HELSINKI UNIVERSITY OF TECHNOLOGY
Faculty of Electronics, Communications and Automation

Antti Piispanen

PERFORMANCE OF COMMUNICATIONS ENABLED BUSINESS PROCESS ENGINE – TESTING PROCESS

Master's thesis submitted in partial fulfillment of the requirements for degree of Master of Science in Technology.

Espoo, 25 May 2009

Supervisor of the Thesis: Professor Raimo Sepponen
Instructor of the Thesis: Kari Immonen, M.Sc.

HELSINKI UNIVERSITY OF TECHNOLOGY Abstract of Master's Thesis

Author:	Antti Piispanen	
Title of the Thesis:	Performance of Communications Enabled Business Process engine – Testing Process	Number of pages: 69
Date:	May 25 2009	

Faculty: Faculty of Electronics, Communications and Automation

Department: Department of Electronics

Professorship: S-66 Applied electronics

Supervisor: Professor Raimo Sepponen

Instructors: Kari Immonen, M.Sc. (Tech.)

Quality assurance is an essential part of software development process. Ensuring the fulfillment of functional requirements is often well defined in testing processes, but definition of systematic activities for ensuring software performance is easily omitted due the lack of time and resources. Performance related problems cause significant costs if detected in late phase of development or in production. Problems of this kind can be avoided or consequences of these problems reduced with proper performance testing process.

In this Master's thesis, performance testing process for Communications Enabled Business Process engine is defined based on requirements defined for performance testing process and system performance. Business process models for most important use cases of system applications are also defined. Defined performance testing process is integrated with company's standard testing processes.

Implementing performance testing as a part of testing processes enables continuous quality assurance through product life cycle. Defined performance testing process enables continuous performance improvement between product versions. Scaling guide document produced by performance testing process enables customer to optimize the installation of productive system which may lead to significant hardware cost savings.

Keywords: Software testing, quality assurance, software performance, performance testing, performance testing process

TEKNILLINEN KORKEAKOULU Diplomityön tiivistelmä

Tekijä: Työn nimi: Päivämäärä:	Antti Piispanen CEBP järjestelmän suorituskyky – testausprosessi 25.5.2009	Sivumäärä: 69
Tiedekunta: Laitos: Professuuri:	Elektroniikan, tietoliikenteen ja automaation tiedekunta Elektroniikan laitos S-66 Sovellettu Elektroniikka	
Työn valvoja:	Professori Raimo Sepponen	
Työn ohjaaja:	Diplomi-insinööri Kari Immonen	

Laadun varmistus on elintärkeä osa ohjelmistotuotekehitysprosessia. Toiminnallisten vaatimusten täyttymisen varmistaminen on usein hyvin määritelty testausprosesseissa, mutta systemaattiset aktiviteetit ohjelmiston suorituskyvyn varmistamiseksi jäävät usein määrittelemättä johtuen ajan ja resurssien puutteesta. Suorituskykyyn liittyvät ongelmat aiheuttavat merkittäviä kustannuksia havaittaessa vasta myöhäisessä vaiheessa tuotekehitysprosessia tai tuotannossa. Tämän tyyppiset ongelmat voidaan välttää tai niiden vaikutuksia vähentää kunnollisen suorituskykytestausprosessin avulla.

Tässä diplomityössä määritellään suorituskykytestausprosessi CEBP (Communications Enabled Business Process) järjestelmälle pohjautuen suorituskykytestausprosessille sekä järjestelmän suorituskyvylle määriteltyihin vaatimuksiin. Myös liiketoimintaprosessimallit määritetään tärkeimmille järjestelmän sovellusten käyttötapauksille. Määritelty suorituskykytestausprosessi integroidaan osaksi yrityksen standardeja testausprosesseja.

Suorituskykytestauksen ottaminen mukaan osaksi testausprosesseja mahdollistaa jatkuvan laadun varmistuksen tuotteen koko elinkaaren aikana. Määritelty suorituskykytestausprosessi mahdollistaa jatkuvan järjestelmän suorituskyvyn parantumisen tuoteversioiden välillä. Prosessin tuottama mitoitusohje mahdollistaa asiakkaiden tuotantojärjestelmäasennuksien optimoidun mitoituksen, mikä voi johtaa merkittäviin laitteistokustannussäästöihin.

Avainsanat: Ohjelmistotestaus, laadun varmistus, ohjelmiston suorituskyky, suorituskykytestaus, suorituskykytestausprosessi

Foreword

I wish to express my gratitude for all the people who have supported me in this work. First, I would like to thank Professor Raimo Sepponen for good feedback during this work and for encouraging me to complete this work in schedule. This work was instructed by M.Sc. Kari Immonen whose interesting ideas, advices and feedback helped me along this work and deserve thanks.

I would also like to thank my beloved wife Tanja for her patience and support during this work. I will thank also my mother for supporting me along my studies. And special thanks for our beautiful daughter Jemina, your smile at the end of every hard day always made all worth it.

Espoo, May 25, 2009

Antti Piispanen

Table of Contents

Abbreviations

CEBP	Communications Enabled Business Process
CPU	Central Processing Unit
CRM	Customer Relations Management
ERP	Enterprise Resources Planning
IP	Internet Protocol
ISDN	Integrated Services Digital Network
LAN	Local Area Network
PDA	Personal Digital Assistant
PSTN	Public Switched Telephony Network
QA	Quality Assurance
RTP	Real-Time Transport Protocol
SIP	Session Initiation Protocol
SPE	Software Performance Engineering
UI	User Interface
VoIP	Voice over Internet Protocol

1 Introduction

Extensive performance testing is often left out from software development process although the consequences of performance related problems may be critical when occurring in production. Late detection of these problems may cause significant costs compared to situation where defects have been detected and corrected in early phase of development. To ensure the proper performance of the software, performance testing has to be introduced in standard testing process.

The correct operation of the software is defined in requirements. Functional requirements for software are typically well defined and correct functionality assured by proper functional testing. Non-functional requirements, performance as one of them, are often loosely defined or definition of these requirements is completely omitted. System performance requirements should be defined early in design phase simultaneous with other key requirements. If these requirements do not exist they can be defined based on information on system's environment of use and workloads.

Performance requirement for software system can be relatively straightforward to define if environment of use, sizing of the system and workload are known. Definition of requirements for software performance becomes more complex when considering product having multiple applications, options for installation and configuration depending on customer. Communications Enabled Business Process engine, considered in this thesis work, is a software product adapted and configured according to customers' communications intensive business needs. Therefore earlier mentioned parameters (environment, sizing, and workload) are not known in advance. The most important applications of the system are defined with business process models which are described in chapter 3.4 in detail. These models describe the environment and events for the system in normal circumstances. Designing performance testing based on these models enables efficient testing implemented by focusing on the scenarios in these models and produces information on the performance of the system in real environment of use. Howev-

er more detailed information on system and its components performance and capability to handle different types of loads is also valuable. Therefore testing based on so called technical processes is also needed in addition to business process model based testing. However, the most important task of performance testing is to ensure the proper operation of the system in all environments and circumstances it has been designed to operate and to provide information on system capabilities and weaknesses.

The Performance testing process for Communications Enabled Business Process system described in this thesis combines these two aspects with objective to achieve balance between business and technical orientated performance testing.

1.1 Objective

The objective of this thesis work is to define performance testing process for the Communications Enabled Business Process engine. The requirements for Performance testing process and for performance of the system are defined. The definition of requirements is based on the literature study and information gathered from the company internally. This internal information consists of facts on business cases, information on needed software performance related information for development (feedback) and information on needed software performance and scalability related information for production (system scaling). Business process models used for business process based performance testing are also defined.

Testing process including test plans and test cases is designed to fulfill the defined requirements. Scheduling for performance testing is defined. Also resources for performance testing, such as personnel needed for test plan execution; environment for test system and tools for load simulation and monitoring system and system performance are described. Integrating the performance testing process into standard testing processes is defined. Performance process produces a scaling guide document based on test results. The document enables optimized sizing of customer productive systems.

1.2 Scope

Scope of this thesis is limited in defining the structure of performance testing process for the Communications Enabled Business Process engine. Definition of the requirements, business process models, test plans, performance testing scheduling and staffing in development process is included.

The detailed description of tests and test results neither detailed performance requirement values specific to actual product are not included in this thesis.

1.3 Structure

In second chapter Communications Enabled Business Process engine is described and methods for performance testing and performance analysis investigated. In third chapter requirements for performance testing process and system performance are defined. Also the most essential business process models describing the environment configuration and events in key use scenarios are defined.

Performance testing process including process description, test plans, test cases, testing tools description and process scheduling is included in fourth chapter. In fifth chapter, deployment of performance testing process into the organization is discussed. Sixth chapter includes conclusions about the defined performance testing process and suggestions for future.

2 Communications Enabled Business Process engine Performance

2.1 Communications Enabled Business Process engine

Performance testing process defined in this document is designed for the Communications Enabled Business Process (CEBP) system similar to one developed in the company the thesis work was done for. This chapter describes the simplified structure and basic features of the system.

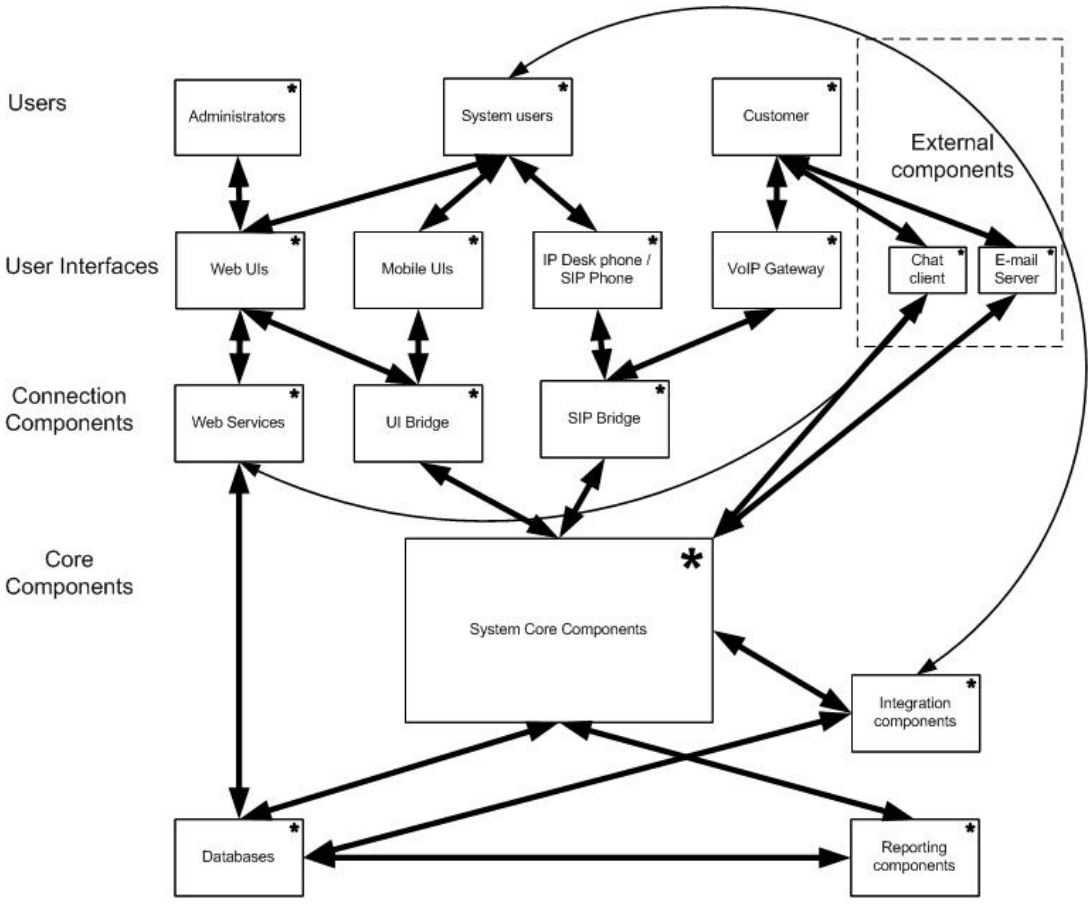

Figure 1 Simplified structure of the Communications Enabled Business Process system

The CEBP system combines customer company's business processes with communications processes by connecting all business related communications events into one controllable system. The implementation of telephone system features is fully based on Voice over Internet Protocol (VoIP) technology. In addition to phone calls, the system also enables other communications channels such as chat and e-mail. The system can be used as a contact center solution in customer service or telemarketing office or enterprise telephony solution in any company. The system enables customers to implement their business communications needs and make communications business processes more effective. The system is easily configurable and adjustable in different kinds of communications intensive business needs. Integrating CEBP system to other business process handling systems enables connecting all business processes together and makes interaction and information transfer between business processes effective.

System users can log in the system using different types of terminals such as web user interface, mobile client user interface and SIP phone (e.g. IP desk phone). These internal system users having an account created in the system are called agents. Users placing calls, e-mails or chat request from outside the system to the system are called customers. As the different terminals offer different functionalities for user they also load the system differently. The UIs connect the system core components through different connection components depending on the type of the UI. The core components are the heart of the system and e.g. take care of call routing and connecting etc. Core components communicate with different parts of the system. Calls from external network are connected to the system through SIP VoIP gateway. It is possible for external customer also to send Chat and E-mail messages to the system, using an external chat client or any e-mail application.

The simplified structure of the system and connections between system components is represented in Figure1. Basic system components and basic features of those are described below.

2.1.1 User interfaces

Web UIs

Main user interface (soft phone) and system administration interfaces are implemented as web browser interfaces. The main user interface contains virtual phone and views for other features such as phone-book directory, chat, e-mail etc. The web user interface connects to the system through UI connection component. User specific data (user settings, information on active features, roles, directory data etc.) is loaded from database through web server and UI connection component when user logs in the system with main user interface. Depending on system configuration, UI connection component encrypts and decrypts the data communications between UI connection component and UI. The quantity of loaded data during the logon depends on user settings, roles, assigned queues, directory size etc. Loading this data can cause significant load at the UI connection component and web server if multiple users log in at the same time.

Other web user interfaces (e.g. system administration interface) does not cause such a load during the logon thus only data related to authorization and current view is loaded.

Mobile UIs

Mobile user interface is an application installed on mobile device (e.g. mobile phone or PDA). The mobile interface enables user to logon the system and use most of features as with Web UI using wireless mobile device. The difference with Web UI is the reduced quantity of data transferred to UI and the fact that data is loaded only when needed (e.g. directory data is loaded from the server only if directory search is performed). The data loaded from database is transferred through web server and UI connection component also in case of Mobile UI.

IP desk phones

Agents can use IP desk phones to connect the system. The phones using SIP protocol are supported. IP desk phones connect through SIP Bridge which communicates with SIP phone and transfers the data to system core components.

Integration user interfaces

If the system is integrated to another system, users may use system features via interface integrated to this another system. Communication events data and applications dependent data is transferred between the systems via integration components.

2.1.2 External Customer User interfaces

Phone in Public Switched Telephone Network (PSTN)

Customer can use any phone connected to PSTN to perform an inbound call to the system. The call will be connected through VoIP gateway and SIP Bridge to the system. SIP Bridge establishes the SIP connection and core components handle the call event and route the call to the correct location (agent, queue) and further handle the event if needed (e.g. allocation from queue to agent).

Chat client

Customer can send a chat request in to the chat channel of the system using an external Chat client. System Core components allocate the request to appropriate contact center agent and connect the conversation. Web server loads the messages between chat client and system user UI. The chat client is typically a browser based application which sends and receives chat messages. The chat client could be for example included in a company web site. Contact center agents use Web UI to handle chat sessions.

E-mail

Customer can send an e-mail in to system's e-mail channel using any e-mail application. The e-mails are handled by external e-mail server and collected in to the inbox. Core components check the inbox at the e-mail server in on frequent interval and load new messages in to the system for parsing and allocate the messages to appropriate e-mail queue.

2.1.3 Connection components

UI Connection component

Connection component establish the connection with user interfaces (handshake messaging) and connects them with core components. Connection component also transfers the data from web server internal web service to UI and vice versa. Connection component also performs the encrypting and decrypting of data communications between the system and UIs if configured to do so.

SIP Bridge

SIP Bridge performs SIP registration activities for connections from SIP Phones and SIP VoIP Gateway and connects these further to the system Core components. In case of SIP phone user with agent account created in then system, authorization info is transferred automatically through SIP Bridge to core components and logon is performed within registration. Registration of contact from VoIP gateway is performed as the call arrives in the gateway

Web server

Web server contains web service components for loading web user interfaces (appearance) and for internal web services responsible for loading requested data from database for UI (directory data, user settings etc.) While user logs in the system using main user interface UI requests all data needed in starting of the UI and for some features to work properly (user settings and roles data, directory data etc.). If there are lots of users logging in the system at the same time (lots of simultaneous data queries) the load at the web server may increase noticeably. Also

simultaneous directory searches, simultaneous chat sessions etc. can produce a significant load to web server.

2.1.4 Other components

Core components

Core components handle most of the events in the system. The transaction routing (e.g. allocating calls, chat sessions, e-mails) and processing is done by core components. Also many other applications such as Interactive Voice Response (IVR) applications, outbound campaign management and dialer are included in core components.

Databases

Databases for user data, user status data, system configuration data, data about current system events, directory data, message data and system events history exist for the system. System components store data and perform queries in order to retrieve required information from database.

Reporting and monitoring components

The system contains tools for reporting system events and users. These features can be used for example for monitoring the quality of service in contact center, for collecting data for customer billing system and for other kind of event data analyzing. Reporting components are connected to system core components where the event data is collected from. The collected event data is stored in the database. System reporting and monitoring tools are used also in performance testing to monitor system events during tests.

Integration components to other systems

System can be integrated to other systems such as customer resources management systems or enterprise resources planning systems via integration components. Integration components enable business communications data to be connected to other business process data (e.g. customer call information connected to

customer data in the CRM system). The integration components interact between Core components, other systems and databases. The type of data transferred through integration component vary (call data, agent status data, customer contact data etc.) depending on which kind of system the tested system is connected.

2.1.5 Hardware

Thus the telephony features are fully based on VoIP there is no need for any telephone switching system. Depending on the type and size of system implementation, the system will be installed on couple servers (e.g. two application servers and database cluster). If back up components are needed the extra server is however not essential as backup system components can be installed crosswise so that backup component for each system component is running on another application server.

VoIP gateway connects the packet switched system to circuit switched external network (ISDN, PSTN). VoIP gateway transfers the communications traffic from external network to internal and to SIP Bridge using SIP protocol.

On client side, depending on user interface, agent needs either computer for Web UI, mobile device or PDA for mobile UI or IP Desk phone to be able to log in the system.

2.2 Standard processes in software quality assurance

Quality assurance (QA) is a continuous process in software development and continues through whole life-cycle of the software product. Different QA activities have to be completed in different phases of development before moving to next phase e.g. acceptance testing has to be completed before releasing the software to customer. Basic software quality assurance processes and activities are well described in literature e.g. [Mye et. al 2004] , [Bur 2003], [Far 2008]. Simplified QA process includes following testing levels: unit testing, integration testing, system testing, acceptance testing. All of these activities have to be completed before re-

leasing the product to customer. In addition to these testing activities quality assurance management includes also planning and test design activities which start in early phase on development process.

2.2.1 Levels of testing

Unit testing tests a small unit or module of the software. The purpose of unit testing is to verify that the operation of developed unit or software component is in compliance with detailed design specifications. Defects related to unit functionality and structure are attempted to detect in unit testing. Unit testing is performed soon after development phase and typically performed by developer, (although it should be preferably executed by someone else e.g. testing specialist or another developer [Bur 2003]).

The goal of integration testing is to detect defects that occur on the interfaces of units [Bur 2003]. Integration testing is performed after completing the unit testing. Units are connected together as working subsystems or a complete system and tests are executed. Integration testing should be performed as an iterative process by adding one unit at the time to the group of already integrated and tested units.

After unit testing for all system components and integration testing with subsystems and complete system have been completed, system testing is performed. In system testing the system is tested as whole. System testing is a complex process including several types of tests divided into two groups, functional testing and non-functional testing. Tests in functional testing are black-box testing, all system functions are tested focusing only on system in and outputs. Functional system testing includes the testing of existing functionality by regression testing and testing of new features.

Non-functional system testing includes tests for verifying that non-functional properties of the system meet the defined requirements. An example of test types which could be included in the system testing: performance testing, stress testing, load testing, recovery testing, reliability testing, scalability testing, configuration

testing, installation testing, security testing, usability testing and accessibility testing.

Acceptance testing tests if the system meets all defined (customer) requirements. The testing is performed after the system testing is completed and detected defects corrected. The testing is done by end-users, customer or client. Typically acceptance testing consists of set of tests performed already in system testing.

Performance testing process introduced in this thesis is included in system testing. However software performance properties can be tested also in other test levels. For example capability of separate system component to handle certain inputs can be tested in unit testing pace.

2.2.2 Quality assurance activities in software lice-cycle.

Several models describing the software development process exist. In common, traditional software development process (Waterfall model, V-model, [PflAtl 2009]) following development life-cycle steps are included: defining requirements for software, defining specifications for software components, designing the architecture of the software, designing and implementing software components. Each of these steps is connected to design and definition of corresponding testing activity. After software implementation step, testing activities on different levels are executed (unit testing, integration testing, system testing, acceptance testing). Figure 2 describes how these testing activities are placed on development life-cycle time line in company's standard software testing process. Figure 3 shows how above described development life-cycle steps are connected to testing activities with slightly modified v-model.

After completing the acceptance testing and validation, the product is ready to be released to customer (from QA perspective). In maintenance phase of the program, testing activities include testing of fixed defects and regression testing.

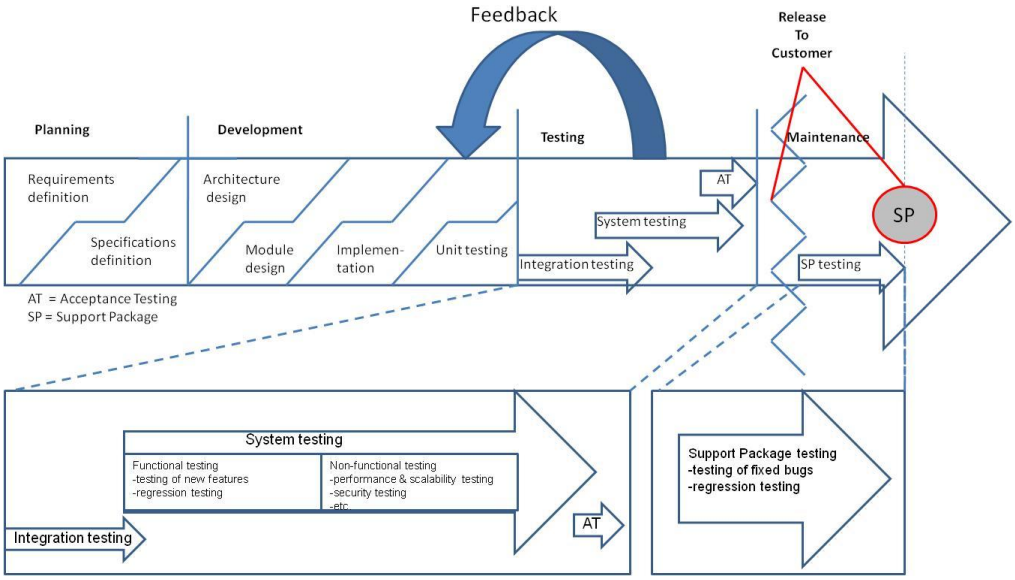

Figure 2 Testing activities in development life-cycle

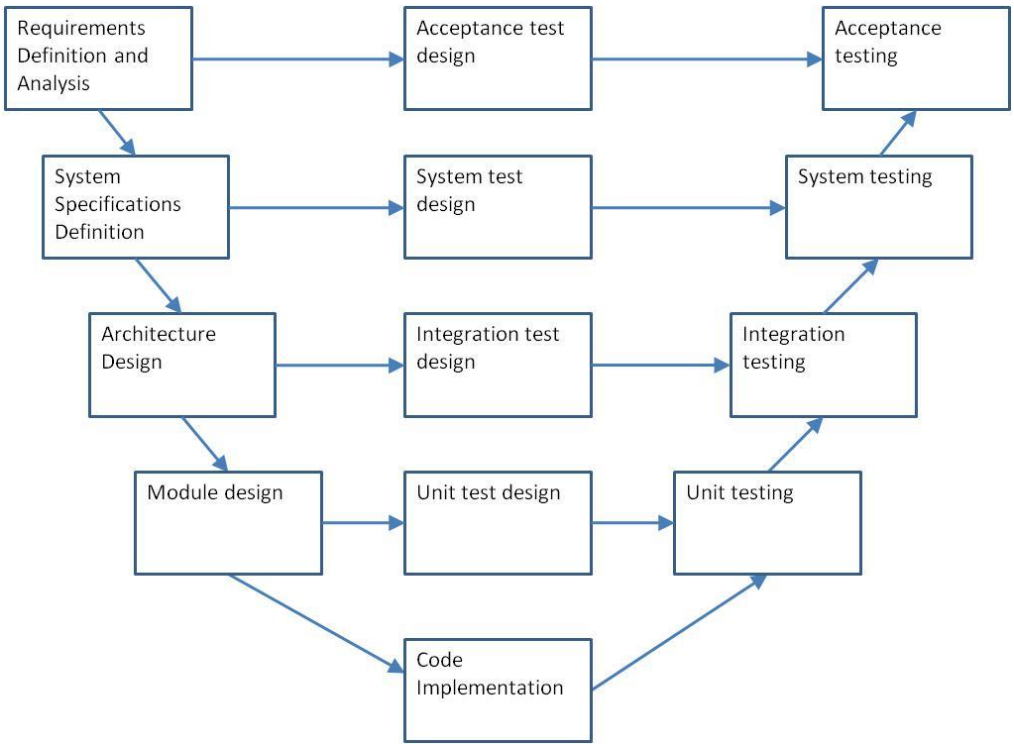

Figure 3 Slightly modified V-model

2.3 Software performance and performance testing

2.3.1 Performance testing in general

Performance of the system describes how well the system operates under certain workload. The performance testing can reveal how much resources the system uses while loaded with certain load. Also other kinds of performance related parameters, such as reliability, speed and scalability can be defined with performance testing. The system can be loaded with different types of loads depending on the type of the test. Typically these loads are created by feeding the system input with multiple simultaneous events and monitoring the output and performance of the system. Monitored metrics vary between tested system and test type. Some measurement results can be defined from in and output of the system (black box testing) and in some of the measurements information on transactions and events inside the system is needed.

Software performance testing is typically divided in load testing and load capacity testing (stress testing). Also failure recovery testing can be considered as a part of performance testing.

Load testing is performed by loading the system with defined level of load. The operation of the system and its performance is monitored during the test runs. The purpose of load test is to ensure the system meets its performance requirements. System performance test results from load tests should also be compared with ones from earlier tests before implementing some changes in the software and therefore an effect on performance can be verified if existing.

In load capacity testing (stress testing) the purpose is to find the maximum level of load under which the system operates and performs well without having any faulty behavior. The testing is done by increasing the test load as long as the system performance is not at the acceptable level any more or any faulty operation occurs. This level of load is called a breaking point.

Failure recovery testing can be a part of stress testing or performed separately by causing the system to fail otherwise. There are two phases in failure recovery testing. In first phase the environment is in a failure state which causes the system to fail. In second phase the environment is restored to normal state. How the system recovers after failure is investigated in this second phase.

2.3.2 Basic performance testing process

Performance testing process can be structured in different ways [Sub 2006], [Far 2008]. Here, the simplified model of performance test process is divided in four steps: preparation, planning, design and implementation, execution and analyzing. In first step preparation, requirements for system performance and common metrics for evaluating the performance are defined. Also acceptance criteria for passing the test and type of test are defined. Second step, planning contains definition of test environment, resources and goals for testing. Second step, test design contains design and implementation of test scenarios and workload models including users, events and transactions according defined requirements and testing goals. Execution and analyzing step contains setting the stress level and configuration for tests, execution of test scenarios and analyzing the results. If the outcome of test is not satisfying, test can be re-executed after changing stress level, configuration or test scenario.

This model describes activities in a single performance testing project but can be exploited also in performance testing process planning.

2.3.1 Performance of Web Based systems

A basic example of web based system contains web user interface which communicates with web server. Web server transfers queries from UI to application server, which contains the system logic, and uploads the requested data to UI. Application server handles the queries from UI, reads requested data from database and writes changes to database. The performance testing of web based systems is quite well studied and documented area of performance testing e.g. [Sub 2006]. Work-

load created in web based systems consists mainly of data queries and data transfer between web user interface, web server, application server and database. The completion time for these transactions in different load scenarios is measured during the tests and results are used for evaluating the performance of the system.

2.3.2 Performance of Communications systems

Communications system implements different types of communications channels for systems users and enables users to communicate via these channels with each other. Communications system can be an enclosed system containing only user interfaces for internal users or also enable communications between users in external network via gateway. Communications system contains logic for communications routing and connecting and can contain lots of different kinds of more advanced features as queues, advanced routing rules etc.

The testing of communications system, as any software system, is performed by loading the system with multiple simultaneous incoming events while monitoring the operation of the system and collecting data related to system performance. In addition to systems resource consumption, other performance indicators such as delays related to connecting and transferring the communications data is measured.

2.3.3 Performance of Communications Enabled Business Process engine

The Communications Enabled Business Process engine discussed in this thesis is system combining the features of web based systems and communications systems. System contains web based user interfaces containing different types of features creating transactions similar to many web based systems such as directory searches. The system also enables communications features and other business process features. More detailed description of the system is in previous subchapter 2.1.

Performance testing of CEBP system combines testing methods of web based systems and communications systems.

2.4 Modeling system performance

The performance of the system, experiencing load caused by simultaneous transactions of certain type, can be modeled using software system modeling techniques. The increase in processor load or memory consumption can be modeled using very simple linear assumptions. In most cases this is accurate enough to estimate the resources usage increase in function of increasing load in a linear area. The behavior after the linear area close breaking point has to be modeled using some different type of method.

Focus of this thesis is in process definition for performance testing. The modeling of the system and system events is only considered by simple example without going further in detail. Using system models for performance estimation is an excellent tool for taking SPE (Software Performance Engineering) as a part of development process already in the beginning of software design process and thereby reduces the possibility of serious performance related problems occurring later in development process or in production. [Tik 2005].

Methods for modeling system events may vary depending on the event. A very basic example of system events modeling is presented below. The system e-mail handling is modeled using simple M/D/1 – queue model [AdaRes 2001].

2.4.1 Queuing theory

Queuing theory is an analytical modeling technique for modeling queuing systems. It is widely used also in computer systems performance analysis. Software system is a queuing system as it contains jobs which share and compete from same resources. While resource, such as CPU, is reserved by a job, other jobs are queuing. The queuing models can be used in estimating for example how long a job spends in a queue or system. Depending on system or system events to be

modeled, different combinations of queues, queuing networks can be used when modeling the system events. In [AdaRes 2001] mean waiting and total times in the queuing system by Pollaczek-Khinchinin mean value equation are given in (1) and (2).

$$E[W] = \frac{\lambda E[S^2]}{2(1-\rho)} \quad (1)$$

$$E[T] = E[S] + \frac{\lambda E[S^2]}{2(1-\rho)} \quad (2)$$

$$\rho = \lambda E[S] \quad (3)$$

where

$E[W] =$ expectation for mean waiting time in queue
$E[S] =$ expectation for service time in queue
$E[T] =$ expectation for total time in queue
$\rho =$ average number of items in queue
$\lambda =$ average arrival range

2.4.2 Model for basic e-mail handling

E-mail handling is implemented so that system core components pick e-mail messages from exchange server inbox, parse messages, upload message data to database and add the message to e-mail queue where it will be allocated to appropriate agent. In simplified model, the system can be assumed to handle one e-mail at the time and other e-mail messages will be queuing during that time. This kind of very simple event can be modeled using single queue. M/D/1- queue is selected as the incoming message distribution is assumed to be exponential, service time distribution deterministic and only one server is assumed to be serving in this example [AdaRes 2001].

Service times for e-mail messages are assumed to be as follows: 40% of messages in 0,2 s and 60% of messages in 0,5s. Messages are assumed to arrive 2 messages / s.

$$E[S] = 0{,}2s \cdot 0{,}4 + 0{,}5s \cdot 0{,}6 = 0{,}38s$$

$$E[S^2] = (0{,}2s)^2 \cdot 0{,}4 + (0{,}5s)^2 \cdot 0{,}6 = 0{,}166s^2$$

And now we get the value for e-mail throughput

$$E[T] = 0{,}38s + \frac{2 \cdot 0{,}166s^2}{2(1 - 2 \cdot 0{,}38s)} = \underline{1{,}7s}$$

2.5 Performance analysis

System performance analysis consists on data collecting, data collation, data analyzing and making conclusions based on analyzed data. System performance and system operation related data is collected during the test runs. The data contains information on system events, points in time the events occur for delay calculations, performance data collected from application and database servers and network traffic information. Also some data collected and analyzed using sensory methods is included (e.g. voice quality check by manual call). The collected data is brought together and presented in manageable form. Then the data is analyzed and some performance indicators are calculated. After the data analysis is completed, the results are evaluated and decided if the test results lead to successful or failed test result, if more testing is required or if the test just brought some information on system performance without verifying any requirements to be fulfilled or not.

2.5.1 Monitored performance indicators

Delays

When considering telephony traffic delays in VoIP system multiple sources of delays can be identified [DavPet 2002]. However, we are not interested on delay components but the total end-to-end delay experienced by end user and having direct impact on quality appearance experienced by customer. If we think about the customer service contact center and events in it, we can find couple delay types having a direct effect on quality appearance of the system for customer. If customer places a call to customer services he or she faces queuing time. This is a time from placing a call to answer. Usually this time is mostly independent of system but while the system is loaded the queuing time can increase due the increased connecting delay. Chat messages and e-mail messages can also suffer from connection delay. Another type of delay affecting calls and chats is a transmission delay. This delay affects the actual conversation data transmission which is RTP stream in case of phone calls and messages containing text in case of chat. The delay is defined to be the time from the RTP package (voice produced by call participant) or the chat message is sent to the receiving the RTP package or the chat message.

With e-mails connection delay is quite meaningless unless it increases dramatically. With calls and chat messages connection delay has to be on reasonable level and has to be monitored during load testing consisting inbound calls and chat requests. Transmission delay cannot increase too much as it immediately affects the quality of the chat or phone calls. In case of phone calls there are even more strict requirements for maximum transmission delay.

These two delay types cover most of the delays interesting from user perspective in communications system. Other types of delays of interest are related to Web UI usage. Web UI login times, loading times for different Web UI views and time for directory searches etc. queries executed by Web UI user define delays Web UI

user experiences while using the system. In CEBP system Web UIs are used by agents, system users who use the system as a working tool. Quality appearance experienced by system users is surely important factor and has to be taken into account in system performance requirements but is still secondary compared delays affecting directly to end customer.

The testing could also include monitoring of some internal delays such as response delays between components. However, only end-to-end delays affecting directly to system usage are considered in this paper.

Server performance

The system components are installed and running on dedicated application servers. The system also contains databases running in other servers. As the system is loaded with some events the workload at the servers naturally increases. The level of load affecting at the servers is measured.

The performance of the system is monitored at the application servers and database servers by using performance monitoring application which stores selected performance related data during the test runs and allows the data to be analyzed after the tests. Windows Performance monitoring application with performance counters can be used in performance data collecting when Windows or Windows Server operating system is in use [Mic 2008]. Depending on the type of test, performance counters are set to collect data related to processor utilization, memory usage, disk read and write actions and network traffic.

On application servers, total processor utilization time is monitored and data collected. Memory usage is monitored e.g. with data on physical memory in use and pages read or written per second. Also data related to each process under interest in a particular test is collected with counters. These counters collect data about the number of threads active in the process, handles opened by the process, processor activity time for the process, and the size of physical and virtual memory allocated by the process. Total and process specific processor and memory related data is collected in all performance tests.

On database servers, the data related to physical disk read and write operations is collected. Counters are set to collect data about the time the monitored disk drive is busy serving read and write requests and the bytes read or written per second. The counters are set on database servers in tests in which the database transaction load is expected during the test

Network traffic

Some system operations may cause significant network traffic between system components. The counters can be set on application servers (and database servers) to collect the information on sent and received data over a network connection. These counters are set to collect data when executing tests including operations which are expected to produce significant network load. In addition to the data sent by system components through the local area network there are also lots of other normal office internal web data moving in the LAN (Local Area Network). The amount of this background traffic should also be estimate in case of test system and also in case of different customer cases.

Test case completion and system operation

As the performance related metrics are monitored during the tests to ensure the appropriate performance under a required amount of load also the correct operation of the system has to be verified regardless the level of workload. The successful completion of all test events (e.g. calls) the system is loaded with during the test is verified but also other basic features are tested during the test run by manual tests.

2.5.2 Acceptance level

Acceptance level is a value of a measurement that is still acceptable but any higher/lower value would not be. Acceptance levels can be defined for different types

of metrics indicating the performance of the system. The acceptance levels of same measure may change somewhat between different types of tests. Test cases and test plans define acceptance criteria for separate tests. Following are listed some acceptance levels for some metrics common for most of the tests. These acceptance levels should not be exceeded in any tests if not separately specified in test description.

Delays

The acceptance levels for delays depend on application of the system. However following acceptance levels has been defined to meet the requirements of most of the applications. Acceptance levels for different types of delays depend mostly on user experience. Users may experience similar delays very differently depending on the situation. For example acceptable time for Web UI to load depends on which type of action is performed. Users may accept longer delays for actions they know normally to take more time to complete. For example load time for Web UI during the logon is acceptable to be much longer than opening another view in UI. Acceptance levels for different types of delays should be defined case specifically as they may vary a lot depending on the action and context. Only transmission delays may be considered quite generic as they have similar affect on quality appearance independently on application. Especially the transmission delay of phone calls shall not exceed the maximum acceptable level.

Acceptance levels for delays also differ depending on the type of delayed contact. Connection delays for calls and chats cannot be too long as the customer is waiting for live conversation and is not expecting too much delay. In case of e-mails the customer normally expects some time to go by before having answer to that e-mail. Connection delays for calls, chats and e-mails in Table 1 are defined for queue contacts connecting customer service queue. In case of normal queues these delays could be at lower level, as direct contact is usually expected to answer much earlier. Transmission delays are critical to keep on required level in case of phone calls, as if the delay increases it becomes perceivable for system user or

customer and affects to quality appearance. With chat messages transmission delay is not that critical but still has to be at reasonable level.

In addition to user experience or customer requirement, acceptance levels for delays can be defined also based on technical limitations. Technical limitations can be estimated using modeling. In subchapter 2.4.2 an example for e-mail connection delay in certain circumstances is modeled using basic queuing model. Examples for maximum acceptable delay and goal delay are shown in Table 1. These acceptance levels are defined based on [Bar 2004], [ITU 2003] and interview of company internal experts. These acceptance limits for different types of delays shown in Table 1. are defined for customer service contact center system. These values may differ between different types of systems.

Table 1 Acceptance levels for different types of delays

Type of delay	**Object of delay**	**Acceptable delay**	**Goal delay**
Connection delays	Calls	5 s *	1 s *
	Chat	5 s	1 s
	E-mail	30 min	1 min
Transmission delays	Calls	150 ms	50 ms
	Chat	5 s	1 s
Web UI loading	Logging in	30 s	10 s
	Opening another view in UI	5 s	3 s
	Performing a search	30 s	15 s

*connection delays of calls are delays for single connection, not for routed calls.

Server performance data

The acceptance levels for server performance are based on information on Microsoft reference guide for performance data counters [Mic 2008] and on interview of company internal experts. Acceptance levels are defined only for most important performance counters.

- Total processor utilization time, constant: 60 %. The processor utilization time may have higher peak values but the level has to settle to lower level between peak values.
- Memory: There should be no noticeable memory leaks
- Disk reads / writes per second: Depends on manufacturers specifications. In general Ultrawide SCSI can handle 50 I/O operations per second.
- Network utilization: Should not exceed 50%

System Operation

All events and tasks described in test case have to be completed successfully. There should not be any faulty behavior of the system caused by the load during the test. The system should not counter any insufficient deceleration in system task execution e.g. directory search or Web UI login should be possible to perform without exceeding the acceptable limits for applicable delays. No relevant error or exception messages should appear in the system log files.

2.5.3 Analyzing test data

After the tests have been completed and test data collected the data needs to be analyzed. The data analysis can lead either in test completion or failure or it can just give information on system behavior under tested circumstances. Test data consists on data provided by system reporting and monitoring components, data from performance counters on system servers, data gathered by testing tools etc.

Collected performance counter data is provided in performance log file. The data is organized by counters and can be easily displayed in graphical format. All important performance indicators e.g. average and maximum CPU utilization rate can be defined from graphical presentation of test results data.

Reporting and monitoring data consists of information on system events. These events are monitored in real time during the test runs. The correct operation of the system during the test can be verified also by checking the reporting data regarding these events afterwards.

Testing tools report contains information on the test events executed by the tool. Test report includes the information on test event successful completion or failure. Report may also include delays for certain events such as chat message sending. If some of data in reports is needed to analyze more, the data should be moved to more usable environment for example to excel table in which the data is more easily manageable.

Log files of testing tool and system components contain information on system internal and external events during the test case run. These log files should be checked for error and exception messages.

3 Requirements

The most important part of performance testing process planning is to define the requirements for the testing process and for the system performance. These requirements will guide the planning of the performance testing process. Different aspects have to be taken into account when defining these requirements. Each of these aspects defines questions and requirements which the performance testing process has to answer and cover. Requirements for performance testing process were defined based on information gathered from company internally by interviewing experts in areas of development, product management, customer support & consulting and quality management; from literature study on software performance testing methods[Bar 2004a], [Far 2008], [Wey 1998], [WeyVok 2000], contact center and office telephony system [Ack 2003], [Eva 2003] and VoIP system performance [Dif et al. 1999], [DavPet 2002] and by evaluating the properties and demands of the type of system described in this document. The requirements for the process include general subjects describing the system performance, methods for collecting performance related information and other demands for performance testing process. Requirements for system performance describe how the system should perform and operate.

Requirements for software performance should be defined extensively and detailed already in design phase of the system. Thus the performance requirement would be considered when architecture and system components are designed. Often these requirements are insufficient, lacking on definitions for normal and stress workloads, acceptable performance values for the system, acceptable delays etc. Consequently the definition of requirements has to be done using other methods. Information for requirements is gathered by evaluating design documentation, from information provided by key stakeholders, from information about user acceptance and personal use and experience [Bar 2004b].

In this chapter requirements for performance testing process and system performance are described and defined. Separate requirements are marked with identifi-

er so that referring to each requirement is easier in later on in testing process description. Requirements for performance testing process are marked with “RP” following with the number of the requirement and requirements for the system performance with “RS” following with the number,

3.1 Requirements for Performance Testing Process

Business Process performance requirements

RP1 The performance of the system in defined business process model shall be verified. Business process models describe the system configuration, events and transactions in the system modeling real productive application. Therefore the business process model based performance testing is an important part of performance testing of the software. The baseline for performance of the business process model shall be defined by testing the performance of the system while the system is loaded with typical level of load for the business process model. Results from upcoming business process model tests shall be compared to baseline and the performance should remain unchanged or improve.

RP2 The performance of the system in defined business process model shall be verified also in more challenging scenarios. An advanced scenario is a combination of basic business process scenario and scenario including high load of events.

Development feedback and Quality Assurance requirements

RP3 Fast feedback shall be given to development team. Development needs feedback from software performance testing in order to get the information on any revealed defects and bottle necks in the system affecting system performance as soon as possible. The feedback has to be given instantly as taking an action in correcting the problem in early phase will decrease amount of work and costs.

RP4 Performance testing shall be started as soon as possible. As any testing, also performance testing should be started in as early phase of development process as possible. The costs caused by fixing the bug can be reduced significantly by revealing the bug in early phase of the development. However, the performance testing described in this document is a part of system testing, thus the testing can be started as soon as most of the system components are ready and integration testing completed.

RP5 Real system events (user actions) shall be used in testing. The performance of the system and its system components shall be tested using real system events in real system. This means that the stimulus in tests should be a real user action or transaction e.g. a call / multiple simultaneous calls and the system should represent the real productive system. Performing performance testing as a part of unit testing is also possible but the focus in this performance testing process will be in system testing.

RP6 Performance testing has shall be executed regularly. Performance of the system shall be verified frequently during the development phase in order assure fast reporting, response and action for encountered performance problems.

RP7 Performance of every software version to be released to customer shall be tested in order to avoid the release to be delivered to customer with any performance problems in the system or in its components.

RP8 Load capacity has to be tested whenever needed. Load capacity of system or its component should be tested after any major changes and whenever noticeable changes in performance are expected.

RP9 Tests must be simple and easily repeatable. As the system performance can be dependent on used configuration and number of possible configurations is extensive, the tests must be performed with basic configuration.

RP10 Also a “worst case” situation shall be tested. A comparative testing for investigating, how much effect on system performance different configuration could

have, shall be performed by testing the system which has so called worst case configuration. This is a configuration causing more load to core components. The configuration consists on more users, more routing rules, more defined queues and channels, basically create situation where, more calculation effort is needed before a call is connected. The configuration of the test system during the tests shall be well documented.

RP11 Also reused components shall be tested. The components which hasn't been changed at all and are added to new version of the software shall be tested as well as the operation may have been changed with new environment [Wey 1998].

Load capacity requirements

RP12 Information on load capacity of the system and its component shall be produced. Information on maximum load capacity of the system and its components is needed and possibly information on mechanisms of failure when system or its components are stressed to breaking point. Also information on system recovery in these heavy stress scenarios should be collected

RP13 Information on system scalability shall be produced. Relation between performance of the system and the level of load should be verified for basic load types from low to near failure load level. The operation and performance of the system while scaled up by adding more resources e.g. new hardware, system component shall be verified also.

Requirements for resources

RP14 The test system shall represent a productive system as well as possible. The system hardware software installation and configuration should be implemented according the installation guidelines provided for customers, if test case does not advice other vice.

RP15 The configuration of the system has to be as simple as possible and should be documented well so that the tests are easily reproducible. All settings, configurations etc. not described by test case or business process model, should be as default.

RP16 People having appropriate knowledge and experience shall be allocated to the performance testing process when needed in order to ensure efficient analysis of the test results and system behavior in advanced circumstances. Test results analysis of certain advanced test cases by experienced people will decrease the time and effort used in this testing phase and will lead to better results.

RP17 Performance testing process shall be integrated with standard testing process.

3.2 Requirements for CEBP system

Service level requirements

RS1 Continuous service shall be ensured. In many applications of the system, such as a customer service or telemarketing call center, the business is strongly dependent on system operation. In some applications persistence round-the-clock service is required. Therefore the continuous operation of the system has to be verified. The operation of any backup systems has to be verified also.

RS2 Continuous service shall be ensured also in case of failure. Therefore it has to be verified that system can perform in an environment and circumstances it is designed to be used. Also system's ability to recover after a failure has to be verified and analyzed.

RS3 Continuous service shall be ensured also in case of maintenance. Maintenance operations defined not to cause system downtime should be tested for correct operation.

Requirements for telephony system

RS4 Delays shall be at acceptable level. Acceptable response times for telephony system are stricter than for systems of other type. Most important delays to monitor are delays the customer can perceive. This kind of delays are end-to-end connecting delay (time between placing the call and call alerting for recipient) and delays in voice stream. Some delay in connecting the call can be acceptable but only very limited delay in voice stream is accepted.

RS5 Voice quality has to be at acceptable level in all calls in all scenarios. As the quality of phone call is most important metric of quality appearance for the user, it has to be ensured. This means that the performance of the system cannot be implemented at expense of call quality.

Performance requirements for communication events.

RS6 Simultaneous incoming inbound events. System having a defined configuration shall be able to handle defined number of events coming in the system at the same time without causing any faulty behavior or insufficient delays. The operation with all supported inbound event types shall be ensured. These events could be calls from external network, chat requests, e-mail messages etc. Successful routing and connecting the events has to be verified. The correct operation of the events loading the system and the system in general shall be verified.

RS7 Simultaneous active inbound events. System having a defined configuration shall be able to handle defined number of events active in the system at the same time. These events could be e.g. phone calls and chat sessions. The correct operation of the events the system is loaded with and the system in general shall be verified.

RS8 Connection reliability. System having certain events (call, e-mail, chat) connected shall maintain these connections until disconnected by the user, if not specified to operate other vice.

RS9 Simultaneous outgoing calls, calls active and connection reliability. The system performance while multiple outgoing calls are performed shall be ensured. The system shall be able to handle defined number of outbound calls placed to external network at the same time, defined number of simultaneous active outbound calls and connected outbound calls shall remain connected until disconnected by user. Three requirements are included in this one and these requirements are similar to ones with inbound events.

RS10 In addition to Delays, system should fulfill also other acceptance criteria defined in chapter 2.5.2 Acceptance levels.

3.3 Workload characterization

Workload is a set of inputs of the tested system is fed during load test run. Defining models for these workloads is an important task to complete as it defines the scenarios the test system will be tested against. Methods in use, when defining workload models for performance testing depend on the type of the software and information provided on software's application and usage in field. Workload characterization can be complex operation if there are no clear definitions for load capacity requirements or knowledge on the loads the software will be facing when taken in productive use. If the system for which the workload models are defined is already in productive use, the information for workload characterization can be gathered by monitoring events and transaction in productive system during a defined period while system is in productive use and loaded with normal usage [Avr et al.2002], [WeyVok 2000]. System usage data is then analyzed and workload models defined based on gathered information.

The system considered in this document does not have a simple use scenario which could be used for workload characterization but adapts and scales according each customer case. There is no possibility to collect any extensive usage data from productive systems either. In this case another approach for workload characterization has to be introduced. Workload model definition will be done based

on requirements described in previous subchapters and business process models described in the following subchapter.

In both cases, the basic goal of defining workload models is to implement performance testing activities which test the system while loaded with load typical in its operating environment and verify the correct operation and performance of the system.

3.4 Business process models

Business process models describe different types of applications of the CEBP system. These business process models define the environments and scenarios which the real business cases of different types are consisted of. These models describe the scenario in normal situation with normal load for each business case. Also some possible advanced scenarios are defined for models.

These business process models are defined for the system described in this document. The planning of business models for performance testing is based on company internal information provided by company consulting team, solution management team and development team about system implementations in real business cases, possible business cases and planned business cases. The information gathered about important business cases is reviewed and used from performance point of view.

The implementations of the system considered in this thesis can be divided in to contact center solutions and enterprise telephony solutions. The systems in real world can be purely contact center or enterprise solutions or some kind of combination of both types. From performance point of view these two basic implementation types have some differences. Contact center has much more resources consuming features as queues, routing, prompts playing, recording, IVR applications etc. Enterprise telephony users mainly use the phone only as one of all business tools with low utilization rate in average whereas contact center agent use their phones as a main tool with high utilization rate. As in contact center case the load

is caused to system by high phone usage level and by multiple simultaneous events, the load in enterprise telephony case is mainly caused by e.g. directory data queries made by the user interfaces as typically much more directory data exists in these cases. Therefore it can be seen that in most cases, single contact center agent causes much more load in the system than single enterprise user in a basic case.

Typically license pricing of VoIP telephony systems makes contact center agent licenses more valuable than enterprise telephony user licenses. This is also one reason why contact center is more important business process also in business point of view.

As the contact center type business processes are more resources consuming and more interesting also from financial viewpoint, the basic installation and configuration of the system used in most tests are based on contact center based business cases. Most of features and scenarios in enterprise telephony will be tested while testing contact center business cases anyway. However, there is still need for verifying the performance of the system in basic enterprise telephony scenarios.

3.4.1 Basic customer service contact center business process model

Basic contact center business case is based on medium sized contact center having a normal level of different kinds of active events and transactions in the system.

There are following kinds of users in the system: Defined number of contact center agents (logged in either with web user interface, SIP IP desk phone or mobile device), customers (call, chat e-mail). The system has phone, chat and e-mail queues configured the agents serving in. There is an IVR application in use in which some of the queues are linked to. Also some special routing rules have been defined.

There are defined number of contact center agents logged in and serving in phone, chat or e-mail queue. The incoming queue calls are routed to free agents, answered, some of calls are hanged up, some of calls are transferred ahead and some

consultation calls are performed. Chat customers send chat requests into the chat queues, chat requests are accepted by free agents and the conversation sessions are stopped. E-mail customers send e-mail messages to e-mail queues, free agents pick e-mails from the queue and answer to some of them.

Examples of key figures for middle sized contact center business process model:

1. 500 Contact center agents, (400 Web UI, 60 SIP IP desk phone and 40 mobile) serving in 20 queues (16 phone queues, 2 chat queues and 2 e-mail queues)

2. 10 000 incoming event per hour (8 000 calls, 1200 chat requests and 800 e-mails)

Examples of key figures for large contact center business process model:

1. 5000 Contact center agents, (4000 Web UI, 600 SIP IP desk phone and 400 mobile)

2. 100 000 incoming events per hour (80 000 calls, 12 000 chat requests and 8 000 e-mails)

3.4.2 Basic enterprise telephony system business process model

Basic enterprise telephony business process model is based on medium or large sized company/office telephony system. Most of the events / transactions in this kind of system are basic, direct internal or external calls.

Enterprise telephony business model includes couple call center agents, office telephony users (logged in the system either with web user interface, SIP IP desk phone or mobile device) and external call participants. The call center part (telephone exchange) in basic enterprise telephony model consists of one queue having couple contact center agents serving it and transferring incoming calls to correct users.

The main difference with contact center and enterprise telephone business models is that the frequency of events per user in contact center model is many times higher than in enterprise model. However, the number of users and the size of directory data (internal and external contacts) are typically much larger than in contact center solution and this could cause some extra load for user interfaces and web servers when UIs make database queries.

Examples of key figures for middle sized enterprise telephony business process model:

1. 5 000 office phone users (logged in the system either with web user interface, SIP IP desk phone or mobile device)
2. 5 contact center agents serving in one queue and connecting incoming calls to office telephone users
3. 100 incoming queue calls per hour
4. 1000 outgoing calls per hour
5. User database containing 30 000 contacts

3.4.3 Basic contact center enterprise telephony combination business process model

As the title predicts, this business process model is a combination of previously described models. In this model the system consists of contact center and office telephony parts. There are both external and internal queues in the system. Customers call to external queues which are connected to contact center. Most of the calls are handled by contact center agents. Some of the calls connected to contact center agents need more expertise and contact center agents transfer the calls to appropriate internal queues where agents working at the office are occasionally answering customer calls. Most of the time agents at the office work in other tasks than customer service but are logged in the queue and answer the queue calls when needed.

Examples of key figures for middle sized combined contact center and enterprise business process model:

1. 500 Contact center agents, (Web UI) serving in 20 queues (16 phone queues, 2 chat queues and 2 e-mail queues)

2. 5 000 office phone users (logged in the system either with web user interface, SIP IP desk phone or mobile device), 500 of them logged in 50 internal queues (40 phone queues, 10 e-mail queues).

3. 10 000 incoming event per hour (8 000 calls, 1 200 chat requests and 800 e-mails)

4. 500 events transferred to internal queue per hour (400 calls, 100 e-mails).

3.4.4 Basic Outbound telemarketing contact center business process model

This business process model describes the system which is used for telemarketing purposes. The users in this business process model are contact center agents (logged in either with web user interface, SIP IP desk phone or mobile device) and customers answering the calls. The system also contains a Predictive dialer (PD) application which is used for calling telemarketing calls according defined call campaigns and connecting calls to free agents. The dialer application can be configured in different ways (predictive or progressive dialing). The system also contains tools for storing data related to each call into the database.

Examples of key figures for middle sized outbound telemarketing contact center business process model:

1. 200 contact center agents (logged in the system with web user interface) logged in the outbound telemarketing call campaign

2. 10 active outbound telemarketing campaigns

3. Predictive dialer application in use configured for predictive dialing (PD calls customer number and if the call is answered by human (not e.g. a fax machine) the call is immediately connected to free agent)

4. In average every other call is connected successfully to agent

5. Call duration is between 20 seconds to 2 minutes.

6. After completing a customer call agents enter the data related to the call in the form which stores the data in the database and start to wait another call.

3.4.5 Basic contact center / enterprise telephony integration business process model

This business process model describes the contact center or enterprise telephony system integrated with some other type of system e.g. an enterprise resources planning (ERP) or a customer relations management (CRM) system. Users, events and basic configuration of this model can be similar to any of above described models with difference of agents using user interface integrated as a part of this "parent" system. The signaling between CEBP system and the parent system consists on communications events (call, chat, e-mail) related signaling, contact information and possibly users' presence information. This business process model enables communications events to be added as a part of other key business processes.

A simple example of this kind of business process model is customer service contact center integrated with customer relations management system. Incoming customer contacts in CEBP system are automatically connected to appropriate customer data in CRM system while contacts arrive in the system. Same examples of key figures as described for basic contact center and basic enterprise telephony business process model can be used also with this model,

3.4.6 Advanced Business process models and scenarios

In addition to these basic business process models, also more advanced models can be defined. These models can have different type of distribution of incoming or outgoing events e.g. balance between calls, e-mails and chats. Also the system may have more specialized configuration, the number of queues may be much larger and more complex routing rules may be in use. However in this process only basic business process models are considered as they cover the most of business cases. If need for performance verification of special business case occur more advanced business process model can be included in the process afterwards.

Example scenarios for these business process models described a basic situation in the system described by the model. The implementation of these business process models with more advanced scenarios is done by simply adding an additional load element in the normal scenario. For example advanced scenario includes e-mail burst of 10 000 incoming e-mail messages per hour and basic scenario having 8000 calls, 1200 chats, and 800 e-mail per hour is changed to advanced scenario having 8000 calls, 1200 chats and 10 000 e-mails per hour and performance is measured. Following advanced scenarios for middle sized systems are defined:

1. 50 000 incoming e-mails per hour
2. 300 incoming calls per second (~10*10000 calls / hour)
3. 100 incoming chat requests per second

3.4.7 Custom Business process models

Above basic business process models cover most of the business cases in which the CEBP system will be implemented, but in addition to those models customized business process models can be added if there is a business need for that. Example of customized business process model could be a customer service contact center having very complex routing rules configured and affecting most of the

calls in the system. This kind of additional computation per event would surely affect the results of performance tests.

4 Performance testing process

4.1 Performance testing process base line

Performance testing process defines all activities related to verification of performance of the software. Performance process description includes the definition of test plans, test cases, staffing and schedules for execution of testing activities. The planning of the process is based on defined requirements. Both, requirements for performance testing process and for the system performance exist. These requirements have to be taken into account when defining the process. Most of the requirements for performance testing process will be covered and fulfilled with test plans. Testing of properties described in requirements for system performance will be covered mostly based on test cases. In some cases system performance requirements will have a direct affect on test plan definition and process requirement on test case definition. Linking requirements to test cases and test plans is described in Figure 4. If test plan or test process description fulfils the requirement defined in 3.1 or 3.2 this is notified with respective requirement identifier in parenthesis.

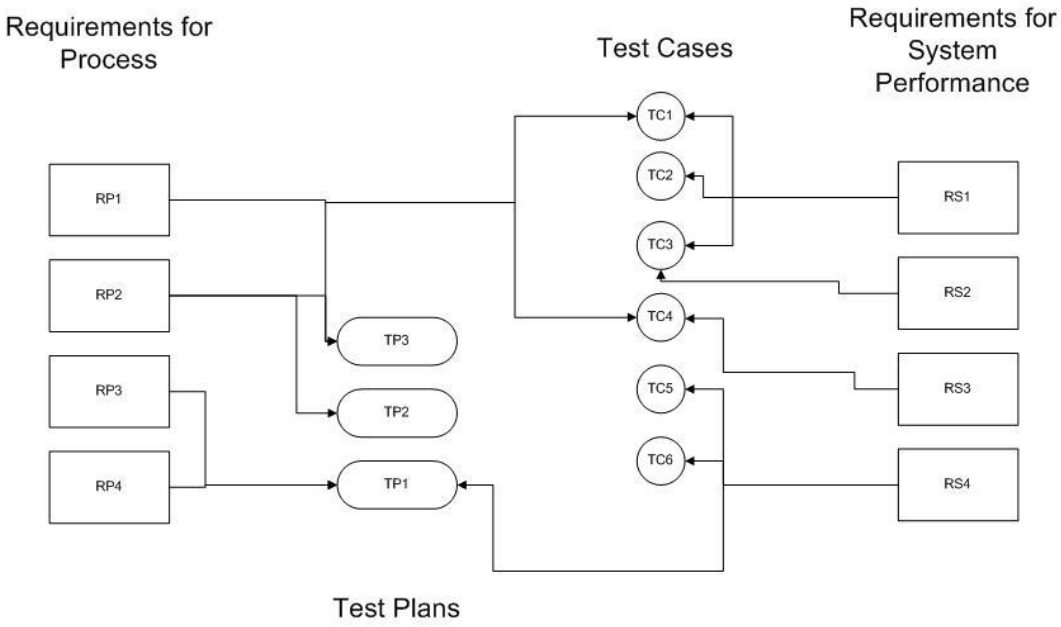

Figure 4 Relation between requirements, test plans and test cases

4.2 Test plans

The test plan prescribes the scope, approach, resources, and schedule of the testing activities [IEEE 1998]. As the standard reveals, the testing activities are defined with test plan. However test plans defined in this chapter do not include all detailed information listed in the standard. These test plans describe the feature or property to be tested, methods for testing the property and acceptance criteria for tests. Also if any special resources are needed in execution of test or in results analysis they are described here. Tests plans contain a set of test cases for testing of the defined area of software properties. These test plans and test cases are designed to fulfill the requirements described in chapter 3.1 and 3.2. Execution of all test plans requires testing specialist to have knowledge on system structure and operation, on performance testing principles and system monitoring and data collection. More advanced expertise may be needed when analyzing test data in complex failure situations. In many test plans, normal and heavy load levels are

used with test cases. These loads are defined for the system and may vary depending on test case. Also acceptable levels for system performance are defined.

System should fulfill all applicable common acceptance criteria for delays and system performance described in chapter 2.5.2 in all tests. All unexpected behavior is documented and reported immediately to development (RP3, RS5).

4.2.1 Load capacity test plan

Load capacity test plan contains complete tests for testing the load capacity of the system and its components. This test plan is planned to be executed only when major changes has been carried out and changes in performance are expected. The test plan could be executed once in every program as early in the development process as possible. The objective of this test plan is to define the maximum quantities for different types of system input event loads the system can handle without failing. Also failing mechanisms are investigated when system is stressed to breaking point.

The test plan contains stress test cases for measuring maximum load limits for the system and its components. Test cases cover the stress testing of core components, each connection component, web components, chat and e-mail components and integration components. Stress testing produces information on system performance, maximum load limits and failing mechanisms when the maximum load limit is crossed. Stress test cases are executed for new components and components which have been changed and there is a reason to expect changes in performance of the component.

The test plan also contains load test cases to test the load capacity by loading the system with defined required level of load near the maximum load level of the component. The load tests may be sufficient for load capacity testing if the component has not been changed much. If significant changes in performance compared to results from tests with previous version have been found in load tests,

more detailed stress testing is also performed. Load capacity testing also includes detailed analysis of test data to ensure the system operates as it should when loaded heavily.

Also testing of worst case configurations is included in this test plan. The testing is performed with common test cases but the system is configured increase the produced system load.

Passing the tests requires system to operate correctly and performance to be on defined, sufficient level while executing tests with defined maximum load the system is required to handle. More testing may be needed if unexpected mechanics are causing the failure while stressing the system to breaking point. Special expertise system components architecture and operation may be needed when evaluating failure mechanisms and operation of the system in heavy stress scenario.

Although the test plan consists on test cases for testing of maximum load capacity of separate system components, stress testing, as all test plans considered in this testing process, are part of system testing and are performed by loading the complete system with set of common system input events. (RP8, RP10, RP11, RP12)

Load capacity test plan consists of test cases for each component with appropriate load input. The plan is a combination of stress and load tests and it is defined case-specifically.

The set of test cases marked in Table 2 is included in the load capacity testing plan.

4.2.2 Scalability test plan

The object of this test plan is to verify the performance of the system e.g. delays, CPU load, memory consumption etc. while the load level is increased step by step. The performance of the system is tested with different types of events.

The test plan also contains testing of the increase in system performance while increasing the resources by adding hardware or software components. The test plan fulfills requirement RP13. The normal and heavy load levels are defined for each sizing level and acceptable performance levels for those are defined also. The test cases with normal and heavy load level are executed separately on system in each sizing level under testing. Passing the tests requires system to operate correctly and performance to be at defined, sufficient level with all test runs with all sizing levels.

If scaling behavior of the system is not known in advanced, the performance baseline is defined by the system performance results from test runs on first sizing level. The testing of other sizing levels is performed by increasing the load level little by little until the performance results are near baseline. This is repeated on following sizing levels. Expertise in system environment maintenance and configuration is needed in this test plan.

Scalability test plan includes a set of test cases marked in Table 2 with different types of loads.

4.2.3 Flexibility and Failure Recovery test plan

If maintenance operations for system required to be executed without causing system downtime exist, flexibility testing plan is used for testing the correct operation of the system in these situations. Certain system component upgrade, system installation or configuration change or hardware resource addition could be such maintenance operation. The testing of system flexibility is done by executing a scenario test case, performing the maintenance operation while there are active events in the system described by scenario test case. Also process based test cases could be used in creating activities if only specific types of events are in focus in certain system maintenance activity. The flexibility tests are passed if the main-

tenance operation is completed successfully and scenario events were not interrupted by the operation.

Failure recovery tests plan contain test cases in which the system components are forced to get unavailable for a moment. The recovery of the system is monitored after the component has been made available again. The downtime of the unavailable component is varied between tests. How the recovery of the system is defined depends on the system component being unavailable and scenario in which the system was as the down time started. Also testing of backup components is included in this test plan. In this case back-up component is configured to activate and take original component's tasks as soon as original component gets unavailable. Passing failure recovery tests requires system to recover and become available after downtime automatically in sufficient time. Failure recovery test cases are marked in Table 2.

Flexibility test plan can contain any steady load or reliability test cases while system maintenance operation is performed. Set of test cases marked in Table 2. can be included.

The Failure recovery test plan can be also combined with stress testing plan or included in business process model tests. Flexibility and failure recovery test plan may require special system architecture and operation knowledge if complex failure states occur, but if tests are passed basic knowledge level is sufficient.

4.2.4 Performance after upgrade test plan – *Business process model approach*

As the title reveals, the goal of this test plan is to measure the performance after upgrade and compare the results with ones from earlier tests with previous version. The changes in performance are monitored and more detailed testing is performed if notable changes are discovered. Test cases included in this test plan are based on defined business process models. These models describe the system configuration, environment, users and active events and transactions in the system.

Test cases included in this test plan describe the conditions and scenario in business process model. Separate test cases for each business process model and for normal and advanced scenarios are included.

This test plan fulfills requirements RP1 and RP2.

Test cases included in this test plan are marked in Table 2.

4.2.5 Performance after upgrade test plan – *Technical process approach*

The test results obtained from tests in this test plan are also compared with results from earlier tests with previous version. An approach to testing is however slightly different. The system is loaded with technical processes which in this case mean isolated events caused by real customer action. The system is loaded with one type of events at the time. Test cases are executed using both normal and heavy level of load. Passing tests in this test plan requires correct operation and sufficient performance while loading the system with normal and heavy load of events described in test cases. There should not be decrease in performance compared to earlier versions either.

This test plan fulfills requirement RP5.

Test cases are included in this test plan are marked in Table 2.

4.2.6 Other test plans

Rough performance and testing tool maintenance testing plan

This test plan will be executed for every build after installing it on testing environment. As the system is developed some interface might be changed so that the load simulator application can no longer use some feature properly with same configuration. The purpose of these tests is to spot any faulty behavior of the si-

mulation tool in early phase so the tool can be developed and updated immediately. These tests can also reveal any bugs having clear effect on the performance of the system in early phase.

This test plan contains only very simple test case requiring only very little effort to execute. The test case included in this testing plan is marked in Table 2.

Customized test plans

Above test plans cover most of the system performance testing needs. However, sometimes other types of testing may be needed as well. Customized testing plans are compiled for any special testing need. Customized test plan could contain a set of standard test cases or custom test cases.

4.3 Test cases

Test cases describe the test scenarios and testing actions in detail. Test cases included in test plans are introduced in this chapter. The same test case described here can be used in different occasions by changing the load level. Test cases defined in this performance testing process are shown in Table 2. The table contains also information on which test cases are included in which test plan.

Test cases can be divided by the type of events load consists, by the quantity of load and by the distribution of load. Different alternatives for these test case properties are described in following subchapters (RS1, RS6, RS7, RS8, RS9).

4.3.1 Event types

Load is a set of events fed into the system during the test run. The system is loaded with sets of different types of events in different test cases. Test cases included in test plans described in previous chapter include testing of system's load handling properties in case of SIP calls, Inbound calls, Outbound calls, Web UI logins, Chats and E-mails.

In SIP call test cases the system is loaded with incoming calls from VoIP gateway to the system. Calls are connected through SIP Bridge which handles the messaging between VoIP gateway and Core components. The successful connection of the calls, connecting delays, queuing times and performance of the system is monitored during the test runs. Incoming SIP calls are produced with simulator application acting as a VoIP gateway. If the test is wanted to be focused on performance of core components (Inbound call tests), system is scaled so that SIP Bridge won't be a bottle neck in these tests.

Web UI login test cases contain multiple simultaneous users logging in the system using Web UI. During the logon user specific data (user settings, user availability information, phone-book directory data etc.) from database is loaded to Web UI via UI connection component and web server.

Chat test cases include chat customers sending chat request to the system. System core components route the chat sessions to the queue and further to free agent. Agent accepts the chat requests. Messages between agent and customer are transferred by web server.

In e-mail tests, multiple simultaneous incoming e-mails are produced to the system. Core components load the e-mails from e-mail exchange server inbox for parsing, add the e-mail into queue and further transfer it to a free agent. System performance and connection delay are monitored.

Test cases with outbound calls include outbound call campaign for creating a mass of simultaneous outbound calls. Testing simulator application (or a dialer application included in the system) places calls to customers in external network and core components allocate these calls to free agents. In addition to call dialer, simulator application simulates also agent and customer actions. The successful connection of the calls, connecting delays, queuing times and performance of the system is monitored during the test runs

4.3.2 Load quantities

In stress test cases the system is loaded with high level of load. The level of load is increased step by steps in test runs until the breaking point of the system is reached. The purpose of these test cases is to find maximum load capacity levels for the system when loaded with different types of events. Stress test cases include testing of system load capacity with all described event types.

In load test cases the system is loaded with specific level of load. The performance and operation of the system is monitored during the test run. The level of load used in these tests depends on the purpose of the test. Performance of the system can be verified on normal or in high load situation. Load test cases include testing of system performance with all described event types.

Scalability test cases include tests in which the relation between system performance and level of load is defined by monitoring the performance indicators of the system at different load levels. Scalability test cases also include tests in which the system is scaled up by adding more resources e.g. system components or hardware and the scaling of system performance is tested.

4.3.3 Load distributions

In burst test cases the system is loaded with multiple events coming in the system at the same time. The purpose of burst tests is to test the performance of the system in case of peak load. Burst loads can be used in both stress and load testing. Burst tests for all described event types are included in test plans.

Steady load test cases include multiple simultaneous events active at the same time. Different from burst tests, the events are not activated at the same time to avoid additional load caused by simultaneous calls connecting, as the purpose is to test the load caused by simultaneous active events.

Connection reliability test cases are similar to steady load test cases but the load levels are typically lower and the length of test run is much longer. The purpose of these tests is to test the connection reliability of different event types. This is done by leaving connected events (call, chat sessions) active for long time e.g. over night or weekend and verifying connections to stay active during the test run.

4.3.4 Other types of tests

Scenario test cases describe the business process model scenarios in detail. The scenarios are simulated in testing environment and defined performance metrics are monitored. The scenarios defined by test cases are based on business process models described in chapter 3.4. Two main types of business process models are contact center and enterprise telephony business process models.

Integration test cases include tests cases similar to above described but users are connected to the system with integration UI via integration components. In addition to call etc. data also some data specific to integration is transferred between integration UI and core components.

Combination test case includes combination of basic tests with all defined event types. The purpose of this test case is to enable to quickly execute test which reveals any clear performance related defects and malfunctioning of testing tools.

Table 2 Test cases included in Test plans

ID	Test case	Load capacity test plan	Scalability test plan	Flexibility and Failure recovery test plan	Performance after upgrade test plan - Business process model approach	Performance after upgrade test plan - Technical process approach	Rough performance and testing tool maintenance testing plan	REQUIREMENT
TC1	Basic customer service contact center scenario				x			RS4, RS5, RS10
TC2	Basic enterprise telephony scenario				x			RS4, RS5, RS10
TC3	Inbound call stress test, call burst	x						RS4, RS5, RS6, RS10
TC4	Inbound call load test, call burst	x				x		RS4, RS5, RS6, RS10
TC5	Inbound call stress test, steady load	x						RS4, RS5, RS7, RS10
TC6	Inbound call load test, steady load	x				x		RS4, RS5, RS7, RS10
TC7	Outbound call stress test, call burst	x						RS4, RS5, RS9, RS10
TC8	Outbound call load test, call burst	x				x		RS4, RS5, RS9, RS10
TC9	Outbound call stress test, steady call load	x						RS4, RS5, RS9, RS10
TC10	Outbound call load test, steady call load	x				x		RS4, RS5, RS9, RS10
TC10	Outbound call load test, steady call load	x				x		RS4, RS5, RS9, RS10
TC11	Inbound call load reliability					x		RS1,RS4, RS5, RS8, RS10
TC12	Outbound call load reliability					x		RS1, RS4, RS5, RS9, RS10
TC13	Inbound call load scalability, steady and burst		x					RS4, RS5, RS6, RS7, RS10
TC14	Outbound call load scalability, steady and burst		x					RS4, RS5, RS9, RS10
TC15	Core components failure recovery testing			x				RS2, RS10
TC16	Web UI login burst stress test	x						RS6, RS10
TC17	Web UI login burst load test	x				x		RS6, RS10
TC18	Web UI login steady load stress test	x						RS7, RS10
TC19	Web UI login steady load test	x				x		RS7, RS10
TC20	UI connection components failure recovery testing			x				RS2, RS10
TC21	SIP call stress test, call burst	x						RS4, RS5, RS6, RS10
TC22	SIP call load test, call burst	x				x		RS4, RS5, RS6, RS10
TC23	SIP call stress test, steady load	x						RS4, RS5, RS10
TC24	SIP call load test, steady load	x				x		RS4, RS5, RS10
TC25	SIP call load test, call reliability					x		RS1, RS8, RS10
TC26	SIP call load scalability, steady and burst		x					RS4, RS5, RS6, RS10
TC27	SIP Bridge components failure recovery testing			x				RS2, RS10
TC28	Web components failure recovery testing			x				RS2, RS10
TC29	Chat stress tests, burst	x						RS6, RS10
TC30	Chat load tests, burst	x				x		RS6, RS10
TC31	Chat stress tests, steady load	x						RS7, RS10
TC32	Chat load tests, steady load	x				x		RS7, RS10
TC33	Chat load reliability					x		RS1, RS8, RS10
TC34	Chat load scalability, steady and burst		x					RS6, RS7, RS10
TC35	E-mail stress tests, burst	x						RS6, RS10
TC36	E-mail load tests, burst	x				x		RS6, RS10
TC37	E-mail load scalability, steady and burst		x					RS6, RS7, RS10
TC38	Integration stress/load tests, burst/steady load	x				x		RS6, RS7, RS10
TC39	Combination test case for basic system performance check						x	RS4, RS10
TC40	Scalability hardware test			x				RS4, RS5, RS10
TC41	Flexibility test			x				RS3, RS10

4.4 Resources for performance testing process

4.4.1 Tools for testing

Performance testing is executed using tools for producing load to the system. The load consists on users performing certain user actions defined in test cases. Also tools for monitoring and measuring the performance of the system are needed.

Load simulator

Simulator is an application which connects the system and simulates actions of system users towards the system according the test cases. Simulator includes adapters which mimic different user interface's or network elements communications towards system Connection components. There are adapters for Web UI, Mobile UI, SIP Phone, SIP Gateway and Chat client. In addition the simulator includes mail generator application which is used in e-mail sending. Also adapter for simulating communications from integration UIs is included.

Simulator performs the set of user actions specified in test cases, collects event data during the test and reports if all parts of test are completed or if there were some errors during the test. The simulator application is used for executing all types of performance test cases. The case scenario with users and events is described for simulator application by writing the required information in test case script file. Test case script files are saved for later use for example for performance after upgrade testing for coming versions. Performing another test using same test case script file should lead to test results which can be compared with previous ones.

Performance monitoring applications

Performance of the system has to be monitored during the tests runs so that performance of the system can be measured and causes for possible problems detected. The performance of the system is monitored on application servers and database servers using Performance monitoring application and performance counters. This is described more closely in chapter 2.5.1.

A monitoring application included in the system viewing system event statistics is used for monitoring active events in the system in real-time while executing the test cases or afterwards from history. Monitoring application provides the number of logged on agents and active calls, chats, e-mails. The correct operation of the system is verified by comparing the monitoring application data with defined test case data and possible problems can be detected once they occur. Monitoring application also collects data about events which can be used when analyzing the test results such as call arrival time, connection time, disconnection time etc.

4.4.2 Staffing for performance testing process

In different phases of performance testing process different type of expertise is required for test execution and testing results analyzing. Like requirement RP16 predicts this kind of resources should be allocated to testing processes so that efficient and productive performance testing plan execution can be achieved and any unnecessary time consumption in seeking of missing information can be avoided. In general, testing activities can be divided in exploratory testing and completion testing. In exploratory testing results are not necessarily clear and need to be investigated more in order to get the needed information and answers. An example of this kind of testing is stress testing where the purpose is to take the system to breaking point and investigate the failure and recovery mechanisms. In this type of testing knowledge from system architecture and operation in detailed is needed to be able to analyze the results efficiently. The personnel having this kind of advanced knowledge on tested area are identified here with name “expert”.

Completion testing contains tests which have more unambiguous results. The acceptance criteria are described in detail for tests of this kind. An example of this kind of testing is performance after upgrade tests which contain test cases loading the system with more manageable level. Some defined performance indicators are monitored and test results compared with ones from earlier tests. This kind of testing requires knowledge on test execution and operation of the system, but deeper knowledge in system and components under testing is not needed as there are only little of test results analyzing to be done. The person having proper knowledge on performance test execution and basic knowledge on system operation and structure will be identified here with name “testing specialist”.

Test plans and activities requiring wider range of expertise are placed in early phase of performance testing. After completing load capacity tests and other tests including advances test results analyzing, rest of tests executed along the system life cycle are more and less standard tests including no detailed test results analyzing. The detailed description which resources are needed in which point is shown in sub chapter 4.5.2 Scheduling.

4.5 Performance testing process and scheduling

The performance testing process is designed based on requirements described in chapter 3. The performance testing process consists of test plans executed according the schedule. Some matters related to performance testing environment and scheduling are described in this chapter. If some requirement is fulfilled by the process definition in this chapter it is mentioned with the requirement identifier in parenthesis.

4.5.1 Environment

The testing environment consists of test system installed on servers dedicated for load testing. Installed test system represents a real productive system and is therefore installed like one having all basic components. The test system is installed according installation guidelines provided for customers if the test does not re-

quire otherwise. The installation of the system may have couple variations described in business process models. However the system structure and configuration will be kept as simple as possible to make reproducing the tests easier and making test results easier to compare. The configuration of the system will be documented to ease the repeatability of test results. The testing is done by executing test cases including only real user events e.g. placing a call, sending an e-mail etc. as a stimulus for the test system. As the testing process includes only activities in system testing, the test system will be used only with this whole system installation. (RP5, RP8, RP9, RP14, RP15)

4.5.2 Scheduling

Requirements for system performance are defined at the beginning of the development process simultaneous with functional requirements. Detailed workload models are designed and description of test plans and test cases specified based on these requirements.

The first performance tests will be executed as soon as first build including most of the system components is provided for system testing (RP4). First testing will consists on basic test with focus on finding any clear performance related issues. The test results are also combined with ones gathered from tests with previous version. This basic testing is covered by executing Performance after upgrading test plan including test cases for testing of all basic functionalities (PAU –T in Figure 5). The test data analysis does not have to be thorough in this context. If any performance issues are detected or there are significant changes in performance test results compared to earlier results, the results are reported to development and more detailed tests are performed if necessary.

As soon as the basic performance of the system is verified the Load capacity testing of the system can begin (LCT in Figure 5). This is done at least once in every program by executing the Load capacity testing plan. As the plan describes, the testing is a combination of tests for components which will be tested just with load testing (fixed, defined load) and for components which will be tested with

stress testing as more information on load capacity is needed. For most of the system components which have not been changed much the load testing with required load level is enough. However new components or components with major changes affecting also the performance has to be tested with stress testing. An expert is needed in data analysis in this test plan when analyzing complex cases of failure

After load capacity testing plan execution Scalability test plan takes place (ST in Figure 5). As the load capacity test plan, also scalability is tested approximately once in a program, Hardware resources are needed in order to test up-scaling. Depending on how well the performance requirements and normal and stress load levels are defined for each scaling level, the testing is performed either by increasing the load in each scaling level until the baseline performance (performance under normal and stress load in basic scaling level) or just by testing each sizing level with defined amount of normal and stress load and comparing the performance results with requirements.

Flexibility and failure recovery plan is also executed in early phase of testing activities (FT in Figure 5). Failure recovery plan can be completely or partially implemented along with load capacity and stress testing plan. An expert is needed in data analysis when investigating failure recovery mechanisms. Flexibility testing is performed by executing certain system maintenance actions while some system event scenario is running active producing events in the system.

After completing scalability, load capacity and failure recovery and flexibility testing, business process model based performance after upgrade – test is performed (PAU –B in Figure 5). After the software has been released to customer the performance testing consists mostly on performance after upgrade type of testing. Performance after upgrade test plan will be executed before every Support Package to be released to customer. If any problem occurs with system performance e.g. some performance metric has changed (increased) significantly, more detailed performance testing for the component will be added. (RP6, RP7)

More detailed and time consuming testing is done at the beginning of the testing process. After that, the testing is planned to consist only little effort consuming test plans. If in any phase some problems occur with system performance more detailed testing will be performed. Figure 5. describes the basic scheduling of the performance testing process integrated to standard testing processes (RP17).

In addition to this schedule also rough performance testing is done by executing the Rough performance and testing tool maintenance testing plan every once in a while. The testing plan contains a set of test cases similar to ones in other test plans. The difference is that only very basic performance data is monitored during the test as the purpose is not to test extensively the system for performance as it is to reveal any clear bugs affecting dramatically to system performance and to detect any changes which could affect the operation of testing tools. (RP7)

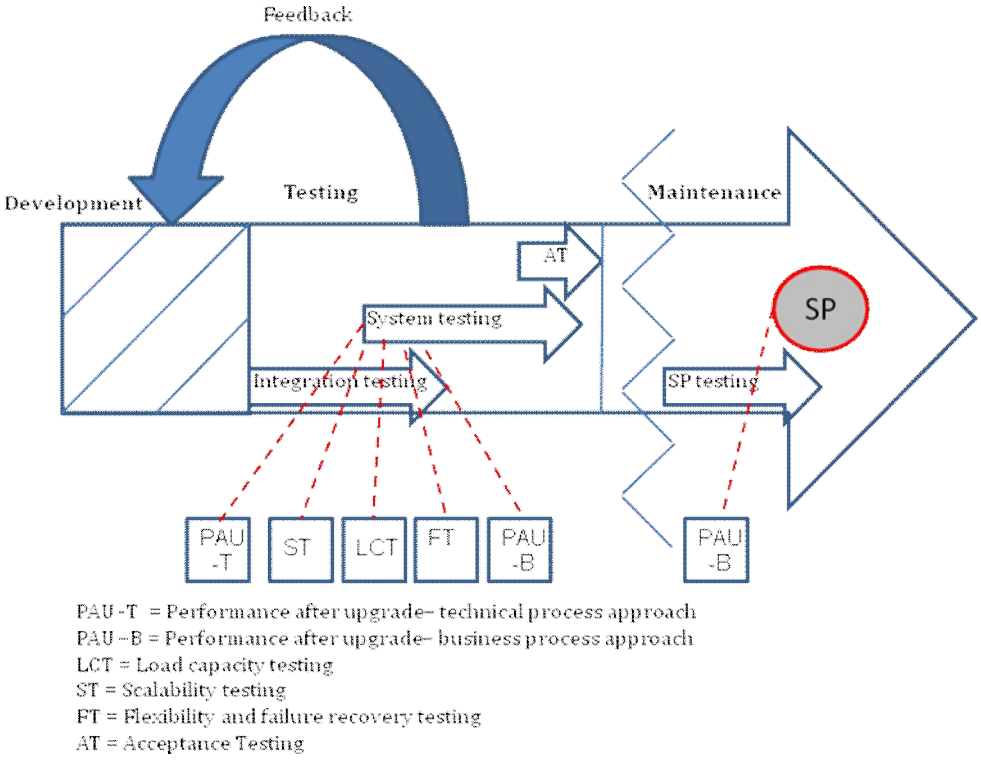

Figure 5 Performance testing process in product life cycle

Table 3 Performance testing process - simple structure

Testing phase	Integration/ System testing	Early system testing	Early system testing	System testing	Support package testing
Test plan	Performance after upgrade – Technical	Load capacity testing	Scalability testing	Flexibility & Failure Recovery	Performance after upgrade - Business
Test effort*	1	5-10	5-10	5-10	1-2
Resources needed	-Testing specialist	-Testing specialist -Expert	-Testing specialist	-Testing specialist -Expert	-Testing specialist

*Test effort includes data analyzing

4.6 Documentation

Documentation is an important tool for collecting and transferring information from different phases of the project between project parties. Yet the amount of documentation has to be kept reasonable so that documentation maintenance won't take too much project time and needed information is easily found from existing documentation. IEEE standard for software test documentation. [IEEE 2008] describes appropriate breadth and depth of test documentation for software with different integrity level. Documents standard suggests for integrity level 3 software are master test plan, level test plan, level test case, level test procedure, level test log and anomaly report. Integrity level indicates the importance of the software to its stakeholders and 3 is a second highest level containing software which malfunction may have serious but still not life-threatening consequences [IEEE 2008]. Master test plan and master test report are common documents for program testing and not separately included in performance testing documentation. Test procedure document describes the steps for executing a set of test cases. In most cases separate document for this purpose is not needed as the information

can be included either in test plan or test case document. Level test log is a document containing chronological information about the events during test run. This information is automatically captured with testing tools and there is no need to document this information further.

Thus, performance testing process includes following documentation activities:

1. Test plan document describing each test plan in detail with information on resources object, scope, scheduling etc.
2. Test case document describing each test in detail
3. Test log produced automatically by testing tools
4. Anomaly report, bug report entered in centralized defect reporting system enabling fast reporting of test results with detailed description and info.

4.6.1 System scaling guide

System scaling guide is a document used by company consulting and customer support teams when planning the system implementation or giving customer recommendations how the system should be installed and in which kind of environment. Information gathered from load capacity testing, scalability testing and flexibility and failure recovery testing is combined into system scaling guide. Recommended maximum values for load which the system can handle while still operating reliably is produced by combining the information from all of these tests. Also recommendations for system up scaling are included in this document. The document consist information which gives answers to below questions:

1. How much load the typical system configuration can handle while operating properly?

2. How much load can separate system component handle while operating correctly?

3. When the capacity of single component runs out, is it possible to add another one? How many additional components can be added? Is it possible to add these additional components to same server with original one or should they be installed to another server?

4. In which point the capacity of the server runs out and another server needs to be added?

5. How does the performance improve while system is sized up with new hardware?

6. How far can we go with the sizing? What is the maximum load capacity of the system?

7. If we know the type of customer system, number of system users and the maximum and normal rate of incoming contacts per hour, how should the system be sized and scaled?

Information from answers to above questions should be combined into equation or graph describing the system and system components scaling as function of load and hardware sizing.

5 Deployment of Performance Testing Process into an Organization

Some issues have to be taken into account when planning the implementation of the performance testing process as a part of standard testing processes. One thing to consider is the resources. In addition to environment and tools for performance testing, completing all performance testing activities requires personnel resources. Also personnel resources with deeper expertise in product architecture and operation will be needed in stress and failure recovery testing. The increase in required work contribution has to be included in testing project plans and resources allocated accordingly.

If allocation of sufficient amount of resources proves to be impossible, testing activities in planned testing process have to be reduced. The reduction of activities has to be allocated carefully so that the risk to miss performance related defects, affecting to key use scenarios, would not increase significantly.

5.1 Present situation and proposition for deployment starting point

In previous programs, load and stress testing of most important software components and features has been completed by developers or testing team. Tests have been performed typically once in a program. Now in current program, the new version of the software has been changed so that earlier used testing tools are no longer usable. New internally developed testing simulator tool has been introduced for performance testing of this new version. Most critical load and stress tests have been executed using this tool but further development and testing is needed in order to implement this performance testing process in its entirety.

As the software version in current program is in design phase, implementation of performance testing process shall start with execution of tests of currently released version for *performance after upgrade* – test plans to obtain reference performance values for comparison in future tests. Testing of testing tools, test cases

and testing environment shall be performed and improvements carried out if needed.

After the implementation phase and unit testing in current program has been completed, performance testing process can be started with basic performance after upgrade tests and executed as a whole.

5.2 Future

In the future, performance testing process shall be fully integrated in standard testing processes and all testing activities described in the process description executed on time. Results from performance after upgrade tests are maintained in centralized testing system and changes in performance detected and reported automatically.

In addition to performance testing activities described in the process description, also other types of methods to estimate products performance in early phase are in use. These methods include modeling system performance in key functions already in design phase and adding performance testing also in unit testing phase. With these actions performance related defects could be detected much earlier and correction costs reduced significantly.

Efficiency of performance testing process has been increased by developing testing tools, test case execution and results data analysis automation so that less effort is needed when compliance with performance requirements is verified and performance properties of software investigated.

6 Conclusions

Performance of the software is an essential factor of quality and should be taken into account already in requirements definition phase so that these requirements could be considered in software design phase as well as in activities and resources planning for performance testing. Requirements for system performance should be extensive and defined in detail so that fulfillment of these requirements could be verified unambiguously with performance testing.

By taking performance testing process into quality assurance processes the company will improve the quality of the product and increase customer satisfactory by preventing any major performance related defects getting to customer. This is accomplished by having a testing process including the testing of performance for each version of the system to be released to customer.

Performance testing process enables continuous performance improvement between product versions. Performance of every release is tested and only test results with unchanged or improved performance are accepted and therefore decrease in performance is not possible with new version.

Defining business process models, described in chapter 3.4, for most important business cases and scenarios and including them in performance testing process enables continuous assurance of system performance in most of the use scenarios in system's real environment of use.

Scaling guide, produced by performance testing process, provides information for productive customer system installations. The guide enables optimized scaling of the system installation and hardware sizing. This may lead to significant savings for customer in hardware purchasing costs. Customer confidence for system performance increases as performance of system installation has been verified.

In future performance testing process should be developed to detect more performance related defects earlier in development phase. With modeling techniques

system performance could be estimated already in planning or design phase. Changes made in this phase of development cause much less extra work and costs compared to changes to already implemented product. Implementing iterative process methods, such as scrum [Sch 2004], in development and QA processes should be investigated and implemented if applicable. Performance testing activities should be added into unit testing phase in addition to testing activities in system testing phase and brought into development sprints in early phase. This would lead to more effective performance testing and earlier detection of defects. However, already defined testing activities in system testing phase should be executed also.

7 List of references

[Ack 2003] Roger Ackerley, Telecommunications performance engineering, The Institution of Electrical Engineers, London, 2003

[AdaRes 2001] Ivo Adan, Jacques Resign, Queuing Theory, Department of Mathematics and Computing Science, Eindhoven University of Technology, 2001

[Avr et al.2002] Alberto Avritzer , Joe Kondek , Danielle Liu , Elaine J. Weyuker, Software performance testing based on workload characterization, Proceedings of the 3rd international workshop on Software and performance, July 24-26, 2002, Rome, Italy

[Bar 2004a] Scott Barber, Creating Effective Load Models for Performance Testing with Incomplete Empirical Data, Proceedings of the Web Site Evolution, Sixth IEEE International Workshop on (WSE'04), p.51-59, September 11-11, 2004

[Bar 2004b] Scott Barber, "Beyond performance testing". IBM DeveloperWorks, Rational Technical Library, 2004, http://www.ibm.com/developerworks/rational/library/4169.html

[Bur 2003] Ilene Burnstein, Practical software testing, Springer–Verlag, New York, 2003

[DavPet 2002] Jonathan Davidson, James Peters (translation: Erkki Huru), Voice over IP (Finnish language edition), IT Press, Helsinki 2002

[Dif et. al 1999] Su Difu; J. Srivastava.; Yao Jey-Hsin, "Investigating factors influencing QoS of Internet phone," *Multimedia Computing and Systems, 1999. IEEE International Conference on*, vol.2, no., pp.541-546 vol.2, Jul 1999 http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=778543

[Eva 2003] Sharon Evans, Telecommunications network modeling, planning and design, The Institution of Electrical Engineers, London, 2003

[Far 2008] Peter Farrell-Vinay, Manage software testing, Auerbach Publications, 2008

[IEEE 1998] IEEE Std 829-1998, IEEE Standard for Software Test Documentation, Institute of Electrical and Electronics Engineers, 1998

[IEEE 2008] IEEE Std 730-2008, IEEE Software and System Test Documentation, Institute of Electrical and Electronics Engineers, 2008

[ITU 2003] ITU-T One-way transmission time. Rec. G.114 INTERNATIONAL TELECOMMUNICATION UNION, 05 / 2003

[Mic 2008] Microsoft TechNet Library, Windows Server 2003 Performance Counters Reference, referred on 11 May 2009, http://technet.microsoft.com/en-us/library/cc776490.aspx

[MonPig 2008] Sergio Montagna and Maurizio Pignolo, Performance evaluation of Load Control Techniques in SIP Signaling Servers, Italtel S.p.A, Proceedings of the Third International Conference on Systems, pages 51-56, 2008

[Mye et. al 2004] Glenford J. Myers, Tom Badgett, Todd M. Thomas, Corey Sandler, The art of software testing, John Wiley & Sons, 2004

[PflAtl 2009] Shari Lawrence Pfleeger, Joanne M. Atlee, Software Engineering: Theory and Practice, Pearson Education (US) 2009

[Sch 2004] Ken Schwaber, Agile project management with Scrum, Microsoft Press, 2004

[Sub 2006] Subraya, B. M., Integrated approach to web performance testing, IRM Press, 2006

[Tik 2005] Tikkanen, Arto, Performance Analysis of Software Systems in Early Life Cycle Steps, Helsinki University of Technlogy, 2005

[Wey 1998] Elaine J. Weyuker, Testing component-based software: a cautionary tale, IEEE Software, Volume 15, Issue 5, Sep/Oct 1998 Page(s):54 - 59

[WeyVok 2000] Elaine J. Weyuker and Filippos I. Vokolos, Experience with Performance testing of Software Systems: Issues, an Approach, and Case Study, IEEE Transactions on software engineering, vol. 26, no. 12, Dec. 2000