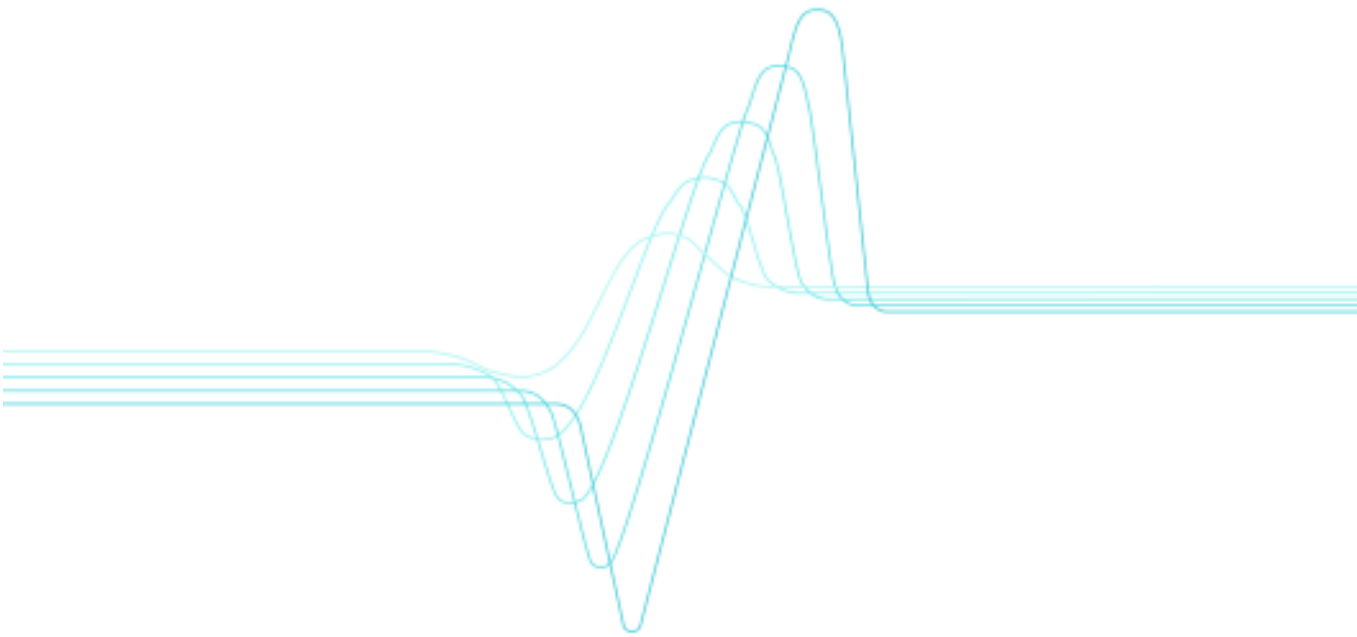
ESPOO 2003 VTT PUBLICATIONS 512

Tarja Kauppi

Performance analysis at the software architectural level



VTT PUBLICATIONS 512

Performance analysis at the software architectural level

Tarja Kauppi

VTT Electronics

ISBN 951–38–6255–0 (soft back ed.)
ISSN 1235–0621 (soft back ed.)

ISBN 951–38–6256–9 (URL: http://www.vtt.fi/inf/pdf/)
ISSN 1455–0849 (URL: http://www.vtt.fi/inf/pdf/)

Copyright © VTT Technical Research Centre of Finland 2003

JULKAISIJA – UTGIVARE – PUBLISHER

VTT, Vuorimiehentie 5, PL 2000, 02044 VTT
puh. vaihde (09) 4561, faksi (09) 456 4374

VTT, Bergsmansvägen 5, PB 2000, 02044 VTT
tel. växel (09) 4561, fax (09) 456 4374

VTT Technical Research Centre of Finland, Vuorimiehentie 5, P.O.Box 2000, FIN–02044 VTT, Finland
phone internat. + 358 9 4561, fax + 358 9 456 4374

VTT Elektroniikka, Kaitoväylä 1, PL 1100, 90571 OULU
puh. vaihde (08) 551 2111, faksi (08) 551 2320

VTT Elektronik, Kaitoväylä 1, PB 1100, 90571 ULEÅBORG
tel. växel (08) 551 2111, fax (08) 551 2320

VTT Electronics, Kaitoväylä 1, P.O.Box 1100, FIN–90571 OULU, Finland
phone internat. + 358 8 551 2111, fax + 358 8 551 2320

Technical editing Marja Kettunen

Otamedia Oy, Espoo 2003

Kauppi, Tarja. Performance analysis at the software architectural level. [Ohjelmistoarkkitehtuuritason suorituskyvyn analysointi]. Espoo 2003. VTT Publications 512. 78 p.

Keywords software quality, analysis methods, software development, mobile phone system, performance

Abstract

This work gives an overview of performance analysis at the software architectural level and methods available for that purpose. Architectural level analysis means analysing the quality of the software in the early development phase based on software architectural designs and estimated timing information. The basic idea related to performance analysis methods is to derive a performance model based on software architecture, which can be analysed and feedback about the performance of the planned software is obtained.

The goal of this work was to analyse the performance of the part of a mobile phone software that is executed on the Symbian operating system during concurrent streaming and multimedia message reception. It was analysed by applying the PASA method, but as a deviation from the method LQN was used for performance modelling. The analysis was conducted mainly by calculating utilisation, residence time and queue length based on the LQN model and estimated execution times. The calculated values were compared to performance objectives and then it was concluded that if the execution times were as estimated in this work then performance objectives would be met on average, but in the worst-case condition performance problems could occur. It was proposed in this work to change the priorities of the related tasks according to RMA principles. Then the deadlines would be met even in the worst-case. In this work performance was analysed only from the processor point of view, so the effect that other hardware resources (such as memory or buses) have on performance was not considered.

Kauppi, Tarja. Ohjelmistoarkkitehtuuritason suorituskyvyn analysointi. [Performance analysis at the software architectural level]. Espoo 2003. VTT Publications 512. 78 s.

Avainsanat software quality, analysis methods, software development, mobile phone system, performance

Tiivistelmä

Työssä esitellään ohjelmistoarkkitehtuuritasolla tapahtuvaa ohjelmiston suorituskyvyn analysointia ja sitä varten kehitettyjä menetelmiä. Ohjelmistoarkkitehtuuritason analyysillä tarkoitetaan ohjelmistokehityksen alkuvaiheessa tehtävää ohjelmiston laadun arviointia perustuen ohjelmistoarkkitehtuurisuunnitelmiin ja arvioituun ajoitustietoon. Tyypillisenä periaatteena suorituskyvyn analyysimenetelmissä on muodostaa ohjelmistoarkkitehtuurin perusteella suorituskykymalli, jota analysoimalla saadaan tietoa ohjelmiston suorituskyvystä.

Työn tavoitteena oli analysoida matkapuhelimen Symbian-käyttöjärjestelmän päällä toimivan ohjelmiston osan suorituskykyä vastaanotettaessa rinnakkaisesti jatkuva-aikaista multimediaa ja multimediaviesti. Analyysi tehtiin PASA-menetelmää soveltaen, mutta siitä poiketen suorituskykyä mallinnettiin LQN-mallinnustavalla. LQN-malli analysoitiin pääosin laskemalla mallin ja arvioitujen suoritusaikojen perusteella prosessorin käyttöaste, prosessorilla oloaika ja jonossa olevien pakettien määrä. Laskettuja parametreja verrattiin määriteltyihin suorituskykytavoitteisiin ja tämän perusteella pääteltiin, että työssä arvioiduilla suoritusajoilla ja oletetuilla pakettien saapumistaajuuksilla suorituskykytavoitteisiin päästäisiin keskimäärin, mutta pahimmassa tapauksessa voisi suorituskykyongelmia ilmetä. Työssä ehdotettiin tehtävien prioriteettien muuttamista RMA-menetelmän periaatteiden mukaisesti, jolloin suorituskykytavoitteisiin päästäisiin jopa pahimmassa tapauksessa. Työssä suorituskykyä analysoitiin ainoastaan prosessorin näkökulmasta, joten esimerkiksi muistin tai väylien vaikutusta suorituskykyyn ei arvioitu.

Preface

This research was performed in the VTT Electronics' Embedded software area's Software architectures group and it relates to two separated projects. The theory part of this research relates to the MOOSE -project funded by TEKES and the practical part relates to a project funded by Nokia Mobile Phones Oulu. MOOSE is a project for studying software engineering methodologies for embedded systems and it relates to this thesis by including a task for gathering software architecture-based performance analysis methods and their characteristics. The project with NMP was for studying performance management and it included a separate task for analysing the performance of Symbian software with concurrent streaming and MMS. In this thesis the PASA (Performance Assessment of Software Architectures) method, LQN (Layered Queuing Network) modelling approach and RMA (Rate Monotonic Analysis) scheduling principles have been applied to concurrent streaming and an MMS case. During the analysis process the instructions given in a white paper introducing the PASA method, a book describing the techniques and principles behind the PASA method and an LQN tutorial have been followed [1, 2, 3].

First I want to thank NMP Oulu for funding and providing such an interesting real-world problem area to be analysed, and Jyrki Leskelä (NMP) and Vesa Pellikka (NMP) for competent technical comments and views concerning the analysis case, and for providing the documents and information needed for the analysis. I would also like to thank Professor Juha Röning for being the superviser of this thesis at the University of Oulu. I am also very grateful to Anu Purhonen for guiding and commenting this work at the VTT. Also Ari Parkkila (VTT) deserves my gratitude for his technical ideas and opinions. Thanks are also due to my family and friends for their support during this challenging and demanding work. Finally, I want to thank you Markku for giving me love and support at home, for your understanding for my long days at work and for a warm dinner waiting for me at home.

Oulu 4.3.2003

Tarja Kauppi

Contents

Abstract 3

Tiivistelmä 4

Preface 5

Abbreviations 8

1. Introduction 9

2. Performance engineering 11

3. Performance analysis methods 15
   3.1 Performance Assessment of Software Architectures 16
   3.2 Layered Queuing Networks 19
   3.3 Labelled Transition System and Queuing Network Model based method 20
   3.4 Coloured Petri Nets 21
   3.5 Rate Monotonic Analysis 23
   3.6 Stochastic Process Algebra 25
   3.7 Summary 26

4. Problem definition 29
   4.1 Requirements for the client terminal 29
   4.2 Target architecture 33
   4.3 Arrival rates 36

5. Method selection and application process 37
   5.1 Method selection 37
   5.2 Methods' application process 39
   5.3 Architectural analysis 41
      5.3.1 Architectural styles 41
      5.3.2 Performance antipatterns 43
      5.3.3 Performance modelling and analysis 45

6. Performance analysis 51
   6.1 The first six PASA steps 51

6.2 PASA step 7: Architectural Analysis 53
6.2.1 Identification of the underlying architectural style(s) 53
6.2.2 Performance modelling and analysis 54
6.3 PASA Step 8: Identification of Alternatives 64
6.4 PASA Step 9: Presentation of Results 65
6.5 Experiences 66

7. Discussion 69

8. Conclusion 73

References 75

Abbreviations

3G	Third Generation Network
3GPP	Third Generation Partnership Project
CPN	Coloured Petri Nets
CPU	Central Processing Unit
CTMC	Continuous Time Markov Chains
DSP	Digital Signal Processing
FIFO	First In First Out
GERAN	GSM Edge Radio Access Network
GSM	Global System for Mobile communications
HTTP	Hypertext Transfer Protocol
I/O	Input/Output
IP	Internet Protocol
ISO	International Standardisation Organisation
LOTOS	Language Of Temporal Ordering Specification
LQN	Layered Queuing Network
LTS	Labelled Transition System
MMS	Multimedia Messaging Service
MMSC	Multimedia Messaging Service Centre
PASA	Performance Assessment of Software Architectures
PSS	Packet Switched Streaming
QNM	Queuing Network Model
RMA	Rate Monotonic Analysis
RMS	Rate Monotonic Scheduling
RTCP	RTP Control Protocol
RTP	Real-time Transfer Protocol
RTSP	Real-Time Streaming Protocol
SMS	Short Message Service
SPA	Stochastic Process Algebra
SPE	Software Performance Engineering
SW	Software
TCP	Transfer Control Protocol
UDP	User Datagram Protocol
UML	Unified Modelling Language
UMTS	Universal Mobile Telecommunication System
UTRAN	UMTS Terrestrial Radio Access Network
WAP	Wireless Application Protocol

1. Introduction

Software architectural level analysis means analysing the quality of the software system in the early software system development phase, when no implementation is available to be measured. Analysis is conducted based on software architectural designs and descriptions. Software architecture means the structure of the system and it consists of software components, the externally visible properties of the components and the relationships among them [4 p. 21]. Since decisions concerning software architectures are made in the early phase of the development, they will have an enormous effect on the system during the whole life cycle. Making changes to the architecture in the later phases is difficult and complex. Architectural level analysis helps in identifying the potential problems in the early phase, when changes are not as complex and expensive to make.

The quality of software can be considered from different quality attribute points of views (such as maintainability, modifiability, security and performance). This work concentrates on performance as an important quality factor. Performance here means the responsiveness of a system, the time to respond to events or the number of events processed in some interval of time [4, p. 79].

Performance has always been problematic, but because the size and complexity of software systems has increased performance problems have become more common than earlier. Therefore, more attention should be paid to performance issues in the early software development as early as the software architectural design phase. Predicting and guaranteeing performance before the system has been built has become an interesting issue. Meeting performance is especially important in real-time systems, because tasks are assumed to be completed by a specified deadline. Missing a hard deadline in a real-time system can result in catastrophic loss of system performance or even loss of life [5]. The main issue in meeting deadlines in a system is the competition for shared resources (e.g. a processor, an input/output device, or a data structure) by multiple tasks [5]. Performance problems may occur if a time critical task does not receive all the needed resources on time, because some other task is controlling the resources.

The importance of early performance analysis has been noticed and many methods for software architectural level performance analysis have been presented in domain conference papers and magazines. Methods have already been applied in

several case studies such as data processing systems [1], mobile phone family [6], database applications [7], web-servers [7], telecommunication systems [7] and multiphase compilers [8].

An example of a highly complex software system is a mobile phone system, which is in fact a multimedia, telecommunication, database and signal processing system providing several different services to its users such as phone calls, multimedia messages, short messages and multi-player games. Some of the services provided require response by a specified deadline, otherwise they may not be accepted by users. Managing and guaranteeing the performance of such a complex system is not easy. The current case studies do not illustrate how applicable available performance analysis methods are for analysing performance of concurrently provided services in such a complex system.

The goal of this research is to find out and illustrate how performance analysis is conducted in practice at the software architectural level, what kind of assumptions have to be made, what kind of information is required during the process and what kind of information is produced as a result. First available software architecture-based performance analysis methods are collected and compared, and then the most potential ones are selected to be applied to analysing the performance of a defined part of a mobile phone system during concurrent streaming and multimedia message reception. The case is a problem from the real world and the work is performed for a customer organisation. To preserve confidentiality this research tries to concentrate on the applicability of the method not the details of the target system. This research may help industries in deciding if the methods are ready to be applied more widely in the real-world and it may also help future analysts in the same problem area in their work.

2. Performance engineering

Performance engineering is describing, analysing and optimising the dynamic time dependent behaviour of a system [9]. Methods and tools have been invented to support performance engineering to guarantee that an implemented software system will meet its performance goals. The idea in performance engineering is to apply methods and tools during the whole development process of a software system starting from the early phase [9]. Performance engineering is not a new concept. Already the developers of early computing systems had to concern performance issues during the software system development, because the hardware was slow and memory capacity was low [10, pp. 4-5]. The performance problems did not disappear even if the hardware grew and became faster, because the complexity and size of the software also grew [10, p. 4]. In 1990s Connie U. Smith created a systematic approach for constructing a software system to meet performance objectives from the early phase of software development. The approach is called the Software Performance Engineering (SPE) method [10] and it has inspired many researchers to create their own methods and tools for performance engineering purposes.

According to Connie U. Smith there are two competing approaches relating to performance issues during a software system development: the traditional approach and the software architecture based approach. Figure 1 [10, p. 8] illustrates how the approaches relate to a typical software development phases. Traditionally performance issues are ignored until the integration testing phase. Whereas, in the software architecture based approach performance issues are considered as early as in the software architectural design phase. The competing approaches mentioned are described more specifically in the next sections. The ideas behind the approaches are based on the description by Connie U. Smith (1990) [10, pp. 4-14], but some own examples of implications and possible advantages and disadvantages have been added to the description. The purpose of this chapter is to provide reasons why the traditional approach is not recommended and why the software architecture-based approach should be used instead.

The traditional approach is based on the idea to first implement the system, then test and measure the performance of the system, and fix the problems if they appear. If no performance problems are revealed, then this approach will be a fast way to build a functional system. However, if performance problems appear, then the

software system is modified to meet the expected performance. Modifying the software system consumes time and money, therefore the product or service may not be completed and released on time and a cost overrun may also occur, because of the additional work and delayed release. The implications mentioned may also have other negative consequences such as lack of customer satisfaction, resulting in the customer changing the service provider because of the missed deadlines and cost overruns.

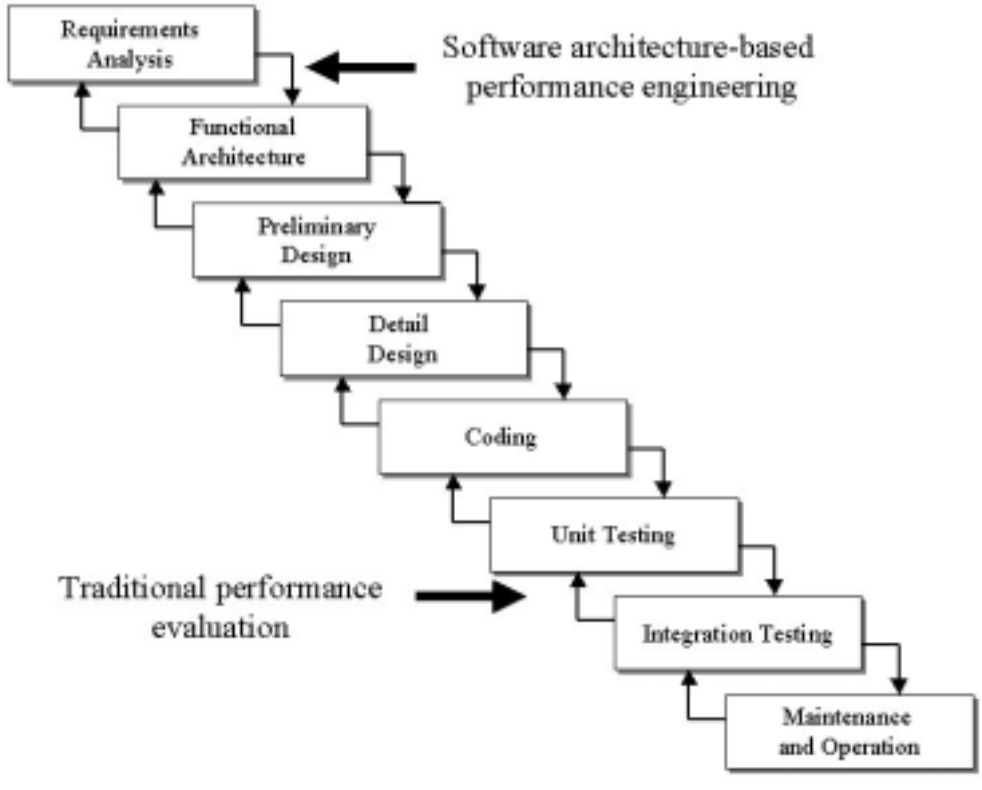


Figure 1. A software development process [10, p. 8].

Typically, the performance is improved, if possible, by modifying the code. If everything goes as hoped, tuning the code solves the performance problems. However, tuning the code may have side effects on other quality factors of the software carefully designed at the beginning of the project. For example reusability of the software may suffer, power consumption may increase and security may get

worse. In a worst case, tuning and modifying the software may not have the hoped impact on the performance of the software and this may have many kinds of implications. Some features or services may have to be left out of the product or software system to meet the performance goals with the implemented software and hardware. If no features or services can be left out, then the software may have to be redesigned and implemented, or the hardware may have to be changed to be faster.

The idea in software architecture-based performance engineering is to design the software system in a way that it meets the performance goals when it is implemented. Available performance engineering methods and tools are used for this purpose. Possibly the most important advantage of the approach is that performance problems are noticed in the early development phase, when changes may not be as complex and time-consuming to perform as in the later phases.

However, applying the approach especially increases the amount of work needed at the beginning of the software development and an implementation of the system is not built as fast as in the traditional approach. However, the system built already meets the expected performance and therefore no work is needed for tuning the code and modifying the system. So, the approach does not delay the software development process, but it changes the time required in different development phases.

The software architecture-based approach for performance issues is still waiting for its breakthrough in industry because, for many years, the approach has not been mature enough to be taken into wider use. Five years ago an industrial case study [6] concerning software architecture-based performance analysis was published and it pointed out that the field was not mature enough from the industrial perspective, because of lacking handbooks, instructions and extensive case studies.

The acceptance of the software architecture-based approach has also been low, because managing the performance is considered to take too much time, and the performance models required by architectural analysis methods are thought to be complex and expensive to construct [2, p. 11]. Also estimating resource requirements in the early software system development phase is often considered to be difficult [2, p. 195].

However, research concerning the software architecture-based performance analysis has been lively during the five last years: new methods [1,7,8] for architectural level analysis have been presented, case studies [1,7] with positive experiences have been published and instructions have been developed for guiding the analysis process [2,3]. So, progress has taken place. The next chapter discusses the available software architecture-based methods for analysing the software architecture of a planned system.

3. Performance analysis methods

Software architecture-based performance analysis methods are applied in the early phase of the development of a software system. The methods take software architecture as an input based on which a performance model is typically drawn (see Figure 2). After this some timing information is added to the model and the analysis can be conducted. The actual analysis is often performed using some tool supporting the method or by calculating some performance metrics. In addition to analysing the performance model architectural styles and performance antipatterns have also been proposed as analysis tools [1]. Architectural styles define a family of systems in terms of a pattern of a structural organisation [11, p. 20]. They define components (such as clients, servers and filters) and connector types (such as procedure calls, event broadcast and pipes), and rules how the components and connectors can be combined. Antipatterns are documented bad solutions to commonly occurring design problems [12, p. 7]. Their use generates negative consequences in the system. They document common mistakes in software development and solutions for these mistakes [2, p. 287]. Performance antipatterns are documented common performance problems and solutions to overcome them [2, p. 287].

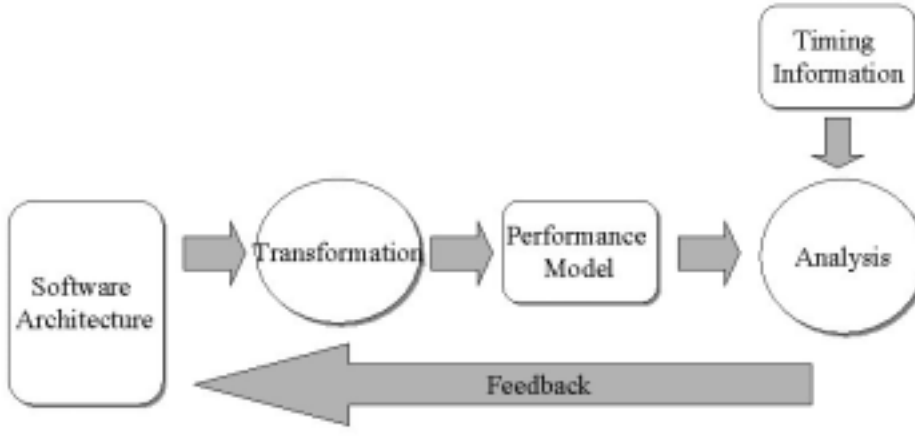


Figure 2. The idea in performance analysis at the architectural level.

Architecture-based performance analysis methods can be used for predicting the performance of a system [1], for guaranteeing that performance goals are met [5, 1] and for comparing different architectural choices from the performance point of

view [13, 8]. The methods can help in finding the bottleneck resources [1] and identifying the potential timing problems even before the system is built [7]. The methods provide useful information on performance related to a planned architecture, and the information can be used as an aid when deciding if the architecture is worth implementation [2].

The analysis methods should be easy to integrate into the conventional software development process, although they are not accepted by industry. Unified Modelling Language (UML) is widely used and it has been adopted as a standard for designing software [14, 2, 7, 9]. Therefore, the analysis methods should support using UML in the design phase.

In the following sections the state-of-the-art software architecture-based performance analysis methods are presented. The preliminary purpose of the methods, application process, tool support and some other characteristics are described. The list of the methods presented here may not be exhaustive, but the methods that are presented were the ones that were found during this research.

3.1 Performance Assessment of Software Architectures

The Performance Assessment of Software Architectures (PASA) method [1] has been presented by L. G. Williams and C. U. Smith. The PASA method uses the same principles and techniques as SPE. The purpose of the SPE is to construct and design a software system to meet performance objectives. Whereas, the PASA method is for finding out if a software system will meet its performance objectives. The PASA method can be applied both in revealing potential problems in a new software system under development and also when upgrading legacy systems to finding out if it is worth committing resources to the existing system. The PASA method provides a framework for the whole performance assessment process starting from a PASA process overview to presenting the results of the analysis. The steps proposed by the method with short descriptions are presented in Table 1 [1].

PASA proposes three different techniques to be applied in the analysis phase: identifying underlying architectural style(s), identifying performance antipatterns and analysing the performance model. The idea is to identify underlying architectural styles and to find out if they are good from the performance point of

view. If deviations from the basic style are found, these points are analysed more specifically by using performance antipatterns, which also document solutions to previously found performance problems. Some points of the software architecture may require more quantitative analysis. Therefore, those points are modelled and the model is analysed.

Table 1. Steps of PASA method [1].

Step	Description
1. Process Overview	The stakeholders are familiarised with the reasons for the assessment, assessment process and the expected outcomes from the process.
2. Architecture Overview	The current or planned architecture is overviewed.
3. Identification of Critical Use Cases	Use cases that are important from the performance point of view are identified.
4. Selection of Key Performance Scenarios	Uses cases typically consist of scenarios that are not all important from the performance point of view. The scenarios that are important to performance are selected.
5. Identification of Performance Objectives	For each selected key performance scenario at least one performance objective is identified. The objectives have to be quantitative and measurable.
6. Architecture clarification and discussion	A more detailed discussion of the architecture is conducted. Especially features supporting the key performance scenarios and problem areas are explored.
7. Architectural Analysis	The architecture is analysed to find out if performance objectives are met or not.
8. Identification of Alternatives	If the architectural analysis reveals a problem, then alternatives for meeting performance objectives are identified.
9. Presentation of Results	Results and proposals are presented to the stakeholders.

PASA proposes two types of models to be drawn: the software execution model (see Figure 15) and the system execution model (see Figure 3). The software execution model is a simple model consisting of execution graphs, which consist of arcs and nodes [2 pp. 72-73]. The arcs represent the order of processing and nodes represent the processing steps. The other option to execution graph is to draw UML sequence diagrams augmented with timing information [2, p. 73]. Solving both types of the software execution models produce optimistic results, because they characterise the resource requirements of the proposed software alone, in the absence of other workloads, multiple users or delays due to contention for resources.

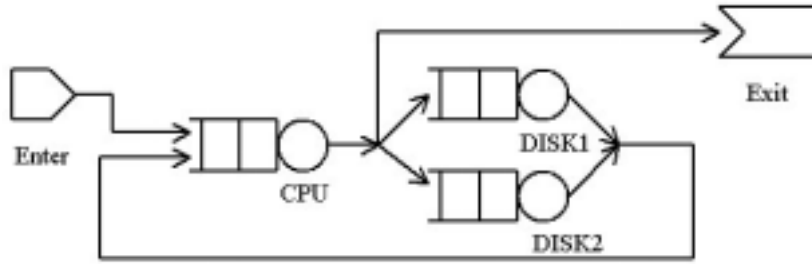


Figure 3. An example of a QNM model.

The system execution model characterises software performance in the presence of factors such as multiple users or other workloads, that could cause contention for resources [2, p. 134]. The system execution model is based on a well-known Queuing Network Model (QNM), which consists of queues, servers and service requests. The requests to the services provided by a server arrive in a queue. The job to be served is selected based on some scheduling policy.

SPE • ED tool [15] is initially developed to support the SPE method, but it can also be used as an aid in the PASA performance assessment process. It automatically generates a QNM based on given information and it also performs quantitative analysis. The PASA method and the techniques used by the method have several advantages: it is easy to apply and to integrate in the software development process [2], the models are simple [2], it provides a framework for the whole assessment process [1] and the assessment process can be speeded up thanks to the available tool support [2].

3.2 Layered Queuing Networks

The Layered Queuing Network (LQN) modelling approach is an extension of QNM, but it is especially developed for modelling concurrent and distributed software systems [7]. LQN modelling notation is not standardised, therefore different notations have been presented [7, 3]. One possible notation of a LQN model can be seen in Figure 4 [3]. The idea in LQN modelling is the same as in QNM: the requests or jobs arrive in a queue to the server. However, in LQN modelling approach the queue is considered to exist, but it is not modelled [3].

A systematic method for transforming UML descriptions of the high level architecture of a system into a LQN model has been presented [7]. The method proposes a LQN model to be built by transforming each architectural pattern related to a system into a performance submodel. Architectural patterns are frequently used architectural solutions [7]. The main aim of the method is to produce a LQN model, that can be analysed with a tool and the results are such things as response time, throughput, queuing delays, and utilisation of different software and hardware components. The analysis reveals the bottleneck resources, and the information obtained from the analysis can be used as an aid in choosing the right changes to the system, so that the system will eventually meet its performance requirements. The method can be used for assessing performance effects of different design and implementation alternatives, from the earliest stages of software development throughout the whole lifecycle.

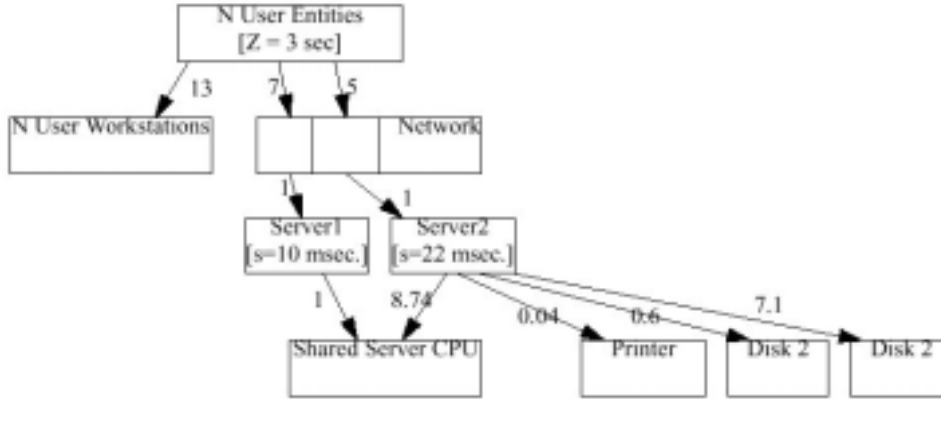


Figure 4. An example LQN model [3].

In addition to the knowledge about software architecture the method also requires the following parameters: occurrence pattern of client classes, average execution times for software components, average service times at the device and average number of visits at the device, average message delays, and scheduling discipline. The parameters mentioned depend on low-level design and implementation decisions. Therefore, at the beginning of the development process the analysis is performed with estimated parameter values. The idea is to reanalyse in the later phases with more accurate parameter values. So, the results of the analysis become more accurate during the process.

The LQN based method has already been applied to several concrete industrial systems such as database applications, web servers and telecommunication systems [7]. There is a LQNS tool available for solving a LQN model [3]. The method has at least the following advantages: it provides insights into performance limitations at software and hardware levels, it suggests performance improvements in different development stages and it can be used as an aid in system sizing and capacity planning [7].

3.3 Labelled Transition System and Queuing Network Model based method

This method [8] provides an approach to evaluating the expected performance of a software architecture. The purpose of the method is to provide a set of measures to compare the performance of competing software architectures at their high abstraction level, and to provide the possibility to interpret the evaluation results in terms of the system development process [8]. The method also provides the designer with a high level framework to making the design decisions in the later development phases [8].

The main phases of the method are presented in Figure 5 [8]. First the software architecture is described and then it is modelled as Labelled Transition System (LTS). In the next phase an algorithm is applied to derive a performance model. The performance modelling approach supported by the method is QNM (see Figure 3). Some additional information about the state annotation and the type of communication among software architecture components is also needed for

defining the complete QNM. The results are obtained by analysing the QNM model and, by evaluating results, feedback about the architecture is obtained.

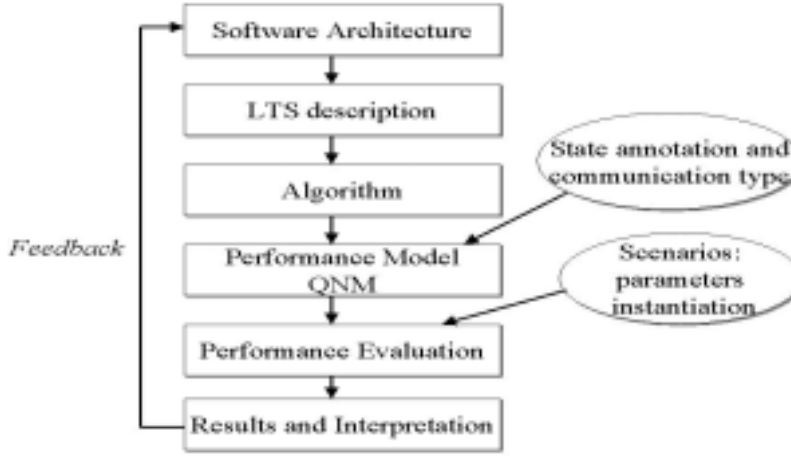


Figure 5. Phases of the approach [8].

The method has been applied at least to a case study of a multiphase compiler [8]. There is no tool available that supports all the steps of the method, but some separate tools can be used [8]. The approach has at least the following advantages: it helps to detect design problems in the early phase of the system development cycle, it provides the designer with a high level framework to derive subsequent design decisions and it does not require information from the detailed design phase or deployment to the hardware [8].

3.4 Coloured Petri Nets

The Coloured Petri Nets (CPN) [6] modelling approach can be used for modelling software architectures. By adding timing information to the model, it can be used for specification and validation of both functional and performance properties of the software system. The CPN model is a hierarchical description consisting of pages and subpages. Therefore, large and complex systems can be modelled in a manageable and

compositional way [6]. An example of the most abstract page of the CPN model related to a mobile phone system is presented in Figure 6 [6].

The CPN analysis is an iterative process as illustrated in Figure 7. The CPN model is created based on the software architecture and the modelling starts when the structure of the new software architecture is designed and the modelling process continues along with the component architecture design and detailed system design. The execution architecture model can be structured by grouping components into sub-pages. The model is improved until it is stable enough. Timing parameters are added to the model and necessary data structures are defined. The CPN model is simulated and analysed using the Design/CPN tool. As a result of the analysis feedback about the architecture is obtained.

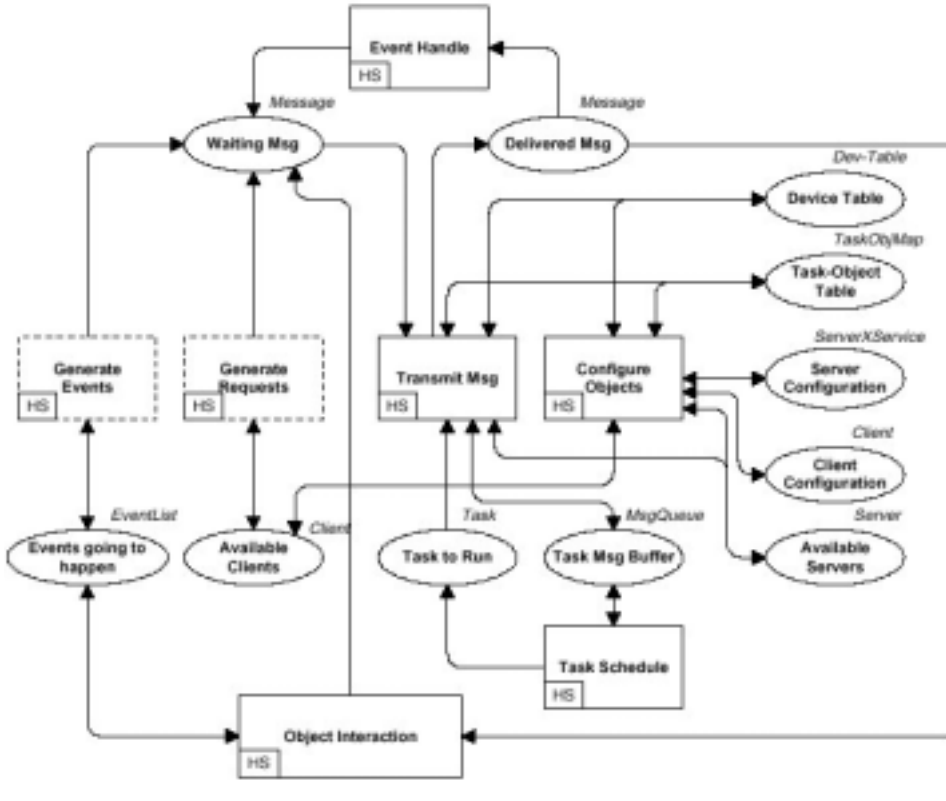


Figure 6. An example of the most abstract page of the CPN model [6].

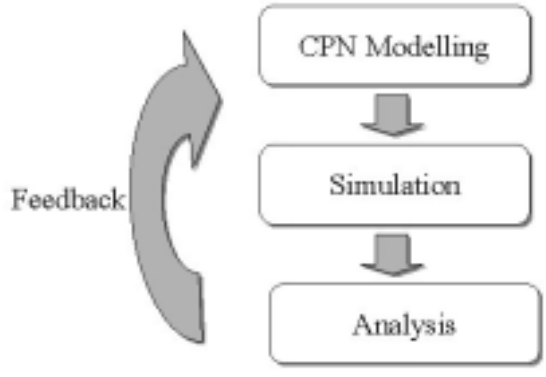


Figure 7. CPN analysis process.

CPN has at least been applied to the software architecture of a mobile phone family [6], but the method has not, as far as we know, been applied to the architectural level analysis after that. The Design/CPN tool supports building an architectural model, simulation and formal analysis [6]. The tool is said to be practical, the modelling approach is proved to be useful and the analysis is said to provide valuable feedback to the development of the architecture [6]. However, developing the correct CPN model requires deep understanding of the system structure and knowledge about CPN [6].

3.5 Rate Monotonic Analysis

Rate Monotonic Analysis (RMA) is a collection of quantitative methods that enable real-time system developers to understand, analyse, and predict the timing behaviour of real-time systems [16]. It provides mathematical formulas and principles that designers can use as a tool to guarantee that when the system is ready and implemented it will meet its timing requirements, even in the worst-case condition [5]. RMA can also be used for identifying potential timing problems even before the system is implemented [5].

RMA is based on the Rate Monotonic Scheduling (RMS) algorithm, which provides principles for scheduling tasks. The idea is to assign the highest priority to the task with the highest rate and the remaining tasks get the priorities in order of their rates. RMA provides many different techniques with different aspects to schedulability.

For example the schedulability of a set of tasks can be analysed with the formula (1) if tasks are periodic, are not synchronised with one another, do not suspend themselves during execution, and are capable of being preempted by higher priority tasks [16, 2].

Tasks will always meet their deadlines if the following precondition are met:

$$\frac{C_1}{T_1} + \frac{C_2}{T_2} + ... + \frac{C_n}{T_n} \leq U(n) = n(2^{1/n} - 1) \quad (1)$$

where

C_i is the execution time for task *i*,

T_i is the period of task *i*,

n is the number of tasks and

U(n) is utilisation limit for *n* tasks.

RMS scheduling principles do not cater for priority inversion, which means the situation in which a low-priority task prevents a high-priority task from executing [16]. Whereas, RMA also reveals these sources for priority inversion. The RMA analysis concentrates on assessing the preemption, execution, and priority inversion that affect each task's ability to meet its deadlines in the worst-case situation [5]. Designers can use the information obtained from the analysis for improving and guaranteeing the performance of the system.

RMA has been applied by several organisations in software development efforts for example by Boeing, General Dynamics, Honeywell, IBM and NASA [5]. The principles of RMA have been adopted in the standards of Ada 9X, Futurebus+, and POSIX [5]. Alsys, DDC-I, Lynx, Sun, Telesoft, Verdix, and Wind River have adopted scheduling protocols that support the use of RMA in building more reliable real-time systems [5]. There are at least two commercial tools supporting the principles of RMA: RapidRMA [17] and TimeWiz [18]. Both of the tools enable performing the analysis right from UML descriptions by adding some timing

information to the descriptions, therefore the analysis can easily be integrated to the software development.

3.6 Stochastic Process Algebra

Stochastic Process Algebra (SPA) [9] is an analytic modelling method and its main purpose is to support performance validation of a system. It uses continuous time Markov chains (CTMC) [19] for performance prediction. The method can be used for modelling and analysing the behaviour of a system. The behaviour can be analysed from the following point of views: functional (such as deadlocks), temporal (such as throughput) and combined properties (such as probability of timeouts) [9]. Stochastic process algebra is an extension of classical process algebra, but performance evaluation features have been added [9]. The goal of the process algebras is to provide a systematic approach to constructing complex systems from smaller parts and to check formally if the systems behave equivalently [9].

The SPA process requires functional information of the target system as an input and the main idea of the process is presented in Figure 8 [9]. Timing information is added to each level of the process. The system is described with a high level language which is an enhanced version of BASIC LOTOS (Language of Temporal Ordering Specification), which is the core language of ISO standard 8807 [9]. The SPA method provides rules (called formal semantics) for translating language expressions automatically into states and transitions of the labelled transition system (LTS). SPA defines equational laws based on which two systems or system components can be compared considering both functional behaviour and timing information.

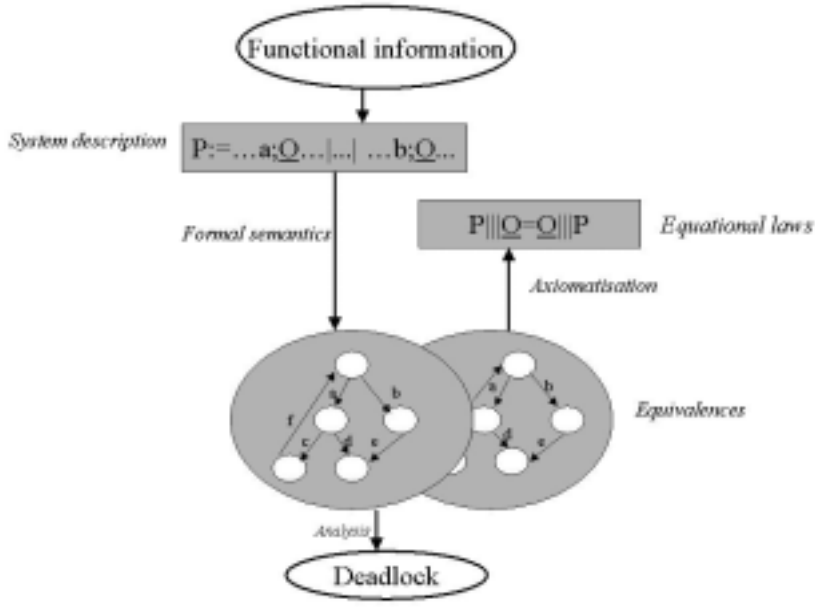


Figure 8. Principal of process algebras [9].

The SPA method has been applied to several domains and several tools supporting stochastic algebras are available such as the PEPA Workbench, Two Towers and TIPPTool, which is a prototype modelling tool containing most of the specification and evaluation features of SPA [9]. The accuracy of the evaluation with the SPA method is from good to high, but the costs of applying the method are high in most cases [9]. The acceptance of SPA has been low because of the unconventional theoretical foundation [9].

3.7 Summary

This section summarises the main properties of the methods that were presented earlier. The idea of most of the methods found is to derive a performance model based on software architecture descriptions and timing information. Feedback about the performance of the designed architecture is obtained by analysing the model. The methods presented some competing approaches to performance modelling. The most popular ones are queuing theory based approaches such as QNM and LQN.

Also Petri nets and stochastic algebras are suggested for use with modelling. The RMA method differs from the other methods in that it does not propose any certain modelling to be used. However, RapidRMA and TimeWiz tools supporting RMA can be used for analysing the schedulability of a system right from UML diagrams augmented with timing information.

Typically, it is assumed that a tool is used for analysing the performance model. The LTS and QNM based method is the only method that had no tool support available for the whole analysis process. The PASA method differs from the other methods presented in that it proposes, in addition to analysing the performance model, also identification of architectural styles and antipatterns to be performed in the analysis phase. The method also provides a framework for the whole assessment process starting from the process overview to presenting the results and observation of the analysis. The most formal of the presented methods seems to be SPA, therefore it has not gained much popularity. The methods with their preliminary purpose, supported modelling approach and some other properties are summarised in Table 2.

Table 2. The summary of the properties of the methods.

Method	Purpose	Performance modelling approach	Other properties
PASA	Predicting performance of the system and uncovering potential problems.	Software execution graph and QNM.	Provides a framework to the whole performance analysis process. Easy to apply and integrate to the development process.
LQN	Revealing possible weaknesses in the architecture, and assessing the performance of different design and implementation alternatives.	LQN	Allows modelling of concurrent scenarios.

LTS & QNM	Comparing performance of different software architectures.	QNM	Has not gained much popularity. Not many case studies available.
CPN	Validation of both functional and performance properties.	CPN	Knowingly applied only at once at the architectural level. Solving the model requires a tool to be used.
SPA	Performance validation of a system.	SPA, LOTOS	Formal approach. Has not gained much popularity.
RMA	Guaranteeing that system will meet its timing requirements. Identifying potential timing problems.	Does not propose any modelling approach to be used.	Does not provide support for collecting the required information. Provides only formulas for counting if the system is schedulable or not.

4. Problem definition

A mobile phone provides several different services to its users. Some of them require real-time or almost real-time response and some of them do not have timing constraints. Time critical services establish performance requirements for the mobile phone system. An example of such service is multimedia streaming. According to the 3GPP streaming service standard [20] streaming is the ability of an application to play synchronised media streams like audio and video in a continuous way while those streams are being transmitted to the client over a data network. So, multimedia streaming content should be handled and rendered on time to look and sound continuous from the user point of view. Even if a multimedia streaming service could be provided without performance problems in the absence of other workloads, problems may appear if some other traffic (such as MMS) is received concurrently. A multimedia message can include multiple media such as audio, video, text and picture [21]. The size of those messages may be much greater than the size of text messages (SMS) [21] and handling them loads the shared resources (such as processor, memory and I/O -devices). This can cause delays to time critical tasks such as processing of received streaming content.

The aim of this research was to analyse the performance of a defined part of a mobile phone system during concurrent streaming and MMS message processing by using some available software architecture based performance analysis method(s). The analysis was performed based on design and architectural documentation, so no implementation was available for measurements or testing. This chapters defines the analysed problem area. First, the requirements to the client terminal defined by standards are described in section 4.1. Then, the target architecture that was analysed is described in section 4.2 and assumed arrival rates for streaming and MMS packets are described in section 4.3.

4.1 Requirements for the client terminal

Open standards define both streaming and MMS services. Both of the services can be implemented by using either the existing circuit-switched network or packet-switched 3G network [22, 23]. In this case study, it was assumed that the content of both streaming and MMS are received in packet-format through the 3G network. 3GPP standards [20, 24, 23, 25] define network architectures, protocols and codecs

that should be supported to enable interoperability with service content providers and other terminals.

The standardised streaming service architecture is presented in Figure 9 [20]. The idea [22] related to the streaming service is that first the session between content provider and client terminal is established according to protocols. Then the streaming server starts sending requested multimedia content to the client terminal, which uncompresses the content data and plays it to the user with output devices such as display and speaker. The client terminal receives the video and audio stream in different packets through either general radio access network (GERAN) or UMTS terrestrial radio access network (UTRAN) [20].

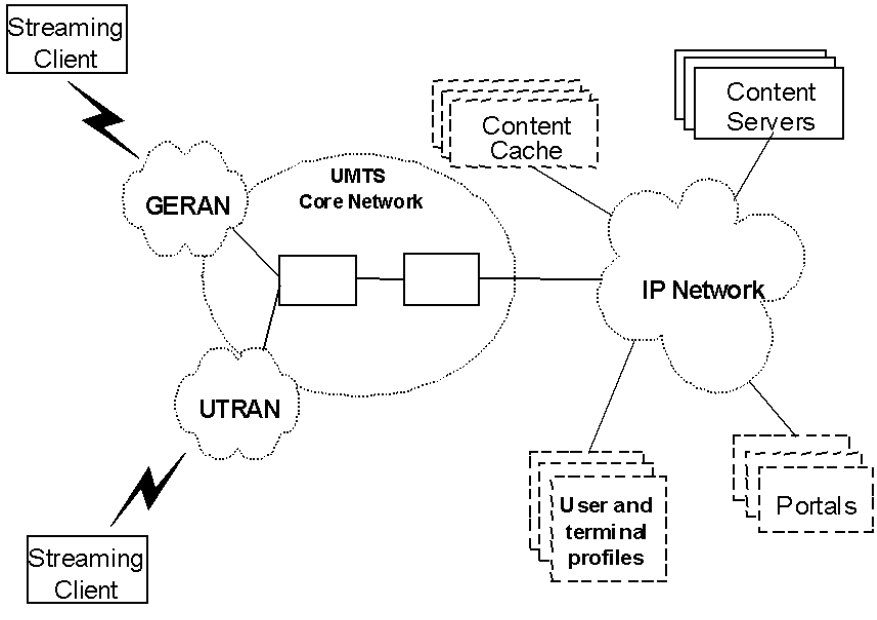


Figure 9. Network elements involved in a 3G packet switched streaming service [20].

The client terminal should include protocols for session establishment, session setup, session control, scene description and data control, and also codecs for speech, audio, video still images and bitmap graphics [20]. The standardised protocol stack related to streaming is presented in Figure 10 [24]. The required

protocols for receiving audio and video stream packets are IP, UDP, RTP and RTCP for payload formats.

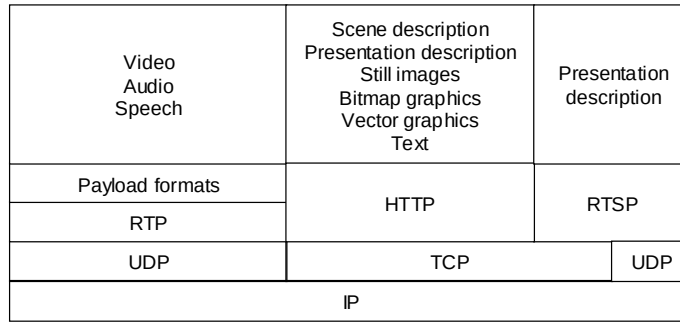


Figure 10. Protocol stack for streaming service [24].

The client terminal needs also to include a media player that manages the connection opening, uncompression of data, sending the video data on the display and audio data to the speakers, controlling media flows, and interfacing with the underlying transport network technology and its specific protocols and data bearers that are dedicated to the service [22]. The benefit of streaming technology is that the content need not to be downloaded to the local disk of the client terminal before starting to presenting the content to the user [22]. The functional components that a streaming client terminal should include are presented in Figure 11 [24].

The standardised MMS network architecture is presented in Figure 12 [21]. The key part of the network is the multimedia messaging service centre (MMSC), which stores and forwards multimedia messages [21]. It enables multimedia messages to be sent with several content types from terminal to terminal, with instant delivery [21]. First a notification of a MMS is sent to the client terminal, which handles the message and notifies MMSC when it is ready to receive MMS content information packets [23].

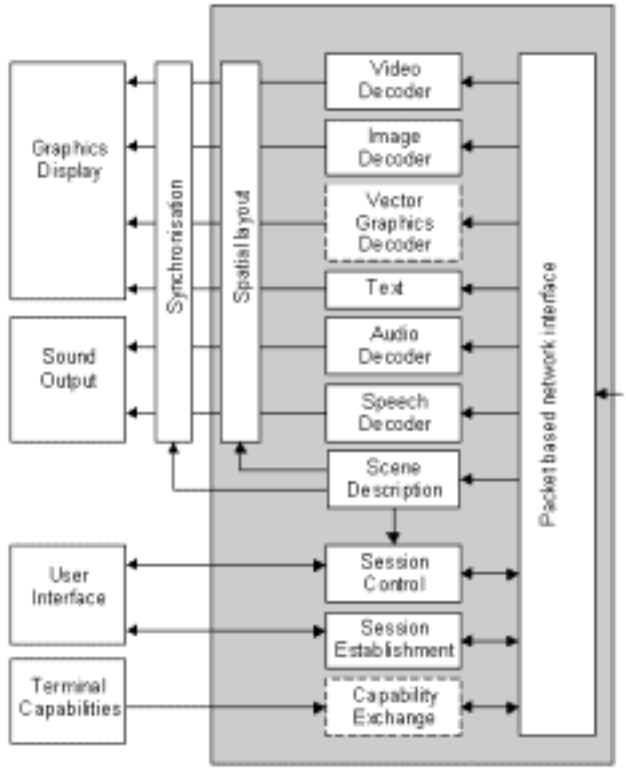


Figure 11. Functional components of a streaming client [24].

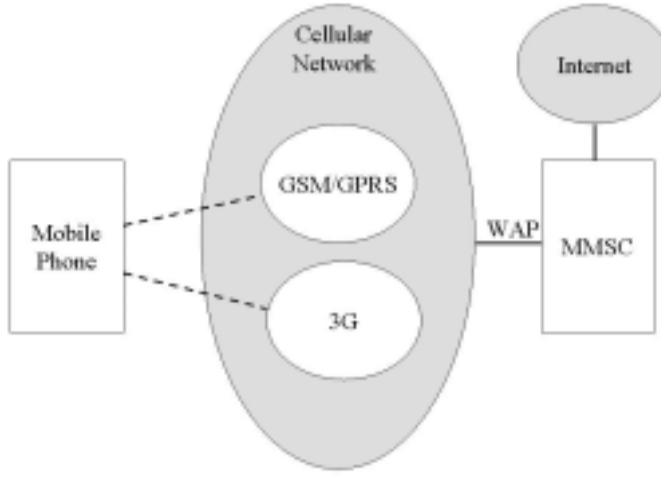


Figure 12. Multimedia Messaging network architecture [21].

MMS client terminals also have to support certain protocols and include codecs for data uncompression. The required protocols for receiving a MMS content packets are IP, TCP and HTTP, and the WAP protocol is needed for receiving notification [23]. In this case study, MMS message was assumed to be received and stored in the local disk of the client terminal during streaming. So, uncompressing and rendering the message to the output devices was not analysed.

4.2 Target architecture

This section describes the target architecture for the analysis, which is based the 3GPP standards [20, 24, 23, 25] described earlier and it is a real industrial planned architecture. The description given in this section is based on internal confidential documents of the customer organisation, therefore they are not presented in the reference list. To preserve confidentiality, the architecture is described in a way that only the essential information for the analysis is presented and some of irrelevant details have been modified or left out.

In this case, the tasks related to streaming and MMS are processed with the three processors presented in Figure 13. The communication and messaging between processors is carried out through buses. Three high level software architectural components take part in receiving streaming and MMS packets (see Figure 14). Symbian software component is running on processor 3, the DSP software component is running on processor 2 and bearer protocols on processor 1. This work concentrated only on analysing the performance of the Symbian software component and other software components were handled as black-boxes. Codecs for uncompressing the data are at the DSP software component and the needed bearer protocols are in the bearer protocols software component.

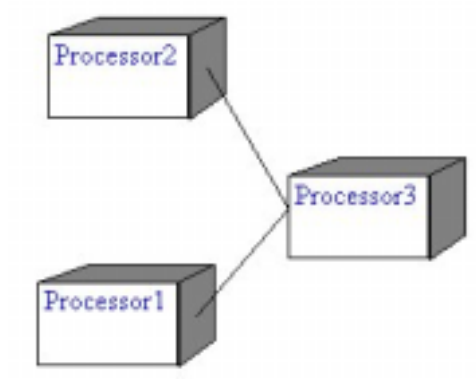


Figure 13. Hardware deployment view.

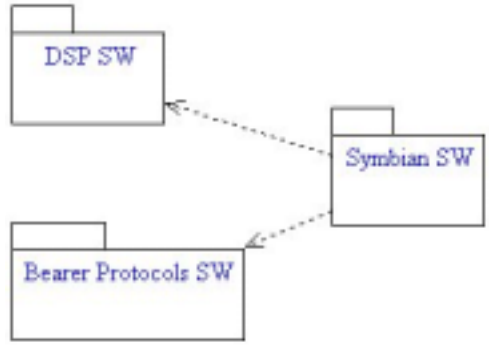


Figure 14. Symbian software context.

As can be concluded from the name, the Symbian software component is on the Symbian operating system platform. The tasks or processes are scheduled based on a preemptive time-sliced round robin scheduling policy [26] and the size of the time-slice is 62.5 ms (16 Hz). The idea of the mentioned policy [27] is explained next. The tasks or processes that are ready to be processed by the processor are handled in a circular queue. The scheduler goes round the queue according to the FIFO (First In First Out) principle and it allocates the processor time to processes. The process gets at maximum the amount of a time-slice and if it is not finished during that time it is preempted and put to the tail of the rounded queue. If the time required by a process is less than the time-slice then it releases the processor voluntarily.

The software architecture related to receiving stream packets and MMS content packets is presented in an execution graph format in Figure 15. The rectangles illustrate processing steps and the arcs denote execution order [2, p. 73]. The software components illustrated with a solid line rectangles belong to the problem area that was analysed in this case.

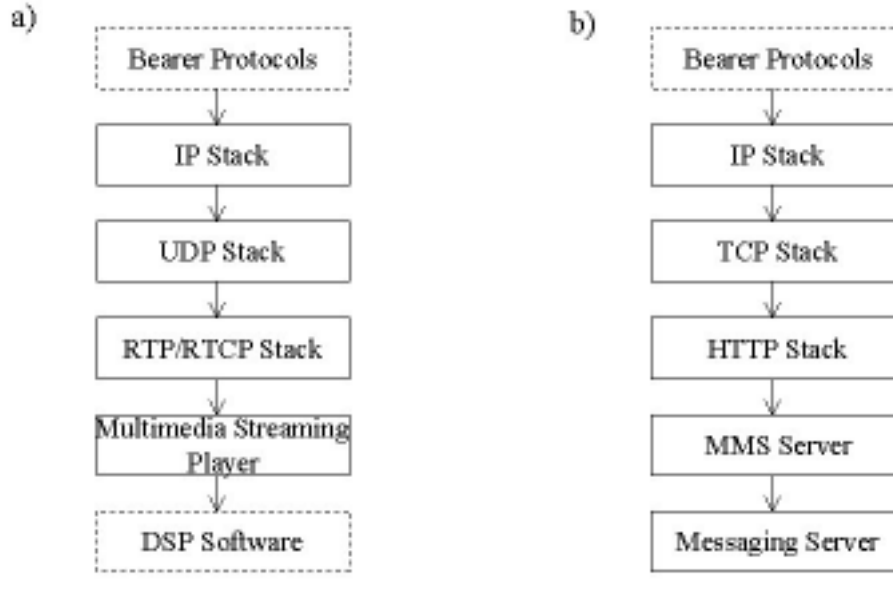


Figure 15. Execution graphs related to a) streaming and b) MMS message.

A thread is a unit of concurrency that can be executed in parallel with other threads inside a process, which represents a flow of control that executes in parallel with other processes [2, p. 135]. The threads that belong to the same process share the same address space [2, p. 135]. In this case, IP Stack, UDP stack and TCP stack are processed in process 1 with thread 1. RTP/RTCP stack and multimedia streaming player are processed in process 2 with thread 2. HTTP stack, MMS server and Messaging server are processed in process 3 with thread 3. All of the priorities of the mentioned threads are the same (Symbian default priorities). Therefore, they are all handled equally in the queuing condition.

4.3 Arrival rates

The arrival rates and the sizes of packets vary, but it was not possible to perform the analysis with all different arrival rates and size values. However, it was noticed that it could be beneficial to analyse worst-case type situation. Based on discussions with the stakeholders from the customer organisation it could be concluded that an interesting situation from the performance point of view would be when the stream data is received with high packet arrival rates and in relatively small packets. The mentioned values are in this case 40 packets/s 200 bytes/packet for video stream and 10 packets/s 150 bytes/packet for audio stream. The delay caused by the MMS message on processing of stream packets is longest when the size of the message is maximum. According to reference [28] the size of the MMS message in the first phase will be between 30 Kbytes and 100 Kbytes. Based on this the size of the MMS message is assumed in this case to be 100 Kbytes. The message is assumed to be received in 100 different packets with an arrival rate of 1 packet/s. This case study was performed based on the mentioned assumed constant arrival rates and packet sizes of streaming and MMS content packets.

5. Method selection and application process

The aim of this work was to analyse the problem defined in the previous chapter with some software architecture based performance analysis method. Many kind of requirements and expectations could be identified for the methods. The properties of the available methods discussed in chapter 3 were compared based on the identified requirements and the most suitable ones were chosen to be applied in this work. The goal was not to create a new method, but to apply the available methods as far as they were applicable. The method selection is discussed in the next section and the compound application process of the selected methods is presented in section 5.2. Section 5.3 describes architectural analysis techniques with details.

5.1 Method selection

The following requirements and expectations were identified for the methods:

1. The method provides support finding out if performance goals are met and it should reveal the bottlenecks in the architecture.

2. It should support outlining the case, so it should not require modelling the whole software architecture.

3. The analysis should be able to be performed without a tool, but it would be useful if there were a tool supporting the method. This is because if the method is decided to be taken in use more widely in the future tool support helps and makes the analysis phase faster.

4. The modelling approach proposed by the method should not be too complicated.

5. There should be enough information available about the application process and it could be useful if there were good case studies available.

6. It could also be useful if the method supported the whole assessment process.

7. The method should also be easy to use and integrate in the software development process.

On the basis of the presented requirements the PASA method seemed to be the most suitable method for solving the presented problem area and therefore it was chosen to be applied in this case study. The PASA method had the following advantages: it is said to be relatively quick and easy to apply, it provides guidance for gathering the required information, it does not require modelling the whole system, but it concentrates on the spots that are the most critical for performance. There is a book available written by C. U. Smith et al. (2002) [2] that provides a detailed description of the principles and techniques related to the PASA method and the book also includes some case studies applying the method in practice. There is also a tool supporting the PASA method. In addition, the mentioned book also presented some formulas for analysing the performance model without tool support.

The LQN modelling approach was selected to be applied for modelling the performance of the software system, even if the PASA method proposed the QNM approach to be used for modelling. The reason is that LQN was seen more suitable for modelling concurrent scenarios and a layered system, because it is especially developed for that purpose. LQN is based on the same queuing theories as QNM therefore it was noticed that the same model solving formulas could be used for analysing LQN as QNM. There is also a LQN tutorial [3] available providing instructions for layered queuing network modelling.

It was mentioned in the book written by C. U. Smith et al. [2 pp. 383-384] that the parameters produced as an output by the PASA method could be used as an input in RMA scheduling. Therefore, RMA was also chosen to be applied to determining the adequate priorities for tasks to meet performance objectives.

The only method that was seen not to be suitable at all for analysing the problem area described in the previous sections was SPA, because it required describing the software architecture of the entire system with a formal description language. The disadvantages of the CPN method considering the case study was that as far as we knew the method has been applied only once on analysing software architecture of a system [6], the modelling approach seamed to be quite complex and the CPN model is assumed to be analysed with CPN/Design tool. The QNM & LTS based method

is a quite a new approach and therefore it was thought that there may not be enough information of applying the method in practice.

5.2 Methods' application process

The steps of the PASA method have already been described briefly in section 3.1 in Table 1, an example of a LQN model has been presented in Figure 4 and RMA has been described in section 3.5. The aim in this section is to complement the description of the PASA method, LQN modelling approach and RMA principles so that it is easy to follow the application process in the next chapter. The first and last steps of the PASA method may not need additional description, therefore the first complemented step of the PASA method is the architecture overview.

The second step of the PASA process is to overview the current or planned architecture. The goal is to obtain a high-level understanding of the architecture before going into its details. If the architecture is documented, then it can be reviewed, and more details can be obtained from discussions with architecture developers. If the architecture is not documented this step may involve a significant discovery phase, concluding the architecture by interviewing developers, examining code and other artefacts.

The third step is to identify critical use cases. A use case is a group of sequences of actions performed by a system and the actions produce a result that can be observed by an actor (e.g. user or an another system) [2, p. 50]. All use cases related to the target system are not important from the performance point of view. Critical use cases are the ones that are important to the operation of the system, or to responsiveness as seen by a user. They can also be those including considerable performance risks, for example, if by not meeting performance objectives the system will fail or be less than successful.

The fourth step is to select key performance scenarios. The critical use cases identified in the previous step typically consist of a group of scenarios, which are not all important from the performance point of view. The key performance scenarios are those that are executed frequently and those that are critical to the user's perception of performance.

The fifth step is to identify performance objectives. At least one performance objective for each selected key performance scenario needs to be identified. Performance objectives have to be quantitative and measurable and they can be described in several ways including response time, throughput and constraints on resource usage. The objectives can be either end-to-end requirements or they can be broken into sub-objectives. In addition to the identified objectives it is important to define the conditions under which the objectives should be achieved.

The sixth step is to clarify and discuss the architecture. This step is needful because architecture descriptions do not usually provide all the required information for assessment. The aim is to learn as much as possible about key portions of the system, interaction between software components and the problem area.

The seventh step is the actual architectural analysis. The PASA method proposes the following techniques to be applied in this step: identification of underlying architectural style(s), identification of performance antipatterns, and analysing the performance model. The mentioned techniques are presented in more detail in section 5.3.

The eighth step is identification of alternatives. This step needs to be taken if performance problems are found during the analysis phase and it is concluded that changes are required in the system to meet performance goals. According to the PASA method the alternatives may be found by finding deviations from the underlying architectural style, identifying alternative interactions between components or refactoring to removing an identified antipattern.

In addition to the techniques proposed by the PASA method RMA scheduling principles can be used for determining if the system is schedulable and if it is then priorities can be assigned in a way that performance objectives are met even in the worst case situation. In this case study it is assumed that the tasks are periodic, are not synchronised with another, do not suspend themselves during execution, and are capable of being preempted by higher priority tasks. Therefore, the formula (1) can be used for analysing the schedulability of the system. The RMA uses the parameters calculated in the PASA application process as an input. If the calculations point out that the system is schedulable, then the priorities can be assigned to the tasks as described in 3.5.

5.3 Architectural analysis

The architectural analysis step produces information about the performance of the architecture as an output. The three techniques proposed by the PASA method are described in the next sections.

5.3.1 Architectural styles

The aim is to identify the underlying architectural style(s) and to find out if it is appropriate also from the performance point of view. If the style is appropriate, but there are deviations from the style in some details, those points may be sources for performance problems and therefore they should be explored more carefully. The exploration can be performed by using performance antipatterns, which are described in the next section, but first some typical architectural styles are presented and described [11]. Many different architectural styles are presented in the books and publications related to this domain, but the aim is not to present all of them in this section, but to give a short overview of some common architectural styles [11, p. 20] that are relevant considering the case analysed in this research.

One very typical style is pipes and filters (see Figure 16) [11, p. 21]. The components of the style are called filters and the connectors between the components are called pipes. The style is especially created for systems handling data streams. The filters read an input stream from pipes and produce an output stream to the pipes. An advantage of the style is that it supports concurrent execution, because filters can be implemented as separate tasks, which can be executed in parallel with other filters [11, p. 22].

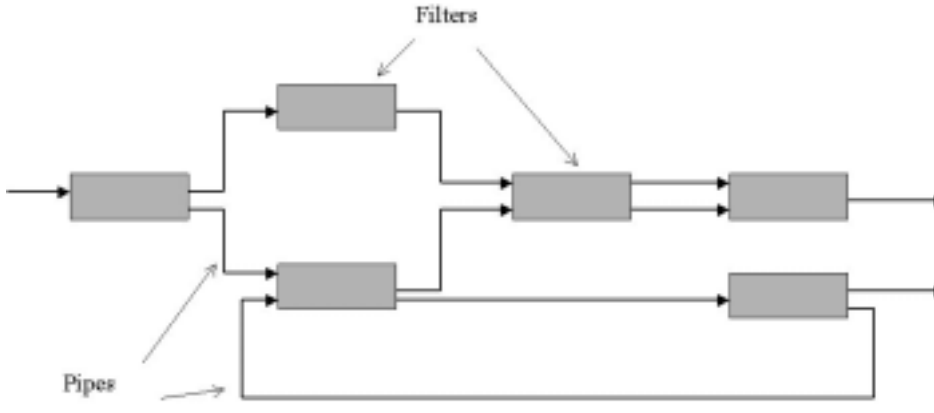


Figure 16. Pipes and Filters [11, p. 21].

The layered style is also a common style (see Figure 17) [11, p. 25]. The architecture of the system consists of layers each of which provides services to the higher layer and acts as a client to the lower layer of the system. The layers are considered to be the components and the interaction between the layers is often defined by protocols. However, the style may not be applicable in systems where high throughput is required [1].

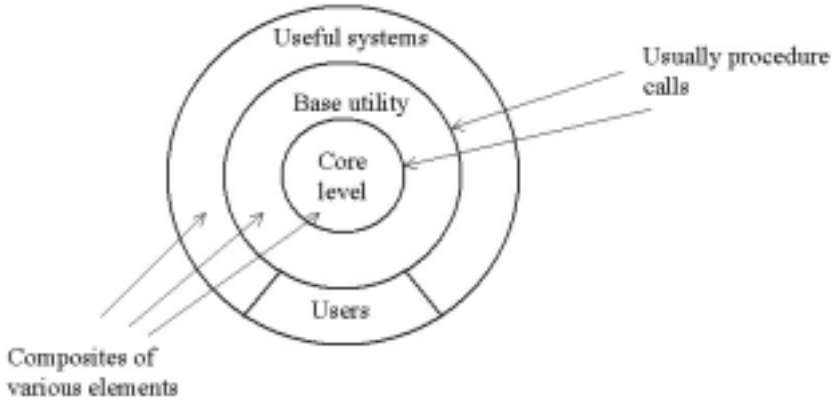


Figure 17. Layered Style [11, p. 25].

The blackboard style [11, p. 26] is in fact a repository style (see Figure 18) consisting of two types of components: a central data structure representing the current state (blackboard) and independent components operating on the central data store. In the blackboard style processes to be executed are selected based on the current state of the central data structure. Knowledge sources in the blackboard model interact with each other only through the blackboard. Knowledge sources make changes to the blackboard.

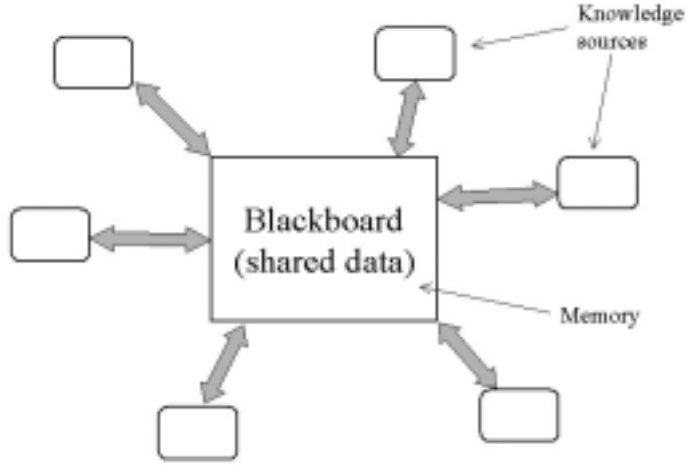


Figure 18. Blackboard Style.

Many other architectural styles have been defined, for example main program and subroutine style, data abstraction and object-oriented organisation style and interpreters [11, p. 20]. However, describing all styles is out of scope of this research, because they are not all essential considering the analysed case.

5.3.2 Performance antipatterns

Sources of possible performance problems can be revealed by analysing the point at which there is a deviation from the underlying architectural style with performance antipatterns. After identifying a performance antipattern, it can be removed, because antipatterns also include solutions to identified performance problems. Some defined performance antipatterns are presented in Table 3 [2, pp. 287-308].

Table 3. Performance antipatterns [2, pp. 287-308].

Antipattern	Problem	Solution
"god" Class	Occurs if one class has too much power. If it either performs all the work of an application or holds all the data of the application. This can cause excessive message traffic that can degrade performance.	Distribute the intelligence evenly between the top-level classes of the application, and keep the related data and behaviour together.
Excessive Dynamic Allocation	Occurs if application unnecessarily destroys and creates large number of objects during the execution. Time consumed at creating and destroying may have negative impact on performance.	Recycle objects rather than creating new ones each time they are needed.
Circuitous Treasure Hunt	Occurs if an object has to look for needed data from many places. Performance problems may happen if each look operation requires large amount of processing.	Change the design to provide alternative access paths.
One-Lane Bridge	Occurs if only one or a few processes may continue to execute concurrently and other processes are delayed because they have to wait their turn.	Alleviating the congestion by constructing multiple lanes, constructing additional bridges, or rerouting traffic.
Traffic Jam	Occurs when one problem causes a traffic jam of jobs that produces wide variability in response time which lasts long after the initial problem has disappeared.	Start by eliminating the original reason for traffic jam. If it is not possible, provide sufficient processing power to handle the worst-case load.

5.3.3 Performance modelling and analysis

Some parts of the architecture may require more quantitative analysis. This kind of parts are modelled and analysed to find out if performance objectives are met. Two types of models are proposed by the PASA method: the software execution model and the system execution model. The software execution model characterises the resource requirements of the software alone, in the absence of other workloads, multiple users or delays due to contention for resources, and analysing the model produces optimistic results [2, p. 72]. Therefore, if performance objectives are not met in an optimistic condition, they are not met in other situations either and then it can be concluded that changes are required to the system to meet performance goals. In that case no further modelling is needed. The software execution model consists of execution graphs and an execution graph is drawn for each selected key performance scenario [2, p. 73]. An example of an execution graph is presented at the left side of the Figure 19 [2, p. 79]. *n* represents the number that *getRequest* and *processRequest* steps are processed and t_n next to the processing steps denote the time that the step requires service from the server (such as processor). Analysing the software execution model means in practice reducing the model as presented in Figure 19 and comparing the resulted value *t* with the maximum allowed value (defined in performance objectives).

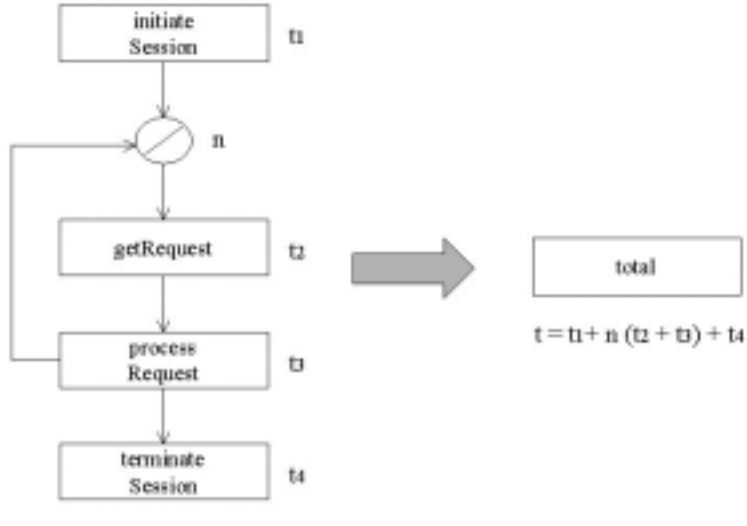


Figure 19. An example execution graph reduction [2, p. 79].

If analysing the software execution model does not reveal performance problems, then the system execution model is drawn, which is modelled as a LQN instead of the QNM proposed by the PASA method. The LQN approach is based on queuing theories which assume the system to consist of servers and requests to the services provided by the servers. The basic elements of a queuing system are illustrated in Figure 20 [2, p. 136]. A job indicates a computation that enters the system, makes requests of one or more computer system resources, and leaves the system upon completion [2, p. 136]. In the queuing system jobs arrive in a queue to the server (such as CPU, disk or I/O-devices). The jobs are selected to be served by the server according to the used scheduling policy. Residence time is the average time that a job spends at the server including the waiting time in the queue and time receiving service from the server [2, p. 137].

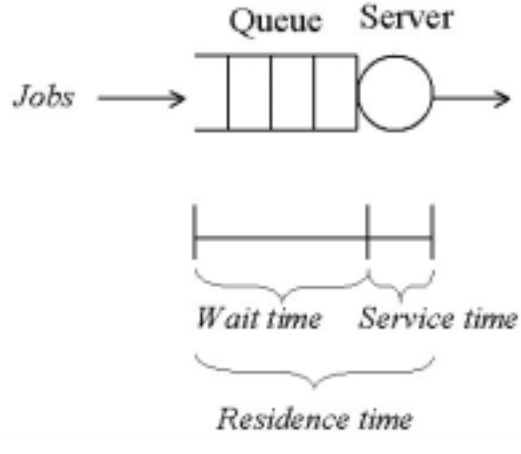


Figure 20. Principle in a queuing system [2, p. 136].

In the layered approach user entities are usually modelled at the highest layer, hardware entities at the lowest level and software entities in the middle layer [3]. The idea in a layered queuing system is that clients send requests to the servers, which can be also clients to other servers. The LQN model consist of the following

elements (see Figure 21): software components (modelled as rectangles), hardware devices (modelled as ellipses) and requests from client to servers (modelled as arrows) [3]. The requests can be either synchronous (a client waits for reply), asynchronous (a client does not wait for reply) or forwarding (server sends a message to an other server) [3]. Software components may have several entries representing different operations it may perform [3].

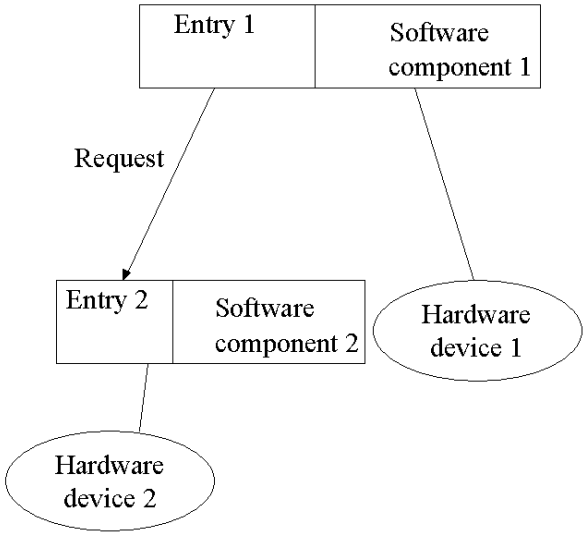


Figure 21. Key elements for a layered queuing system.

The LQN model is typically analysed by using a LQNS tool. However, in this case study no tool support is available and therefore the LQN model is analysed by calculating some performance metrics such as average residence time, utilisation, throughput and queue length [2]. Utilisation is the average percent of time that the server is busy providing service, throughput means the average rate at which jobs complete service and a queue length is the average number of jobs at the server [2, p. 137]. The idea is to compare the calculated performance metrics with performance objectives. Then based on the comparison it can be concluded if changes are required to the system.

Next the formulas for calculating the mentioned performance metrics are given [2 pp. 138-140]. However, here should be noticed that the formulas are said to be valid when the scheduling policy is first-come-first served or priority scheduling [2 pp. 136-137], but because in this case study the scheduling policy is not as assumed

the validity of the formulas needs to be considered. The validation discussion is presented after the formulas.

The formula for utilisation is as follows:

$$U = \frac{B}{T}, \qquad (2)$$

where

B is the time the server is busy providing service and

T is measurement period.

Throughput can be calculated with the following formula:

$$X = \frac{C}{T}, \qquad (3)$$

where

C is number of completed jobs and

T is measurement period.

If the system can be assumed to be fast enough to handle the arrivals, and thus the completion rate or throughput equals the arrival rate, then the following formulas are valid [2 pp. 139-140].

Utilisation is

$$U = XS, \qquad (4)$$

where

X is throughput and

S is mean service time for jobs.

Mean service time is the average time that a job spends receiving service from the server [2, p. 138] and it can be calculated with the following formula:

$$S = \frac{B}{C}, \quad (5)$$

where

B is the time that the server is busy providing service and

C is the number of completed jobs.

Residence time is

$$R = \frac{S}{1-U}, \quad (6)$$

where

S is the time the server is busy providing service and

U is utilisation percent of the server.

Queue length is

$$N = XxR, \quad (7)$$

where

X is throughput and

R is residence time.

According to Silberschatz & Galvin [27, p. 135] the round robin scheduling algorithm is similar to first-come-first-served scheduling, but preemption is added to switch between processes. In the first-come-first-served scheduling policy the first arrived job is served first [27, p. 129]. In this case the priorities of the threads related to streaming and MMS are the same, therefore the threads are not preempted by each other. Each thread achieves a 62.5 ms (time-slice) time at maximum and if the thread is not completed before that it is put to the tail of the queue. If the maximum service time required by a thread is in this case less than 62.5 ms, then the jobs are selected from the queue just as in the first-come-first-served scheduling policy. Because the formulas are said to be valid [2 pp. 136-137]

to the first-come-first-served scheduling, then they should also be valid in this case if the time-slice is longer than required service time by the threads. However, it should be noticed that the formulas may not be valid if priorities of the threads were different from each other and if the time required by the thread is more than the size of a time-slice.

6. Performance analysis

This chapter describes step by step the performance analysis process of concurrent streaming and an MMS message. The aims of each step are not described in this chapter, because they were already briefly described in section 3.1 and a detailed description of the methods' application process was given in sections 5.2 and 5.3. The aim of this chapter is to describe how the performance analysis process was conducted in practice, what kind of assumptions had to be made, what kind of information was needed during the process, how it was obtained, what were the results of each step and what could be concluded based on the analysis. The experiences obtained during the process are collected in section 6.5.

6.1 The first six PASA steps

This section describes how the first six steps of PASA were conducted in this work. The first PASA step was process overview. In this work the application process with the aims of the steps were overviewed in technical meetings with stakeholders both from the customer organisation and VTT. Also the reasons for the analysis were discussed. The analysis was conducted to see how the process is performed in practice, but also to find out what is the performance of Symbian software with concurrent streaming and an MMS message.

The second step of PASA was to overview the architecture. In this work a lot of architectural documentation related to the problem area was luckily available and it was provided by the customer organisation. Therefore, the architecture could be overviewed based on the mentioned documentation. As a result of this step it was found that the architecture related to this problem area is as presented in section 4.2. High level software architecture related to receiving streaming and MMS messages was presented in Figure 14.

The third PASA step was to identify critical use cases. As described in chapter 4 the aim of this research was to analyse concurrent streaming and MMS. Therefore, the streaming and receiving a MMS message were identified to be the critical use cases in this case. Streaming is a time-critical use case and the MMS message was seen to be the most likely to cause performance problems for streaming. Other traffic during streaming could have been for example SMS message, but it is not as

likely to cause performance problems, because it does not load the shared resources as much as receiving and handling a MMS message.

The fourth PASA step was to select the key performance scenarios. As described in section 4.1 streaming consists of several scenarios such as establishing a session to the streaming content provider, tearing down the session and receiving multimedia stream packets from the content provider. The first two scenarios mentioned are executed only once, but stream packets are received frequently. Delays related to handling and rendering stream packets can be noticed by the user. For the reasons mentioned receiving stream packets (audio and video) were selected to be the key performance scenarios in this case study. Receiving a MMS message consists of two separate scenarios: receiving a notification of the message and receiving MMS content packets. The notification is received only once per one MMS message. The impact that the notification has on the performance of streaming was considered to be insignificant. The actual content of the MMS message consists of multiple packets and the competition of the same resources with streaming may cause performance problems. Therefore, receiving MMS content packets was also selected to be one of the key performance scenarios.

The fifth step was to identify performance objectives for the analysis. The streaming requires almost real-time behaviour, but receiving a MMS message does not have real-time requirements. As defined in section 4.3 the arrival rate of video stream packets is 40 packets/s and the corresponding value for audio stream packets is 10 packets/s. In this research it was assumed that the objective is that at maximum only one of each packet type is at the Symbian software (processor 3). Based on the assumption mentioned it was concluded that the maximum allowed response time for video packets is 25 ms and for audio packets 100 ms. If these objectives are not met,then more than one audio or video packet is waiting for service from the processor at the same time. This may cause performance problems, because of the contention for the resources (such as processing time). However, the user may tolerate some video frames or audio to be lost without even noticing it. Therefore, it was assumed in this case study that 95 % of the streaming packets are expected to be handled and rendered on time. The defined performance objectives should also be met when a MMS message is received concurrently. Even if receiving a MMS message does not require real-time response the number of MMS content packets at processor 3 affects the performance of streaming. Therefore, it was defined that there should not be more than one MMS content

packet at the processor. Hence, the maximum response time for an MMS content packet is 1000 ms.

The sixth step was architecture clarification and discussion. Some of the architectural details were not clarified based on the available documentation. This kind of issues were discussed in technical meetings with the customer organisation. For example the communication between different software components and software deployment to the hardware needed clarification. In this case study, each message between the software components related to streaming and MMS was assumed to be sent once per one arrived packet and they were assumed not to wait for a reply message. The architecture that was concluded based on the architecture overview step and this step was presented in section 4.2.

6.2 PASA step 7: Architectural Analysis

The seventh step was to analyse the performance of the architecture. The analysis was conducted mainly based on the information gathered in the previous steps, but in addition some timing information needed to be obtained. The timing values are based on estimations, because no measurements could be done with the target system. However, the estimations were based on measurements with a similar kind of implemented features executed on a different system. The values were scaled to correspond to the values in the target system. The analysis phase was conducted based on the principles and techniques presented in section 5.3. However, most of the time was consumed in performance modelling and the analysis technique.

6.2.1 Identification of the underlying architectural style(s)

Two underlying architectural styles related to the architecture in this case could be identified: pipe and filter style, and layered style. The first of the mentioned styles could be identified by analysing the communication between software components related to the target architecture. The communication between the high abstraction level software components was noticed to be carried out as defined in the pipe and filter style (see Figure 15). Because the mentioned style is especially designed for systems handling data streams and it supports concurrent processing, it could be concluded that the style is appropriate in this case. The second style was identified

by considering the structure of the whole system. The software system was noticed to consist of layers. The highest layer is application software, it is on Symbian platform and the lowest layer is the hardware. Even if the layered style is not considered to be efficient from the performance point of view, it could not be changed in this case study, because it would have required too extensive changes to the whole system. As far as we know all the architectural details related to this case were based on the two styles mentioned, thus no deviations could be identified. The aim would have been to analyse deviation points by using performance antipatterns, but in this case this did not need to be done, because no deviations were found. However, the absence of performance problems was not yet proved, therefore quantitative analysis was needed. It is discussed in the next section.

6.2.2 Performance modelling and analysis

In this case study the following three key performance scenarios were selected: receiving video stream packets, receiving audio stream packets and receiving MMS content packets. Thus, the software execution model consists of three execution graphs (see Figure 22, Figure 23 and Figure 24). The timing values next to the processing steps present the estimated service time required from the processor 3 by the software component. The response time at the Symbian software in the absence of other workloads for one received video stream packet could be calculated by summing the required service times as presented in Figure 22. It is in this case 1.85 ms and the maximum allowed value is 25 ms. Thus, no performance problems were revealed so far.

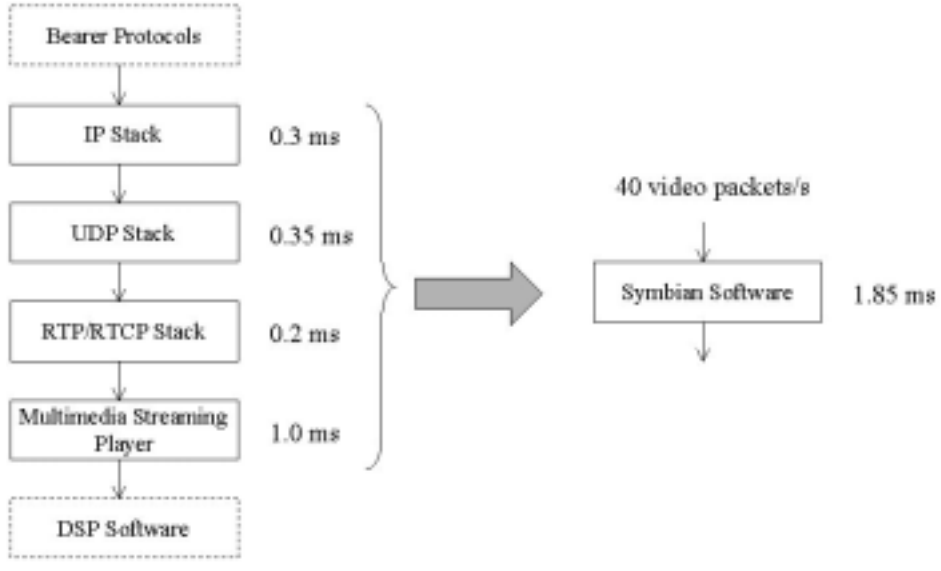


Figure 22. Execution graph for receiving video stream packets.

The response time at the Symbian software for one received audio stream packet is 1.85 ms (see Figure 23). It is less than defined in the performance objectives (100 ms). However, it could not be concluded would the objectives also be met if other workloads were processed concurrently.

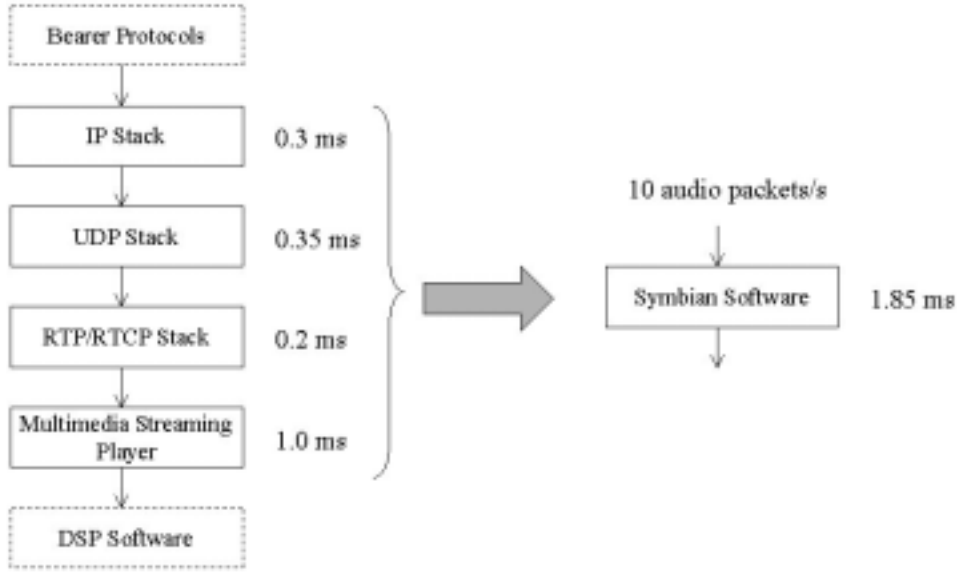


Figure 23. Execution graph for receiving audio stream packets.

The execution graph for receiving the MMS content packet can be seen in Figure 24. Processing one MMS content packet at Symbian software takes 49.25 ms. The time between MMS content packet arrivals is 1000 ms, so the packet is processed before the next one arrives, at least if no other workloads are present.

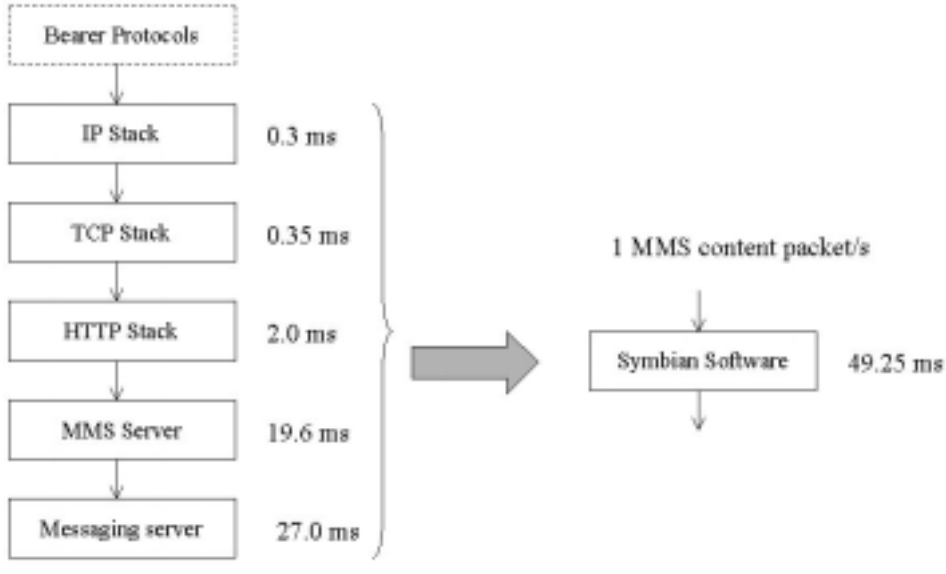


Figure 24. Execution graph for receiving MMS content packets.

The system execution model of the target system needed to be created and analysed, because the absence of performance problems could not be proved by analysing the software execution model. The LQN model related to this case is presented in Figure 25. It is based on the notation proposed in a LQN tutorial [3]. The numbers next to the arcs mean the number of visits at the software component per one received packet. The numbers inside the rectangles in the middle layer are estimated service times required from processor 3 per one request. They are assumed to also include the execution time required at the lower layers of the system and message delays. The numbers inside rectangles at the highest layer are the arrival rates of the packets.

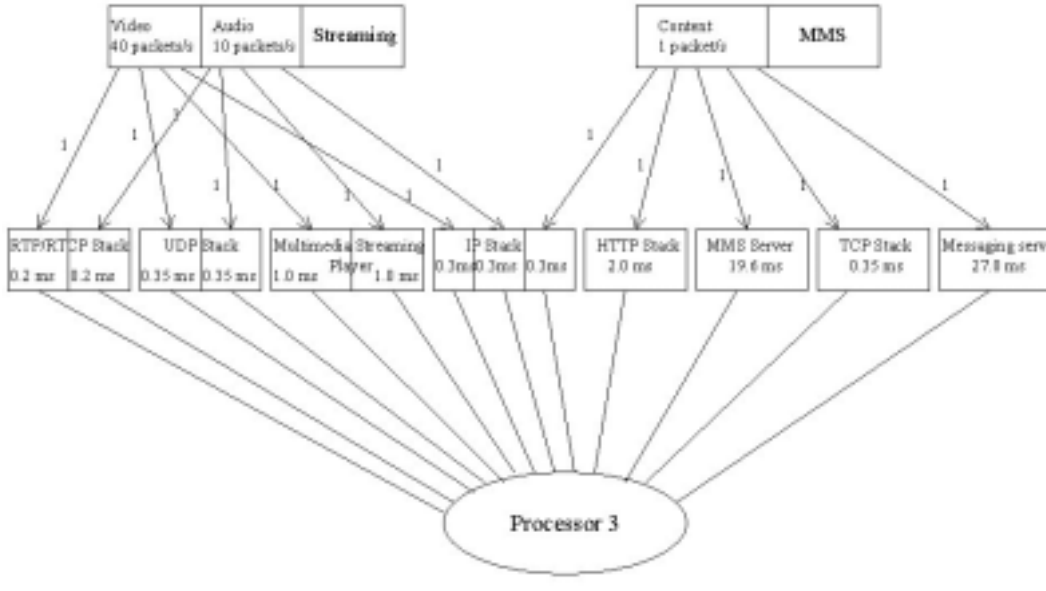


Figure 25. LQN model.

In this case study the formulas presented in section 5.3.3 could be used, because the maximum service time required by the packets from processor 3 (49.25 ms) is less than the amount of a time-slice (62.5 ms). The performance metrics (utilisation, residence time and queue length) were calculated first for each scenario separately and then total values were concluded. In the calculations it was assumed that the system is fast enough to handle the packet arrivals before the next arrival of the same packet type occurs. So, it was assumed that the throughput equals the arrival rate. Formula (4) was used for calculating utilisation, formula (6) for residence time and formula (7) for queue length. The results of calculations are collected in Table 4, Table 5 and Table 6. The job was assumed to be a video stream, audio stream or MMS content packet and it was assumed that packets are not broken into pieces at the data path.

According to calculations the utilisation of processor 3 during a video stream was noticed to be quite low (see Table 4). The maximum allowed utilisation for three tasks could be calculated with formula (1) and it was in this case study 77.9 %. Based on the information mentioned it could be concluded that there is also processing power for other processes in addition to video streaming. Concurrent

processing of streaming and MMS delays the processing of video stream packets 0.31 ms/packet, which is not much. The residence time for one video stream packet at the Symbian software is 2.16 ms and the corresponding maximum allowed value (defined in performance objectives) is 25 ms. Therefore, it could be concluded that on average performance objectives of video stream packets are met also during concurrent processing of streaming and MMS. On average there is less than one video packet at the server. However, the calculated values are averages and it could not be concluded would the performance objectives be met also when the MMS message was received in worst possible time from the streaming point of view.

Table 4. Performance metrics for video stream.

Performance metric	Value
Utilisation (U)	7.4 %
Throughput (X)	40 packets/s
Mean service time (S)	1.85 ms/packet
Residence time (R)	2.16 ms/packet
Queue length (N)	0.09 (packets)

The utilisation percent related to the audio stream was also considered to be quite low (see Table 5). The concurrent processing also delays audio stream packet handling 0.31 ms/packet and on average there is less than one audio packet in the processor queue. On average handling one audio stream packet at processor 3 during concurrent processing takes 2.16 ms, which is less than defined in performance objectives. It could be concluded that on average there seems to be no performance problems related to audio streaming.

Table 5. Performance metrics for audio stream.

Performance metric	Value
Utilisation (U)	1.9 %
Throughput (X)	10 packets/s
Mean service time (S)	1.85 ms/packet
Residence time (R)	2.16 ms/packet
Queue length (N)	0.02 (packets)

The performance metrics related to MMS content packets are presented in Table 6. Based on them it could be concluded that, in addition to MMS, there is processing power also for other processes. The delay caused by concurrent processing on average is not long enough to cause performance problems.

Table 6. Performance metrics for MMS content packets.

Performance metric	Value
Utilisation (U)	4.9 %
Throughput (X)	1 packet/s
Mean service time (S)	49.25 ms/packet
Residence time (R)	57.38 ms/packet
Queue length (N)	0.06 (packets)

Total values for utilisation percent, service time and queue length are presented in Table 7. The total values were calculated by summing video stream, audio stream and MMS content packet values together. The total utilisation percent of processor 3 during concurrent streaming and MMS is 14.2 %, which is a lot less than the maximum allowed utilisation for three tasks (77.9 %). Therefore, it could be concluded that on average there is also processing power for other activities. The total service time required by streaming and MMS from processor 3 is 52.95 ms, which is less than the maximum response time constraint for audio stream or MMS content packets. So, even if the audio stream packet handling or MMS content packet handling lasts the mentioned total time the performance objectives would still be met. However, if the residence time of a video stream packet at processor 3 is 52.95 ms then more than one video packet would arrive during that time. This may be noticed as a performance problem, but not necessarily.

Table 7. Total performance metrics.

Performance metric	Value
Utilisation (U)	14.2 %
Total service time (S)	52.95 ms
Queue length (N)	0.16 (packets)

It was seen to be interesting in this case study to find out what is the maximum queue length and residence time of packets during concurrent streaming and MMS. The queuing theories that the QNM and LQN are based on provide only formulas for calculating average values. Unfortunately, the formulas for calculating maximum values were not found during this research. However, the worst-case situation could be concluded without formulas because the policy that is used for selecting a job from the queue, occurrence pattern of jobs, required service time per job and processes and threads, and their priorities were all known.

The number of selected key performance scenarios is 3. The mentioned scenarios can occur at the same time, so the maximum queue length can be at least 3 packets. The total service time required by the scenarios is 52.95ms (see Table 7). In that

time three video packets, one audio packet and one MMS content packet may arrive, so the maximum queue length is from 3 to 5 packets. If all 5 packets were at processor 3 simultaneously, then the maximum residence time would be 56.65 ms (= 1 x 1.85ms + 3 x 1.85 ms + 1 x 49.25 ms). If the residence time is that high, then performance problems may occur. However, all the three video packets may not be able to reside in the queue simultaneously even in the worst-case condition, because the earlier arrived packets may be already processed before the next ones arrive. Therefore, more detailed examination was needed.

The packet handling is executed in processes, which are processed in parallel. The Symbian scheduler schedules the processing time for the processes in the queue. The processes related to handing video, audio and MMS content packet at the processor 3 with their service time requirements are presented in Figure 26. The processes related to each packet type handling are presented in the order they arrive in the queue to processor 3. The figure mentioned also illustrates that three video stream packets may arrive during one MMS content packet handing.

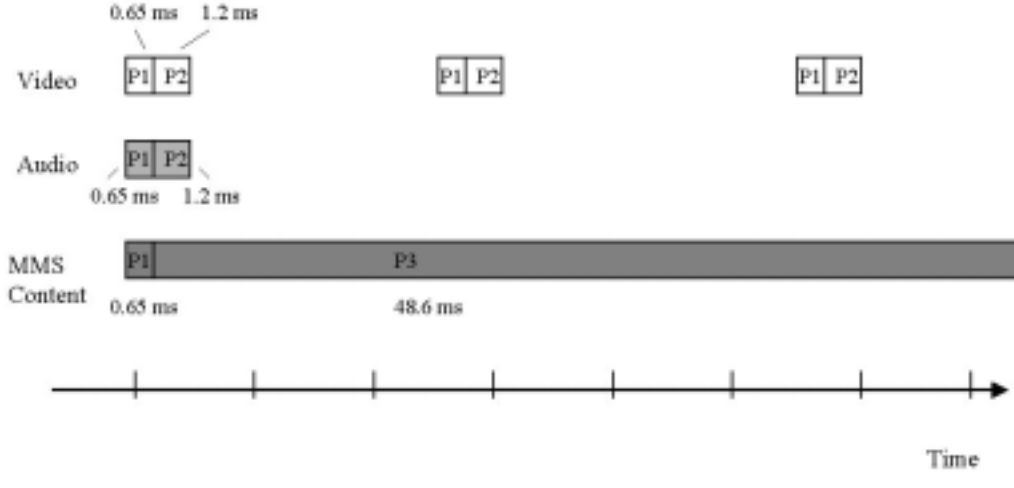


Figure 26. Processes.

Video stream packet handling has the most critical deadline, because of the highest arrival rate. The worst case situation is when MMS and audio packet handling delays the processing of video stream packet the most. All of the processes in this case have the same priorities and therefore the processes in the queue are served in the arrival order. The delay caused by MMS and audio packets to the video processing is the longest when all the three packets arrive in the processor 3 queue almost at the same time, so that MMS arrives a little moment before audio, and audio arrives a moment before the video packet. The worst-case situation is illustrated in Figure 27.

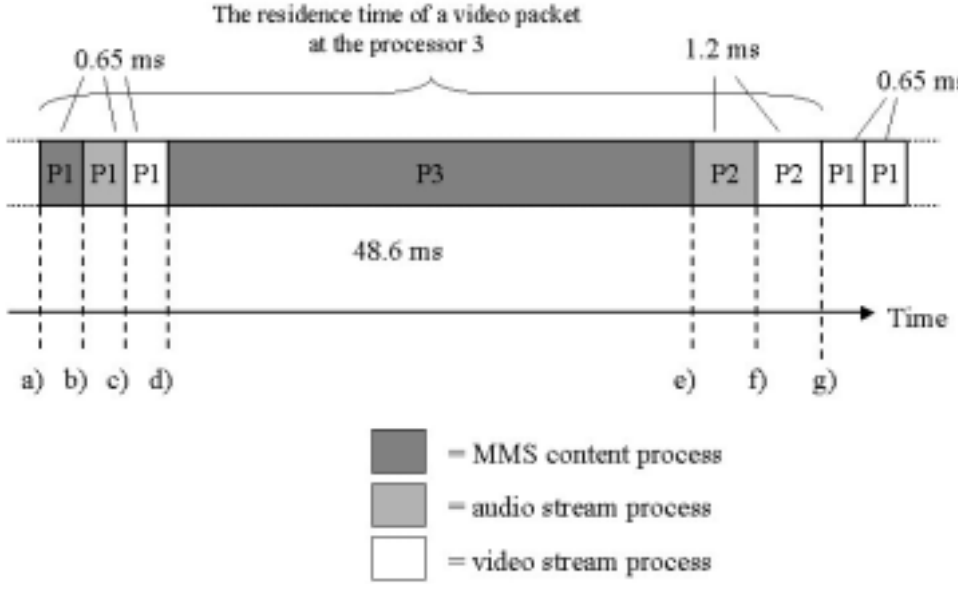


Figure 27. Worst-case situation.

The worst case scenario proceeds as follows, the letters a)-g) refer to Figure 27:

a) MMS content packet, audio packet and video packet arrive in the queue to processor 3. MMS process P1 is processed first, because it has requested service

first. Audio packet processing and video packet processing have to wait until MMS process P1 is processed.

b) MMS process P1 is processed and it requests process P3 to be processed, but it has to wait until audio and video processes P1 are processed, because they were in the queue first. At the moment there are three packets in the queue. Audio packet process P1 is started to be processed next.

c) Audio process P1 is processed and it requests audio process P2. However, the request has to wait in the queue because video process P1 and MMS content process P3 have requested service first. So, video process P1 is started to be executed. At this moment there is three packets at the queue.

d) Video process P1 execution is completed and video process P2 is requested, but the request is put in the queue. MMS process P3 is started to be executed. During the processing two video packets arrive. So, at the end of processing of P3 there are five different packets (three video packets, one MMS packet and one audio packet) in the queue.

e) MMS process P3 is completed and the last audio process is started to be executed. Now, four packets are in the queue (one audio and three video packets).

f) Audio packet is processed completely and the last process of the first video packet is started to be executed. At this moment there are three packets in the queue.

g) The processing of the first arrived video packet is complete.

It could be calculated that the residence time for the video packet at processor 3 is in the worst-case condition 52.95 ms and the maximum queue length was five packets. Because the residence time is more than the maximum allowed value (25 ms), it could be concluded that in a worst-case condition performance problems may occur. However, the kind of situation described earlier may be very rare.

6.3 PASA Step 8: Identification of Alternatives

The eighth step was to identify alternatives for improving performance. The PASA method proposed different techniques to be applied for that purpose. However, the proposed techniques did not provide change proposals in this case, because no deviations from the underlying architectural style were identified, the pipe and filter style was seen appropriate style from the performance point of view and layered style could not be changed in this case study. However, RMA was used for schedulability analysis and assigning priorities. The parameters needed for the RMA schedulability analysis are presented in Table 8. The execution times for packets were obtained as an output from the PASA method.

Table 8. Rate Monotonic Analysis.

Task	Execution time	Period	Deadline
Video packet	1.85 ms	25 ms	25 ms
Audio packet	1.85 ms	100 ms	100 ms
MMS content packet	49.25 ms	1000 ms	1000 ms

The utilisation bound was calculated as follows with formula (1):

$$\frac{1.85ms}{25ms} + \frac{1.85ms}{100ms} + \frac{49.25ms}{1000ms} \leq U(3) = 3(2^{1/3} - 1) = 0.779.$$

The total utilisation of these tasks is 0.142, and the utilisation bound for three tasks is 0.779. Because the total utilisation is less than the bound, it could be concluded that the tasks are schedulable. That is, they will meet their deadlines if threads related to video packets are given the highest priority, audio packets are given the next highest priority, and MMS are given the lowest priority.

It was also noticed in this research that another way to improve performance of video streaming could be to reduce the amount of a time-slice. Then video packet processes would not have to wait until the end of the MMS process P3, because it would be preempted and the processor would be allocated to video or audio processes. However, reducing the time-slice has also a negative effect on performance, because the smaller time-slice increases the time required for context switches between processes. Therefore, reducing the time-slice is not recommended in this research.

6.4 PASA Step 9: Presentation of Results

The ninth step was to present the results to the stakeholders. In this analysis the following results were obtained. They were presented in document format to the stakeholders from the customer organisation and the document was reviewed in a technical meeting and in a steering group meeting of the project. It could be concluded based on the analysis that on average there will not be performance problems if the required service times from processor 3, arrival rates and packet sizes were as estimated and assumed in this analysis. However, here it should be noticed that the effect that other hardware resources (such as memory or buses) have on performance was not analysed even if they may also become bottleneck resources.

The processor 3 utilisation is only 14.2 % during concurrent streaming and MMS reception, which is a lot less than 77.9 %, which is the maximum allowed value for three tasks. On average there is less than one packet in the queue to processor 3. The residence times for different packet types are less than defined in performance objectives. However, in a worst-case condition performance problems may occur. It was noticed that in a worst possible condition MMS content packets and audio stream packets processing delays the processing of the video packet so that it is not completed before the next video packet arrives. However, the worst-case condition is rare. If two packets of 50 received streaming packets per one second are not rendered on time and other 48 packets are rendered on time, then still 96 % of all packets related to streaming are rendered on time. This is still more than defined in the performance objectives (95 %), so the rendering is tolerable. So, changes are not necessarily needed to the system, but by changing priorities according to RMA principles the deadlines would be met even in the worst-case situation. It is

proposed here that video stream packet handling processes should be assigned the highest priority, audio stream processes the next highest priority and MMS content packet processes the lowest priority.

6.5 Experiences

The framework provided by the PASA method really helped during the performance analysis process. PASA helped in concentrating on the essential matters, outlining the case and identifying the objectives for the analysis. Without any kind of support gathering the required information for the analysis would have been a lot harder and it would have been difficult to decide where to start and what kind of information is needed. Other methods described in chapter 3 do not provide support for collecting information and outlining problem area.

Despite the framework provided by the PASA method gathering the required data for the actual analysis took about 2.5 months. Because the work was conducted outside the customer organisation, it took some time to find out who to ask and what kind of information is available. The software of a mobile phone system and its structure was not previously familiar to the author, nor was applying performance analysis methods in practice. It was also problematic to decide what is an adequate abstraction level of knowledge. The instructions related to the PASA method application were generic and they did not give specific instructions related to this case. The chosen abstraction level was the level of information that was obtained from architectural and functional documents. A lot of material concerning streaming and MMS was obtained, therefore it took time to glance through all of those documents and find the right information. Also because the documents were written by specialists some of the details had been left out, because they were taken as granted by the authors of the documents.

The PASA method proposed identifying underlying architectural styles, identifying performance antipatterns and analysing performance model to be applied in the analysis phase. The first two techniques did not provide much benefit in this case, because the target architecture was quite small and therefore no deviations from the underlying style were identified. So, most of the time consumed in the analysis phase was used for performance modelling and analysing the model.

Drawing the LQN model at the level presented was not difficult, because the needed information was already gathered in the previous steps. Several different notations for drawing a LQN model have been published. Even the inventors of the LQN approach, Woodside and Petriu, use several different notations. Therefore, it was a bit difficult to decide what notation should be used in this case.

Analysing performance of a LQN model was difficult, because no tool support was available in this case and it was assumed both in instructions related to the PASA method [2, 1] and LQN [3] that a tool is used for analysing and solving the performance model. So, in most of the available case studies a tool was used for analysing the model. However, simple formulas for analysing a performance model without a tool were given by Smith and Williams in their book [2], but they were applicable only for systems with a priority or first-come-first-served scheduling policy. In this case, the scheduling policy was pre-emptive time-sliced round-robin scheduling. However, the mentioned formulas were applied, because the system is assumed to act like a first-come-first-served scheduled system, because the priorities of the processes related to the problem area were the same and time-slice is longer than the service time required by the processes.

Performing the analyses was also difficult, because there were no example analysis of applying PASA or LQN approaches to a corresponding system. Also the available example case studies found during this research did not present all the intermediate phases of the method application process. If the case studies were based on some industrial real world system, then most of the interesting details would have been left out. Because of lacking good examples, what was calculated and how in the analysis phase, and what could be concluded from the calculations depended on the analyst.

The PASA method did not provide any useful change proposals for this case. However, it produced the parameters as output that were used as an input for the RMA method. Using RMA was easy, because the needed inputs were already available and the problem area was specifically defined and outlined in the PASA application process. The RMA principles were noticed to be very useful and applicable.

All of the problems described had to be clarified to be able to perform this analysis as it is presented in this report. It was noticed that the analysis produces valuable

information about the performance of the system. The most important thing is that the analysis can be conducted before the software system is implemented. The accuracy of the results depend on the accuracy of the timing parameters and also the reality of the performance model. So, the target system needs to be understood well to be able to draw a model that corresponds to the real system. It is still unknown how accurately the timing parameters can be estimated in the early software system development phase.

7. Discussion

This research had two main purposes. The first one was to find out and illustrate how applicable performance analysis methods are in practice in the mobile phone problem area, and what kind of results the methods produce. The second goal was to find out what is the performance of concurrent streaming and MMS. Both of the goals were achieved in this research by analysing the performance of the Symbian software part of a mobile phone during concurrent streaming and MMS with the PASA method, by LQN modelling approach and RMA scheduling principles, but because the time for this study was limited some outlining had to be made to be able to perform the analysis on time and also some assumptions were made, which may affect the generalisation of this work.

The processor was the only hardware resource that was concerned in this case even if in reality some other resources (such as memory or buses) could also be bottleneck resources. For example even if the processing power was sufficient, but memory overflows, then performance problems may occur. Some video frames or audio may be lost and it can be seen as a gap between video frames or heard as non-continuous audio. Also, even if both the memory capacity and processing power are sufficient, performance problems may still occur if buses are not able to pass on the data at the required rate. Therefore, to obtain more certain prediction of performance also the usage of other hardware resources should be studied.

In this research only the performance of the Symbian software was analysed even if the performance is in real terms the result of the functioning of the whole system. Even if the response time at the Symbian software is less than defined in performance objectives, performance problems will be seen if other components do not meet their performance goals.

The performance of the network also affects the performance noticed by the user, but in this case the jitter of packet arrivals caused by the network was not considered. The packet arrival rates and packet sizes were assumed to be constants. If the packet arrival rates or packet sizes were higher than assumed in this case, then the probability of performance problems would increase. However, the arrival rate and packet size values used in this research were bad from the performance point of view, so normally the condition will not be as bad as assumed in this case. Therefore, the performance objectives will likely to be met also with other arrival

rate and packet size values. It was also assumed that packets are not broken into pieces at the Symbian software and therefore queuing parameters were calculated for packets. So, if the packets broke into pieces in reality then some other data unit that endures through the whole data path needs to be found and queuing parameter calculations should be done for that data unit.

In this case study, even if the scheduling policy is preemptive time-sliced round-robin scheduling the system was assumed to act like a first-come-first-served scheduled system, because the priorities of the processes were the same and service time requirements of the processes were less than the amount of a time-slice. Therefore, the formulas for a first-come-first-served scheduled system were applicable. However, it should be noticed that the formulas for utilisation, residence time and queue length applied in this research may not be applicable if the system does not act as in a priority based or first-come-first-served scheduled system. If the formulas were not applicable, then some other formulas should be used instead.

To be accepted and to be taken in wider use in the industry, the analysis should produce reliable results and the analysis should not be too time-consuming. It was noticed that the accuracy of the results depends on the accuracy of the performance model and timing estimates. According to specialists from the customer organisation the LQN model presented in this research may be too simple for illustrating the real system. The LQN model included only the application layer and hardware layer, but not the layer between because of lacking information. However, the LQN approach allows modelling many different layers. So, the model presented here could be expanded if more knowledge was available. Therefore, it might be best if the performance models would be drawn by specialists in the future. Then the models would correspond better with reality and all essential information would be modelled. This would also minimise the time spent at gathering the required data for the analysis, because the specialists are familiar with the software architecture and relevant details.

This research may help future analysts in the mobile phone problem area in their work, because the whole application process is described in detail, the assumptions made during the process are presented, and the problems appearing and solutions to the problems are also presented. As far as we know no such completely presented case study from this problem area has been published before.

If all hardware resources and the components of the whole system were included in the same model, then the model would become much more complex than in this case. Analysing such a complex model may be difficult and therefore it could be reasonable to use a tool to performing the analysis in the future.

Before taking the LQN modelling approach into wider use it should be found out what is the LQN modelling notation that is recommended by the developers and what is the modelling approach required by the available tool. It should also be found out if the LQN tool is able to solve the performance of a system scheduled with different scheduling policies or does it restrict the policy. Because the available tool is based on the queuing theories that provide formulas only for average values, it may not solve maximum queuing values. The only possibility to obtain maximum values may be by simulating the system.

The biggest problem in early performance analysis may be obtaining reliable timing values. In this case study, the analysis was performed based on estimated execution time values. However, it is difficult to say how accurate the values are in reality, because no implementation is available for measuring and comparing the values in the real target system. It was noticed that if a similar kind of system with the same features is available the timing values can be measured and scaled to corresponding values in the target system. Then the values are possibly more accurate. Based on this research it was noticed that if it is possible to obtain reliable timing values then early performance analysis produces valuable knowledge about the performance of a designed system.

If it is decided to take the early performance analysis into wider use then it could be beneficial to include performance models in architectural level documents. A performance model should be drawn of each critical use case. Then it would be easy to analyse the performance of different combinations of scenarios using a tool and it could be concluded if the performance is adequate during their concurrent processing. Tool support also facilitates performing the analysis with several different arrival rates and timing values.

Performance analysis should be re-conducted when more accurate timing information is available or if changes are made to the architecture. Because a performance model (such as LQN-model) illustrates the software at the high abstraction level, the models may also be reusable in other systems with different

hardware platforms including the same features. Then the timing parameters (such as execution time) could be estimated for that platform.

8. Conclusion

In this research, the available state-of-the-art software architecture based performance analysis methods were gathered and six competing methods were found. They were compared and the PASA method was selected to be applied to analysing the performance of the Symbian software part of a mobile phone during concurrent streaming and MMS message reception. The PASA method was found to be a good framework for collecting the required data for the analysis, outlining the problem area and selecting the performance objectives and it was also quite easy to apply.

The basic idea related to the methods is to derive a performance model based on software architectural descriptions. The PASA method proposed the QNM modelling approach to be used for performance modelling, but in this research a LQN approach was applied instead, because it is more applicable for modelling concurrent scenarios and layered system. The LQN modelling approach was quite simple and it could be easily seen from the model how the total throughput time is built up.

Typically the performance model was proposed to be solved by using a tool, but in this case study no tool support is available. Therefore, the analysis of the performance model was conducted by calculating some performance metrics such as utilisation, residence time and queue length. The calculated values were compared to the performance objectives and it could be concluded that if timing values are as assumed the performance objectives will be met on average, but performance problems may occur in the worst-case situation.

In addition to analysing performance modelling the PASA method also proposed architectural styles and performance antipatterns to be used as an aid in the analysis phase. However, in this case study no deviations from the underlying architectural styles were found and therefore performance antipatterns were not used.

Changes are not necessarily needed to the system, but by using the RMA scheduling principles the priorities of the tasks can be changed in such a way that the performance objectives will be met even in the worst-case condition. The suggested priorities are as follows: the video stream packet handling processes gets the highest priority, audio the next highest priority and MMS content the lowest

priority. Hence, RMA was noticed to be very useful for solving this kind of limited problem.

There were some limitations to this study. First, the time was for the analysis case was limited to 4.5 months. Therefore, the problem area had to be outlined to be able to perform the work in the given time. A second limitation of the present study is that the author had no earlier experience of the target system or applying performance analysis methods in practice. Third, the work was conducted outside the customer organisation. Therefore, the author had no access to the internal databases and it was not possible to have face to face conversation with streaming and MMS specialists.

The conclusion of this work is that performance analysis can be applied to this problem area. If it is possible to create a performance model corresponding to reality and obtain accurate timing values, then it is possible to obtain reliable results. Further research is needed for constructing a more extensive performance model including all the high-level software components in the system that take part in streaming and MMS packet handling, and all related hardware resources. This model could be analysed by using a tool and then it would also be found out how applicable the tool is in practice.

References

1. Williams L.G. & Smith C.U. (2002). PASA: A method for the Performance Assessment of Software Architectures. In: Proceedings of the Third International Workshop on Software and Performance (WOSP '2002), July 24-26, Rome, Italy. Pp. 179-189.

2. Smith C.U. & Williams L.G. (2002). Performance Solutions: A Practical Guide to Creating Responsive, Scalable Software. Addison-Wesley, Boston, 510 p.

3. Woodside M. (10.3.2003). Tutorial Introduction to Layered Modeling of Software Performance. Edition 3.0. Carleton University. URL: http://www.sce.carleton.ca/rads/lqn/lqn-documentation/tutorialf.pdf.

4. Bass L., Clements P. & Kazman R. (1998). Software Architecture in Practice. Addison-Wesley, Massachusetts. 452 p.

5. Obenza R. (1994). Guaranteeing Real-Time Performance Using RMA. Embedded Systems Programming, May, pp. 26-40.

6. Xu J. & Kuusela J. (1998). Analyzing the execution architecture of mobile phone software with colored Petri nets. International Journal on Software Tools for Technology Transfer, Vol. 2, No. 2, pp. 133-143.

7. Petriu D., Shousha C. & Jalnapurkar A. (2000). Architecture-Based Performance Analysis Applied to a Telecommunication System. IEEE Transactions on Software Engineering, Vol. 26, No.11, pp. 1049-1065.

8. Aquilani F., Balsamo S. & Inverardi P. (2001). Performance analysis at the software architectural design level. Performance Evaluation, Vol. 45, No. 2-3, pp. 147-178.

9. Herzog U. & Rolia J. (2001). Performance validation tools for software/hardware systems. Performance Evaluation, Vol. 45, No. 2-3, pp. 125-146.

10. Smith C. U. (1990). Performance Engineering of Software Systems. Addison-Wesley, Massachusetts. 570 p.

11. Shaw M. & Garlan D. (1996). Software architecture: Perspectives on an Emerging Discipline. Prentice Hall, New Jersey. 242 p.

12. Brown W., Malveau R., McCormick III H. & Mowbray T. (1998). Anti Patterns: Refactoring Software, Architectures, and Projects in Crisis. John Wiley & Sons, New York. 309 p.

13. Smith C.U. & Williams L.G. (1993). Software Performance Engineering: A Case Study Including Performance Comparison with Design Alternatives. IEEE Transactions on software engineering, Vol. 19, No. 7, pp. 720-741.

14. Balsamo S. & Simeoni M. (10.3.2003). Deriving Performance Models from Software Architecture Specifications. Research Report, CS-2001-04, Dipartimento di Informatica Università Ca' Foscari di Venezia,. URL: http://www.dsi.unive.it/~balsamo/saladin/bal-sim.2.01.pdf.

15. SPEED Performance Modelling Tool (11.3.2003). Performance Engineering Services, A Division of L&S Computer Technology, Austin. URL: http://www.perfeng.com.

16. Klein M.H., Ralya T., Pollak B., Obenza R. & Harbour M.G. (1993). A Practitioner's Handbook for Real-Time Analysis: Guide to Rate Monotonic Analysis for Real-Time Systems. Kluwer Academic Publishers, Massachusetts. Pp. 1-1...9-67.

17. Products (19.2.2003). Tri-Pacific Software Incorporation, Alameda. URL: http://www.tripac.com/html/prod-toc.html.

18. TimeWiz (19.3.2003). TimeSys Incorporation, Pittsburgh. URL: http://www.timesys.com/index.cfm?hdr=tools_header.cfm&bdy=tools_bdy_model.cfm.

19. Bolch G., Greiner S., de Meer H. & Triveldi K. (1998). Queueing Net works and Markov Chains. Modelling and Performance Evaluation with Computer Science Applications. John Wiley & Sons, New York. 726 p.

20. 3GPP TS 26.233 V4.2.0 (2002). Transparent end-to-end packet switched streaming service (PSS): General description. 3rd Generation Partnership Project, Technical Specification Group Services and System Aspects, Release 4, Sophia Antipolis, France, 11 p.

21. Nokia Multimedia Messaging: Show what you mean. (22.10.2002). Nokia White Paper. URL: http://nds1.nokia.com/press/pdfs/mmsA4.pdf .

22. Multimedia Streaming (22.10.2002). Nokia White Paper. URL: http://nds1.nokia.com/press/pdfs/streaming_wp.pdf.

23. 3GPP TS 23.140 V4.8.0 (2002). Multimedia Messaging Service (MMS): Functional description, Stage 2. 3rd Generation Partnership Project, Technical Specification Group Terminals, Release 4, Sophia Antipolis, France. 78 p.

24. 3GPP TS 26.234 V4.4.0 (2002). Transparent end-to-end packet switched streaming service (PSS): Protocols and codecs. 3rd Generation Partnership Project, Technical Specification Group Services and System Aspects, Release 4, Sophia Antipolis, France. 39 p.

25. 3GPP TS 23.140 V4.2.0 (2002). Multimedia Messaging Service, Stage 1. 3rd Generation Partnership Project, Technical Specification Group Services and System Aspects, Service Aspects, Release 4, Sophia Antipolis, France. 12 p.

26. Tasker M., Allin J., Dixon J., Forrest J. Health M., Richardson T. & Shackman M. (2002). Symbian Programming: Mobile Solutions on the EPOC Platform. Wrox Presss, Birmingham. 1031 p.

27. Silberschatz A. & Galvin P. (1999). Operating System Concepts, Fifth Edition. John Wiley & Sons, New York. 888 p.

28. MMS: Multimedia Messaging Service FAQ (24.3.2003). Canvas Dreams. Beaverton.UnitedStates.
URL: http://www.canvasdreams.com/viewarticle.cfm?articleid=1186.

Published by

Series title, number and
report code of publication

VTT Publications 512
VTT–PUBS–512

Author(s)
Kauppi, Tarja

Title

Performance analysis at the software architectural level

Abstract

This work gives an overview on software architecture -based performance analysis methods. PASA method was applied to analysing performance of the part of a mobile phone software that is executed on the Symbian operating system during concurrent streaming and multimedia message reception. As a deviation from the method, LQN approach was used for performance modelling. The model was analysed by calculating utilisation, residence time and queue length. By comparing the calculated values to performance objectives it could be concluded that performance objectives are met on average, but in the worst-case condition performance problems may occur. It was proposed in this work to change the priorities of the related tasks according to RMA principles. Then the deadlines would be met even in the worst-case. In this work performance was analysed only from the processor point of view, so the effect that other hardware resources (such as memory or buses) have on performance was not considered.

Keywords
software quality, analysis methods, software development, mobile phone system, performance

Activity unit
VTT Electronics, Kaitoväylä 1, P.O.Box 1100, FIN–90571 OULU, Finland

ISBN	Project number
951–38–6255–0 (soft back ed.) 951–38–6256–9 (URL:http://www.vtt.fi/inf/pdf/)	E2SU00054

Date	Language	Pages	Price
November 2003	English, Finnish abstr.	78 p.	B

Name of project	Commissioned by
MOOSE	Tekes, Nokia Mobile Phones

Series title and ISSN	Sold by
VTT Publications 1235–0621 (soft back ed.) 1455–0849 (URL: http://www.vtt.fi/inf/pdf/)	VTT Information Service P.O.Box 2000, FIN–02044 VTT, Finland Phone internat. +358 9 456 4404 Fax +358 9 456 4374

Julkaisija

Julkaisun sarja, numero ja raporttikoodi

VTT Publictions 512
VTT–PUBL–512

Tekijä(t)
Kauppi, Tarja

Nimeke

Ohjelmistoarkkitehtuuritason suorituskyvyn analysointi

Tiivistelmä

Työssä esitellään ohjelmistoarkkitehtuurin suorituskyvyn analysointimenetelmiä. PASA -nimistä analysointimenetelmää sovellettiin matkapuhelimen Symbian-käyttöjärjestelmän päällä toimivan ohjelmiston osan analysoimiseksi matkapuhelimen vastaanottaessa rinnakkaisesti jatkuva-aikaista multimediaa ja multimediaviestiä. PASA -menetelmän ehdottamasta tavasta poiketen ohjelmiston suorituskykyä mallinnettiin LQN -mallinnustapaa käyttäen.. Suorituskykymalli analysoitiin pääasiassa laskemalla prosessorin käyttöaste, prosessorilla oloaika ja jonossa olevien pakettien määrä. Vertaamalla laskettuja arvoja asetettuihin suorituskykytavoitteisiin huomattiin, että keskimäärin suorituskykytavoitteisiin päästään, mutta pahimmassa tapauksessa voi suorituskykyongelmia ilmetä. Työssä ehdotetaan taskien prioriteettien muuttamista RMA:n periaatteiden mukaisesti, jolloin suorituskykytavoitteisiin päästäisiin jopa pahimmassa tapauksessa. Työssä rajoituttiin tarkastelemaan suorituskykyä ainoastaan prosessorin näkökulmasta, joten väylän ja muistin käytön vaikutusta ei tarkasteltu.

Avainsanat
software quality, analysis methods, software development, mobile phone system, performance

Toimintayksikkö
VTT Elektroniikka, Kaitoväylä 1, PL 1100, 90571 OULU

ISBN			Projektinumero
951–38–6255–0 (nid.) 951–38–6256–9 (URL: http://www.inf.vtt.fi/pdf/)			E2SU00054
Julkaisuaika Marraskuu 2003	Kieli englanti, suom. tiiv.	Sivuja 78 s.	Hinta B
Projektin nimi MOOSE		Toimeksiantaja(t) Tekes, Nokia Mobile Phones	
Avainnimeke ja ISSN VTT Publications 1235–0621 (nid.) 1455–0849 (URL: http://www.inf.vtt.fi/pdf/)		Myynti VTT Tietopalvelu PL 2000, 02044 VTT Puh. (09) 456 4404 Faksi (09) 456 4374	

VTT PUBLICATIONS

492 Himanen, Mervi. The Intelligence of Intelligent Buildings. The Feasibility of the Intelligent Building Consept in Office Buildings. 2003. 497 p.

493 Rantamäki, Karin. Particle-in-Cell Simulations of the Near-Field of a Lower Hybrid Grill. 2003. 74 p. + app. 61 p.

494 Heiniö, Raija-Liisa. Influence of processing on the flavour formation of oat and rye. 2003. 72 p. + app. 48 p.

495 Räsänen, Erkki. Modelling ion exchange and flow in pulp suspensions. 2003. 62 p. + app. 110 p.

496 Nuutinen, Maaria, Reiman, Teemu & Oedewald, Pia. Osaamisen hallinta ydinvoimalaitoksessa operaattoreiden sukupolvenvaihdostilanteessa. 2003. 82 s.

497 Kolari, Sirpa. Ilmanvaihtojärjestelmien puhdistuksen vaikutus toimistorakennusten sisäilman laatuun ja työntekijöiden työoloihin. 2003. 62 s. + liitt. 43 s.

498 Tammi, Kari. Active vibration control of rotor in desktop test environment. 2003. 82 p.

499 Kololuoma, Terho. Preparation of multifunctional coating materials and their applications. 62 p. + app. 33 p.

500 Karppinen, Sirpa. Dietary fibre components of rye bran and their fermentation *in vitro*. 96 p. + app. 52 p.

501 Marjamäki, Heikki. Siirtymäperusteisen elementtimenetelmäohjelmiston suunnittelu ja ohjelmointi. 2003. 102 s. + liitt. 2 s.

502 Bäckström, Mika. Multiaxial fatigue life assessment of welds based on nominal and hot spot stresses. 2003. 97 p. + app. 9 p.

503 Hostikka, Simo, Keski-Rahkonen, Olavi & Korhonen, Timo. Probabilistic Fire Simulator. Theory and User's Manual for Version 1.2. 2003. 72 p. + app. 1 p.

504 Torkkeli, Altti. Droplet microfluidics on a planar surface. 2003. 194 p. + app. 19 p.

505 Valkonen, Mari. Functional studies of the secretory pathway of filamentous fungi. The effect of unfolded protein response on protein production. 2003. 114 p. + app. 68 p.

508 Parviainen, Päivi, Hulkko, Hanna, Kääriäinen, Jukka, Takalo, Juha & Tihinen, Maarit. Requirements engineering. Inventory of technologies. 2003. 107 p.

507 Rosqvist, Tony. On the use of expert judgement in the qualification of risk assessment. 2003. 48 p. + app. 82 p.

509 Sallinen, Mikko. Modelling and estimation of spatial relationships in sensor-based robot workcells. 2003. 218 p.

512 Kauppi, Tarja. Performance analysis at the software architectural level. 2003. 78 p.

Tätä julkaisua myy	Denna publikation säljs av	This publication is available from
VTT TIETOPALVELU PL 2000 02044 VTT Puh. (09) 456 4404 Faksi (09) 456 4374	VTT INFORMATIONSTJÄNST PB 2000 02044 VTT Tel. (09) 456 4404 Fax (09) 456 4374	VTT INFORMATION SERVICE P.O.Box 2000 FIN–02044 VTT, Finland Phone internat. +358 9 456 4404 Fax +358 9 456 4374

ISBN 951–38–6255–0 (soft back ed.)
ISSN 1235–0621 (soft back ed.)

ISBN 951–38–6256–9 (URL: http://www.vtt.fi/inf/pdf/)
ISSN 1455–0849 (URL: http://www.vtt.fi/inf/pdf/)