Process Handbook primarily consolidates all the templates, checklists and guides to a single document to which a test engineer could refer and utilize in the project. Following are some common objectives it enlists as examples:

- Measure End-to-End transaction response time and demonstrate that the system functions to specifications with acceptable response times while processing the required transaction volume
- Demonstrate that the system meets the requirements for transaction throughput and response times simultaneously
- Measure server components performance under various loads
- Monitor system resources under various loads
- Measure the network delay between the server and clients

It also provides the basic guidelines to identify the workflows to be considered for performance testing. Following is the excerpt from the corresponding section of the guide. The transactions selected for the performance test are a small subset of the system or functional test transactions.

The transactions are selected based on
1) Business criticality,
2) High volume, or
3) Resource intensive

Under project lifecycle, which is further divided into following performance testing project phases:
1) Requirement Gathering and Analysis Phase
2) Test Planning and Designing
3) Test Development
4) Test Execution and Delivery
5) Project Closure

Each phase is detailed with the expected activities in the phase with the reference to the corresponding supporting documentation i.e. templates, guides, checklists, example reports etc.

5.4 Handbook to Use Web Analyzer Tool for Log Analysis

This handbook in intended to support the test engineer in the case where application historical data is not available or need to be collected and analyzed.

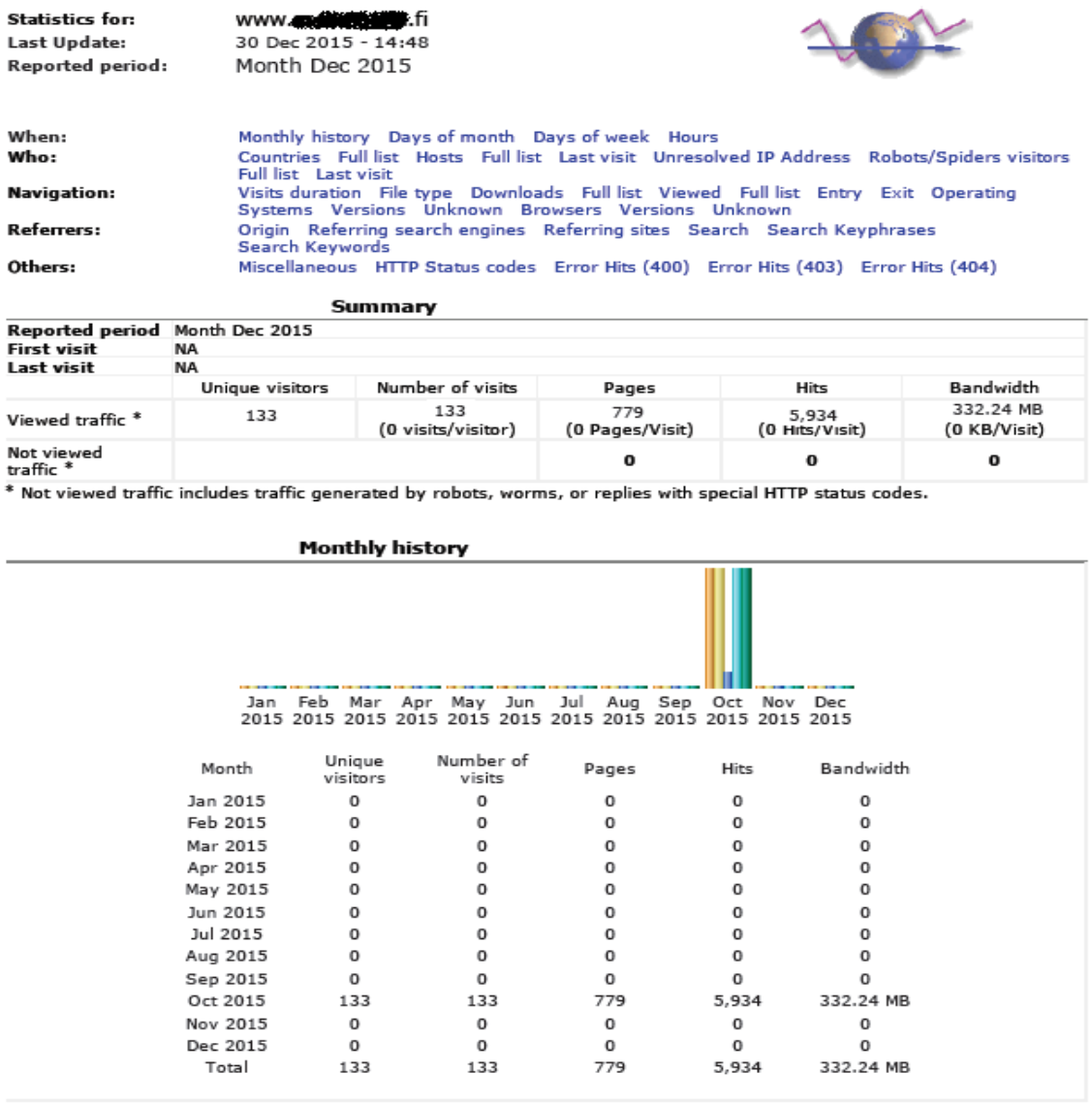
The target of the workload model is to generate as real as possible a production load during the performance testing. Designing an accurate workload model by using the historical usage data from production helps the performance test engineer:
1. In reproducing same load (as production) with exact caching and think time behavior.
2. In distributing user load based on geographic locations and across different types of browsers.
3. In understanding the exact business steps that end users perform - business flow.

Web server log analyzers and web analytics tools could be used to analyze the historical data from production. Log analyzers parses web access log files obtained from web servers and derives indicators about who, when, and how a web server is visited. Analytics tools on the other hand integrate with browser components (Java-script, cookies, etc.) and make it possible for the tool to present the exact user behavior.

In organizations, some projects are actively utilizing the web analytics tools to understand the system load and collect the usage statistics. While undergoing the performance testing of such projects the availability of data for such projects is easy and the study does not emphasize such projects. As described in the problem statement, the main issue arises while organizing the performance testing of the projects with no defined performance requirements or without application usage statistics. As a solution to collect application usage statistics is to analyze web server access logs, various tools were studied briefly and were compared for the suitability in the shared service.

AWSTATS, an open source log analyzer tool, was studied in detail to setup a common solution in the performance testing shared service. As an outcome, a handbook for AWSTATS log analyzer is created for internal use illustrating the step by step procedure to install and setup the tool. It also provides the detailed instructions on how to analyze the historical application usage data for WebLogic access logs.

Following (Figure 5) is an example report generated based on the web access logs of a University website:



Statistics for: www.[redacted].fi
Last Update: 30 Dec 2015 - 14:48
Reported period: Month Dec 2015

When: Monthly history Days of month Days of week Hours
Who: Countries Full list Hosts Full list Last visit Unresolved IP Address Robots/Spiders visitors Full list Last visit
Navigation: Visits duration File type Downloads Full list Viewed Full list Entry Exit Operating Systems Versions Unknown Browsers Versions Unknown
Referrers: Origin Referring search engines Referring sites Search Search Keyphrases Search Keywords
Others: Miscellaneous HTTP Status codes Error Hits (400) Error Hits (403) Error Hits (404)

Summary

Reported period	Month Dec 2015				
First visit	NA				
Last visit	NA				
	Unique visitors	Number of visits	Pages	Hits	Bandwidth
Viewed traffic *	133	133 (0 visits/visitor)	779 (0 Pages/Visit)	5,934 (0 Hits/Visit)	332.24 MB (0 KB/Visit)
Not viewed traffic *			0	0	0

* Not viewed traffic includes traffic generated by robots, worms, or replies with special HTTP status codes.

Monthly history

Jan 2015 Feb 2015 Mar 2015 Apr 2015 May 2015 Jun 2015 Jul 2015 Aug 2015 Sep 2015 Oct 2015 Nov 2015 Dec 2015

Month	Unique visitors	Number of visits	Pages	Hits	Bandwidth
Jan 2015	0	0	0	0	0
Feb 2015	0	0	0	0	0
Mar 2015	0	0	0	0	0
Apr 2015	0	0	0	0	0
May 2015	0	0	0	0	0
Jun 2015	0	0	0	0	0
Jul 2015	0	0	0	0	0
Aug 2015	0	0	0	0	0
Sep 2015	0	0	0	0	0
Oct 2015	133	133	779	5,934	332.24 MB
Nov 2015	0	0	0	0	0
Dec 2015	0	0	0	0	0
Total	133	133	779	5,934	332.24 MB

Figure 5: AWSTAT Summary Report

Figure 5 includes the statistics of a university website and its usage during a single month. Other important application usage details which could also be collected using AWSTAT tool are as follows:

- Hourly usage statistics to collect the peak usage data per hour as an input for peak load test
- 'Visits Duration' section could help in calculating average and peak session length for as an input for work load model

- 'Browsers (Top 10)' could reveal the main load generating browsers to further refine the load scenario by simulating the browser behavior during test
- Domains/countries of hosts' visitors (pages, hits etc.)
- Most viewed, entry and exit pages, to identify the most used workflows, Etc.

The information collected using AWSTAT could be used to design effective workload model based on historical data analysis of the application in production. In the cases where logs are not enabled, it is recommended to engineer to make a request to enable logs for some time in production environment and utilize the collected logs for analysis in the tool using the handbook.

5.4.1 Workload Modelling Guide

This section provides the insight in to the guide, which is created to support performance engineer working in the shared service in order to design an effective workload modelling for the application under performance testing.

The target of performance testing is to simulate the real world load but practically there is always something, which is either too much costly or not possible to have in lab environments. One the one hand it is impossible to achieve the 100% real production load in the lab environments but on the other hand, the idea is to have the near production load in order to identify the production performance issues.

There are numerous factors, which should be taken care of to define the load for the web application under testing, but following are some salient factors, which should be derived carefully as these are the main load defining factors for any applications.

5.4.2 Number of Concurrent Sessions

The number of concurrent factors is one of the most prominent factors, which define the load as every user logged on to the server consumes server resources.

Some projects try to achieve a high rate of transactions by reducing the pacing and think time (also called as user pause time between 2 user actions) for example to minimize

performance testing tool license cost. Usually the cost of commercial performance testing tools license is based on the number of virtual users. It is not a good practice in case application utilizes persistent sessions. It is not an issue for the web service based applications, which does not maintain persistent user session.

Every user logged into web server consumes some resources for example each session reserves some memory from the memory pool of the application to execute its thread originating from the corresponding user or client. The session state information is stored on the web server in its memory pool and it remains in the memory till the time user is logged in to the application which is usually represented by removal of session from the session pool. If rate of requests is increased by reducing the think time between user actions and pacing, which is pause duration before next iteration starts, it would result into a lower number of user sessions on the web server which means less memory would be consumed on the server. Using this workload, it is evident that the memory associateted issues would not surface out in testing whereas the issue might occur in production.

Secondly, the number of concurrent users would seek parallel connections to the web server as well as to the database server. In the event of running the load with lower number of virtual users, the issue with lower parallel connection cannot be exposed whereas it would lead to higher response time in production, as the thread would wait for connection at web or application or database layer.

5.4.3 Session Length

Session length plays a significant role in defining the number of concurrent sessions, which, in turn, plays an important role in defining the overall load on the server as each session consumes some server resource. A longer session length would lead to large number of concurrent sessions in the application as well as could lead to higher response times due to queuing for the resources for example memory allocation or CPU thread queue apart from the queue at connection pool.

Aggressive session length leads to better application performance with the given server resources but it also leads to unpleasant user experience since users are logged out

frequently and asked to login again. In this case a due balance in the session length is required considering both aspects.

Average session length should be derived by mining the historical data for example web access logs by using web log analysers or by using web analytics tools to gather the statistics.

5.4.4 Rate of Transactions

The rate of transactions is also commonly referred to as TPS (transactions per second). This is the main contributor to the overall load for any application. Primarily, the key transactions should be identified and their rate of occurrence should be derived from the historical usage of the application. This could be achieved using application access logs or using web analytics tool implemented to the application.

For the applications without usage history, the anticipated rate of key transactions should be derived with business stakeholders based on expectation from the application.

5.4.5 Other Factors and Best Practices:

Some other aspects which should also be taken into consideration when designing a workload model for the performance testing of an application are as follows:

- Hits per second, which describes the rate of requests received by the application for processing and in turn define the arrival rate of the load.
- Understanding cache settings, the areas where application or database uses cache to provide response to the requests. The cached response would of course be faster than the actual data retrieval from the source or database.
- Understand the load distribution on each application components or transaction.
- A small duration smoke test should be executed before running the main load to check if the designed load model achieves the target transactions per second.

The suggestive list of recommended points and best practices to design an effective workload model is presented to the service and recommended to be kept updated.

5.4.6 Requirement Traceability Matrix Template

This document was created to target the projects which has their non-functional requirement are defined properly. This template suggests to create a traceability matrix in order to find out the gap and any missing requirement which has not been covered in any test. If the traced requirements are high level then the same should be refined further using Goal, Objective and Target methods explained above.

The template primarily contains the ‘Performance Objective’ column which contains performance requirements or test objectives mapped to column ‘Test Name’ which list out the test which supposedly ensure the validation of the requirement or goals.

6 Case Study

The application under test is an organization wide single sign on system called ‘WEB SSO’ for which the request is raised to access its performance after it was migrated to different infrastructure as a part of datacentre migration project. At the time of raising the request it is used by around 70 applications and approximately 2-3 applications were getting integrated on monthly basis.

The following sections present the further details of the performance test project executed by the performance testing and monitoring shared service.

6.1 Test Setup Description

The following sections present the further details of the test setup conducted on the case project.

6.1.1 Test Tool Setup

The web single sign on system is tested using the Hewlett Packard’s Application Life Cycle Management Performance Center version 11.50 tool (HP ALMPC v11.50). Performance Test engineer used the standard organization workstation and Hewlett Packard Virtual User Generator Tool from organization intranet or from Collaborator location to create the protocol based performance test scripts. This is done by manually repeating the designed performance test use cases while Hewlett Packard Virtual User Generator Tool records the client-server traffic (usually http or https). Connection between workstation and the target system is needed on End User point of view.

Virtual User Generator version 11.5 is used to create scripts for the identified use cases. Scripts are used in Performance Center which simulates the desired number of users accessing the script. A certain number of scripts together with timing settings make up a scenario. A scenario, run according to timing settings, makes up a test. The load that a test generates is created using the Load Generators. During the testing, information about the run is gathered by Performance Center tool. Results are then analysed and compared with Load Runner Analysis tool.

6.1.2 Application Test Setup

Following is the approach of geographical distribution in conjunction with the application-distributed architecture; the resultant overall architecture post migration is as follows (cf. Figure 6):

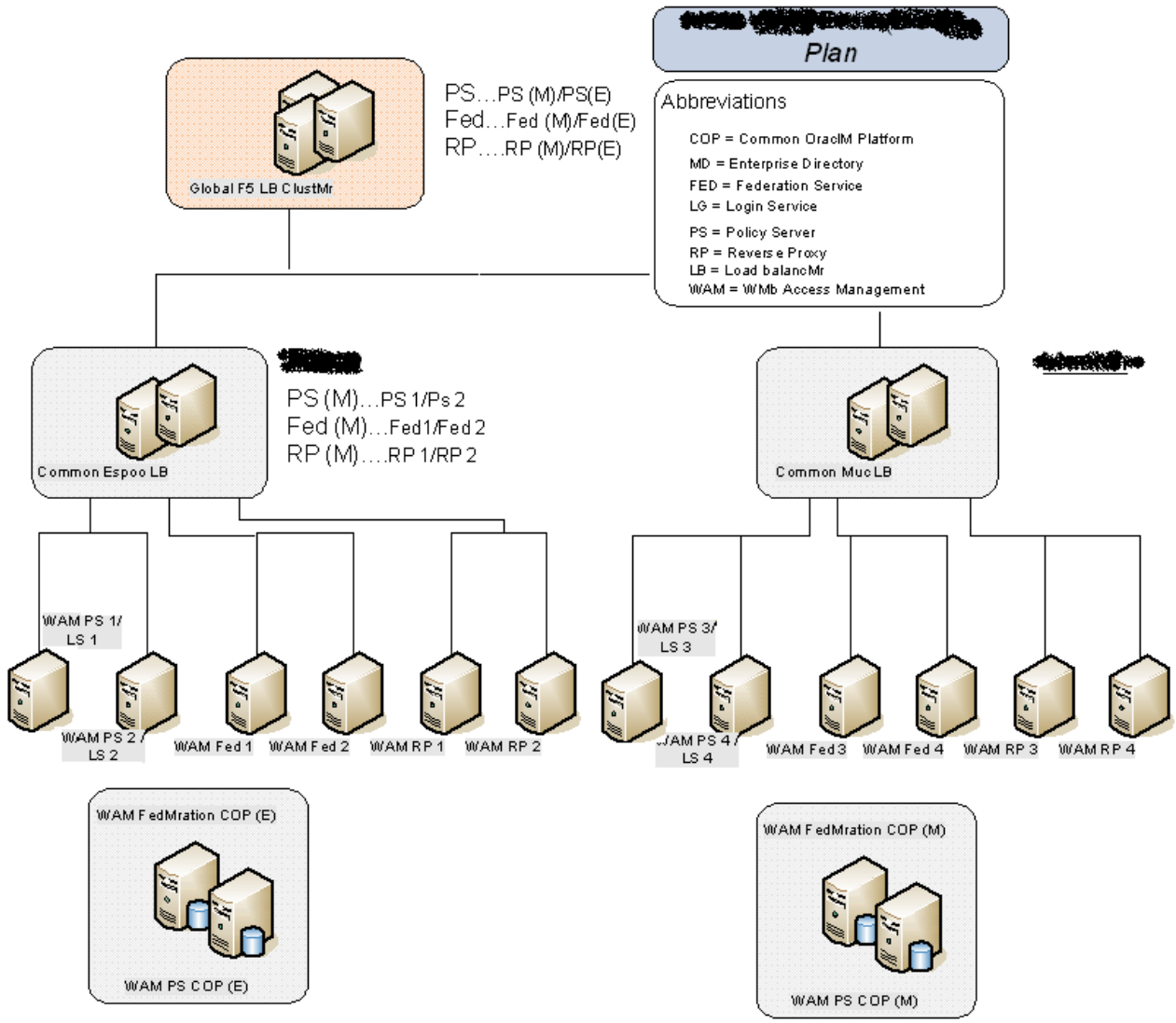


Figure 6: Network Diagram of Test Tool & Web SSO

In the diagram shown in Figure 6 above, each datacenter hosts two instances of policy / login server, federation server and reverse proxy using the local load balancers. The load balancing across datacenters is based on IP load balancing using DNS load balancer.

SSO services rely on authentication against Enterprise Directory (ED) for internal user and on NEDI-B/CE Siteminder directory for external user.

6.1.3 Resource Monitoring Setup Details

The system resource monitoring i.e. CPU utilization and memory consumption performance counters are monitored using HP Site Scope agentless monitoring tool. The architectural and integration with performance Center is detailed section ‘Performance Test Tool Setup and architecture’ above. The polling interval for all monitors is set to 30 seconds.

6.2 Performance Test Goals

Based on the goal specified in the service request document and the corresponding discussion based on the proposed methodology following goals were identified:

1) User experience should not be affected due to migration

2) Redundancy must be improved so that outages have less impact

3) Analyse and improve the performance of the production system

The above mentioned goals are further elaborated in terms of objectives and targets of each test as follows (cf. Table5):

Table 5: Case Project - Performance Test Goals

Priority	Goal / Requirement	Test objective	Target Test(s)	Test Target
1	User experience should not be affected due to migration	To compare the performance in terms of user response time at full load before and after the migration	Load test- Pre Migration	To collect the average transaction response time of the key transactions when system is subjected to 2526 login requests/minute
		To compare the performance in terms of user response time at full load before and after the migration	Load test- Post Migration	The performance should be either equal or better after migration. The performance is measured in terms of average transaction response time of the key transactions which should be either low or equal when system is subjected to 2526 login requests/minute
2	Redundancy must be improved so that outages have less impact	To validate the DNS load balancing across two different datacenters present in different geographical locations in case one datacenter is not reachable.	Site failover test	Web SSO service should be available if one site in a datacenter is not available without affecting the user login response time when system is subjected to 2526 login requests/minute
		To validate the functioning of local load balancer	Application failover test	Web SSO service should be available if one node is not available without affecting the user login response time when system is subjected to 2526 login requests/minute
3	Analyze and improve the performance of the production system so that consistent performance is ensured	To analyze the performance of the application to determine if the system can sustain the high load for long duration.	Endurance Test	Web SSO service should be available without affecting the user login response time when system is subjected to 2526 login requests/minute for 48hours
		To determines the robustness of the application	Stress Test	To determine the system breakdown point i.e. the number of login requests it could handle with the existing infrastructure.

The objective of tuning the application configuration parameters for the hardening of the system and gain the performance improvement is agreed to be treated as a separate project if the results shows that tuning is required. This is considered not in scope of the study at present. The main areas of performance tuning with their priorities would be identified based as the results of load, failover, endurance, stress tests apart from the recommendations from the product vendor.

6.3 Performance Test Cases

This section describes the test workflows included in scope based on the business analyst recommendation. Historical data and analysis was not required in this case as the application had very limited workflows.

1. The following workflows were considered for Policy Server for performance testing:

   i. User tries to access protected application, Site Minder agent checks if user has valid cookie. If user has valid cookie, user can access the server, else user is redirected to the login page.

   ii. Browser loads the new page with the correct parameters (user fills in credentials to new page and press submit).

   iii. Login takes place and policy server authenticates user against ED.

   iv. Authentication successful is returned to the agent and agent creates cookie for the user's browser and redirects user to the original URL.

   v. User accesses protected application with the cookie – Agent authenticates the cookie against policy server and lets the user access the protected resource.

2. The following workflows were considered for Proxy Server for performance testing:

   i. User tries to access protected application, Site Minder agent checks if the user has a valid cookie. If the user has a valid cookie, the user can access the server, else the user is redirected to the Cookie provider.

ii. Cookie provider checks if the user has got a valid cookie for the master domain and if the user has a valid cookie for company website the user is redirected back to the server with the data that the agent uses to create a cookie to the new domain. If the user does not have a valid cookie, he is redirected back to the agent which in turn redirects the user to be authenticated.

iii. Browser loads the new page with the correct parameters – The user fills in credentials to a new page and presses submit.

iv. Login takes place and policy server authenticates the user against ED. Authentication successful is returned to the agent and the agent creates a cookie for the user's browser for the company domain and redirects the user to the original URL.

v. User access protected application still without suitable cookie. Site Minder redirects the user to the cookie provider. If the user has a valid cookie for the company domain -> he is redirected back to the server with the data that the agent uses to create a cookie for the abcd.net domain where the proxy server is and the cookie provider redirects the user back to the original resource.

vi. Agent authenticates the cookie against policy server and lets the user access the protected resource.

3. The following workflows were considered for Federation Server performance testing:

i. Login process to the policy Server.

ii. The URL that is provided in the first request is one that points to the federation server and that will initiate the actual login process and redirection would follow after successful login.

6.4 Load Model- Full Load Test

This section describes the load model for the peak load test. There is a different load model for each specific test in scope for the application but in this section only most important test is covered which is designed to achieve the primary business goal. The

results of the test (i.e. peak load test) are discussed in detail in Section 6.5 as it is a common test which was executed in the earlier release and is discussed in the initial state analysis, Section 4.2.

Figure 7 below shows the load scenario settings i.e. user load distribution per script during peak load.

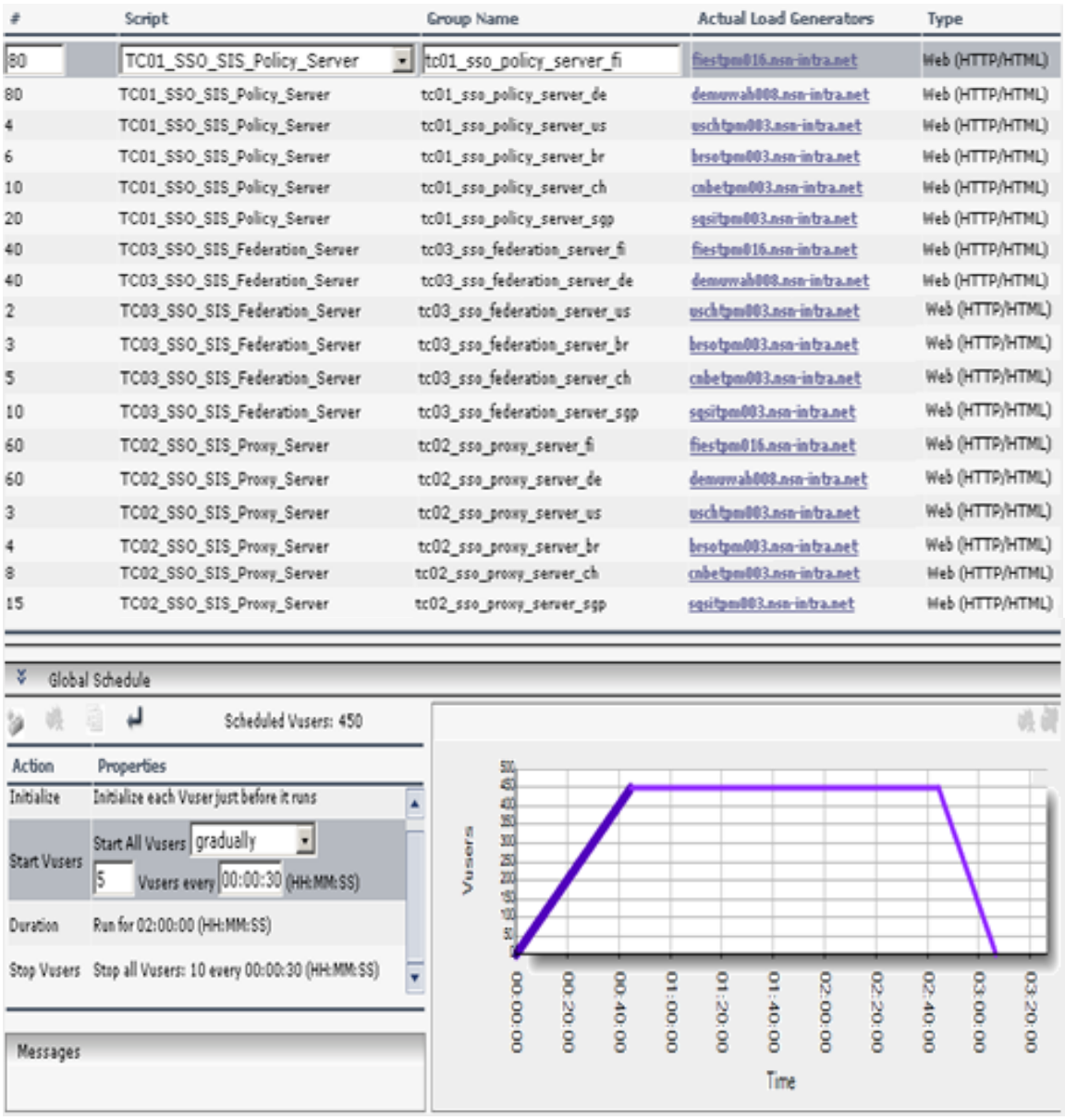


Figure 7: Load Model – Peak Load Test

In the figure above, each script represents one user workflow and includes the details of the number of users assigned to execute each script. It includes the load generator assigned to execute each script apart from the rate of user ramp.

6.5 Test Results

Following are the peak load test results obtained from test execution to meet the main objective, objective priority 1 set during test planning phase, of the performance test:

Table 6 presents the statistics obtained during the tests executed in the old data center as well as the new data center:

Table 6: Statistics Summary of Peak Load Test in Old Data Center

Statistics	Old Data Center	New Data Center
Maximum Running Virtual Users:	450	450
Total Throughput (bytes):	35,132,393,747	47,483,775,452
Average Throughput (bytes/second):	3,022,661	4,080,063
Total Hits:	5,382,918	6,619,259
Average Hits per Second:	463.126	568.763

As shown in Table 6, there is a significant gain in the average throughput resulting into overall throughput obtained during test. Faster response from the servers enabled more requests by the virtual users which resulted into higher number of hits per seconds and eventually higher total hits.

Figure 8 presents the pattern of running virtual users during both the tests:

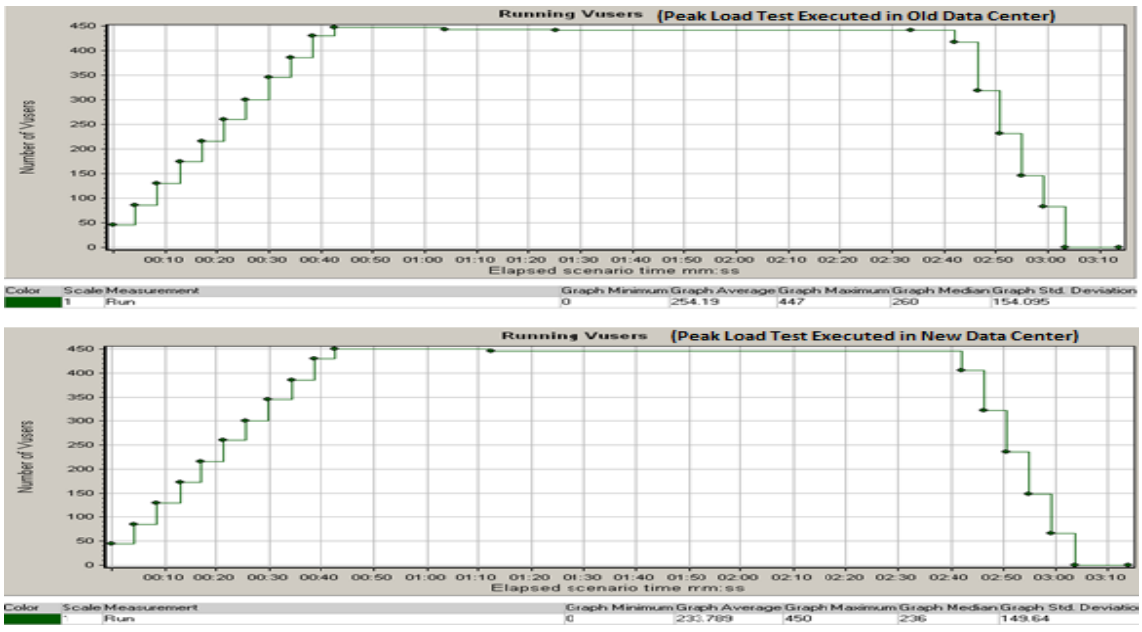


Figure 8: Peak Load Test - Running Virtual Users

As shown in Figure 8, the tests were exactly identical in terms of running virtual users. The total test duration, user ramp-up & ramp-down was also identical in both the tests which would lay the basis of fair comparison.

Figure 9 presents the pattern of throughput obtained during both the tests:

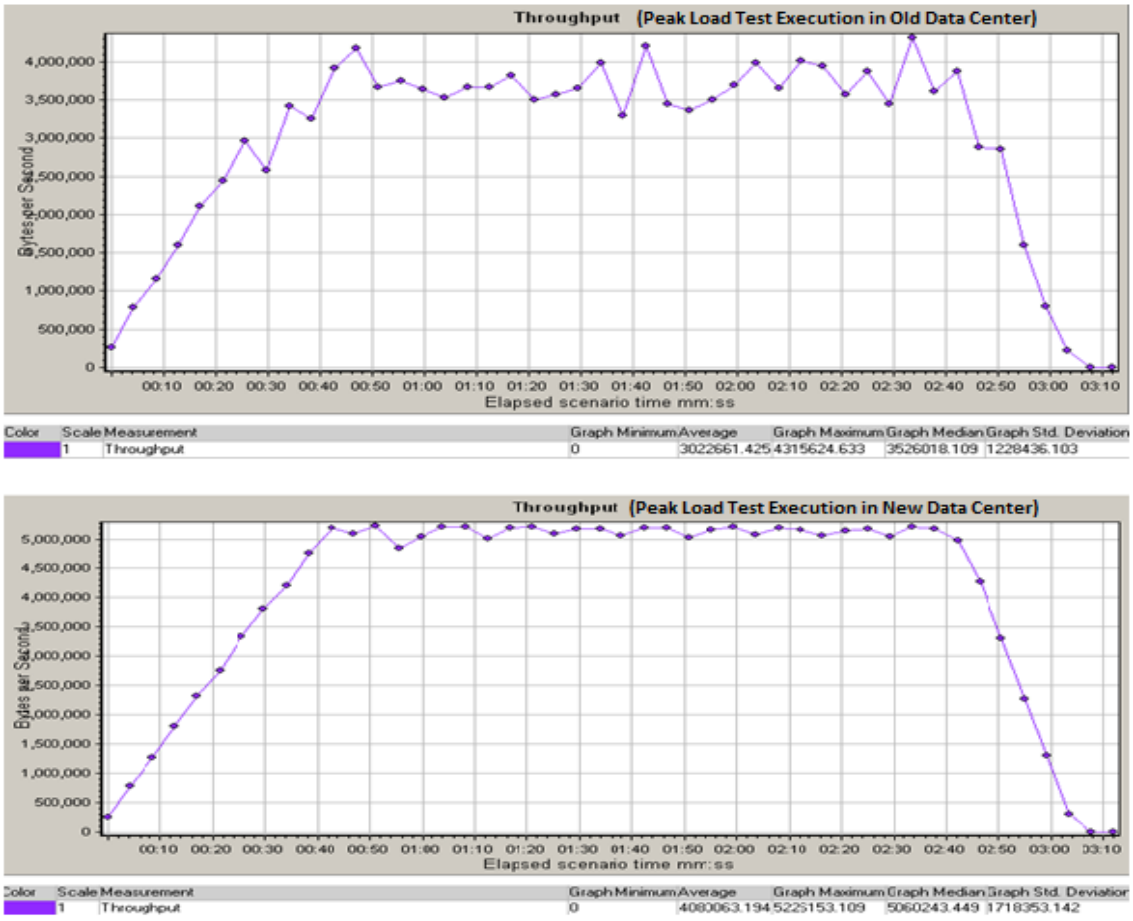
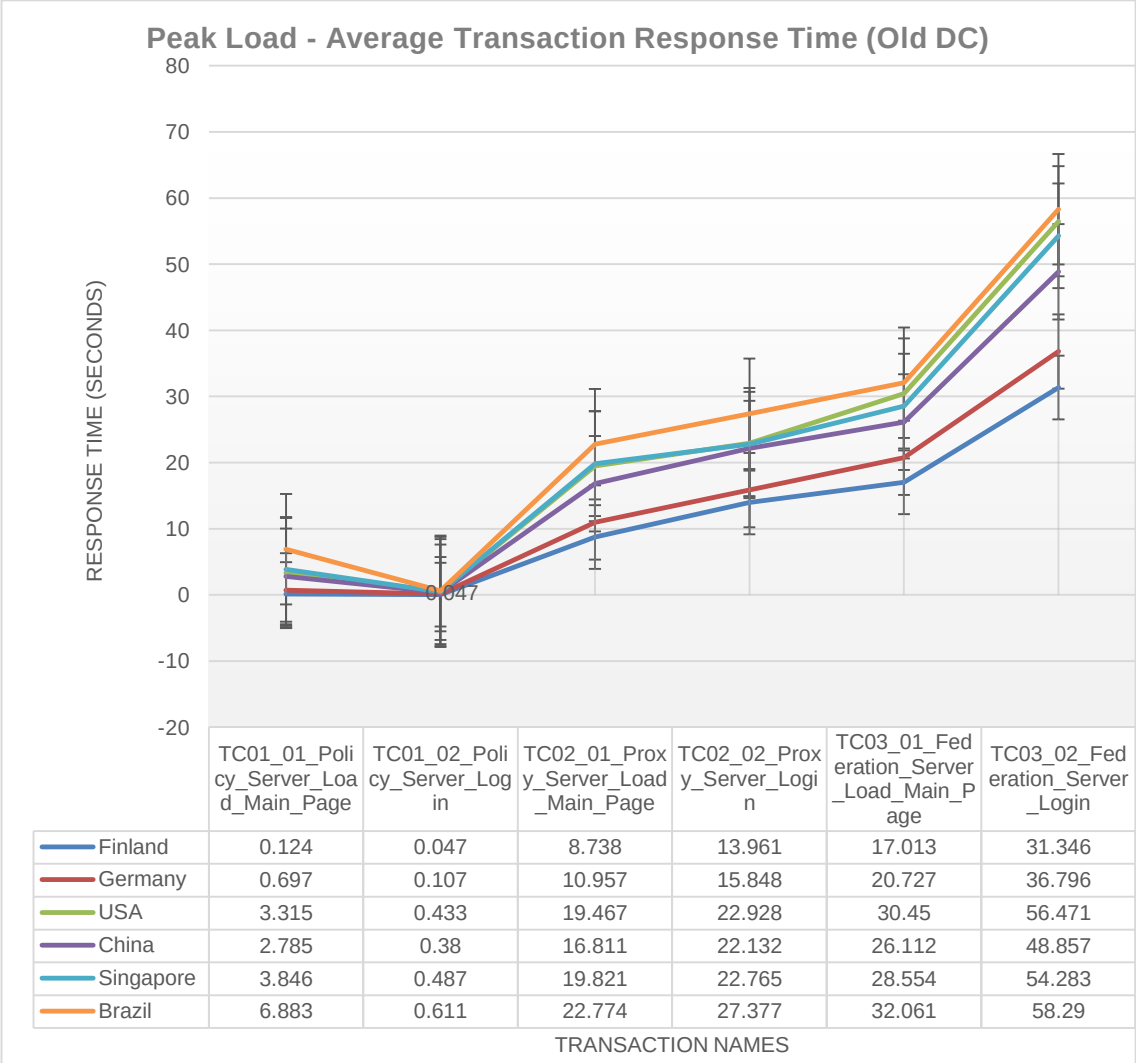


Figure 9: Peak Load Test – Throughput

CAs seen in Figure 9, it is evident that the web servers provided higher and consistent throughput in the new data center throughout the test duration for the exactly identical load model.

The next two figures (Figures 10 and 11) present the average response times obtained from various locations during peak load tests executed in old data center before migration and in new data center post migration. Figures 10 and 11 primarily indicate the variation in user experience from various locations due to the quality of network connectivity since the response is delivered from the same servers.

Figure 10 presents the average response time obtained during a peak load test in the old data center. The location of the servers in the old data center was in Espoo, Finland.

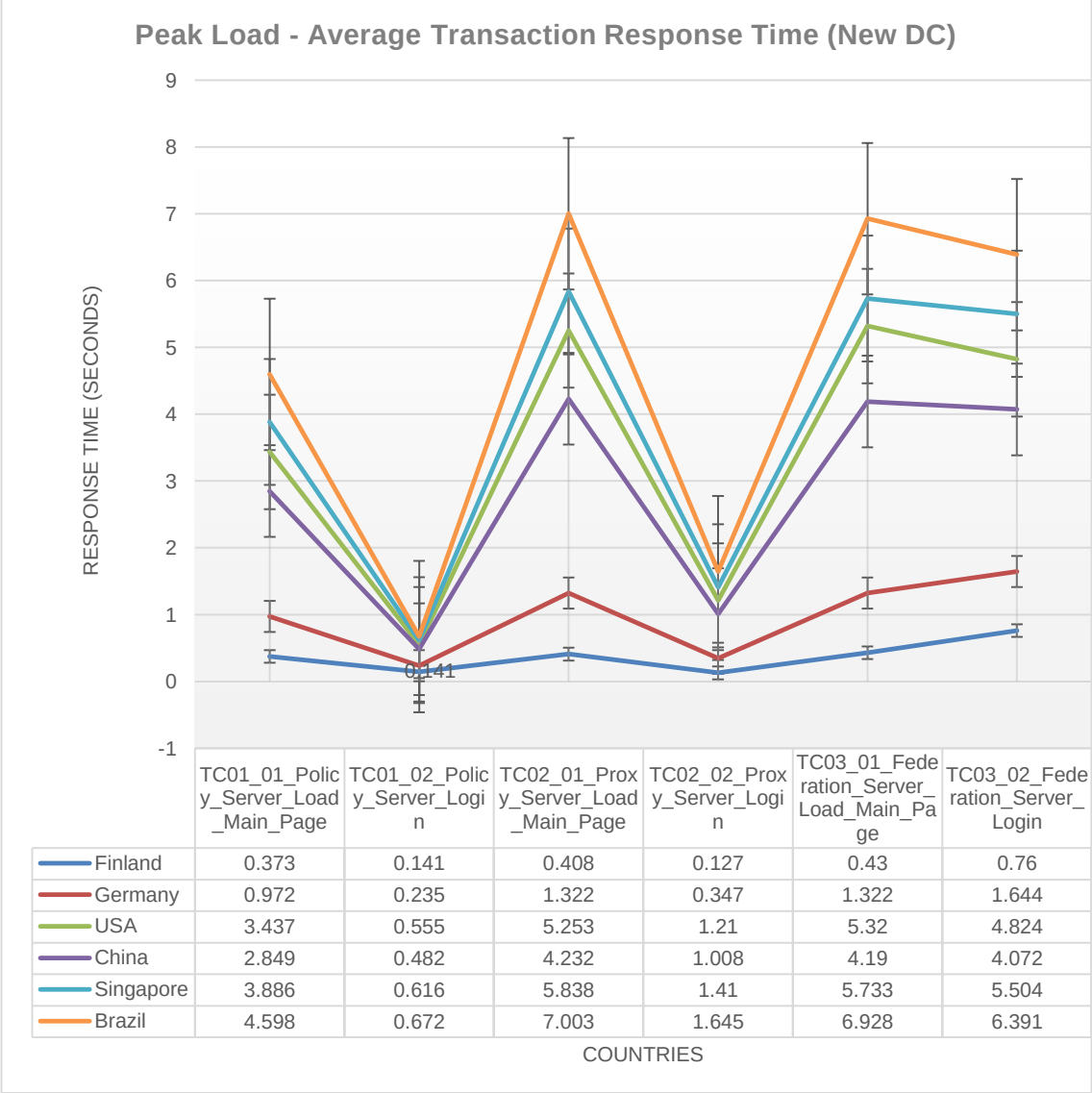


	TC01_01_Policy_Server_Load_Main_Page	TC01_02_Policy_Server_Login	TC02_01_Proxy_Server_Load_Main_Page	TC02_02_Proxy_Server_Login	TC03_01_Federation_Server_Load_Main_Page	TC03_02_Federation_Server_Login
Finland	0.124	0.047	8.738	13.961	17.013	31.346
Germany	0.697	0.107	10.957	15.848	20.727	36.796
USA	3.315	0.433	19.467	22.928	30.45	56.471
China	2.785	0.38	16.811	22.132	26.112	48.857
Singapore	3.846	0.487	19.821	22.765	28.554	54.283
Brazil	6.883	0.611	22.774	27.377	32.061	58.29

Figure 10: Peak Load - Average Transaction Response Time from Old Data Center

Figure 10 represents the average response times obtained for the transactions from various locations. The response times were captured, in order of their performance results (Fastest Location First), from Espoo, Finland; Munich, Germany; Beijing, China; Singapore; Chicago, USA and Sao-Paulo, Brazil. The highest time was taken from Brazil & USA.

Similarly, Figure 11 presents the average response times obtained from the same locations. The location of the servers in the new data center is distributed in Espoo, Finland and Munich, Germany.



	TC01_01_Policy_Server_Load_Main_Page	TC01_02_Policy_Server_Login	TC02_01_Proxy_Server_Load_Main_Page	TC02_02_Proxy_Server_Login	TC03_01_Federation_Server_Load_Main_Page	TC03_02_Federation_Server_Login
Finland	0.373	0.141	0.408	0.127	0.43	0.76
Germany	0.972	0.235	1.322	0.347	1.322	1.644
USA	3.437	0.555	5.253	1.21	5.32	4.824
China	2.849	0.482	4.232	1.008	4.19	4.072
Singapore	3.886	0.616	5.838	1.41	5.733	5.504
Brazil	4.598	0.672	7.003	1.645	6.928	6.391

Figure 11: Peak Load - Average Response Times from New Data Center

Following is the performance of the locations (lowest last), as shown in the Figure 11:

1. Espoo, Finland
2. Munich, Germany
3. Beijing, China
4. Chicago, USA
5. Singapore
6. Sao-Paulo, Brazil

The location of the servers explains the fastest performing locations to be Finland & Germany during both the tests. Table 7 provides further insight to the change in performance from various locations:

Table 7: Load Test Response Times Comparison – Old vs New Data Center

Transaction Name	Old DC	New DC	Response Time Difference	Change Percentage
Finland Location				
TC01_01_Policy_Server_Load_Main_Page	0.124	0.373	0.249	200.806
TC01_02_Policy_Server_Login	0.047	0.141	0.094	200
TC02_01_Proxy_Server_Load_Main_Page	8.738	0.408	-8.33	-95.331
TC02_02_Proxy_Server_Login	13.961	0.127	-13.834	-99.09
TC03_01_Federation_Server_Load_Main_Page	17.013	0.43	-16.583	-97.473
TC03_02_Federation_Server_Login	**31.346**	**0.76**	-30.586	-97.575
Germany Location				
TC01_01_Policy_Server_Load_Main_Page	0.697	0.972	0.275	39.455
TC01_02_Policy_Server_Login	0.107	0.235	0.128	119.626
TC02_01_Proxy_Server_Load_Main_Page	10.957	1.322	-9.635	-87.935
TC02_02_Proxy_Server_Login	15.848	0.347	-15.501	-97.81
TC03_01_Federation_Server_Load_Main_Page	20.727	1.322	-19.405	-93.622
TC03_02_Federation_Server_Login	**36.976**	**1.644**	-35.332	-95.554
USA Location				
TC01_01_Policy_Server_Load_Main_Page	3.315	3.437	0.122	3.68
TC01_02_Policy_Server_Login	0.433	0.555	0.122	28.176
TC02_01_Proxy_Server_Load_Main_Page	19.467	5.253	-14.214	-73.016
TC02_02_Proxy_Server_Login	22.928	1.21	-21.718	-94.723
TC03_01_Federation_Server_Load_Main_Page	30.45	5.32	-25.13	-82.529
TC03_02_Federation_Server_Login	**56.471**	**4.824**	-51.647	-91.458
China Location				
TC01_01_Policy_Server_Load_Main_Page	2.785	2.849	0.064	2.298
TC01_02_Policy_Server_Login	0.38	0.482	0.102	26.842
TC02_01_Proxy_Server_Load_Main_Page	16.811	4.232	-12.579	-74.826
TC02_02_Proxy_Server_Login	22.132	1.008	-21.124	-95.446
TC03_01_Federation_Server_Load_Main_Page	26.112	4.19	-21.922	-83.954
TC03_02_Federation_Server_Login	**48.857**	**4.072**	-44.785	-91.665
Singapore Location				
TC01_01_Policy_Server_Load_Main_Page	3.846	3.886	0.04	1.04
TC01_02_Policy_Server_Login	0.487	0.616	0.129	26.489
TC02_01_Proxy_Server_Load_Main_Page	19.821	5.838	-13.983	-70.546
TC02_02_Proxy_Server_Login	22.765	1.41	-21.355	-93.806
TC03_01_Federation_Server_Load_Main_Page	28.554	5.733	-22.821	-79.922
TC03_02_Federation_Server_Login	**54.283**	**5.504**	-48.779	-89.861
Brazil Location				
TC01_01_Policy_Server_Load_Main_Page	6.883	4.598	-2.285	-33.198
TC01_02_Policy_Server_Login	0.611	0.672	0.061	9.984
TC02_01_Proxy_Server_Load_Main_Page	22.774	7.003	-15.771	-69.25
TC02_02_Proxy_Server_Login	27.377	1.645	-25.732	-93.991
TC03_01_Federation_Server_Load_Main_Page	32.061	6.928	-25.133	-78.391
TC03_02_Federation_Server_Login	**58.29**	**6.391**	-51.899	-89.036

NOTE: Green color represents difference in response times whereas Red color represents response time increase.

As shown in Table 7, there are some locations which are impacted negatively as well but the data center migration overall has brought more performance gains than loss. Germany gained significantly as the network traffic was redirected locally to the servers hosted in Munich data center. The user experience from USA also improved significantly after the data center migration.

Figure 12 presents the comparison of transactions 90 percentile response time, which represents the user experience of 90% users during test executions:

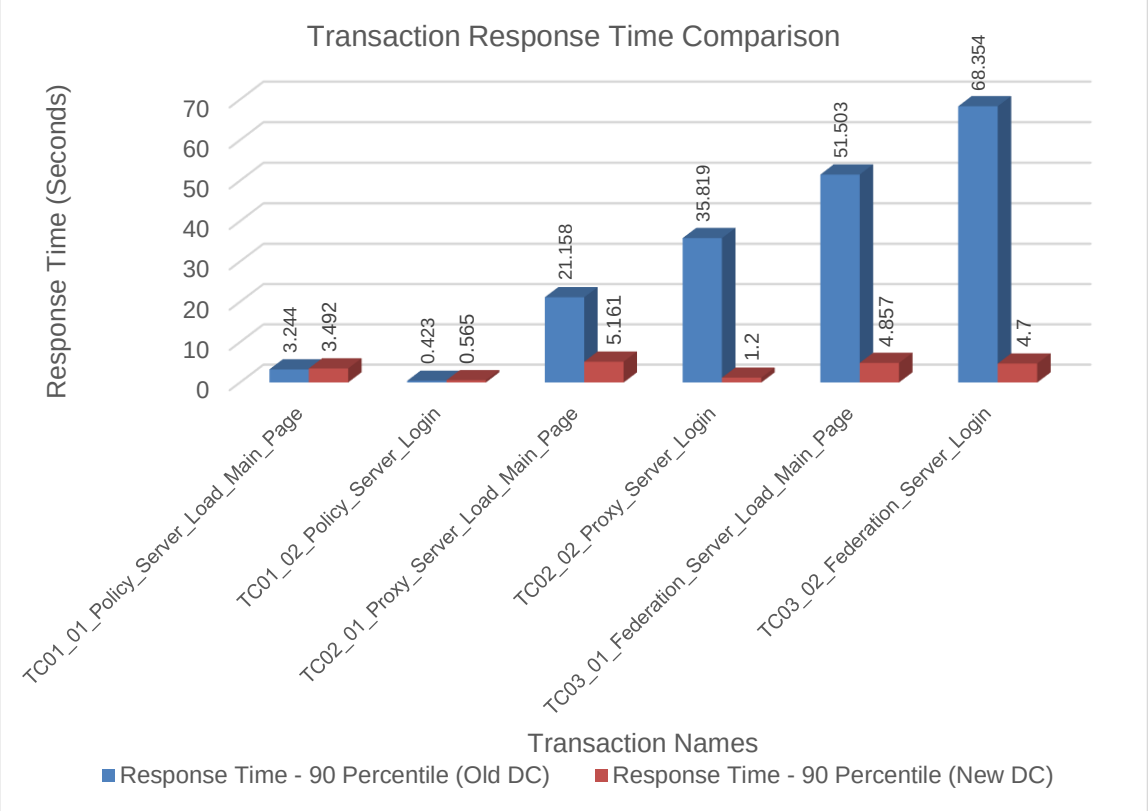


Figure 12: Transaction Response Time Comparison

As evident in the Figure 12 above, there is a significant performance improvement in Web SSO after datacenter migration. Both proxy and federation servers gained performance significantly. Though policy servers show performance degrade in fraction of seconds which can be ignored as it does not impact the user experience.

7 Results Comparison

This section provides the brief details of the results available from the old performance test(s) conducted on the Web SSO application and the detailed results of the recent test executions performed as a part of the case study apart from their comparative analysis.

7.1 Brief Details of Old Test Results

As specified in the initial state analysis of the case project, the following test types were executed to meet the single goal mentioned in the service document i.e. to analyse the performance of the application after the new release:

1) Shakedown test- Supplementary test to ensure scripts and environment readiness.
2) Baseline Test – Supplementary test carried out with less than 20% of peak load to baseline the performance of the application.
3) Load Test – The Load test with 400 virtual users for the duration of 2-3 hours. The load was generated from Finland location only.

The load test was the main test planned to meet the target or customer request to evaluate the performance of the system at full load. There is no documentation available about how the conclusion of 400 virtual users was drawn. Also there is no evidence or documentation about historical data analysis or how the load model was derived. One of the aspects to test after the new release to compare the performance with old release in order to establish whether performance is degraded with the new release and to measure how much was not included in the scope.

7.2 Summary of New Test Results

This section presents the recent test execution results on the single sign on system after implementing the methodology proposed in the solution section which consists of a series of process trainings and documentation within the case department of the organization.

For the goals identified and listed in Section 4.1.5 the tests below were executed with their brief result summary details:

1) Shakedown test: This test is executed to verify that the test scripts and environment is working properly. The test is executed successfully with 2 virtual users per script i.e. 18 scripts in total for 30 mins and without any script errors.

2) Smoke test: This test is executed before main load test to verify that the designed load model achieves the desired rate of key transactions as well as intended rate of HTTP request to the server. After couple of executions and tuning the values of think time and pacing within different scripts the rate of login request was attained to around the target value of 2526 logins per minutes excluding ramp and down period i.e. during steady state test duration of 1 hour.

3) Load Test- Pre Migration: In order to collect the baseline response times for the key transactions before migration for the comparison purposes. The test was successfully conducted using 450 users for 18 scripts and by attaining peak load on 2526 requests/minute. The test duration is 2 hours excluding ramp up and down periods.

4) Peak Load Test- Post Migration: This test is exactly identical to load test before migration. The details and comparison is captured in Section 6.5.

5) Site Failover Testing: The primarily target of the test was to validate the functioning of DNS load balancing between data centers for the SSO application. One of the application load balancer was shut down for 10 minutes, which was acting as one node to receive traffic from DNS load balancer. This test was successful as the servers underneath the restarted application load balancer started receiving login requests through DNS load balancer.

6) Application Failover Testing: The primarily target of the test was to validate the functioning of application load balancer. During this test one node of proxy, policy and federation services were brought down for 10 minutes and started again. The test was successful for federation and proxy servers and as the servers underneath the restarted application load balancer started receiving login requests from corresponding load balancer on restarting.

Core finding of application failover test: This test failed and it was found out that only single node was handling all the login requests from the load balancer and the failover server was not sharing the load on successful restart. The load balancer misconfiguration was identified and resolved before further test was planned.

7) Endurance Testing: The test was executed for 48 hours, over the weekend so that the test does not affect other applications. Approximately, 2600 login requests/minute were maintained throughout the test duration and no major issues were reported. The system was able to handle the peak load consistently for 48 hours.

8) Stress Testing: The test is planned in steps by increasing 100 virtual users in each step on top of 450 peak load to find out the system limits in terms of number of requests it could handle. Also, the test could reveal the symptoms of the system before crashing. The test has not been completed till completion of the study.

The detailed test results of peak load test have been captured in Section 6.5 whereas all the remaining tests have been summarized above. Application failover test identified an issue with load balancer configuration.

7.3 Comparative Analysis of Results

There are the two main following comparison levels in the case study project of single sign on system performance tested before and after implementing the new methodology and the documentation produced as a part of the study:

7.3.1 Project Level Comparison

The project level comparison primarily includes the comparison of the method of deriving performance testing requirements or goals in order to conclude the number of tests with their objectives and targets. Following is the summary (Table 8):

Table 8: Project Level Comparison

Comparison Aspect	Description	Previous Test	New Test	Comparison Remarks
Requirements Identified	Number of requirements or goals identified during requirement gathering phase of the project	One	Three	Apart from the primary goal mentioned in the service request, the questionnaire and checklist helped to identify secondary and tertiary goals also identified. It lead to major difference in the project execution and affected the shape of project totally i.e. the number of tests and their benefits which would have easily missed otherwise as in the case of previous release.
Quality of load model	Usage of historical data in designing the load model	The load model was not based on historical data analysis	Historical data analyzed to design workload	The load model affects performance prediction of the production system and the test executed previously was not based on the historical data analysis
Test types included in scope	Incl. only main tests (Excl. supporting tests like shakedown, baseline and smoke tests	One [Load Test- Post Release]	Six [Pre, Post Migration Load tests; Site, Application failovers; Endurance and Stress Test]	Identification of detailed requirements lead to higher number and types of tests which helps in analyzing the performance of the production system

As evident from the comparison table above, the requirements were gathered properly in the similar project by using the methods defined in the new methodology section which resulted in to higher number of requirements and more test executions.

7.3.2 Test Level Comparison

Test level comparison includes the differences observed in the quality of a single test i.e. load test which is common in both the projects. Following is the summary (Table 9):

Table 9: Test Level Comparison

Test Type	Previous test	New Test	Comparison Remarks
Load generating locations	One [Finland]	Six [Finland, Germany, Singapore, Brazil, USA and China]	This helped in understanding the response times of the key transactions from global locations when application is under peak load
Quality of load model	The basis of load is not documented and was not based on historical data analysis	Load model is based on the historical data from the production systems	The new methodology emphasize on the usage of historical data for load modeling which results into a load model close to production load means realistic results and better prediction of production performance
Number of test cases	One	Three	Due to some scripting/technical challenges only single script was executed in the previous test as compared to three scripts for each type of authentication was used in the new test
Performance comparison method	The comparison was based on baseline test executed at 20% of the peak load	The comparison was based on the baselines collected from the old data center at the same/peak load using the exact same scenario	The method used in the new test provide clear comparison of the performance and clear isolation of the fact if the performance is improved or degraded

It is evident from the table above that the new methodology brought the quality change in the project in terms of results obtained from individual test executions. During the case study a very clear comparison was made whereas in the previous test execution baseline was not executed in the prior release for comparison with new release performance. Also, many new tests were introduced in scope using new methodology which would have otherwise not executed.

8 Summary

This section contains the summary of the study, practical implications for the proposed method and the next possible steps for its future operational implementation. It also contains the evaluation of the study by comparing the outcome with the initial research objective. Finally, validity and reliability of the study are discussed.

The main goals of the study was to explore and propose the possible ways to improve the quality of the requirement gathering process being followed in the performance testing service of the case company. The other goals of the study were to provide the supporting documentation to improve the initial steps involved in any performance testing projects to support the performance engineers in deriving the effective performance requirements as well as utilize those requirement to further identify the test types and their load models.

The study produced a series of documentation ranging from the new methodology presentations, various checklists and questionnaires to be used during the requirement gathering process, a handbook to use the tool in order to analyze the application historical data to design the effective load model and a guide including reference to all available documentation for an overall performance test process.

As per the feedback from peers and the management, there is no requirement to modify the overall service level process as it starts quite early in the development life cycle being followed in the case company which gives enough time to performance engineers to implement the new methods and documentation.

The implementations show very promising results (as summarized in Section 7.3). The identification of two new goals based on the usage of new methodology and documentation lead to major change in the course of project. It affected all the following stages of the projects until its completion and major improvements in the results of the test executions. It resulted in the derivation of 7 more tests, the inputs from each test execution helped in better predicting the performance of the application in its production environment. The application failover test which might not have been in the scope of the study if only the primary goal mentioned in the service request would have been considered. It means that the test would have been failed in identifying the policy server load balancing misconfiguration issue.

The proposed methodology would require some basic changes in the way of working within the shared service, the new methodology introduces many new documents to be followed and also emphasizes reviews after each milestone. It would take some time for the management to decide as well as people to adapt to it before it could be fully implemented.

This study helps the service management to understand for example how the methods and documentation could help in improving the quality of the performance testing service being offered. As per the initial discussion about its implementation, it could be divided into two parts i.e. further trainings on how to use the new methods and in later stage again implement it for another project before including it fully in to the process.

This study could be further expanded outside the existing scope for web applications only to cover the other systems. Also, from the case project perspective the study does not include the details of the second part of the project about performance tuning of the single sign on system. This exercise in general is expected to maximize the gain one could see from performance tests.

This section evaluates the outcome of the project compared against the research objective defined at the beginning. Additionally, validity and reliability of the study are evaluated and compared to the requirements defined in Section 4.3.

Following is the summary of the targeted requirements and their mapping with the outcome of the study (Table 10):

Table 10: Requirement Fulfillment Status

S. No.	Requirement	Status	Remarks
1.	Study and propose the improvements for performance requirement gathering process being followed at service level	Fulfilled	As an outcome service level process remains unchanged but method of requirement gathering is proposed to change as detailed in section-5
2.	Study and propose the method to improve load modelling method being followed in the service	Fulfilled	Section 5.5 to guide how to derive effective workload model and section 5.4 about newly created tool handbook to analyze web logs would help in fetching the data based input for effective load modelling
3.	Study and implement the proposal to some specific project to evaluate the effectiveness	Fulfilled	Section 6 and 7 covers the details of the case study and the corresponding results

To increase the reliability and validity of the outcome, more interviews and feedback would definitely benefit the evaluation and might lead to more concrete recommendations for the shared service. It was not, however, possible due to time constraints and other business factors to include such statistics in the present study.

Overall, the performance is truly a big question in the constantly evolving IT world and doing the performance testing right is a crucial topic. Hopefully the present study added a step up in this direction.

References

1 [1] Avritzer, A., J. Kondek, D. Liu, and E. J. Weyuker, "Software Performance Testing Based on Workload Characterization," in Proceedings of the 3rd International Workshop on Software and Performance, pp. 17-24, Rome, Italy, Jul 2002.

2 [2] Chih-Wei Ho, North Carolina State University and Laurie Williams, North Carolina State University “Deriving Performance Requirements and Test Cases with the Performance refinement and Evaluation Model” dissertation August 2008

3 [3] Avritzer, A., ATandT Labs and E.J. Weyuker, ATandT Labs “Deriving Workloads for Performance Testing”, Software -- Practice and Experience, Vol. 26, No. 6. June 1996, pp. 613-633.

4 [4] Avritzer, A. and E. J. Weyuker, "Generating Test Suites for Software Load Testing," in Proceedings of International Symposium on Software Testing and Analysis, pp. 44-57, Seatle, WA, Aug 1994.

5 [5] Balsamo, S., A. D. Marco, and P. Inverardi, "Model-Based Performance Prediction in Software Development: A Survey," IEEE Transactions on Software Engineering, vol. 30, no. 5, pp. 295-310, May 2004.

6 [6] Daniel A. Menasce and Virgilio A.F. Almeida; “Capaciry Planning for Web Perfomance Metrics, Models, and Methods”, Prentice Hall, PTR, New Jersey 07458, 1998.

7 [7] Menasce D.A. and Dowdy L.W., “Capaciry Planning and Performance Modeling: From Mainframes or Clienr-Server Systems’’, Prentice Hall, NJ, 1994.

8 [8] Bill Jaegcr, “Minimizing Risk with Proactive Performance Testing”, E-Business Advisor Article, April 1999.

9 [9] “Microsoft Site Server 3.0 Commerce Server,” Capacity and Performance Analysis manual.

10 [10] Z. Alzamil. Application of the operational profile in software performance analysis. In Proceedings of the 4th international workshop on Software and performance, pages 64–68, Redwood Shores, California, 2004. ACM. ISBN 1-58113-673-0.

11 [11] S. Barber. Creating Effective Load Models for Performance Testing with Incomplete Empirical Data. In Web Site Evolution, 2004. WSE 2004. Proceedings. Sixth IEEE International Workshop, pages 51–59. IEEE Computer Society, 2004. ISBN 0-7695-2224-6.

12 [12] Berry, D. M. and E. Kamsties, "Ambiguity in Requirements Specification," in Perspectives on Requirements Engineering, J. C. S. P. Leite and J. Doorn, Eds.: Kluwer, 2004, pp. 7- 44.

13 [13] J.D. Meier, Carlos Farre, Prashant Bansode, Scott Barber, and Dennis Rea, Microsoft Corporation, “Performance Testing Guidance for Web Applications” September 2007

14 [14] Todd DeCapua, Chief Technology Evangelist, Hewlett Packard Enterprise and Technology Leader, TechBeacon “State of Performance Engineering” 2015-2016 Edition

15 [15] D. G. Feitelson, Workload Modeling for Computer Systems Performance Evaluation. Cambridge University Press, 2015

Glossary

The following definitions are used throughout this guide. Every effort has been made to ensure that these terms and definitions are consistent with formal use and industry standards; however, some of these terms are known to have certain valid alternate definitions and implications in specific industries and organizations.

Table 11: Glossary

Term/Concept	Description
Latency	Latency is a measure of responsiveness that represents the time it takes to complete the execution of a request. Latency may also represent the sum of several latencies or subtasks.
Metrics	Metrics are measurements obtained by running performance tests as expressed on a commonly understood scale. Some metrics commonly obtained through performance tests include processor utilization over time and memory usage by load.
Pacing	The duration of pause at the end of any iteration before starting the new iteration
Response time	Response time is a measure of how responsive an application or subsystem is to a client request.
Scenarios	In the context of performance testing, a scenario is a sequence of steps in your application. A scenario can represent a use case or a business function such as searching a product catalogue, adding an item to a shopping cart, or placing an order.
Think Time	The user pause between two user actions on the web page. The referred in this case results into a request to the server.
Throughput	Throughput is the number of units of work that can be handled per unit of time; for instance, requests per second, calls per day, hits per second, reports per year, etc.
Transaction	The transaction means a user action, which resulted into a web/http request to the server.
Resource utilization	Resource utilization is the cost of the project in terms of system resources. The primary resources are processor, memory, disk Input/Output, and network Input/Output.
Virtual User	A software process or thread which simulate the user actions (in terms of web requests) to the web server
Workload	Workload is the stimulus applied to a system, application, or component to simulate a usage pattern, about concurrency and/or data inputs. The workload includes the total number of users, concurrent active users, data volumes, and transaction volumes, along with the transaction mix. For performance modelling, you associate a workload with an individual scenario.