Linköping Studies in Science and Technology

Dissertation No. 1238

Processes and Models for Capacity Requirements in Telecommunication Systems

by

Andreas Borg

Linköping University
INSTITUTE OF TECHNOLOGY

Department of Computer and Information Science
Linköpings universitet
SE-581 83 Linköping, Sweden

Linköping 2009

ISBN 978-91-7393-700-9
ISSN 0345-7524

Printed by LiU-Tryck, Linköping 2009

Abstract

Capacity is an essential quality factor in telecommunication systems. The ability to develop systems with the lowest cost per subscriber and transaction, that also meet the highest availability requirements and at the same time allow for scalability, is a true challenge for a telecommunication systems provider. This thesis describes a research collaboration between Linköping University and Ericsson AB aimed at improving the management, representation, and implementation of capacity requirements in large-scale software engineering.

An industrial case study on non-functional requirements in general was conducted to provide the explorative research background, and a richer understanding of identified difficulties was gained by dedicating subsequent investigations to capacity. A best practice inventory within Ericsson regarding the management of capacity requirements and their refinement into design and implementation was carried out. It revealed that capacity requirements crosscut most of the development process and the system lifecycle, thus widening the research context considerably. The interview series resulted in the specification of 19 capacity sub-processes; these were represented as a method plug-in to the OpenUP software development process in order to construct a coherent package of knowledge as well as to communicate the results. They also provide the basis of an empirically grounded anatomy which has been validated in a focus group. The anatomy enables the assessment and stepwise improvement of an organization's ability to develop for capacity, thus keeping the initial cost low. Moreover, the notion of capacity is discussed

and a pragmatic approach for how to support model-based, function-oriented development with capacity information by its annotation in UML models is presented. The results combine into a method for how to improve the treatment of capacity requirements in large-scale software systems.

Acknowledgements

This work has been funded by the Swedish Foundation for Strategic Research through the Research center for Integrational Software Engineering (RISE), by the KK foundation through the research school for industrial IT research at Linköpings universitet, by Ericsson AB, and by Vinnova.

This thesis is the concluding result from my years of doctoral studies. Even though I am pleased with the accomplishment, I do not see how it would have been possible without the input and contributions from several very competent and appreciated advisors.

First and foremost, I want to express my deepest gratitude to Prof. Kristian Sandahl: For being an outstanding supervisor, always willing to share his time and vast knowledge to give useful advice, for being patient with my progress during parental leaves, and for being both a colleague and a friend far beyond the duties of a primary supervisor.

I am also truly grateful for the essential contributions by Lic. Eng. Mikael Patel: For arranging so that I could spend two autumns as his colleague at Ericsson AB, for sharing his impressive knowledge and creative mind, for all the inspiration and guidance, and for being a much appreciated travelling companion when attending conferences.

I am also indebted to Dr. Pär Carlshamre for arranging my first stay at Ericsson, for raising my interest for and putting me on track with non-functional requirements, and for serving as a secondary supervisor. Thanks also to Dr. Joachim Karlsson for letting me combine my first years of

doctoral studies with an employment at Focal Point AB and for serving as a secondary supervisor during the same years.

In addition to the group of supervisors, I am much indebted to the 40 anonymous industrial practitioners of Ericsson AB, SMHI, and Saab AB that have generously spent their time and shared their expertise for me to gain valuable industrial data.

I would also like to express my gratitude to past and present Pelab colleagues for their friendship and the very entertaining coffee break discussions. I am particularly grateful to Jens Gustavsson, Levon Saldamli, and John Wilander for co-organizing our interesting study circle on research methodology.

Finally, I have many reasons to be grateful to my beloved wife Kristin and our wonderful children Axel, Klara, and Saga. The reason most relevant to the results herein, though, is the admirable effort Kristin put up to take care of our three months old twin daughters and our 2.5 years old son when I attended RE'06 in Minneapolis – and for doing it again during my stays at another three conferences within a year from then. I also want to thank my parents Kristina and Håkan, Kristin's mother Els-Mari, and my aunt Birgitta for their generous and extensive support to Kristin and our children during these conference trips.

Andreas Borg
Rimforsa, February 2009

Table of Contents

1 Introduction

This chapter presents the background of the thesis and the research objectives it responds to. Furthermore, a brief description of the papers included in the thesis is provided, the applied research method and related issues are described, and the overall contributions are summarized.

1.1 Background and motivation

The complex context of large-scale software engineering is critically dependent on well-managed requirements on all levels and in all phases: From overall system level to the level of the smallest sub systems and from elicitation of requirements to system verification and maintenance. A way of coping with complexity is to apply processes to bring order and to facilitate the coordination of people, tasks, artifacts, etc. Such processes, for example the Rational Unified Process (RUP) [37] supported with UML modeling tools, have been successful in industry as regards functional requirements (FRs). However, non-functional requirements (NFRs) crosscut the system structure [7] and do not easily lend themselves to smooth refinement in functional models. Hence, specialized methods are needed to also comprise successful treatment of NFRs.

The term "non-functional requirement" is wide and there is an ongoing debate regarding the term's usefulness and regarding its definition [20] (which is discussed in Chapter 2). However, regardless of the exact borders of the set denoted "non-functional requirements", there is no doubt that quality factors like usability, performance, reliability, maintainability, etc.

are normally considered as subsets of NFRs. The point-of-view taken herein is that each quality factor needs to be studied separately in order to gain an in-depth understanding of the quality factor in scope and to allow different quality factors to have different properties. Naturally, for instance usability and reliability share properties, both crosscut the functional model, but there are also numerous differences to consider.

The NFR type of special interest in this thesis is *capacity*[1]. It is an important property of large-scale telecommunication systems as well as of other systems with high transaction intensity (such as bank systems, decision support systems, etc.) and it differs from other quality factors in that it is relatively easy to specify and measure. For example, we know how many subscribers a mobile telecommunication system needs to support, how many simultaneous phone calls that the system must handle, what response times that are acceptable, etc., and these properties can be measured.

Capacity provides yet another illustration of how NFRs crosscut the functional model: A software system's capacity cannot be isolated to a single system module. Instead, capacity must be built into the system's architecture and design, which means that capacity requirements must be articulated and present when needed and that organizational issues and power structures are as important as technical aspects. On the other hand, it can be argued that it is possible to cope with capacity as if it was isolated to the system's hardware. There is limited need of addressing capacity issues if newer and better hardware can be bought to compensate for poor system architecture. However, relying solely on upgrading hardware is risky. There may be a limit where better hardware does not significantly improve capacity and there may be another limit where upgrades are simply too expensive for the system to be competitive.

The complex challenge of a telecommunication system is to provide systems with the lowest cost per subscriber and transaction, but also with the highest availability, 24/7 systems with 99.999+ % uptime, and at the same time allow for scalability, that is, the network size and the number of subscribers to grow. The circumstance that the delivered systems must meet the needs of today's tele and data communication networks as well as tomorrow's means that more capacity is always needed, both in terms of bandwidth and transactions per second. Thus, improving capacity is an issue during the entire lifecycle of the system and within each development project, and it must be addressed in all development phases. To achieve

[1] The meaning of capacity is explained in Section 2.4

this, the improved capacity of a new increment is often the combination of both faster hardware *and* better software.

The presented research has been conducted in cooperation with the telecommunication systems provider Ericsson AB. It considers NFRs in general as an introduction, but its major part concentrates on capacity and arrives at a method for improved treatment of such requirements in large-scale software engineering. There are contributions regarding the notion of capacity and how to annotate UML models with capacity information. Moreover, a capacity plug-in to the OpenUP software development process has been constructed and a way of assessing and improving capacity processes using an anatomy has also been suggested. The contributions are empirically grounded as described in Section 1.4.6, and most of the results have been published within the Requirements Engineering community (see Section 1.3).

1.2 Research objectives

Several research questions have been formulated during the research project. To start with, the overall research objective is described by the following research question (Q) and the applied research method is described by the method hypothesis (H) below:

Q **How can capacity requirements be treated so that they are available when needed and influence all phases of large-scale software system development?**

H **It is possible to learn, improve, feed back, and evaluate knowledge regarding NFR/capacity management in large, developing, and administering organizations by the means of industrial case studies.**

The research question is based on the assumption that overall capacity requirements are generally known in large-scale software engineering, but that they are not always transformed into the representations needed to fully influence the architecture, design and testing of the system. This assumption was derived from the investigation of the following closely related explorative research questions:

Q_1 How are NFRs managed in large, developing, and administering organizations?

Q_2 How are capacity requirements managed in large, developing, and administering organizations?

Finally, the suggested improvements regarding capacity procedures were guided by the following research questions:

Q_3 How can the routines regarding capacity requirements and development for capacity be improved in large, developing, and administering organizations characterized by long product life cycle and many releases of the same product?

Q_4 How can capacity be modeled in large-scale software development characterized by long product life cycle and many releases of the same product so that capacity requirements are refined to design and implementation?

1.3 Overview of papers

The research objectives stated in the previous section are responded by Papers I-VI in the second part of the thesis. Each paper is described briefly below to give an early overview and serve as input to the research methodology discussion in the following section.

Paper I: The Bad Conscience of Requirements Engineering: An Investigation in Real-World Treatment of Non-Functional Requirements

Andreas Borg, Angela Yong, Pär Carlshamre, Kristian Sandahl

In the proceedings of the 3rd Conference on Software Engineering Research and Practice in Sweden (SERPS'03), pp. 1-8, Lund, Sweden, 2003.

The first paper is an explorative study that concentrates on the real-world treatment of NFRs. 14 practitioners within two software developing organizations (Ericsson OSS and SMHI) are interviewed regarding NFRs, their treatment, difficulties related to NFRs, and problems that arise due to the difficulties. The objectives are to provide empirical data to support or

challenge the literature, and to identify potential research opportunities for the PhD project. A list of difficulties is assembled, analyzed, and discussed, and the most tangible problems are identified. The reasons to NFR-related problems are found in the nature of NFRs and in hierarchical organization structure.

Dr. Carlshamre and I designed the interview series. The interview series, including the analysis of protocols, were carried out by me and Ms. Yong. Dr. Carlshamre and Prof. Sandahl contributed to the analysis results. I wrote the paper.

Paper II: Good Practice and Improvement Model of Handling Capacity Requirements of Large Telecommunication Systems

Andreas Borg, Mikael Patel, Kristian Sandahl

In the proceedings of the 14th IEEE International Requirements Engineering Conference (RE'06), pp. 245-250, Minneapolis/S:t Paul, 2006.

The scope is narrowed to only consider capacity requirements in the second paper. An interview series regarding the treatment of capacity requirements and related issues was conducted within Ericsson. Focus was on how difficulties related to capacity are overcome and to what extent modeling is used to document capacity information. A number of good practices are identified and put into a methodological context regarding what is needed to be able to develop for capacity. 19 *capacity sub-processes*[2] *(CSPs)* are presented related to the capability areas *Estimation and prediction*, *Specification*, *Measurement and tuning*, and *Verification*.

I conducted the interview series which was co-designed by me and Lic. Eng. Patel. We jointly analyzed the results, with Lic. Eng. Patel's expertise in telecommunication capacity and the development activities within Ericsson as prerequisites for putting the results into their methodological context. I wrote most of the paper.

[2] Only 18 CSPs were presented in the original version of this paper. Editorial revisions have been made so that all papers in the thesis present 19 CSPs.

Paper III: Integrating an Improvement Model of Handling Capacity Requirements with the OpenUP/Basic Process

Andreas Borg, Mikael Patel, Kristian Sandahl

In the proceedings of the International working conference on Requirements Engineering: Foundations for Software Quality (REFSQ'07), pp. 341-354, Trondheim, Norway, 2007.

The third paper proceeds from the CSPs presented in Paper II. The Eclipse Process Framework (EPF) [15] is applied to transfer the CSPs into a so called *method plug-in* to the OpenUP/Basic software development process [46]. This is done via a series of workshops involving all co-authors of the paper. The method plug-in facilitates the feedback of Paper II results within Ericsson (EPF and OpenUP/Basic can be regarded as open and free variants of the Rational Model Composer and RUP that is used within Ericsson) and it also makes the communication with other researchers smoother. The receiver of the method plug-in is typically a process engineer who can choose to extend a process with support for capacity development.

Lic. Eng. Patel suggested the idea of a method plug-in to represent the CSPs as a process extension accessible to both Ericsson employees and other researchers. The analysis of how to implement the capabilities in a method plug-in was carried out jointly by me, Lic. Eng. Patel, and Prof. Sandahl. I did most of the actual plug-in construction and I also wrote most of the paper with contributions from Prof. Sandahl.

Paper IV: Extending the OpenUP/Basic Requirements Discipline to Specify Capacity Requirements

Andreas Borg, Mikael Patel, Kristian Sandahl

In the proceedings of the 15th IEEE International Requirements Engineering Conference (RE'07), pp., 328-333, Delhi, India, 2007.

Paper IV is based on the same foundation as Paper III but concentrates solely on the requirements perspective. The requirements discipline of OpenUP/Basic and how our method plug-in can support the specification of capacity requirements is described. Our approach is compared to another independent process initiative – called the W project – related to capacity improvement within Ericsson. The approaches are estimated to be around 80 percent similar and we get confirmation on our major ideas:

modeling real-life capacity, using time budgets, and defining sub-system tests.

I wrote most of the paper. Lic. Eng. Patel is responsible for the presentation of W project material.

Paper V: A Case Study in Assessing and Improving Capacity Using an Anatomy of Good Practice (extended version)

Mikael Patel, Andreas Borg, Kristian Sandahl

The 6th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE'07), pp. 509-512, Dubrovnik, Croatia, 2007.

This paper proposes an Anatomy of Capacity Engineering (ACE). An *anatomy* is constructed from the CSPs in which their internal relations and ordering are made visible. ACE involves four steps for how an organization can assess and improve its capacity abilities. The initial two ACE steps, the assessment activity and visualizing the results in the anatomy, are tried in three case studies and the results are briefly discussed. The paper as presented in this thesis has been extended to provide a more detailed description of ACE.

Lic. Eng. Patel was the main architect behind the construction of the anatomy and he also carried out the assessments of the Ericsson-internal cases of the case studies. Prof. Sandahl performed the assessments on the OpenUP/Basic process. I wrote the paper.

Paper VI: A Method for Improving the Treatment of Capacity Requirements in Large Telecommunication Systems

Andreas Borg, Mikael Patel, Kristian Sandahl

Submitted to Requirements Engineering Journal.

The final paper is of a special kind. Its major contribution is that it describes the progress from Paper I through Paper V and how the papers fit together. Thus, contents from all the previous papers can be found in this paper too, but it also includes the description of a pragmatic approach to annotating UML models with capacity information. I wrote the paper.

1.4 Research methodology

1.4.1 SOLVING REAL-WORLD PROBLEMS

The research approach taken in this project adheres to a problem-oriented paradigm, in which *relevance* and *usefulness* in a relatively near future are important properties. Thus, research that might lead to a major break-through in twenty years is less preferred than research that has reasonable chances to create something useful in the perspective of one to five years.

Applying the concepts of relevance and usefulness means, in the case of software engineering, to try and solve problems that are faced by software engineering practitioners in software engineering industry. Consequently, such problems need to be identified in order to perform the described research, which can be done indirectly by reading research papers describing real-world problems. However, the obvious alternative is to be there, in software engineering industry, to meet with practitioners and identify problems directly in what is sometimes called *industry-as-laboratory* [47]. Fortunately, this opportunity was given several times during the research project.

1.4.2 CASE STUDIES AND FOCUS GROUPS

The empirical experience presented herein has mainly been acquired from various organizations within Ericsson AB. However, the explorative problem inventory of Paper I also involved the Swedish Meteorological and Hydrological Institute (SMHI), and it is described in Paper VI that representatives from Saab AB were involved in the validation of ACE. The research methods that have been applied to acquire this industrial experience are those entitling this subsection.

Case studies are described in Paper I and Paper V. The problem inventory of Paper I is a case study in which an interview series from one case was replicated in a second case (the fact that it is built around an interview series indicates that the case study type is *survey* [39]), and ACE – the capacity assessment and evaluation method proposed in Paper V – is tried out in three different *experimental* case studies [39] to investigate the method's validity.

The *focus group* is a qualitative research method that can be used for several purposes. The method originates from market research where companies can evaluate their ideas and products in groups of carefully selected representatives of the target customer [16]. In market research, the

typical focus group is video-taped and consists of eight to ten participants and a moderator.

The focus group described in Paper VI follows the focus group design as described by Hedenskog [25]: A few people, preferably four to get a good balance between quantity and depth in discussion, share their knowledge and thoughts regarding a limited set of questions prepared by the researcher, and discussion is optimally facilitated by somebody not involved in the study. Each participant has to think each question over individually and account for his/her opinion orally one by one before plenary discussion is allowed (to avoid bias). The researcher takes notes and records the discussion using audio or video equipment and performs protocol analysis on transcripts.

The focus group was used to assess the transferability of ACE to organizations outside of Ericsson. It was conducted strictly according to Hedenskog's example, that is, four participants (from the defense and avionics company Saab AB), an outside moderator, and audio recordings that were transcribed and analyzed.

1.4.3 ACTION RESEARCH

Parts of the work presented in this thesis can be characterized as *action research*. Different views of action research are presented by Cronholm and Goldkuhl [10] and a short summary is provided below.

An action research project involves both researchers and practitioners and they collaborate to reach common goals. Thus, the researchers work together with practitioners to accomplish some kind of business change. This contrasts with a *participatory observation* approach which allows researchers to be present in industrial contexts, but only to observe procedures from a "fly-on-the-wall" perspective.

Cronholm and Goldkuhl [10] point out that the action researcher must be interested in both the action and the research. (A consultant could collaborate with practitioners but is probably only interested in the action.) The actual change of procedures constitutes the action whereas the research is about generating new knowledge, which means that reflecting upon the business change process is an important research activity. McCay and Marshall [40] have formalized these dual aims into an action research process that consists of two interlinked cycles: The aim to improve a real-world situation and the aim to generate new knowledge based on the research question. This view enables the possibility to apply both a

research and a business change perspective regarding interest, method, and result respectively.

Cronholm and Goldkuhl take the above one step further. The dual cycles of McCay and Marshall are renamed *practices* (research and business respectively) and the intersection in between them is recognized as a practice on its own: the business change practice or the empirical research practice depending on the perspective.

The methodological context for how to develop for capacity that emerged from the second interview series described in Paper II is an example of how researchers and practitioners work together to accomplish an improvement. This kind of work continued in the development of a capacity method plug-in (Paper III and Paper IV) and the development of ACE (Paper V). Finally, Paper VI uncovers the research process and presents reflections of the research process.

It is important to notice that the collaboration regarding business change in our context have been carried out on a process level (within the group responsible for Ericsson's software development processes, methods, and tools) and that we have co-authored research papers on our findings.

1.4.4 QUALITATIVE RESEARCH

The research methods that have been described above have been applied qualitatively in the research project. Strauss and Corbin [53] describe qualitative research in the following way:

> By the term "qualitative research", we mean any type of research that produces findings not arrived at by statistical procedures or other means of quantification.

There are several situations when qualitative research methods are the most suitable to gain knowledge. The most valid in the context of this research project is [53]:

> … to get out in the field and finding out what people are doing and thinking.

The quote above motivates the choice of research method in important parts of the research project. The explorative study of Paper I and the best practice inventory of Paper II are based on interview series with "finding out what people are doing and thinking" as principal objective.

Qualitative research consists of three major components [53] and interviews are a good example of the first component: *data*. Observation

and reading documents are other examples of how to acquire data. Thus, the data component of qualitative research is very similar to the elicitation stage of requirements engineering.

The second component of qualitative research consists of the *procedures* to analyze data. This step is often denoted *coding*, and involves for instance conceptualizing and reducing data and constructing categories with respect to properties and dimensions. In Papers I-II, the dominating procedures to analyze interview data were to summarize interviews into minutes-of-meeting (commented upon by respondents) and to perform protocol analysis, whereas transcription from audio to text preceded the protocol analysis of the focus group described in Paper VI. The papers are also the primary instantiations of the final component of qualitative research: *written and verbal reports*.

1.4.5 GROUNDED AND MULTI-GROUNDED THEORY

Grounded theory (GT) is a well-known approach to qualitative research which originates from within the field of sociology. GT, as it was initially proposed, is strictly inductive. This means that theory is built solely from the analysis of empirical data without considering existing literature until after data has been gathered and analyzed. In fact, original GT explicitly advises against reading literature regarding other theories until a new theory induced from the data has been built. The rationale is to be able to keep an open mind and not let existing theory prejudice the mind of the researcher. Thus, a theory that is "grounded" according to orthodox GT is grounded in empirical data.

From a practical point-of-view, there is an evident objection to GT and its reluctance to consider relevant literature; the probability that the wheel is reinvented increases. However, GT has evolved [53] and its extension into *multi-grounded theory* (MGT) has been suggested. The following quote from Goldkuhl and Cronholm [22] explains how MGT relates to GT:

> There is much GT in our MGT approach. We would like to see it as an extension to or modification of GT. We think that Strauss & Corbin (1998) have taken important steps away from a pure inductivist position. We will continue this move away from pure inductivism. This should not be interpreted as we reject an empirically based inductive analysis as is performed in the coding processes of GT. To have an open-minded attitude towards the empirical data is one of the main strengths in GT and this is incorporated in MGT.

The primary extension in MGT is that the empirically-driven analysis of GT is complemented with theory-driven analysis. In other words, the new theory represents a combined view of what is induced from empirical data and what can be deduced from existing theory. In more detail, MGT suggests two grounding processes – *theoretical grounding* and *internal grounding* – in addition to the original process of empirical grounding. The grounding processes of MGT are illustrated in Figure 1 below.

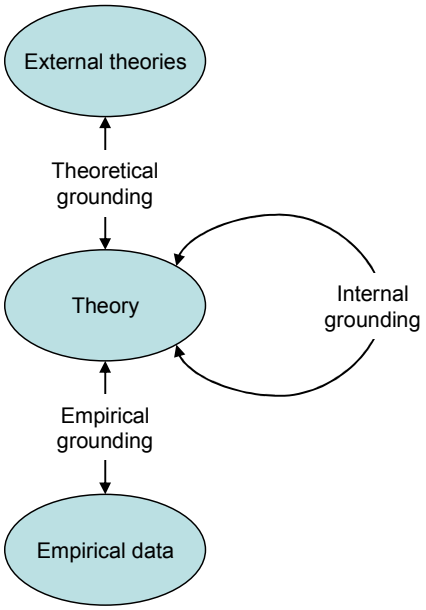

Figure 1: The grounding processes of MGT according to Goldkuhl and Cronholm [22]

The traditional grounded theory is achieved by the analysis of empirical data – the empirical grounding. Naturally, this analysis shall be as inductive as possible. This is true for the first step (inductive coding) of MGT too ("It is harder to introduce an open mind later if one has explicitly used some pre-categories early in the process for interpretation of the data"). However, existing literature is allowed to play a part in the successive steps of the empirical grounding (conceptual refinement, building categorical structures, theory condensation). Moreover, MGT claims that even empirically grounded theories need to be explicitly and systematically checked to ensure its empirical validity.

The theoretical grounding of MGT studies relevant published theories to make use of existing knowledge and to make the new theory coherent with existing theories. Practically this is achieved with *theoretical matching*, in which the evolving theory is confronted with other theories. If there is full conformity between the new theory and existing theories the former is explicitly grounded theoretically. However, the comparison with existing theory may lead to an adaptation of the evolving theory and/or criticism towards existing theories.

Finally, MGT also incorporates internal grounding to explicitly address the consistency within the theory, that is, to evaluate the *theoretical cohesion* of the new theory.

1.4.6 THE GROUNDING OF THIS THESIS

The research process described herein is not rigorous to such an extent that it fully complies with the theoretical description of MGT. Nevertheless, all three grounding processes of MGT are represented in the research project.

First, the empirical grounding is obvious: Papers I and II explore industrial practice as important input to the research project, and it can also be noticed that the focus group of Paper VI represent empirical validation.

Second, the theoretical grounding is almost as obvious. The empirical findings are related to existing literature within the field, and one of the objectives in Paper I was to "corroborate or challenge" what was available in the literature. However, the existing theories were considered in a too early stage to conform to MGT.

Finally, the work with the capacity method plug-in and the method to assess and improve capacity processes (ACE) presented in Papers III-V constitute the internal grounding of the research. Representing the capacity sub-processes of Paper II as a method plug-in and an anatomy forced thinking into terms of internal coherence and consistency. The capacity method plug-in required the transformation of CSPs into a set of roles, tasks, and artifacts that resulted in more detailed knowledge regarding development for capacity. Moreover, constructing an anatomy required thorough thinking regarding the relations between capabilities and how they contribute to each other.

1.4.7 A FEW NOTES ON RESEARCH IN INDUSTRIAL SETTINGS

When Potts proposed the "industry-as-laboratory" approach to replace the "research-then-transfer" approach he made the following statement [47]:

> Industry-as-laboratory research sacrifices revolution, but gains steady evolution.

There are a number of issues to tackle in order to gain this steady evolution. For example, what distinguishes in-house process improvement from research? One of the answers given by Potts is how general the results are. If new knowledge can be gained from the lessons learned within one organization that proves useful in another organization (the less adaptation needed the better) there is clearly relevant research. The focus group described in Paper VI is an example of how to demonstrate a method's general relevance.

Research projects in industrial settings face hindrances of practical kinds as well. An example is how results can be made publically available if the conducted research is concentrated around a company's business secrets. Naturally, most companies are reluctant to share their secrets with their competitors and to expose problems and failures to potential customers. This was not a big problem in this research project since the research was concentrated to methods and processes rather than products.

Other threats are reduced budgets, projects being closed, and that key persons leave (to another company or department). All these threats were calculated risks that were accepted to gain the benefits of being able to perform industrial research and to make unique research findings.

1.4.8 PROTOCOL ANALYSIS

Protocol analysis – how verbal data can be analyzed – has been thoroughly described within the field of cognitive psychology [17], and a good example of how verbal data can be gathered and analyzed within the field of Requirements Engineering is provided by Karlsson et al. [33]. Such techniques have been used and protocols have been produced and analyzed in three parts of the research project.

First, each interview of the interview series described in Paper I was summarized immediately after each interview session based on minutes of meeting. If need for clarifications arose when producing an interview summary the interviewee was asked to redeliver his/her message.

The procedure of the second interview series (see Paper II) was identical to the first with one exception; this time each interviewee read the

interview summary to ensure that it was correct. All interview summaries were accepted by the respective interviewee and only minor changes were made. The advantages of letting interviewees read and comment are that any misunderstandings can be corrected and that improved articulation of vague wordings can be achieved. However, there is also a possibility that an interviewee wants to change his/her statement in a matter, which can then be regarded as another data point to the previous ones.

The focus group that was used to validate ACE in Paper VI was the most rigorous approach to creating protocols, since the discussion was transcribed from audio to text.

The actual protocol analysis techniques applied to the summaries from the two interview series were straightforward. The set of questions provided a structure for coding and analyzing the summaries into themes and responses could be compared per question. The analysis of the focus group protocols was somewhat different since the verbal material was a discussion, not interviews. However, the discussion was structured according to the contents of the anatomy in focus and participants suggested anatomy design improvements while assessing the transferability of each CSP. Thus, themes could be easily identified regarding the transferability of ACE and regarding the design of the anatomy as such.

1.5 Contributions

The contributions reported of herein correspond well to the collection of papers and can be summarized as follows:

- An industrial survey and empirical data on real-world NFR problems.
- An industrial survey and empirical data regarding how capacity requirements are treated within Ericsson.
- A set of CSPs that is useful when developing for capacity in large-scale telecommunication systems.
- A capacity method plug-in that can be used (and adapted) in conjunction with the OpenUP/Basic software development process.
- A method for how to assess and improve capacity processes (ACE) validated in a focus group.
- A heuristic suggestion for how to include capacity information in UML models.
- An integrated method for how to treat capacity requirements in large-scale telecommunication systems based on the above contributions.

1.6 Related publications not included in the thesis

Borg, A., J. Karlsson, S. Olsson, and K. Sandahl. "Supporting Requirements Selection by Measuring Feature Use", in the proceedings of the *10th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'04)*, pp. 77-82, Riga, Latvia, June 7-8, 2004.

Borg, A., J. Karlsson, S. Olsson, and K. Sandahl. "Measuring the Use of Features in a Requirements Engineering Tool-An Industrial Case Study", in the proceedings of the *Fourth Conference on Software Engineering Research and Practice in Sweden (SERPS'04)*, pp. 101-110, Linköping, Sweden, October 21-22, 2004.

Borg, A. *Contributions to Management and Validation of Non-Functional Requirements*. Licentiate thesis no. 1126, Department of Computer and Information Science, Linköpings universitet, Sweden, 2004.

Gorschek, T., M. Svahnberg, A. Borg, J. Börstler, M. Eriksson, A. Loconsole, and K. Sandahl. "A Controlled Empirical Evaluation of a Requirements Abstraction Model", *Information and Software Technology*, Vol. 49, No 7, pp. 790-805, July 2007.

Sandahl, K., M. Patel, and A. Borg. "A Method for Assessing and Improving Processes for Capacity in Telecommunication Systems", in the proceedings of the *Seventh Conference on Software Engineering Research and Practice in Sweden (SERPS'07)*, Göteborg, Sweden, October 24-25, 2007.

Svahnberg, M., T. Gorschek, M. Eriksson, A. Borg, K. Sandahl, J. Börstler, and A. Loconsole. "Perspectives on Requirements Understandability: For Whom Does the Teacher's Bell Toll?", in the proceedings of the *Third International Workshop on Requirements Engineering Education and Training (REET'08)*, Barcelona, Spain, September 9, 2008.

Borg, A., M. Patel, and K. Sandahl. "Modeling Capacity Requirements in Large-Scale Telecommunication Systems", in the Proceedings of the *Eighth Conference on Software Engineering Research and Practice in Sweden (SERPS'08)*, Karlskrona, Sweden, November 4-5, 2008.

2 Frame of Reference

This chapter provides an overview of issues that form the frame of reference. The meanings of requirements, non-functional requirements, and capacity are described, as is the case with processes and process improvements.

2.1 Background

The contents of this thesis originate from within the field of Requirements Engineering. The research project started with an initial interest in NFRs and evolved into investigating capacity requirements. This evolvement is reflected in the frame of reference. A brief description of requirements in general is provided to start with, followed by guidance to NFRs before we end with a detailed description of what is capacity. Processes and process improvement are also described. Related work is pointed out and discussed along the way.

2.2 Software requirements

2.2.1 REQUIREMENT DEFINITIONS

Many definitions of the term "requirement" have been proposed. In this section some well-known suggestions are described in order to provide basic domain information.

The general objective of RE is to capture the ideas and needs of various stakeholders and transform these needs into a solid basis for system development. Harwell et al. [24] emphasize this when formulating the purpose of requirements:

> … to reproduce in the mind of the reader the intellectual content which was in the mind of the writer.

Even though this takes into account the transformation of the ideas and needs of various stakeholders into a proper representation, it does not define the term "requirement" (and it is also narrowed to the communication between readers and writers as noted by Carlshamre [6]). Furthermore, the explanation assumes that the writer has the correct picture of the requirement(s). Singer [51] provides a more general definition of the term:

> A requirement is a portrait of a user's needs.

Although excluding all stakeholders but users, this definition nicely encompasses that requirements can be explicit as well as implicit. Explicit requirements are those that stakeholders ask for and can express, whereas implicit requirements are those requirements that are unspoken. The reason for implicit requirements may be that stakeholders simply do not know all their needs and/or that requirements are so obvious to stakeholders that they take them for granted.

A widely adopted "truth" regarding requirements is that they shall focus entirely on what is needed, leaving any how-aspect for designers to handle. This seems natural recalling that the requirements should provide a detached "portrait of users' needs". However, *how* is sometimes inseparable from *what* and the questions mean different things to different people. This is discussed by Davis [13] (page 17) who also provides a useful requirement definition emphasizing what will go into the product (by the formulation "external to that system"):

> [A requirement is] a user need or a necessary feature, function or attribute of a system that can be sensed from a position external to that system.

Kotonya and Sommerville [35] moves even further away from user centered requirements definitions when defining requirements

> …as a specification of what should be implemented. They are descriptions of how the system should behave, application domain information, constraints on the system's operation, or specifications of a system property or attribute. Sometimes they are constraints on the development process of the system.

Observing that even this small sample of definitions provide a rather disharmonious picture of requirements, it is easy to understand that no universal definition is available so far. However, the IEEE's definition [28] of software requirements is widely spread and accepted and concludes this section: A requirement is:

> (1) A condition or capability needed by a user to solve a problem or achieve an objective.
> (2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents.
> (3) A documented representation of a condition or capability as in (1) or (2).

Note that this definition includes both the user's perspective and other system characteristics. Moreover, the definition uses *requirement* for a user need as well as for its corresponding documented representation (that is, the user need is a requirement even before it is documented).

2.2.2 FUNCTIONAL VS. NON-FUNCTIONAL REQUIREMENTS

There are many ways of classifying requirements. However, a very common way of separating requirements – which is also the most valid classification for the topic of this thesis – is into functional requirements (FRs) and non-functional requirements (NFRs).

Functional requirements are characterized by their exclusive devotion to the already mentioned what-aspect of the system. Revisiting the IEEE glossary [28], a functional requirement is defined as:

> A requirement that specifies a function that a system component must be able to perform.

"Function" in the above definition can be regarded as semantically equivalent with the mathematical notion of a function:

$$y = f(x)$$

The mathematical function defines the relation between the variables x and y for every possible x, and a specific value of x defines the value of y deterministically. A software function does the same: input variables or states are deterministically transformed into their corresponding output variables or states. The following is an example of a functional requirement:

> Pressing the "Calculate BMI" button shall result in the correct BMI[3] value for the current entries being calculated and displayed.

The example requirement clearly describes a function of the forthcoming system. However, there are further considerations that must be made and specified to transform this piece of intended functionality into a usable implementation. What if:

- Most or all intended users do not know the meaning of the BMI value or how to interpret it?
- The interface location of the button and/or the displaying of the result are unknown to most intended users?
- The calculation takes a month or two?

Thus, the functional requirement can be literally met, but completely useless if one or several of these (or similar) situations occur. It is evident from the above scenario that additional properties of the requirement need to be specified, and such requirements are often referred to as non-functional. The following is a non-functional requirement addressing one of the considerations listed above:

> The time elapsed between pressing the "Calculate BMI" button and the result being displayed shall be less than 0.1 seconds.

Non-functional requirements are described in the following section.

[3] BMI (Body Mass Index) uses body mass and body length to indicate overweight or underweight in a simple but common way.

2.3 Non-functional requirements

2.3.1 BACKGROUND

There is a general opinion that NFRs are difficult to capture as well as to define, and that a major reason is their vague nature (which is supported by our own research, see Paper I). This vagueness tempts requirements writers to use words like "easy", "optimal", "flexible", etc. which do not properly describe what is wanted or how requirements should be tested. Requirements of the type *"The user shall find it easy to …"* is a typical example. Another example is found in a publically available requirements specification of a system ordered by the Swedish police authority (directly translated from Swedish):

> A 3.16.5 The response-, access- and processing times of the D-System and other factors that are significant to the D-System from a user's perspective **shall** be optimal. Moreover, variations adhering to different load conditions may not occur in such a way that the D-System is perceived as slow, inconsistent or unrhythmic from a user's perspective.

The example requirement is subjectively stated (the user's perspective, optimality, etc.) and makes use of words that are impossible to interpret correctly ("unrhythmic"). It cannot be sufficiently tested and it is hard to imagine how it could be satisfied.

NFRs are also generally considered difficult to test. A good example is usability that often requires either lots of time, extensive effort, many people, or expertise (or even all of them) to be tested (Carlshamre [6] provides an overview), whereas subjective evaluation can be done at a significantly lower cost. The vague nature of NFRs also makes it difficult to write measurable and unambiguous requirements. Furthermore, the majority of existing processes and techniques focus on FRs and are not well-suited for NFRs.

According to Kotonya and Sommerville [35] most existing RE methods do not adequately cover NFRs simply because it is very difficult to do so. Reasons are, for instance, that certain constraints are unknown at the requirements stage, that some constraints need very complex empirical evaluations to be determined, and that NFRs tend to conflict each other. Furthermore, they argue that separating NFRs and FRs makes it difficult to see dependencies between them, whereas functional and non-functional considerations are difficult to separate if all requirements are stated

together. Finally they claim that it is difficult to determine when NFRs are optimally met, since it is almost always possible to refine solutions. Despite these difficulties, there are approaches that address treatment of NFRs in various ways, although not yet standardized in for instance mainstream RE text books and methods.

Chung et al. [7] state that *"two basic approaches characterize the systematic treatment of non-functional requirements"*, which are referred to as *product-oriented* and *process-oriented*. Product-oriented approaches received most of the attention to begin with (see Keller et al. [34] for an overview), whereas process-oriented (or goal-oriented) approaches have gained a lot of interest the past decade. The main difference between the approaches is that the product-oriented approach aims at determining to what extent the conclusive software system fulfils its NFRs, whereas the process-oriented approach tries to deal with NFRs during the development process, and make sure that NFRs *will* be fulfilled by the conclusive system. Product-oriented and process-oriented approaches are described in sections 2.3.3 and 2.3.4 respectively.

2.3.2 TERMINOLOGY

Considering the vagueness of NFRs as described above it seems logical that vagueness applies to the actual term "non-functional requirement" as well. Nevertheless, the term "non-functional requirement" (NFR) is widely accepted and is used throughout this thesis to denote what is also called *extra-functional requirement* [26], *non-behavioral requirement* [13], and *quality requirement* [34] in related literature. The highly associated term *quality attribute* [56] is generally equivalent to *NFR type* (for example performance, maintainability, usability), and sometimes the terms *goal* and *constraint* are used as well to label various kinds of NFRs. The definition of "functional requirement" (see Section 2.2.2) does not have a corresponding definition of "non-functional requirement" in the quoted glossary, instead "functional requirements" are claimed to contrast with *"design requirements, implementation requirements, interface requirements, performance requirements, and physical requirements"*. Thayer and Thayer re-formulate this in their RE glossary [55], explaining the non-functional requirement as:

> In software system engineering, a software requirement that describes not what the software will do, but how the software will do it, for example, software performance requirements, software external interface requirements, software design constraints, and software quality attributes.

This description presents a common view but it still leaves room for interpretations ("for example"). The division between what the system does (FRs) and how the system behaves (NFRs) can also be questioned. Glinz explains why in a recent paper on NFRs [20], which is begun with the following sentences:

> If you want to trigger a hot debate among a group of requirements engineering people, just let them talk about non-functional requirements. Although this term has been in use for more than two decades, there is still no consensus about the nature of non-functional requirements and how to document them in requirements specifications.

Glinz presents and analyzes a list of definitions (see Table 1) and arrives at the conclusion that the problems regarding the notion of non-functional requirements manifest in their definition, classification, and representation. He suggests that the traditional classification of functional and non-functional requirements is replaced by a faceted classification which separates the concepts of representation, kind, satisfaction, and role. This conforms well to the suggested aspect-oriented representation of requirements and their definition as a number of concerns. System requirements are divided into four concerns: *functional*, *performance*, and *quality* concerns complemented with *constraints*. Quality (the "-ilities") and performance are combined into *attributes*, but are treated separately since they are "typically treated separately in practice". The reason is explained to be that there is a consensus for how to measure performance (time, volume, and volume per time unit) but that no such consensus is available for other quality factors. This means that the definition of non-functional requirements – if we want to stick to that term – according to Glinz is:

> A non-functional requirement is an attribute of or a constraint of a system.

To conclude, this allows requirements to be classified by applying four simple rules in the following order. If a requirement was stated to specify (1) *"some of the system's data, input, or reaction to input stimuli – regardless of the way how this is done"*, then it is a functional requirement. If it was stated to specify (2) *"restrictions about timing, processing or reaction speed, data volume, or throughput"*, then it is a performance requirement. If it was stated to specify (3) *"a specific quality that the system or a component shall have*", then it is a specific quality. Finally, if it was stated to specify "*any other restriction about what the system shall do, how it shall do it, or any prescribed solution or solution element*", then it is a constraint.

Table 1: "Non-functional requirement" definitions compiled by Glinz [20]

Source	Definition
Antón [1]	Describe the nonbehavioral aspects of a system, capturing the properties and constraints under which a system must operate.
Davis [13]	The required overall attributes of the system, including portability, reliability, efficiency, human engineering, testability, understandability, and modifiability.
IEEE 610.12 [28]	Term is not defined. The standard distinguishes design requirements, implementation requirements, interface requirements, performance requirements, and physical requirements.
IEEE 830-1998 [29]	Term is not defined. The standard defines the categories functionality, external interfaces, performance, attributes (portability, security, etc.), and design constraints. Project requirements (such as schedule, cost, or development requirements) are explicitly excluded.
Jacobson, Booch and Rumbaugh [30]	A requirement that specifies system properties, such as environmental and implementation constraints, performance, platform dependencies, maintainability, extensibility, and reliability. A requirement that specifies physical constraints on a functional requirement.
Kotonya and Sommerville [35]	Requirements which are not specifically concerned with the functionality of a system. They place restrictions on the product being developed and the development process, and they specify external constraints that the product must meet.
Mylopoulos, Chung and Nixon [41]	"... global requirements on its development or operational cost, performance, reliability, maintainability, portability, robustness, and the like. (...) There is not a formal definition or a complete list of non-functional requirements."
Ncube [42]	The behavioral properties that the specified functions must have, such as performance, usability.
Robertson and Robertson [48]	A property, or quality, that the product must have, such as an appearance, or a speed or accuracy property.
SCREEN Glossary [49]	A requirement on a service that does not have a bearing on its functionality, but describes attributes, constraints, performance considerations, design, quality of service, environmental considerations, failure and recovery.
Wiegers [56]	A description of a property or characteristic that a software system must exhibit or a constraint that it must respect, other than an observable system behavior.
Wikipedia: NFRs [57]	Requirements which specify criteria that can be used to judge the operation of a system, rather than specific behaviors.
Wikipedia: Requirements Analysis [58]	Requirements which impose constraints on the design or implementation (such as performance requirements, quality standards, or design constraints).

2.3.3 PRODUCT-ORIENTED APPROACHES

When applying a product-oriented approach to NFR treatment the conclusive software system is considered. Measuring and verifying the performance of features before releasing a product is a basic example of this. Thus, the ability to specify testable quality requirements is essential and with that metrics are placed in focus.

A product-oriented approach requires some kind of formal framework that describes the quality attributes that need to be measured and which metrics to use when evaluating to what extent the quality attributes are met. An early and well-known example of such a framework was accounted for already in 1990 by Keller et al. [34] based on the extensive work of the Rome Air Development Center (RADC). The quality attributes are classified into a structure and metrics are used to provide visibility to decision makers, adherence to documented standards, and to serve as input to prediction models. The framework as such is "a hierarchical metrics structure in which metrics are organized into metric-aggregates", which means that metrics on one level are computed from metrics of another level.

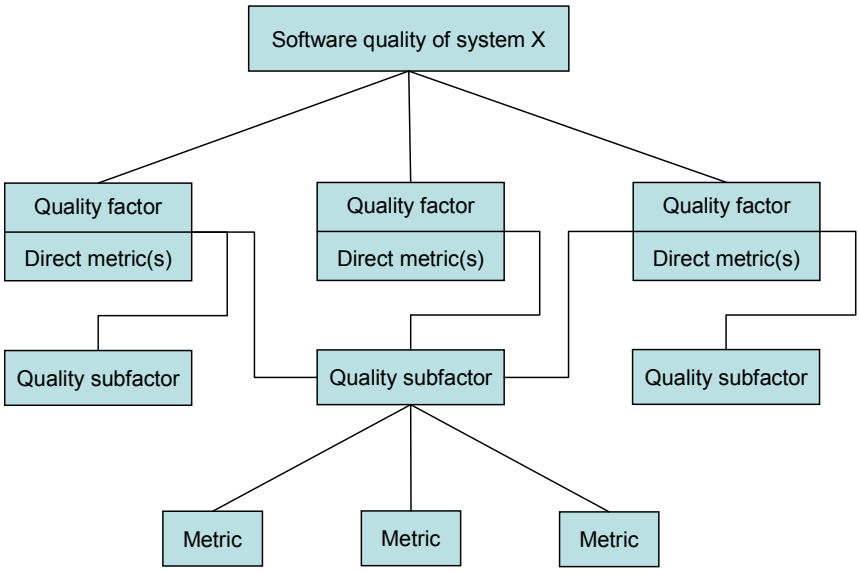

Figure 2: Software quality metrics framework as presented in the "IEEE standard for a software quality metrics methodology" [27]

The *IEEE Standard for a Software Quality Metrics Methodology* [27] is similar to the above in several ways. It shares the approach of applying metrics on several levels and an example is shown in Figure 2. The standard also comprises a methodology that "is a systematic approach to establishing quality requirements and identifying, implementing, analyzing, and validating the process and product software quality metrics for a software system".

2.3.4 PROCESS-ORIENTED APPROACHES

In contrast to product-oriented approaches, process-oriented approaches focus on the actual software development process. The idea can be described as to let the positive and negative contributions of design decisions with respect to NFRs drive the development process. Thus, these contributions can imply that certain NFR aspects are met or describe why they are not.

Process-oriented approaches are often called *goal-oriented* methods as well, due to the fact that they focus on the goals (such as "the system shall be secure", "serve more subscribers", etc.) of the software system. These approaches do not concentrate on NFRs exclusively; both FRs and NFRs are derived from the stated goals.

Goal-oriented requirements engineering, its basic principles, and its approaches and frameworks have been described by van Lamsweerde [38], and his description is still informative. Four major approaches, which are briefly described below, can be distinguished in goal-oriented RE. These have many interconnections and three of them actually originate from the Knowledge Management Laboratory[4] of the University of Toronto.

A goal-oriented method that is concentrated on NFRs is the *NFR Framework* that has been developed by Chung et al. [7], which is also one of the most comprehensive approaches to NFRs. The method is based on the decomposition of a few general NFRs (security and performance for example) that are considered important, using so called *Softgoal Interdependency Graphs* (SIGs) and catalogued design knowledge. The term "softgoal" denotes a specific non-functional goal and is used to point out that such a goal has no clear-cut criteria to whether it is satisfied or not. Similarly, the term "satisfice" (can be read as "sufficiently satisfied") is used to indicate the same thing, that is, stating that a goal is satisficed means that it is sufficiently satisfied. The decomposition goes all the way

[4] See http://www.cs.toronto.edu/km/. Accessed Feb 16, 2009.

from the initial softgoals to design decisions and implementation suggestions ("operationalizations") using AND/OR refinement. The framework models ambiguities, tradeoffs and priorities as well as interdependencies between softgoals and operationalizations.

The *KAOS*[5] [12] approach is complementary to the NFR Framework. The NFR Framework is a qualitative framework oriented towards satisficing quality goals (the use of negative and positive contributions to drive the design process is clearly qualitative). In contrast, KAOS can be described as a formal framework concentrated on goal satisfaction and how to build complete requirements models with no internal conflicts. The approach extends requirements modeling beyond traditional *what* statements to also include the aspects of *why*, *who*, and *when*. Roughly, goals are identified and refined, and objects and actions are also identified from the goal refinement procedure. Requirements on objects and actions are derived that explains how constraints can be met, and these constraints, objects, and actions are assigned to the agents of the system. However, it can also be noted that efforts have been made to make the NFR Framework quantitative: The *Attributed Goal-Oriented Requirements Analysis Method* (AGORA) [32] is an attempt to add metrics, basically by assigning values to the positive and negative contributions mentioned above.

The *i** framework is claimed to extend goal-oriented RE as described by Yu and Mylopoulos [60], particularly regarding the softgoal concept that continues from the techniques applied in the NFR Framework. However, *i** is an *agent-oriented* approach useful in RE as well as in business process modeling that consists of several autonomous parties. An agent can be described as a non-human actor that is:

- Situated – it senses and changes the environment
- Autonomous – it has control of its actions and can act without human intervention
- Flexible – it responds to environmental changes
- Social – it can interact with humans and other agents

The above approaches are mainly directed to the requirements phase of system development. *Tropos* is an agent-based software development methodology and framework that reuses the notions of actor, goal, and

[5] See http://www.info.ucl.ac.be/~avl/ReqEng.html. Accessed Feb 16, 2009.

dependency from *i** and proceeds from requirements to architecture and detailed design.

Finally, in addition to modeling requirements and providing basis for design decisions, process-oriented approaches can serve as requirements elicitation techniques as well. Decomposing high-level goals and properties means adding more refined requirements that need to be discovered. For instance, decomposing the top-level requirement *"the system shall be secure"* requires further specification (in several steps) regarding the system's security and how it is to be achieved.

2.4 Capacity

2.4.1 CAPACITY, PERFORMANCE, AND EFFICIENCY

Terms like capacity, performance, and efficiency are used with slight differences in practice and in the literature. Hence, before the meaning of capacity in the context of this research project is described in the next section a few definitions from the literature are provided.

The *Software Quality Characteristics Tree* [5] has been influential to software quality and provides the foundation to successive quality models, such as the one described in the ISO/IEC 9126-1:2001 standard. In both these models, *efficiency* is the quality factor that contains capacity issues. The factor is built from the sub factors accountability, device efficiency, and accessibility in the former model and consists of time behavior and resource behavior in the latter model. Put in text, efficiency is the following in the ISO/IEC standard:

> A set of attributes that bear on the relationship between the level of performance of the software and the amount of resources used, under stated conditions.

Davis's [13] definition of capacity is simple: *capacity, timing constraints*, *degradation of service*, and *memory requirements* are subsets of *efficiency* [5]. Capacity is stated to respond to the question "How many?" and also to take into account peak versus normal periods.

The efficiency definition above rely on the concept of *performance*, which is described in the following way in the *IEEE Standard Glossary of Software Engineering Terminology (Std 610.12-1990)* [28]:

> The degree to which a system or component accomplishes its designated functions within given constraints, such as speed, accuracy, or memory usage.

Performance is, in conformity with efficiency, often regarded to embrace both timeliness and memory usage (for example [7]). However, the approach of Software Performance Engineering (SPE) [52] concentrates primarily on timeliness and states that "*performance is the degree to which a software system or component meets its objectives for timeliness"*. Furthermore, performance is stated to have two dimensions: *responsiveness ("the ability to meet objectives for response time or throughput")* and *scalability* (the ability to meet these objectives "… *as the demand for software functions increases*").

It shall be noticed that there is a live community around SPE and other aspects of software and performance, and the *International Workshop on Software and Performance* (WOSP[6]) has been organized seven times since 1998 so far. The objective of the first workshop was to join the forces of the software engineering community and the performance analysis community. One program co-chair was appointed from within each community and the general chair was Connie U. Smith, the author of "*Performance Solutions: A Practical Guide to Creating Responsive, Scalable Software"* [52]. She has been part of the program committee ever since and the overall goal of the workshop remains the same: to build predictable performance into software by quantitative means. Modeling tools and techniques have been important areas of interest from the start and in WOSP 2008 the program co-chair positions were dedicated to software and modeling respectively. Apart from Dr. Smith, other well-known researchers that frequently have been part of the workshop's program committee are, for example, Elaine Weyuker, Bran Selic, Vittorio Cortellessa, Lawrence Chung, Paola Inverardi, Simonetta Balsamo, Reiner Dumke, and José Merseguer.

2.4.2 CAPACITY IN THIS RESEARCH PROJECT

Capacity belongs to the quality factors generally referred to as *characteristics* at Ericsson and it can be described concisely as a system's ability to provide *good-enough service to a large-enough number of subscribers*. This indicates a trade-off; that the minimum service-level maximizes the possible number of subscribers and vice versa. It also illustrates that capacity is more than the concept responding to the question "How many?" Instead, we perceive the general use of "capacity" as being close to "efficiency" as described in the previous section. This means that performance can be regarded as a subset of capacity.

---

[6] See http://www.inf.pucrs.br/wosp/. Accessed Feb 16, 2009.

The capacity view that we have adopted reflects the ambition to undertake the research effort of improving the treatment of capacity requirements and refining these into design and implementation using terminology that reduces the risk of misunderstandings. Thus, our capacity view focus the parts considered especially important for representing requirements when developing for capacity, and can be depicted as in Figure 3 below (using the AND tree notation from the NFR Framework [7]). This is the view that is used throughout the remainder of this thesis.

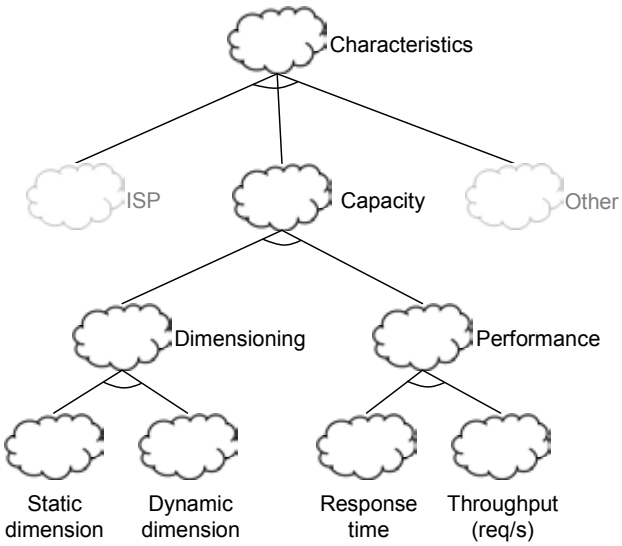

Figure 3: The adopted capacity view

Capacity is one of several quality factors within Ericsson which are normally referred to as *characteristics*. The most important quality factor is a combination of reliability and availability denoted *in service performance* (ISP), which can be described as the system's ability to be up-and-running and providing service. However, capacity and several other quality factors (such as security and maintainability) are considered important as well.

Capacity consists of two parts: *dimensioning* and *performance*. Dimensioning is equivalent to Davis's description of capacity [13] with one important extension: The difference between static and dynamic dimension is made explicit. The typical example is to specify both the maximum number of subscribers (static) in the network and the maximum number of *active* subscribers, for example placing a call simultaneously

(dynamic). The adopted capacity view emphasizes the elaboration of capacity requirements before design and/or implementation. The implementations of these requirements are typically in memory, disk, and processor clock cycles. The limits of the resources are also plain requirements. The dynamic dimensioning in a telecommunication system often reflects upper limits of communication resources. This means, in the case of real-time operating systems, dynamic dimensioning resources such as number of processes, sockets, I/O buffers, switching hardware, etc.

Performance, on the other hand, is similar to how it is described in the *Software Performance Engineering* (SPE) [52] approach; both response time and throughput need to be considered, but maximizing throughput (transactions per second) is emphasized over minimizing response time. What really counts is throughput with respect to groups of transactions, which is what customers measure and care about. It is, however, very important to notice that optimizing throughput on an individual transaction level can actually reduce overall throughput. The reason is simple and relies on the relation between *latency* and parallelism. The response time of a transaction can be defined in the following way:

$$T_{response} = T_{latency} + T_{processing}$$

Latency is the "idle" time elapsed between the time of a request being issued and the start of the actual processing. The throughput is at its top if latency is at its bottom if each transaction is considered respectively. This means that latency should be minimized if processing is strictly sequential. However, longer latency at the individual transaction level is acceptable if it facilitates parallelism among transactions. When managing several transactions in parallel, internal queues are introduced and with that longer latency. This means that it might be beneficial for overall throughput to purposely slow down single transactions by the introduction of internal queues. If the response time of an individual transaction is set to its maximum limit it will also allow for maximum latency regarding the transaction in scope. Maximum latency means maximum number of parallel transactions and thus maximum overall throughput. Hence, response time can be regarded as an upper limit for what is acceptable for a single transaction, and maximizing capacity is then all about maximizing the number of transactions within the response time limit. In other words, response time contributes to "good-enough service level" and throughput to "large-enough number of subscribers".

Space (memory) has, as is also the case in SPE [52], deliberately been left aside in this view since it to a great extent can be regarded as a direct function of the dimensioning requirements. However, it still needs to be considered when making capacity trade-offs as described in Paper VI.

The dimensionless unit *Erlang* is often used to describe the volume of telecommunications traffic [19]. Traditionally the goal has been to minimize system cost while achieving the maximum quality of service and profit.

2.4.3 MODELING CAPACITY

Research questions Q_4 and Q_5 of section 1.2 are aimed towards modeling in general and UML modeling in particular. They respond to the wish to remedy the established requirements traceability failure by accomplishing a full chain of capacity refinement in models. The principal source of inspiration to this approach was the work by Dimitrov et al. [14], in which several constructions of annotating UML diagrams with performance information were suggested and compared.

The release of the *UML Profile for Schedulability, performance, and time* (SPT) [45] launched a new platform for suggestions regarding performance and capacity. Unlike the requirements engineering perspective of this research project, most of the work that has been done on the modeling of NFRs and capacity/performance originates from within the modeling community. However, Bernardi and Petriu [4] have compared SPT and its annotation techniques with the NFR annotations of the broader *UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms* (QoS) [44].

Model-based performance prediction in general is surveyed by Balsamo et al. [2]. One of the problems that have received a lot of interest in the new millennium is how formal performance models can be derived from UML design. Bernardi and Merseguer [3] have compiled the most important works in the field and contribute with a method for how to transform UML design into a Stochastic Well-formed Net (SWN) performance analysis model from which metrics such as response time and throughput can be calculated.

Cortellessa et al. [9] observe that available methodologies tend to take a transformational approach to create performance models from software/hardware models and how to automate these (the SWN approach above). In their own approach they propose a framework to integrate a software model with a hardware model from which a

performance model can be built. The integrated model is annotated with performance data such as operational profiles and resources as an intermediate step towards the performance model. They also make the following interesting statement when describing the framework:

> The idea of integrating a software model with platform specifications for performance validation goals can be accepted by software developers only if the integration does not bring changes to their development practices.

The attitude imposed by this quote is totally conformant with the attitude to change that is embraced in this thesis: to improve capacity with minimal interference of current procedures. The means differ, however. It becomes evident from the examples above that there is a lot of work done on the modeling, analysis, and prediction of capacity/performance. The major problem that is addressed herein is not only to investigate how capacity can be modeled, analyzed, and predicted. Instead, the focus is on how to improve the development for capacity, the treatment of capacity requirements, and how to make them available for architects, designers, programmers, and testers to consider. Thus, the ability to predict capacity is important but only one part of the problem treated herein.

The attitude of this thesis – that new procedures are more easily adopted if changes to current procedures are minimal rather than extensive – does not mean that a paradigm shift to for example model-based architecture can never happen. However, it is likely to believe that risks are bigger when implementing radical changes compared to achieving change via many small steps. Nevertheless, situations may occur when this risk is acceptable in relation to the time it takes to perform "many small steps".

2.4.4 REQUIREMENTS AND ARCHITECTURE

The overall objective with the improved modeling of capacity is for requirements to influence the choice of architecture, design alternatives, and implementation solutions (even though an existing architecture is often present in an Ericsson context). The relation between requirements and architecture has been described in the following way [43]:

> Compelling economic arguments justify why an early understanding of stakeholders' requirements leads to systems that satisfy their expectations. Equally compelling arguments justify an early understanding and construction of software-system architecture to provide a basis for discovering further requirements and constraints, evaluating a system's technical feasibility, and determining alternative design solutions.

A set of selected requirements may reduce the set of candidate architectures, and respective candidate architecture may prevent specific requirements from being implemented. The "Twin Peaks" model offers a trade-off between requirements and architecture as depicted in Figure 4 below:

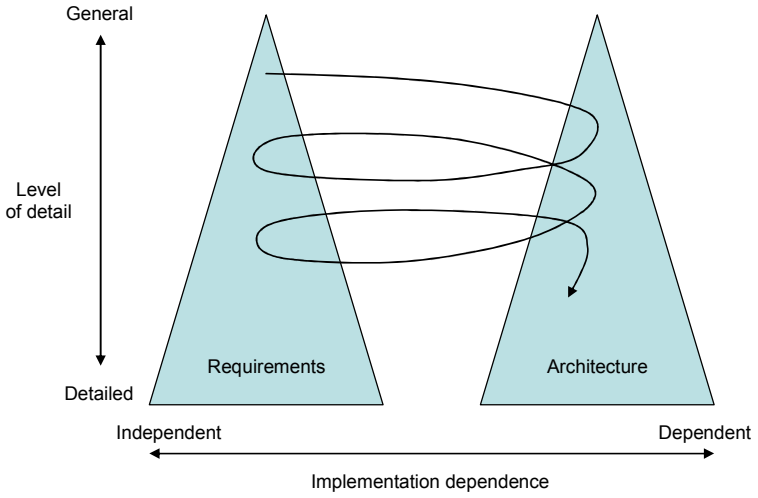

Figure 4: The Twin Peaks model as depicted by Nuseibeh [43]

The Twin Peaks model illustrates that an iterative spiral model can be extended to involve both requirements and architecture, in order to incorporate both early requirements and architectural thinking into the system development process.

Grünbacher et al. propose a refined method that builds on the Twin Peaks model which has been tested in a real-life case-study [23]. The approach introduces a third peak between the twin peaks of Figure 4 as an intermediary representation that is generated from the requirements. A knowledge base with mappings from the intermediary representation to architectural styles guides the designer in finding the most appropriate alternatives for system design.

2.5 Processes and process improvement

2.5.1 EPF AND OPENUP

The *Eclipse Process Framework* (EPF) [15] is an open framework in which a process engineer can compose and reconfigure process descriptions that is similar to the commercial tool *Rational Model Composer* (RMC) that is used in Ericsson today:

> The Eclipse Process Framework (EPF) aims at producing a customizable software process engineering framework, with exemplary process content and tools, supporting a broad variety of project types and development styles.[7]

Moreover, the process framework project is stated to have the following goals:

- To provide an extensible framework and exemplary tools for software process engineering - method and process authoring, library management, configuring and publishing a process.
- To provide exemplary and extensible process content for a range of software development and management processes supporting iterative, agile, and incremental development, and applicable to a broad set of development platforms and applications.

EPF can be downloaded with method libraries for the OpenUP, XP, and Scrum processes, and there is also a method library available for the OpenUP contents packaged as independent practices. Papers III and IV describe how the findings from the interview series of Papers II have been represented as a method plug-in to the OpenUP/Basic 0.9 in order to make it possible to communicate results within Ericsson as well as with other researchers and industrial practitioners.

OpenUP/Basic is, as the name implies, the most basic version of OpenUP. The process is simply referred to as OpenUP in later versions 1.0 and 1.5, to contrast with the currently available[8] plug-ins OpenUP/DSDM (Dynamic Systems Development Method) and OpenUP/ABRD (Agile Business Rule Development). The most important news in OpenUP 1.0 are that contents have been rewritten as a result of field testing reports and that many content areas have been removed to

[7] www.eclipse.org/projects/project_summary.php?projectid=technology.epf
[8] February 11, 2009

make the OpenUP core even more general and extensible. This means that more specific content areas are supposed to be added as plug-ins extending OpenUP, just like the capacity plug-in presented in this thesis. The major news of OpenUP 1.5 is that practices are available and that a first process using these practices has been published.

Regardless of version number, OpenUP is an iterative open-source software development process that is intended to be minimal, although complete, and with extensibility as an important characteristic. It is organized into four major areas of content (Communication and Collaboration, Intent, Solution, and Management), and it is based on four core principles:

- Collaborate to align interests and share understanding
- Balance competing priorities to maximize stakeholder value
- Focus on articulating the architecture
- Evolve to continuously obtain feedback and improve

The most central concepts in OpenUP are *roles*, *tasks* (organized by discipline), *activities* (time oriented sets of tasks), and *artifacts* (organized by domain). A role performs tasks that use, modify, and create artifacts. Each concept has attributes, often consisting of links to other concepts, texts and external files, such as templates and guidelines. The resulting process description is published as a set of web pages with hypertext links viewed from different perspectives. The most dominating perspective is the four phases of iteration: Inception, Elaboration, Construction and Transition. Another perspective is constituted by the disciplines and the process model can also be used to generate project plans, for example Microsoft Project documents.

2.5.2 PROCESS IMPROVEMENT

Both the efforts made in EPF and OpenUP as well as the effort of creating the Anatomy of Capacity Engineering count as process improvements. The CSPs and the CSPAs presented in Paper II are transformed into ACE in Paper V and both contents and terminology has a flavor of CMMI.

The *Capability Maturity Model Integration* (CMMI), and before that the *Capability Maturity Model* (CMM), is a well-known and influential process improvement model. A CMMI model describes the characteristics of good processes, how appraisal can be conducted within an organization and what can be done to improve to a higher level. Taking CMMI for

development as an example [50], it contains 22 *process areas* (such as configuration management, risk management, project planning, etc.). CMMI distinguishes between *continuous representation* and *staged representation;* the former is described by six *capability* levels and the latter by five *maturity* levels as depicted in Table 2 below:

Table 2: Comparison of Capability and Maturity Levels by the SEI [50]

Level	Continuous Representation Capability Levels	Staged Representation Maturity Levels
Level 0	Incomplete	N/A
Level 1	Performed	Initial
Level 2	Managed	Managed
Level 3	Defined	Defined
Level 4	Quantitatively Managed	Quantitatively Managed
Level 5	Optimizing	Optimizing

The capability levels are the means for improving the processes within a single process area whereas the process improvement achievements across several process areas are reflected by the maturity level. The levels 2 through 5 have intentionally identical names (if all process areas reach the "managed" capability level this is also the maturity level).

The capacity process improvement model and the anatomy presented herein are not intended to also be maturity models. However, there are two ways of relating the capacity process improvement model to CMMI. First, the presented CSPAs correspond well to the process area concept of CMMI and it would be possible to apply capability levels to the CSPAs. To achieve a likewise corresponding maturity model, maturity levels would need to be defined across the CSPAs which have not been done. Instead, the process improvement model has been further detailed by making the relations between CSPs explicit in an anatomy. The other way to relate the capacity process improvement model to CMMI is to actually use it together with CMMI to aid in process improvement efforts when climbing the capability and maturity levels. In such a perspective, different parts of the anatomy would correspond to the CMMI maturity levels 2, 3, and 4 respectively.

2.5.3 THE ANATOMY CONCEPT

The use of anatomies stem from changing the procedures to utilize integration and verification resources more efficiently. In traditional system

development test crews and test facilities will have little to work with until a system is designed and implemented. A remedy is to let integration drive development, which means that the question "What can we give to the testers tomorrow?" is in focus from day one. To be able to answer this question – how testers can be fully occupied all the time – some kind of map is needed that states what is dependent of (or even requires) something else. This map is the *anatomy*, a body of many parts represented with boxes and arrows. The anatomy shows where to start the development and it has the positive side effect that the most important system parts have been tested many times when the system is finally released. Thus, the anatomy supports an evolutionary approach to system development

According to Taxén and Lilliesköld [54] the anatomy approach has been used successfully in Ericsson development practice in more than 250 projects since the early 1990s, and they describe the anatomy approach as below:

> The system anatomy, or "anatomy" for short, is a comprehensive picture – on one page – of how the system is working. It shows the functional dependencies in the system from start-up to an operational system. The gist of the anatomy approach is to develop, integrate and verify the system in the same order as it "comes alive". In order to do so, two types of plans are defined based on the anatomy: the increment plan and the integration plan. The increment plan structures the development work in verifiable steps – increments – that can be successively integrated. The integration plan shows sub-project responsibilities and dates for deliveries of increments.

The evolutionary aspects of anatomies are further discussed by Jönsson [31], who distinguishes between *product* anatomy and *project* anatomy. A product anatomy represents an entire system that is going to evolve over several releases and a project anatomy is a subset of a product anatomy that describes the contents of one particular project that will lead to one particular release of the system.

Anatomies are currently used within Ericsson not only to represent system structure but also methodology and organizations' abilities. The notion of an anatomy in this thesis is the same as this notion within Ericsson. The anatomy of Paper V has been constructed to comply with – in structure and shape – an anatomy describing the capabilities needed when developing systems. This general anatomy is called *Anatomy of Excellent Development* (AED) and one of its specified capabilities is to know the system's characteristics. The capacity anatomy described in Paper V responds to this particular need with respect to capacity.

3 Discussion

This chapter presents my view of some of the issues described in the thesis. Since the discussion is based on material presented in the papers it is recommended to read those – or at least Paper VI – prior to this chapter.

3.1 On the acquisition of empirical data

3.1.1 Background

The empirical grounding of this thesis is the data acquired in two interview series (Paper I and Paper II) and the focus group described in Paper VI. The first interview series involved 14 interviewees from two different organizations and the second interview series involved 17 interviewees. In addition, it is described in Paper IV that one of the organizations that participated in the second interview series was revisited, and five new practitioners contributed in a discussion. Finally, the focus group described in Paper VI involved another four practitioners, which means that 40 industrial practitioners have contributed to the empirical data.

Did the interviews provide any new knowledge? If so, would half the number of people have been enough to gain this knowledge and/or would we have learnt even more if 100 industrial practitioners had been interviewed? A large number of interviewees will enable the possibility to perform quantitative (statistical) analyses on the interview material so that statements of the type "22 percent of the respondents think that …" can

be made, whereas each respondent has a greater impact on the result if there is only a few of them. It is also possible that a single interviewee could have provided all of the information in the interview series results if this person has enough experience and insight. However, deducing the results from several or many interviewees is a considerably stronger result. To be able to respond to the above questions in more detail we have to recapitulate what was gained from the interview series.

3.1.2 THE INTERVIEW STUDY ON NFR-RELATED DIFFICULTIES

The results of the interview series on difficulties related to NFRs were to a great extent conformant with our expectations and with existing literature. This means that the results did not pinpoint any new and never reported NFR-related problems, but provided empirical evidence to support commonly accepted knowledge regarding difficulties related to NFRs. The premier contribution of the interview series was in that respect the empirical evidence as such, underpinning the commonly accepted knowledge with real facts. This is essential and the foundation of what research is about – to build theory from what we actually know, not what we think or think we know. This attitude is sententiously formulated in the famous quote "*In God we trust, all others bring data*" usually referred to William Edwards Deming.

Besides providing empirical data regarding NFR-related difficulties, the interview study revealed an important source of difficulties that has significantly influenced the perspective of the research effort: the socio-economical aspect of company organization. It became very clear when interviewing practitioners from one hierarchical organization and from one less hierarchical that NFR difficulties relate to organizational structures. Hence, to successfully manage NFRs it is required that competence, interest, and authority is present on a high-enough level in hierarchically structured organizations to make NFRs influence all system parts. This also means that the decisions regarding NFRs on a high-enough hierarchical level needs to be propagated to all relevant sub systems to gain the needed impact.

Interviewing people from two organizations – two different cases – was necessary in order to highlight the differences in organizational structures. The number of interviewees, seven in each case, was sufficient to gain reasonable diversity in responses. How interviewees are selected is also important and analyzing results is very different if a homogeneous group of interviewees provides very heterogeneous results compared to a

heterogeneous group of interviewees that provides homogeneous results. This is further discussed in Section 3.1.5.

3.1.3 THE INTERVIEW SERIES ON CAPACITY

We learned more from performing two interview series than only one, since the second series had a shift of focus compared to the initial series: The interest was narrowed from NFRs in general to the specific quality factor capacity. Still, the capacity interview series provided new insights regarding one of the findings from the NFR interview series. It was stated in the findings from the NFR series that one of the most tangible difficulties is that NFRs are not discovered, which was true on that level of abstraction. However, it is interesting to note that moving on to reach a more profound description of this difficulty provided new findings and made further research interesting. The preparations and realization of the capacity series led to the reformulation of this statement of non-discovered NFRs: Overall capacity requirements are known and the principal difficulty lies in making these requirements cause the intended impact on architecture, design, implementation, and test. Thus, the reason that capacity requirements appear to be non-discovered is that capacity requirements are not adequately refined from overall requirements to refined requirements, design decisions, implementation solutions, test cases, etc. It is likely to believe that a more focused treatment of other difficulties identified in the NFR interview series can give rise to similar enrichments of findings on more detailed levels.

17 practitioners were interviewed in the capacity series, which was sufficient to identify several good practices and to provide the required information regarding how capacity issues are tackled. The number was also sufficient to serve as the foundation for the methodological context for how to develop for capacity. The presence of expertise regarding Ericsson development made it easier to analyze the interview material and to compare and relate statements from different interviewees. On the other hand, it can be argued that an external analyst would have been better suited to analyze the results and come up with a general solution that is less targeted towards an Ericsson context. This is addressed in the focus group that is described as part of Paper VI.

It can be noticed from the results that the intersection between interview results is relatively small: Only two CSPs are applied in all the organizations that participated in the interview series. Moreover, additional seven CSPs are applied by some organizations, which means that around

half of the CSPs were not in practice in the participating organizations at the time of the study. However, several CSPs were discussed with interviewees although not yet in practice, especially those related to modeling and how to represent and refine capacity in models. The remaining CSPs were added during analysis in order to form the complete methodological context for how to develop for capacity.

3.1.4 REVISITING THE METHOD HYPOTHESIS

It was hypothesized in Section 1.2 that "*it is possible to learn, improve, feed back, and evaluate knowledge regarding NFR/capacity management in large, developing, and administering organizations by the means of industrial case studies.*" The discussion presented above shows that it was possible to learn which difficulties related to NFRs that occur in industrial contexts and how capacity issues are tackled. The captured empirical findings were analyzed and a suggestion of improved procedures was made when CSPs were compiled in the methodological context for how to develop for capacity. The CSPs and their context were evaluated and fed back when revisiting one of the participating organizations to present the CSPs and the method plug-in. This evaluation also provided useful input to the later work with the capacity anatomy.

All in all, the above means that the method hypothesis has been valid and useful and that all its parts have been successfully conducted in the research project. However, the method depends on the interviewees and their abilities to reason about capacity on an abstraction level beyond product-specific technicalities. Thus, the biggest threats to a case study based approach as this are that findings are too scattered and/or too product-specific. The latter implies the former, but findings can be scattered on a higher abstraction level as well; if each interviewee brings up new issues that are not relevant to the other interviewees there are no good chances to produce any useful findings.

3.1.5 THREATS TO VALIDITY

The previous section calls for a more comprehensive treatment of potential validity threats. Wohlin et al. [59] describe four categories of validity in the context of software engineering experiments (originally presented by Cook and Campbell [8]): Conclusion, internal, construct, and external. These categories and the most relevant validity threats within each category are presented below.

Conclusion validity addresses the relationship between the treatment of an experiment and its outcome and is sometimes referred to as *statistical* conclusion validity. This type of validity is threatened if there are hindrances to drawing the correct conclusions, for instance if wrong type of statistical tests are chosen. The statistical part of these threats is not directly applicable to the qualitative research presented in this thesis, but other parts are:

- **Fishing and the error rate.** This is applicable in all interview settings that are not strictly structured and with that free from discussion. The researcher may be looking for a certain type of result and is thus more likely to find that result. However, the opposite happened during the capacity interview series. We set out to improve capacity treatment by the means of UML, but the interviews showed that a much larger context needs to be considered.
- **Reliability of measures.** For instance poor wordings in questions and subjective measurements. In contrast to the above threat, this threat was mitigated by semi- or unstructured interviews since misunderstandings due to poor wordings can be corrected immediately by the interviewer. The risk of subjective measurements was addressed by involving an Ericsson expert that did not participate in the interview sessions.
- **Random heterogeneity of subjects.** If the interviewees are very heterogeneous it will affect the conclusion validity negatively, since variations based on individual differences can be larger than variations due to the treatment. On the other hand, very homogeneous interviewees will reduce the external validity as described below. The interviewees of the first interview series were selected to cover several aspects of software development, and the interviewees of the second interview series were selected since they had been working with capacity improvements lately.

Conclusion validity is achieved if a statistical relationship with a given significance between treatment and outcome can be observed. *Internal validity* is concerned with causality; that the outcome is actually caused by the treatment. The threat is that the outcome can be caused by a factor beyond the researcher's knowledge or control. The most relevant internal validity threats in the present context are the following:

- **Instrumentation.** This is connected to the reliability of measures as described above and is concerned with the quality of the artifacts used in the experiments. This means primarily the interview questions and protocols in this thesis. The interview questions have already been commented upon and the interview protocols of the capacity interview series were sent to all interviewees for confirmation.
- **Selection.** The results of an experiment can vary based on who participates, which is applicable in the interview series. It has already been mentioned that the interviewees of the first interview series were selected to cover several aspects of software development, and the interviewees of the second interview series were selected since they had been working with capacity improvements lately.
- **Interactions with selection.** There is always a risk that interviewees are biased with the thoughts – or presumed thoughts – of the researcher. Semi-structured interviews have their drawbacks and advantages, but they offer a useful trade-off between the threat of poor wordings in questions and the threat of interacting too much with interviewees. The prepared questions drive the discussion and reduce the risk of biasing interviewees.

Construct validity is concerned with the generalization of experiment results to theory and concepts. The most relevant threats in this context are:

- **Inadequate preoperational explication of constructs.** That constructs are not sufficiently defined before used in experiments. This is not a major issue in explorative interviews, even though different notions of capacity constituted a potential threat. However, it is relatively easy to detect and discuss this type of matters in a face-to-face interview setting, and we believe the threat did not cause any harm to the data.
- **Hypothesis guessing.** Interviewees may try to guess the hypotheses and intentions of the researcher and act or respond accordingly. This is connected to "Fishing …" above, and – once again – the results from the capacity interview series indicate that we succeeded in avoiding this threat. An Ericsson expert taking part of the analysis of interview protocols made it possible to identify symptoms of the threat during analysis; the answers of the interviews could be compared with the expert's knowledge about each organization's situation.

- **Evaluation apprehension.** Interviewees may feel that they are being graded or assessed and try to look better than they actually are. The main decision taken to avoid this threat in the capacity interview series was to not involve Ericsson experts in the interviewer's role. We believe that the answers were more open-hearted with me as the only interviewer than would have been the case otherwise, which were considered outweighing the drawbacks of being only one interviewer.
- **Experimenter expectancies.** Relates to "Fishing …" above and is concerned with the fact that the researcher may have expectations on results and bias the analysis – consciously or unconsciously.

Finally, the *external validity* is concerned with how general the results are and if they can be transferred to industrial practice. There are three types of interactions to be considered:

- **Interaction of selection and treatment.** This means to select the right people in the sense that they constitute a representative sample of the population that we want to generalize. As has been described, interviewees were selected to be as representative as possible.
- **Interaction of setting and treatment.** This means to make use of the right material and tools to make results transferrable to industrial practice. Since the interview series were explorative this was not a big issue. However, it can be discussed to what extent results are applicable to domains outside telecommunication systems. The focus group of Paper VI is a direct response to this.
- **Interaction of history and treatment.** This means to consider the time of an experiment so that results are not affected. Wohlin et al. [59] gives the example of conducting a questionnaire on safety-critical systems only a few days after a major software-related crash. However, this was used to our advantage in the capacity interview series since it was possible to select interviewees that had been involved in recently conducted capacity improvement projects.

3.2 From refinement to process improvement

3.2.1 THE INTENTIONS BEHIND THE CAPACITY RESEARCH EFFORT

It was shown in the NFR interview study that NFRs are not always present when needed during system development. It was hypothesized when planning the research effort on capacity that the major reason for this is that capacity requirements are not always refined to design and implementation. Moreover, it was also hypothesized that it is possible to annotate UML models with capacity information to support the refinement of capacity requirements to be available whenever needed. Since Ericsson development is use case-oriented to a great extent it is appealing to also make capacity concerns part of the use cases and the underlying model.

Based on the above hypotheses, the capacity interview series included questions explicitly dedicated to the use of modeling and how capacity issues can be tackled by the means of modeling.

3.2.2 THE NEED OF A LARGER CONTEXT

During interviews, discussions, and analysis of interview material it became evident that the task of improving the representation and refinement of capacity requirements in UML models cannot be separated from the context where new procedures shall be introduced. A trivial example is that UML must be used throughout the system development process for a UML approach to make impact in all its parts. Some organizations within the Ericsson context would be able to adopt such an approach as an extension of current procedures, whereas most organizations currently do not use modeling to an extent that short-term benefits from such an approach can be gained. However, if a developing organization uses modeling to some extent and intends to move towards model-based development, an approach that builds on UML modeling can act as a vehicle to support that intention. The experience from the capacity interview series shows that three out of ten represented organizations use modeling to an extent that is fairly close to being model-based; use case diagrams, sequence diagrams, and class diagrams are the most frequent diagrammatic representations in these cases and they also make good use of code generation from models. Four out of ten use only basic modeling for some aspects of development (typically use case diagrams) and the remaining three are somewhere in between.

The conclusion drawn from the above is that it is not feasible just to present an approach for how to refine capacity into various design and implementation models. The reason is that the abilities to manage capacity requirements and to develop systems with respect to capacity require a larger context to be considered. Of course, an approach to refine capacity requirements to design and implementation that is complete, consistent, correct, useful, etc. is desirable and would be a welcome scientific contribution. However, if the primary goal of the research is to improve current procedures and understand what capacity development is all about, then optimizing an isolated part of the development process may cause sub-optimization of limited value to the overall goal. In other words, it is better to improve all development stages before improvements at a detailed level of optimization are feasible and useful. This is the core idea of maturity models like CMMI; it is of limited value to improve in one area unless related areas are mature enough to benefit from improvements.

The overall system development within Ericsson is certainly mature enough to motivate focus in a single area like capacity. However, the lessons learned from the interviews on capacity are that capacity improvements address a wide range of issues and that improved refinement of capacity in models belongs to a larger context. This context must be understood before an organization can fully benefit from improvements in refinement and specification of capacity requirements. Thus, the focus shifted from specificational concerns only to the overall ability to develop for capacity (of which refinement in models is one part). The primary objective became to describe what is needed to be able to develop for capacity and how to improve an organization's abilities. With this shift of perspective we allow different organizations to improve in different ways. Some organizations are well suited to improve the specification of capacity requirements whereas other organization may be better suited to start with improving their abilities to verify capacity requirements.

3.2.3 SOCIO-ECONOMICAL ASPECTS

Capacity has to be implemented as a result of many trade-offs and the socio-economic perspective is highly present. It *does* matter how a developing organization is structured, where decisions are made, by whom, etc. The example of specificational or verificational approach builds on the two main types of attitudes that can be distinguished from the interview series: The attitude that better specification and refinement is a feasible

approach to the improvement of capacity processes, and the attitude that the best way to improve is to enable short "feedback loops" from testers to developers so that developers get real data to react upon.

I strongly believe, with support in the interview series, that it is not specification *or* verification. Instead, the overall context must be considered and support must be provided for organizations to prioritize the most needed improvement actions. Capacity in large-scale telecommunication systems is achieved in a shared environment; organizational aspects and where competence, interest, and authority regarding capacity are located play important roles.

The capacity interview series point out many actions of improvement that can be implemented, but few organizations have the time and budget to implement all actions at once. Thus, the CSPs developed from the interview series must be accompanied by a method for how to assess an organization's capacity abilities and to recommend the vital few CSPs that contribute the most to overall improvement. This was the driving objective behind the research described in Papers III-V.

3.3 Revisiting the research questions

The method hypothesis has already been revisited and stated valid in section 3.1.4. This section revisits the research questions, one by one, starting with the research questions labeled Q_1 through Q_4 in section 1.2 before the overall research question (Q) is summarized and discussed.

3.3.1 Q_1: The management of NFRs

The first interview series (see Paper I) presents and discusses the management of NFRs and related difficulties at Ericsson OSS and SMHI. Even though two organizations constitute only a small sample, it is evident from the study that the management of NFRs and related difficulties are well conformant with common knowledge and what is stated in the literature. Their vague nature makes NFRs harder to incorporate in methods and tools, which in turn makes FRs focused in everyday system development. The management of NFRs as perceived in Paper I is briefly summarized and commented upon below.

The *elicitation* of NFRs is a vital step in order to build quality into a system. Undiscovered NFRs cause an incomplete requirements set, which may affect system architecture negatively and with that result in poor runtime properties. There are in the general case no specific activities

dedicated to the elicitation of NFRs. Instead, the development process trusts the organizations' abilities to build good-enough systems and to cope with NFRs as they occur (which they do as consequences of FRs or as requirements from related systems in the Ericsson OSS case), even if some NFRs are elicited too late for them to influence the architecture. The improvements proposed by interviewees are to focus NFRs at an early stage and SMHI respondents stress the benefits of involving the customer/user (which is usually not possible in the context of Ericsson OSS). The use of NFR checklists that are applied to all FRs has been suggested in the literature [11, 48], and the approach can be complemented with prototyping, scenarios and other elicitation techniques [21]. If using the NFR Framework or similar approaches, refining overall NFRs is an elicitation process in itself.

The *analysis and specification* of NFRs is where the consequences and impact of NFRs are investigated, estimations are made, and where the NFRs are represented as testable requirements. It is considered tedious and difficult to quantify NFRs, despite the fact that metrics are available for any kind of NFRs [18]. The main remedies proposed by interviewees are the well-known advice to involve testers when analyzing and specifying the NFRs to ensure their testability, to use prototyping techniques, and to create theoretical models to analyze what is needed for an NFR to be considered fulfilled. The latter means – in a performance context – to apply techniques such as SPE [52] to ensure that needed performance will be possible to achieve. It was also suggested to relate NFRs to design patterns to know in advance to which extent various NFRs will be fulfilled if a certain design pattern is chosen. Again, this type of reasoning is well conformant with the NFR Framework and other goal-oriented methods since these specifically address the fulfillment of non-functional goals (that may be in conflict with each other).

The *verification* of NFRs generally follows the same procedures as the verification of FRs. What is in the requirements specifications gets verified, which means that the specification of requirements is crucial for verification as well as for design. Ericsson OSS sometimes uses a *statement of compliance* (SOC) document to "SOC away" requirements that are not fulfilled in the tested system version. Some respondents think that NFRs are disregarded this way more often than FRs, whereas other respondents claim the opposite: A required feature that is not there has to be admitted in the SOC, but NFRs are often vaguely stated which makes it easier to argue that they have been met in some interpretation. Both Ericsson and SMHI, however, find it relatively easy to measure and verify performance

requirements, and they also agree that the testability differs a lot among NFR types. Many NFRs – such as availability, usability, and security requirements – are tedious and difficult to verify even though they are well specified. An example is the availability requirement that "The down-time of the system may not exceed three hours in a year", which is a quantified requirement that still is difficult to verify before releasing the system. Finally, Ericsson and SMHI respondents also agree that the most important improvement with respect to NFR verification is the sufficient specification and quantification of NFRs: Vaguely stated NFRs are difficult to test.

The overall conclusion of how NFRs are managed is that FRs is the primary focus and that NFRs receive no more interest than is needed. Reasons are both the vague nature of NFRs and that industrial methods are generally better suited for FRs than NFRs (which may also be because of the vague nature of NFRs). However, the interviewees show that they understand NFR related difficulties and they present remedies to the most important of those difficulties. It is reasonable to believe that a lot of improvement could be achieved if NFRs were paid more attention. Workshops dedicated to the elicitation, analysis, and specification of NFRs constitute a simple remedy that can be accompanied with guidelines for how to quantify NFRs and refine them to design and implementation.

The organizational structure may be an obstacle for dealing with NFRs. A hierarchically structured organization needs to consider NFRs on a high-enough hierarchical level in order for NFRs to be reflected in architecture, design, etc. and for NFRs to be distributed over sub-systems and design units. This means that NFR interest, authority and competence must be present at a high enough level, regardless of designers' skills. Even if the designers of a design unit are very skilled, they do not possess the required overview. This means that two things must apply for the successful treatment of NFRs in large and hierarchically structured organizations:

1. There must be enough competence and interest regarding NFRs on a system management level.
2. System level NFRs need to be identified, analyzed, and refined to be represented in architecture, design, implementation, and test.

The subsequent investigations on capacity conclude that the refinement of capacity requirements from system level to sub systems and design units is not straight-forward. This complexity is discussed in the capacity context when revisiting Q_4 in Section 3.3.4, but its findings apply to NFRs in

general: The process of refining NFRs is iterative and involves numerous design decisions on various levels of detail.

Finally, the primary target of a software system vendor is to make money. Thus, investing time and effort in improving NFR management needs to be the result of an ROI-calculation that analyzes improved quality against increased cost. Some of the findings herein, especially the anatomy approach presented in Paper V, lower the investment threshold of capacity improvements.

3.3.2 Q_2: THE MANAGEMENT OF CAPACITY REQUIREMENTS

The second interview series revealed how capacity requirements are treated within Ericsson. Capacity requirements are generally not paid extra attention unless overall capacity requirements seem to be challenged.

An interesting finding arose from one of the organizations that participated in the interview series. The organization in scope takes care of everything that is not included in use cases using so called *base scenarios* that are specified in a base scenario specification (which all requirements that relate to characteristics refer to). Around 15 base scenarios are created for the system in scope and these are split into around 100 *event scenarios* that are associated with a few requirements each. The NFRs that are derived from the combination of FRs and event scenarios are documented in supplementary specifications, and each supplementary specification is based on a single base scenario. Thus, there is one set of NFRs for each base scenario. These scenarios and specifications serve as a basis for requirements as well as testing, but the point made by one of the interviewees – as described in the papers – was that the 1000 pages of characteristics requirements "are more useful to testers than designers", since there is simply too much information to take into consideration. This is a good illustration of the assumed requirement traceability failure of Paper II, even though it is caused by information overflow rather than lack of information in this example.

Even though the hypothesis was supported the interview series also reminded of the socio-economical aspect that was mentioned above. The successful treatment of capacity requirements involves both specification and refinement, but it also involves the ability to predict the capacity of a new system. The circumstance that Ericsson develops a new release of an existing system more often than a system is developed from scratch enables the opportunity to consider the capacity of the current release and how it is used in operation. However, the successful treatment of capacity

requirements also involves the ability to test for capacity and let architects and developers know when required capacity is not going to be reached. Moreover, some improvements regarding capacity can be achieved by optimizing and tuning the system. This means that system capacity is achieved in cooperation between people responsible for requirements and people responsible for testing, which has also relations to the improvement of an existing system version based on measurement and tuning and how to predict system capacity. Thus, many people and various organizational units contribute to the capacity of a forthcoming system and the capacity requirements need to be traded against other requirements of the system.

There is a concluding point to make regarding the requirement that a new release of a system must have at least the same performance as its predecessor. This requirement stands to reason: People do not expect to perceive a new version of something as slower as and less capable than the previous one. However, it is important to notice that this requirement has to apply to a new release that typically has a number of new features. These features need resources and, unless addressed, more functionality will share the same set of resources and with that lead to a performance loss. Thus, what seems as a modest requirement at first sight – to maintain the acquired performance level – means improved performance in reality.

3.3.3 Q_3: IMPROVING THE CAPACITY ROUTINES

Each one of Papers II-V presents strategies of how to improve overall capacity routines. The foundation is the findings from the capacity interview series; the CSPs which are beneficial when developing for capacity. I believe that any large-scale software developing organization that implements most of the presented CSPs are capable of delivering systems with the right capacity in the right time, even though some of the CSPs need to be further investigated and adapted with respect to the organization's context. However, efforts were made to make the acquired knowledge more accessible and easier to comprehend in Papers III-V.

First, the results were packaged as a method plug-in to the OpenUP process component of the EPF project for process engineers to include in an OpenUP-compatible process. It is possible for the process engineer to customize the process and adapt the capacity contents to adequately reflect the context of the organization in scope.

Second, the results were packaged in the shape of an anatomy which was associated with a corresponding method (ACE). ACE provides support on how to improve capacity procedures in the sense that ACE

includes a step to select the most important CSPs to be included in an improvement effort. Thus, the anatomy shows how CSPs relate to each other and contribute to improved organizational capacity capability, and support is provided for capacity capability assessment as well as identification of the CSPs that are likely to generate the best ROI.

The assessment of an organization's current status with respect to the CSPs of the anatomy is essential. An assessment workshop can be carried out in a few hours and the discussion as such brings capacity issues in focus and has the potential to increase the capacity awareness. Moreover, when discussing and assessing CSPs it is probable that workshop participants intuitively feel which CSPs that need to be prioritized.

The support for prioritization of candidate improvement efforts is necessary for practitioners to accept the method, and it was an explicitly stated requirement by the organization of the second interview series that validated the CSPs and the method plug-in.

3.3.4 Q_4: MODELING CAPACITY

It has already been noticed that one of the research goals was to model capacity and to benefit from such models in order to mitigate the "requirements traceability failure" described in Paper II. A pragmatic approach to represent capacity information in UML is described in Paper VI along with the thoughts behind it. The objective of the approach is to enter capacity information into a model (important, not only a set of diagrams) and to show how overall capacity requirements need to be reflected in diagrams. This requires a strategy for how to use regular UML constructs and how to make capacity information visible in diagrams and traceable in models. We suggest a `<<capacity>>` stereotype with associated tagged values for the latter and Figure 5 shows how this stereotype extends the UML metaclass "Class".

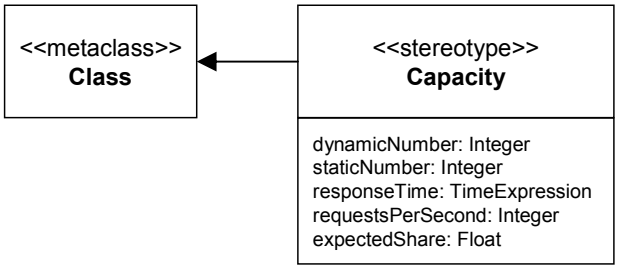

Figure 5: The `<<capacity>>` stereotype

Dimensioning and throughput requirements are represented with plain integers, whereas response time is represented using `TimeExpression` from the `SimpleTime` package of UML. In addition, the `expectedShare` tag has been included in the stereotype to enable the possibility to model operational profiles, primarily in the context of state machine diagrams and activity diagrams. The circumstance at Ericsson – and in many other developing organizations – that development usually starts from an existing system version improves the chances to feed the `expectedShare` tag with reasonable figures.

The instance specification of the stereotype is depicted in Figure 6 below:

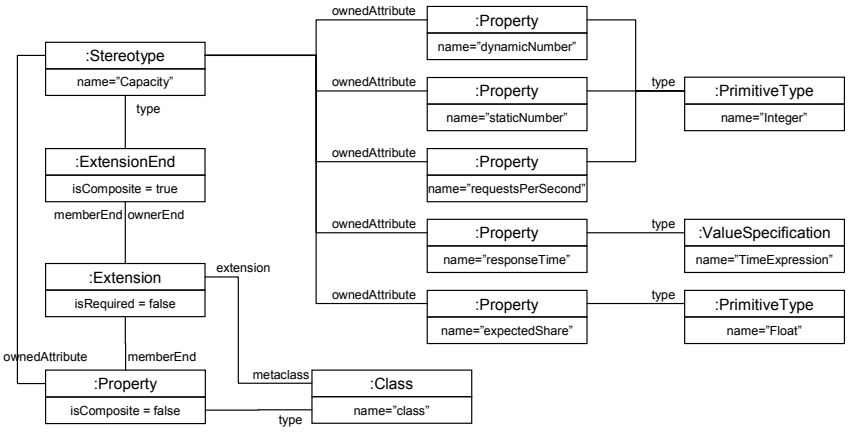

Figure 6: Instance specification of the `<<capacity>>` stereotype

The `<<capacity>>` stereotype can be applied to any UML element which enables a pragmatic and straight-forward approach to representing capacity: Capacity aspects are separated from functional requirements and are represented as requirements of their own. Capacity figures are represented as tagged values that can be further refined in later stages, for instance time budgets in sequence diagrams and defined multiplicity in class diagrams.

What really motivates a UML approach to representing and refining capacity requirements is if it can be shown that capacity requirements actually influence the development to a sufficient extent (by being present at all levels when needed). However, it is also necessary to acknowledge that it is not only "refinement" it is all about. The process of achieving "refinement" is iterative and involves complex design decision making that

needs to conform to the current deployment structure. Each iteration of refinement includes design choices that constrain the further refinement of requirements etc. For example, describing the distribution of time budgets in sequence diagrams requires decisions regarding components, classes, and objects. This means that the requirements analyst must perform his/her work in close cooperation with architects and designers and that conscious design decisions and traceability to the capacity requirements is a better description of reality than "refinement".

3.3.5 Q: THE OVERALL RESEARCH QUESTION

Recall the overall objective of this research project:

Q How can capacity requirements be treated so that they are available when needed and influence all phases of large-scale software system development?

The response to this question involves parts from all papers in the thesis and is best described by the integrated method presented in Paper VI, which also serve as a summary of the thesis as a whole. The method [36] consists of the following elements:

- Capacity requirements as the method's underlying model,
- EPF and UML as applied languages,
- the method plug-in as process model, and
- ACE as the method's guidance.

It is my opinion that the integrated method contains the knowledge needed to initiate an improvement program within the domain of large-scale software engineering. I also believe that ACE is a useful vehicle to conduct and customize improvement efforts and it has been shown in a focus group (reported of in a working paper) that the ACE method – as well as the anatomy contents – is sound and transferable from the telecommunications domain to the domain of avionics. However, as with any improvement program there must be an interest from management level and I have repeatedly pointed out that there is a need of competence, interest, and authority regarding capacity (and NFRs in general) to gain full impact of capacity requirements (and NFRs in general).

Is it feasible to initiate a capacity improvement effort, then? Is it possible to "engineer" capacity or is capacity an emergent property of the system that cannot be controlled even if we try? If so, software engineering must be different from for instance mechanics; no engineer would build a bridge without having performed the calculations showing that the construction is strong enough and I do believe that such calculations goes without saying in mature software engineering as well. SPE [52] is a good example of that it is possible and the contents of this thesis place such calculations in a larger context. Thus, capacity engineering is a valid discipline within software engineering, but my impression is that capacity is often treated as being an emergent property (that is addressed when needed) in reality anyhow. However, this impression is neither limited to large-scale software engineering nor capacity: NFRs in general in software engineering are often given the least possible attention and are prioritized only if they are absolutely crucial to product success.

References

[1] Antón, A. I. *Goal Identification and Refinement in the Specification of Software-Based Information Systems,* Doctoral thesis, Georgia Institute of Technology, 1997.

[2] Balsamo, S., A. D. Marco, P. Inverardi and M. Simeoni. "Model-Based Performance Prediction in Software Development: A Survey", *IEEE Transactions on Software Engineering,* Vol. 30, No 5, pp. 295-310, 2004.

[3] Bernardi, S. and J. Merseguer. "Performance evaluation of UML design with Stochastic Well-formed Nets", *Journal of Systems and Software,* Vol. 80, No 11, pp. 1843-1865, 2007.

[4] Bernardi, S. and D. C. Petriu, "Comparing two UML profiles for non-functional requirement annotations", in the proceedings of the *International Workshop on Specification and Validation of UML Models for Real Time and Embedded Systems,* 2004.

[5] Boehm, B., J. R. Brown, H. Kaspar, M. Lipow, G. J. Macleod and M. J. Merrit. *Characteristics of Software Quality*. North-Holland, Amsterdam, 1978.

[6] Carlshamre, P. *A Usability Perspective on Requirements Engineering: From Methodology to Product Development,* Doctoral thesis, Dissertation No 726, Linköping University, 2001.

[7] Chung, L., B. A. Nixon, E. Yu and J. Mylopoulos. *Non-Functional Requirements in Software Engineering.* Kluwer Academic Publishers, Boston, 2000.

[8] Cook, T. D. and D. T. Campbell. *Quasi-Experimentation: Design and Analysis Issues for Field Settings.* Houghton-Mifflin, Boston, 1979.

[9] Cortellessa, V., P. Pierini and D. Rossi, "Integrating Software Models and Platform Models for Performance Analysis", *IEEE Transactions on Software Engineering,* vol. 33, pp. 385-401, 2007.

[10] Cronholm, S. and G. Goldkuhl, "Understanding the practices of action research", in the proceedings of the *European Conference on Research Methods in Business and Management (ECRM 2003),* pp. 81-91, 2003.

[11] Cysneiros, L. M., J. C. S. d. P. Leite and J. d. M. S. Neto. "A Framework for Integrating Non-Functional Requirements into Conceptual Models", *Requirements Engineering Journal,* Vol. 6, No 2, pp. 97-115, 2001.

[12] Darimont, R., E. Delor, P. Massonet and A. van Lamsweerde, "GRAIL/KAOS: An Environment for Goal-Driven Requirements Engineering", *Proceedings of the 1997 (19th) International Conference on Software Engineering,* pp. 612-613, 1997.

[13] Davis, A. M. *Software Requirements: Objects, Functions and States.* Prentice Hall, Upper Saddle River, NJ, 1993.

[14] Dimitrov, E., A. Schmietendorf and R. Dumke. "UML-Based Performance Engineering Possibilities and Techniques", *IEEE Software,* Vol. 19, No 1, pp. 74-83, 2002.

[15] Eclipse Process Framework, http://www.eclipse.org/epf/, Accessed Feb 16, 2009.

[16] Edmunds, H. *The Focus Group Research Handbook.* NTC Business Books, Lincolnwood, 1999.

[17] Ericsson, K. A. and H. A. Simon. *Protocol Analysis: Verbal Reports as Data.* MIT Press, Cambridge, MA, 1993.

[18] Fenton, N. E. and S. L. Pfleeger. *Software Metrics: A Rigorous and Practical Approach.* PWS Publishing Company, Boston, 1997.

[19] Freeman, R. L. *Telecommunication System Engineering.* Fourth ed., Wiley, Hoboken, NJ, 2004.

[20] Glinz, M. "On Non-Functional Requirements", in the proceedings of the *15th IEEE International Requirements Engineering Conference (RE '07),* pp. 21-26, 2007.

[21] Goguen, J. A. and C. Linde, "Techniques for requirements elicitation", in the proceedings of the *1st IEEE International Symposium on Requirements Engineering,* pp. 152-164, 1993.

[22] Goldkuhl, G. and S. Cronholm, "Multi-grounded theory – adding theoretical grounding to grounded theory", in the proceedings of the *Second European Conference on Research Methods in Business and Management (ECRM 2003),* Reading, UK, pp. 177-186, 2003.

[23] Grünbacher, P., A. Egyed and N. Medvidovic. "Reconciling Software Requirements and Architectures with Intermediate Models", *Journal of Software and Systems Modeling*, Vol. 3, No 3, pp. 235-253, 2004.

[24] Harwell, R., E. Aslaksen, I. Hooks, R. Mengot and K. Ptack. "What is a Requirement?", in *Software Requirements Engineering,* IEEE Computer Society, Los Alamitos, CA, 2000.

[25] Hedenskog, Å. *Perceive those things which cannot be seen: A cognitive systems engineering perspective on requirements management,* Doctoral thesis, Dissertation No 1054, Linköping University, 2006.

[26] Hochmüller, E. "Towards the Proper Integration of Extra-Functional Requirements", *The Australian Journal of Information Systems,* Vol. 7, 1999.

[27] IEEE_Std_1061-1998, "IEEE standard for a software quality metrics methodology".

[28] IEEE_Std_610.12-1990, "IEEE standard glossary of software engineering terminology".

[29] IEEE_Std_830-1998. "IEEE Recommended Practice for Software Requirements Specifications", in *Software Requirements Engineering,* IEEE Computer Society, Los Alamitos, CA, 2000.

[30] Jacobson, I., G. Booch and J. Rumbaugh. *The Unified Software Development Process.* Addison-Wesley, Reading, 1999.

[31] Jönsson, P., "The anatomy - an instrument for managing software evolution and evolvability", in the proceedings of the *Proceedings of the Second International IEEE Workshop on Software Evolvability,* Philadelphia, USA, pp. 31-37, 2006.

[32] Kaiya, H., H. Horai and M. Saeki, "AGORA: Attributed goal-oriented requirements analysis method", in the proceedings of the *IEEE Joint International Conference on Requirements Engineering,* Essen, Germany, pp. 13-22, 2002.

[33] Karlsson, L., Å G. Dahlstedt, B. Regnell, J. N. o. Dag and A. Persson. "Requirements engineering challenges in market-driven software development - An interview study with practitioners", *Information and Software Technology,* Vol. 49, No 6, pp. 588-604, 2007.

[34] Keller, S. E., L. G. Kahn and R. B. Panara. "Specifying Software Quality Requirements with Metrics", in *System and Software Requirements Engineering,* IEEE Computer Society Press, Los Alamitos, 1990.

[35] Kotonya, G. and I. Sommerville. *Requirements Engineering: Processes and Techniques.* John Wiley & Sons, Chichester, 1998.

[36] Kronlöf, K., A. Sheehan and M. Hallmann. "The Concept of Method Integration", in *Method integration: concepts and case studies.* Wiley, New York, pp 1-18, 1993.

[37] Kruchten, P. *The Rational Unified Process: An Introduction.* Addison-Wesley, Reading, Massachusetts, 2000.

[38] Lamsweerde, A. v., "Goal-oriented requirements engineering: A guided tour", in the proceedings of the *5th IEEE International Symposium on Requirements Engineering (RE'01),* pp. 249, 2001.

[39] Lindvall, M. *An Empirical Study of Requirements-Driven Impact Analysis in Object-Oriented Software Evolution,* Doctoral thesis, Dissertation No 480, Linköping University, 1997.

[40] McKay, J. and P. Marshall. "The dual imperatives of action research", *Information Technology & People,* Vol. 14, No 1, pp. 46-59, 2001.

[41] Mylopoulos, J., L. Chung and B. A. Nixon. "Representing and Using Nonfunctional Requirements: A Process-Oriented Approach", *IEEE Transactions on Software Engineering,* Vol. 18, No 6, pp. 483-497, 1992.

[42] Ncube, C. *A Requirements Engineering Method for COTS-Based Development,* Doctoral thesis, City University London, UK, 2000.

[43] Nuseibeh, B. "Weaving together requirements and architectures", *Computer,* Vol. 34, No. 3, pp. 115-119, 2001.

[44] OMG, "UML profile for modeling quality of service and fault tolerance characteristics and mechanisms", OMG, Tech. Rep. OMG Adopted Specification, ptc/2004-06-01, 2004.

[45] OMG-UML-SPT. *UML Profile for Schedulability, Performance, and Time.* 2005.

[46] OpenUP Component, http://www.eclipse.org/epf/openup_component/openup_index.php, Accessed Feb 12, 2009.

[47] Potts, C. "Software-Engineering Research Revisited", *IEEE Software*, Vol. 10, No 5, pp. 19-28, 1993.

[48] Robertson, S. and J. Robertson. *Mastering the Requirements Process.* Addison-Wesley, Harlow, 1999.

[49] Glossary of EU SCREEN Project, http://cordis.europa.eu/infowin/acts/rus/projects/screen/glossary/glossary.htm, Accessed Feb 12, 2009.

[50] SEI, "CMMI for development", Software Engineering Institute, Carnegie Mellon, Pittsburgh, USA, Tech. Rep. CMMI-DEV, V1.2, 2006.

[51] Singer, C. A. *A requirements tutorial. Quality systems and software requirements*, Special Report SR-NWT-002159, Bellcore, 1992.

[52] Smith, C. U. and L. G. Williams. *Performance Solutions: A Practical Guide to Creating Responsive, Scalable Software.* Addison Wesley Longman Publishing Co, Redwood City, USA, 2002.

[53] Strauss, A. and J. Corbin. *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory.* Second ed., Sage Publications, Thousand Oaks, California, 1998.

[54] Taxén, L. and J. Lilliesköld, "Manifesting shared affordances in system development - the system anatomy", in the proceedings of the *The 3rd International Conference on Action in Language, Organisations and Information Systems (ALOIS 2005),* Limerick, Ireland, pp. 28-47, 2005.

[55] Thayer, R. H. and M. C. Thayer. "Software Requirements Engineering Glossary", in *Software Requirements Engineering,* IEEE Computer Society, Los Alamitos, CA, 2000.

[56] Wiegers, K. E. *Software Requirements.* Microsoft Press, Redmond, 1999.

[57] Non-Functional Requirements, http://en.wikipedia.org/wiki/Non-functional_requirements, Accessed Feb 12, 2009.

[58] Requirements Analysis, http://en.wikipedia.org/wiki/Requirements_analysis, Accessed Feb 12, 2009.

[59] Wohlin, C., P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell and A. Wesslén. *Experimentation in Software Engineering - an Introduction.* Kluwer Academic Publishers, Boston, 2000.

[60] Yu, E. and J. Mylopoulos, "Why goal-oriented requirements engineering", in the proceedings of the *4th International Workshop on Requirements Engineering: Foundations of Software Quality,* Pisa, Italy, pp. 15-22, 1998.

Department of Computer and Information Science
Linköpings universitet

Dissertations

Linköping Studies in Science and Technology

No 14 **Anders Haraldsson:** A Program Manipulation System Based on Partial Evaluation, 1977, ISBN 91-7372-144-1.

No 17 **Bengt Magnhagen:** Probability Based Verification of Time Margins in Digital Designs, 1977, ISBN 91-7372-157-3.

No 18 **Mats Cedwall:** Semantisk analys av process-beskrivningar i naturligt språk, 1977, ISBN 91-7372-168-9.

No 22 **Jaak Urmi:** A Machine Independent LISP Compiler and its Implications for Ideal Hardware, 1978, ISBN 91-7372-188-3.

No 33 **Tore Risch:** Compilation of Multiple File Queries in a Meta-Database System 1978, ISBN 91-7372-232-4.

No 51 **Erland Jungert:** Synthesizing Database Structures from a User Oriented Data Model, 1980, ISBN 91-7372-387-8.

No 54 **Sture Hägglund:** Contributions to the Development of Methods and Tools for Interactive Design of Applications Software, 1980, ISBN 91-7372-404-1.

No 55 **Pär Emanuelson:** Performance Enhancement in a Well-Structured Pattern Matcher through Partial Evaluation, 1980, ISBN 91-7372-403-3.

No 58 **Bengt Johnsson, Bertil Andersson:** The Human-Computer Interface in Commercial Systems, 1981, ISBN 91-7372-414-9.

No 69 **H. Jan Komorowski:** A Specification of an Abstract Prolog Machine and its Application to Partial Evaluation, 1981, ISBN 91-7372-479-3.

No 71 **René Reboh:** Knowledge Engineering Techniques and Tools for Expert Systems, 1981, ISBN 91-7372-489-0.

No 77 **Östen Oskarsson:** Mechanisms of Modifiability in large Software Systems, 1982, ISBN 91-7372-527-7.

No 94 **Hans Lunell:** Code Generator Writing Systems, 1983, ISBN 91-7372-652-4.

No 97 **Andrzej Lingas:** Advances in Minimum Weight Triangulation, 1983, ISBN 91-7372-660-5.

No 109 **Peter Fritzson:** Towards a Distributed Programming Environment based on Incremental Compilation,1984, ISBN 91-7372-801-2.

No 111 **Erik Tengvald:** The Design of Expert Planning Systems. An Experimental Operations Planning System for Turning, 1984, ISBN 91-7372-805-5.

No 155 **Christos Levcopoulos:** Heuristics for Minimum Decompositions of Polygons, 1987, ISBN 91-7870-133-3.

No 165 **James W. Goodwin:** A Theory and System for Non-Monotonic Reasoning, 1987, ISBN 91-7870-183-X.

No 170 **Zebo Peng:** A Formal Methodology for Automated Synthesis of VLSI Systems, 1987, ISBN 91-7870-225-9.

No 174 **Johan Fagerström:** A Paradigm and System for Design of Distributed Systems, 1988, ISBN 91-7870-301-8.

No 192 **Dimiter Driankov:** Towards a Many Valued Logic of Quantified Belief, 1988, ISBN 91-7870-374-3.

No 213 **Lin Padgham:** Non-Monotonic Inheritance for an Object Oriented Knowledge Base, 1989, ISBN 91-7870-485-5.

No 214 **Tony Larsson:** A Formal Hardware Description and Verification Method, 1989, ISBN 91-7870-517-7.

No 221 **Michael Reinfrank:** Fundamentals and Logical Foundations of Truth Maintenance, 1989, ISBN 91-7870-546-0.

No 239 **Jonas Löwgren:** Knowledge-Based Design Support and Discourse Management in User Interface Management Systems, 1991, ISBN 91-7870-720-X.

No 244 **Henrik Eriksson:** Meta-Tool Support for Knowledge Acquisition, 1991, ISBN 91-7870-746-3.

No 252 **Peter Eklund:** An Epistemic Approach to Interactive Design in Multiple Inheritance Hierarchies,1991, ISBN 91-7870-784-6.

No 258 **Patrick Doherty:** NML3 - A Non-Monotonic Formalism with Explicit Defaults, 1991, ISBN 91-7870-816-8.

No 260 **Nahid Shahmehri:** Generalized Algorithmic Debugging, 1991, ISBN 91-7870-828-1.

No 264 **Nils Dahlbäck:** Representation of Discourse-Cognitive and Computational Aspects, 1992, ISBN 91-7870-850-8.

No 265 **Ulf Nilsson:** Abstract Interpretations and Abstract Machines: Contributions to a Methodology for the Implementation of Logic Programs, 1992, ISBN 91-7870-858-3.

No 270 **Ralph Rönnquist:** Theory and Practice of Tense-bound Object References, 1992, ISBN 91-7870-873-7.

No 273 **Björn Fjellborg:** Pipeline Extraction for VLSI Data Path Synthesis, 1992, ISBN 91-7870-880-X.

No 276 **Staffan Bonnier:** A Formal Basis for Horn Clause Logic with External Polymorphic Functions, 1992, ISBN 91-7870-896-6.

No 277 **Kristian Sandahl:** Developing Knowledge Management Systems with an Active Expert Methodology, 1992, ISBN 91-7870-897-4.

No 281 **Christer Bäckström:** Computational Complexity

of Reasoning about Plans, 1992, ISBN 91-7870-979-2.

No 292 **Mats Wirén:** Studies in Incremental Natural Language Analysis, 1992, ISBN 91-7871-027-8.

No 297 **Mariam Kamkar:** Interprocedural Dynamic Slicing with Applications to Debugging and Testing, 1993, ISBN 91-7871-065-0.

No 302 **Tingting Zhang:** A Study in Diagnosis Using Classification and Defaults, 1993, ISBN 91-7871-078-2.

No 312 **Arne Jönsson:** Dialogue Management for Natural Language Interfaces - An Empirical Approach, 1993, ISBN 91-7871-110-X.

No 338 **Simin Nadjm-Tehrani**: Reactive Systems in Physical Environments: Compositional Modelling and Framework for Verification, 1994, ISBN 91-7871-237-8.

No 371 **Bengt Savén:** Business Models for Decision Support and Learning. A Study of Discrete-Event Manufacturing Simulation at Asea/ABB 1968-1993, 1995, ISBN 91-7871-494-X.

No 375 **Ulf Söderman:** Conceptual Modelling of Mode Switching Physical Systems, 1995, ISBN 91-7871-516-4.

No 383 **Andreas Kågedal:** Exploiting Groundness in Logic Programs, 1995, ISBN 91-7871-538-5.

No 396 **George Fodor:** Ontological Control, Description, Identification and Recovery from Problematic Control Situations, 1995, ISBN 91-7871-603-9.

No 413 **Mikael Pettersson:** Compiling Natural Semantics, 1995, ISBN 91-7871-641-1.

No 414 **Xinli Gu:** RT Level Testability Improvement by Testability Analysis and Transformations, 1996, ISBN 91-7871-654-3.

No 416 **Hua Shu:** Distributed Default Reasoning, 1996, ISBN 91-7871-665-9.

No 429 **Jaime Villegas:** Simulation Supported Industrial Training from an Organisational Learning Perspective - Development and Evaluation of the SSIT Method, 1996, ISBN 91-7871-700-0.

No 431 **Peter Jonsson:** Studies in Action Planning: Algorithms and Complexity, 1996, ISBN 91-7871-704-3.

No 437 **Johan Boye:** Directional Types in Logic Programming, 1996, ISBN 91-7871-725-6.

No 439 **Cecilia Sjöberg:** Activities, Voices and Arenas: Participatory Design in Practice, 1996, ISBN 91-7871-728-0.

No 448 **Patrick Lambrix:** Part-Whole Reasoning in Description Logics, 1996, ISBN 91-7871-820-1.

No 452 **Kjell Orsborn:** On Extensible and Object-Relational Database Technology for Finite Element Analysis Applications, 1996, ISBN 91-7871-827-9.

No 459 **Olof Johansson:** Development Environments for Complex Product Models, 1996, ISBN 91-7871-855-4.

No 461 **Lena Strömbäck:** User-Defined Constructions in Unification-Based Formalisms,1997, ISBN 91-7871-857-0.

No 462 **Lars Degerstedt:** Tabulation-based Logic Programming: A Multi-Level View of Query Answering, 1996, ISBN 91-7871-858-9.

No 475 **Fredrik Nilsson:** Strategi och ekonomisk styrning - En studie av hur ekonomiska styrsystem utformas och används efter företagsförvärv, 1997, ISBN 91-7871-914-3.

No 480 **Mikael Lindvall:** An Empirical Study of Requirements-Driven Impact Analysis in Object-Oriented Software Evolution, 1997, ISBN 91-7871-927-5.

No 485 **Göran Forslund**: Opinion-Based Systems: The Cooperative Perspective on Knowledge-Based Decision Support, 1997, ISBN 91-7871-938-0.

No 494 **Martin Sköld**: Active Database Management Systems for Monitoring and Control, 1997, ISBN 91-7219-002-7.

No 495 **Hans Olsén**: Automatic Verification of Petri Nets in a CLP framework, 1997, ISBN 91-7219-011-6.

No 498 **Thomas Drakengren:** Algorithms and Complexity for Temporal and Spatial Formalisms, 1997, ISBN 91-7219-019-1.

No 502 **Jakob Axelsson:** Analysis and Synthesis of Heterogeneous Real-Time Systems, 1997, ISBN 91-7219-035-3.

No 503 **Johan Ringström:** Compiler Generation for Data-Parallel Programming Langugaes from Two-Level Semantics Specifications, 1997, ISBN 91-7219-045-0.

No 512 **Anna Moberg:** Närhet och distans - Studier av kommunikationsmmönster i satellitkontor och flexibla kontor, 1997, ISBN 91-7219-119-8.

No 520 **Mikael Ronström:** Design and Modelling of a Parallel Data Server for Telecom Applications, 1998, ISBN 91-7219-169-4.

No 522 **Niclas Ohlsson:** Towards Effective Fault Prevention - An Empirical Study in Software Engineering, 1998, ISBN 91-7219-176-7.

No 526 **Joachim Karlsson:** A Systematic Approach for Prioritizing Software Requirements, 1998, ISBN 91-7219-184-8.

No 530 **Henrik Nilsson:** Declarative Debugging for Lazy Functional Languages, 1998, ISBN 91-7219-197-x.

No 555 **Jonas Hallberg:** Timing Issues in High-Level Synthesis,1998, ISBN 91-7219-369-7.

No 561 **Ling Lin:** Management of 1-D Sequence Data - From Discrete to Continuous, 1999, ISBN 91-7219-402-2.

No 563 **Eva L Ragnemalm:** Student Modelling based on Collaborative Dialogue with a Learning Companion, 1999, ISBN 91-7219-412-X.

No 567 **Jörgen Lindström:** Does Distance matter? On geographical dispersion in organisations, 1999, ISBN 91-7219-439-1.

No 582 **Vanja Josifovski:** Design, Implementation and

Evaluation of a Distributed Mediator System for Data Integration, 1999, ISBN 91-7219-482-0.

No 589 **Rita Kovordányi**: Modeling and Simulating Inhibitory Mechanisms in Mental Image Reinterpretation - Towards Cooperative Human-Computer Creativity, 1999, ISBN 91-7219-506-1.

No 592 **Mikael Ericsson:** Supporting the Use of Design Knowledge - An Assessment of Commenting Agents, 1999, ISBN 91-7219-532-0.

No 593 **Lars Karlsson:** Actions, Interactions and Narratives, 1999, ISBN 91-7219-534-7.

No 594 **C. G. Mikael Johansson:** Social and Organizational Aspects of Requirements Engineering Methods - A practice-oriented approach, 1999, ISBN 91-7219-541-X.

No 595 **Jörgen Hansson:** Value-Driven Multi-Class Overload Management in Real-Time Database Systems, 1999, ISBN 91-7219-542-8.

No 596 **Niklas Hallberg:** Incorporating User Values in the Design of Information Systems and Services in the Public Sector: A Methods Approach, 1999, ISBN 91-7219-543-6.

No 597 **Vivian Vimarlund:** An Economic Perspective on the Analysis of Impacts of Information Technology: From Case Studies in Health-Care towards General Models and Theories, 1999, ISBN 91-7219-544-4.

No 598 **Johan Jenvald:** Methods and Tools in Computer-Supported Taskforce Training, 1999, ISBN 91-7219-547-9.

No 607 **Magnus Merkel:** Understanding and enhancing translation by parallel text processing, 1999, ISBN 91-7219-614-9.

No 611 **Silvia Coradeschi:** Anchoring symbols to sensory data, 1999, ISBN 91-7219-623-8.

No 613 **Man Lin:** Analysis and Synthesis of Reactive Systems: A Generic Layered Architecture Perspective, 1999, ISBN 91-7219-630-0.

No 618 **Jimmy Tjäder:** Systemimplementering i praktiken - En studie av logiker i fyra projekt, 1999, ISBN 91-7219-657-2.

No 627 **Vadim Engelson:** Tools for Design, Interactive Simulation, and Visualization of Object-Oriented Models in Scientific Computing, 2000, ISBN 91-7219-709-9.

No 637 **Esa Falkenroth:** Database Technology for Control and Simulation, 2000, ISBN 91-7219-766-8.

No 639 **Per-Arne Persson:** Bringing Power and Knowledge Together: Information Systems Design for Autonomy and Control in Command Work, 2000, ISBN 91-7219-796-X.

No 660 **Erik Larsson:** An Integrated System-Level Design for Testability Methodology, 2000, ISBN 91-7219-890-7.

No 688 **Marcus Bjäreland:** Model-based Execution Monitoring, 2001, ISBN 91-7373-016-5.

No 689 **Joakim Gustafsson:** Extending Temporal Action Logic, 2001, ISBN 91-7373-017-3.

No 720 **Carl-Johan Petri:** Organizational Information Provision - Managing Mandatory and Discretionary Use of Information Technology, 2001, ISBN-91-7373-126-9.

No 724 **Paul Scerri:** Designing Agents for Systems with Adjustable Autonomy, 2001, ISBN 91 7373 207 9.

No 725 **Tim Heyer**: Semantic Inspection of Software Artifacts: From Theory to Practice, 2001, ISBN 91 7373 208 7.

No 726 **Pär Carlshamre:** A Usability Perspective on Requirements Engineering - From Methodology to Product Development, 2001, ISBN 91 7373 212 5.

No 732 **Juha Takkinen:** From Information Management to Task Management in Electronic Mail, 2002, ISBN 91 7373 258 3.

No 745 **Johan Åberg:** Live Help Systems: An Approach to Intelligent Help for Web Information Systems, 2002, ISBN 91-7373-311-3.

No 746 **Rego Granlund:** Monitoring Distributed Teamwork Training, 2002, ISBN 91-7373-312-1.

No 757 **Henrik André-Jönsson:** Indexing Strategies for Time Series Data, 2002, ISBN 917373-346-6.

No 747 **Anneli Hagdahl:** Development of IT-suppor-ted Inter-organisational Collaboration - A Case Study in the Swedish Public Sector, 2002, ISBN 91-7373-314-8.

No 749 **Sofie Pilemalm:** Information Technology for Non-Profit Organisations - Extended Participatory Design of an Information System for Trade Union Shop Stewards, 2002, ISBN 91-7373-318-0.

No 765 **Stefan Holmlid:** Adapting users: Towards a theory of use quality, 2002, ISBN 91-7373-397-0.

No 771 **Magnus Morin:** Multimedia Representations of Distributed Tactical Operations, 2002, ISBN 91-7373-421-7.

No 772 **Pawel Pietrzak:** A Type-Based Framework for Locating Errors in Constraint Logic Programs, 2002, ISBN 91-7373-422-5.

No 758 **Erik Berglund:** Library Communication Among Programmers Worldwide, 2002, ISBN 91-7373-349-0.

No 774 **Choong-ho Yi:** Modelling Object-Oriented Dynamic Systems Using a Logic-Based Framework, 2002, ISBN 91-7373-424-1.

No 779 **Mathias Broxvall:** A Study in the Computational Complexity of Temporal Reasoning, 2002, ISBN 91-7373-440-3.

No 793 **Asmus Pandikow:** A Generic Principle for Enabling Interoperability of Structured and Object-Oriented Analysis and Design Tools, 2002, ISBN 91-7373-479-9.

No 785 **Lars Hult:** Publika Informationstjänster. En studie av den Internetbaserade encyklopedins bruksegenskaper, 2003, ISBN 91-7373-461-6.

No 800 **Lars Taxén:** A Framework for the Coordination of Complex Systems´ Development, 2003, ISBN 91-7373-604-X

No 808 **Klas Gäre:** Tre perspektiv på förväntningar och förändringar i samband med införande av informa-

tionsystem, 2003, ISBN 91-7373-618-X.

No 821 **Mikael Kindborg:** Concurrent Comics - programming of social agents by children, 2003, ISBN 91-7373-651-1.

No 823 **Christina Ölvingson:** On Development of Information Systems with GIS Functionality in Public Health Informatics: A Requirements Engineering Approach, 2003, ISBN 91-7373-656-2.

No 828 **Tobias Ritzau:** Memory Efficient Hard Real-Time Garbage Collection, 2003, ISBN 91-7373-666-X.

No 833 **Paul Pop:** Analysis and Synthesis of Communication-Intensive Heterogeneous Real-Time Systems, 2003, ISBN 91-7373-683-X.

No 852 **Johan Moe:** Observing the Dynamic Behaviour of Large Distributed Systems to Improve Development and Testing - An Emperical Study in Software Engineering, 2003, ISBN 91-7373-779-8.

No 867 **Erik Herzog:** An Approach to Systems Engineering Tool Data Representation and Exchange, 2004, ISBN 91-7373-929-4.

No 872 **Aseel Berglund:** Augmenting the Remote Control: Studies in Complex Information Navigation for Digital TV, 2004, ISBN 91-7373-940-5.

No 869 **Jo Skåmedal:** Telecommuting's Implications on Travel and Travel Patterns, 2004, ISBN 91-7373-935-9.

No 870 **Linda Askenäs:** The Roles of IT - Studies of Organising when Implementing and Using Enterprise Systems, 2004, ISBN 91-7373-936-7.

No 874 **Annika Flycht-Eriksson:** Design and Use of Ontologies in Information-Providing Dialogue Systems, 2004, ISBN 91-7373-947-2.

No 873 **Peter Bunus:** Debugging Techniques for Equation-Based Languages, 2004, ISBN 91-7373-941-3.

No 876 **Jonas Mellin:** Resource-Predictable and Efficient Monitoring of Events, 2004, ISBN 91-7373-956-1.

No 883 **Magnus Bång:** Computing at the Speed of Paper: Ubiquitous Computing Environments for Healthcare Professionals, 2004, ISBN 91-7373-971-5

No 882 **Robert Eklund:** Disfluency in Swedish human-human and human-machine travel booking dialogues, 2004. ISBN 91-7373-966-9.

No 887 **Anders Lindström:** English and other Foreign Linquistic Elements in Spoken Swedish. Studies of Productive Processes and their Modelling using Finite-State Tools, 2004, ISBN 91-7373-981-2.

No 889 **Zhiping Wang:** Capacity-Constrained Production-inventory systems - Modellling and Analysis in both a traditional and an e-business context, 2004, ISBN 91-85295-08-6.

No 893 **Pernilla Qvarfordt:** Eyes on Multimodal Interaction, 2004, ISBN 91-85295-30-2.

No 910 **Magnus Kald:** In the Borderland between Strategy and Management Control - Theoretical Framework and Empirical Evidence, 2004, ISBN 91-85295-82-5.

No 918 **Jonas Lundberg:** Shaping Electronic News: Genre Perspectives on Interaction Design, 2004, ISBN 91-85297-14-3.

No 900 **Mattias Arvola:** Shades of use: The dynamics of interaction design for sociable use, 2004, ISBN 91-85295-42-6.

No 920 **Luis Alejandro Cortés:** Verification and Scheduling Techniques for Real-Time Embedded Systems, 2004, ISBN 91-85297-21-6.

No 929 **Diana Szentivanyi:** Performance Studies of Fault-Tolerant Middleware, 2005, ISBN 91-85297-58-5.

No 933 **Mikael Cäker:** Management Accounting as Constructing and Opposing Customer Focus: Three Case Studies on Management Accounting and Customer Relations, 2005, ISBN 91-85297-64-X.

No 937 **Jonas Kvarnström:** TALplanner and Other Extensions to Temporal Action Logic, 2005, ISBN 91-85297-75-5.

No 938 **Bourhane Kadmiry:** Fuzzy Gain-Scheduled Visual Servoing for Unmanned Helicopter, 2005, ISBN 91-85297-76-3.

No 945 **Gert Jervan:** Hybrid Built-In Self-Test and Test Generation Techniques for Digital Systems, 2005, ISBN: 91-85297-97-6.

No 946 **Anders Arpteg:** Intelligent Semi-Structured Information Extraction, 2005, ISBN 91-85297-98-4.

No 947 **Ola Angelsmark:** Constructing Algorithms for Constraint Satisfaction and Related Problems - Methods and Applications, 2005, ISBN 91-85297-99-2.

No 963 **Calin Curescu:** Utility-based Optimisation of Resource Allocation for Wireless Networks, 2005. ISBN 91-85457-07-8.

No 972 **Björn Johansson:** Joint Control in Dynamic Situations, 2005, ISBN 91-85457-31-0.

No 974 **Dan Lawesson:** An Approach to Diagnosability Analysis for Interacting Finite State Systems, 2005, ISBN 91-85457-39-6.

No 979 **Claudiu Duma:** Security and Trust Mechanisms for Groups in Distributed Services, 2005, ISBN 91-85457-54-X.

No 983 **Sorin Manolache:** Analysis and Optimisation of Real-Time Systems with Stochastic Behaviour, 2005, ISBN 91-85457-60-4.

No 986 **Yuxiao Zhao:** Standards-Based Application Integration for Business-to-Business Communications, 2005, ISBN 91-85457-66-3.

No 1004 **Patrik Haslum:** Admissible Heuristics for Automated Planning, 2006, ISBN 91-85497-28-2.

No 1005 **Aleksandra Tešanovic:** Developing Reusable and Reconfigurable Real-Time Software using Aspects and Components, 2006, ISBN 91-85497-29-0.

No 1008 **David Dinka:** Role, Identity and Work: Extending the design and development agenda, 2006, ISBN 91-85497-42-8.

No 1009 **Iakov Nakhimovski:** Contributions to the Modeling and Simulation of Mechanical Systems with Detailed Contact Analysis, 2006, ISBN 91-85497-43-X.

No 1013 **Wilhelm Dahllöf:** Exact Algorithms for Exact Satisfiability Problems, 2006, ISBN 91-85523-97-6.

No 1016 **Levon Saldamli:** PDEModelica - A High-Level Language for Modeling with Partial Differential Equations, 2006, ISBN 91-85523-84-4.

No 1017 **Daniel Karlsson:** Verification of Component-based Embedded System Designs, 2006, ISBN 91-85523-79-8.

No 1018 **Ioan Chisalita:** Communication and Networking Techniques for Traffic Safety Systems, 2006, ISBN 91-85523-77-1.

No 1019 **Tarja Susi:** The Puzzle of Social Activity - The Significance of Tools in Cognition and Cooperation, 2006, ISBN 91-85523-71-2.

No 1021 **Andrzej Bednarski:** Integrated Optimal Code Generation for Digital Signal Processors, 2006, ISBN 91-85523-69-0.

No 1022 **Peter Aronsson:** Automatic Parallelization of Equation-Based Simulation Programs, 2006, ISBN 91-85523-68-2.

No 1030 **Robert Nilsson:** A Mutation-based Framework for Automated Testing of Timeliness, 2006, ISBN 91-85523-35-6.

No 1034 **Jon Edvardsson:** Techniques for Automatic Generation of Tests from Programs and Specifications, 2006, ISBN 91-85523-31-3.

No 1035 **Vaida Jakoniene:** Integration of Biological Data, 2006, ISBN 91-85523-28-3.

No 1045 **Genevieve Gorrell:** Generalized Hebbian Algorithms for Dimensionality Reduction in Natural Language Processing, 2006, ISBN 91-85643-88-2.

No 1051 **Yu-Hsing Huang:** Having a New Pair of Glasses - Applying Systemic Accident Models on Road Safety, 2006, ISBN 91-85643-64-5.

No 1054 **Åsa Hedenskog:** Perceive those things which cannot be seen - A Cognitive Systems Engineering perspective on requirements management, 2006, ISBN 91-85643-57-2.

No 1061 **Cécile Åberg:** An Evaluation Platform for Semantic Web Technology, 2007, ISBN 91-85643-31-9.

No 1073 **Mats Grindal:** Handling Combinatorial Explosion in Software Testing, 2007, ISBN 978-91-85715-74-9.

No 1075 **Almut Herzog:** Usable Security Policies for Runtime Environments, 2007, ISBN 978-91-85715-65-7.

No 1079 **Magnus Wahlström:** Algorithms, measures, and upper bounds for satisfiability and related problems, 2007, ISBN 978-91-85715-55-8.

No 1083 **Jesper Andersson:** Dynamic Software Architectures, 2007, ISBN 978-91-85715-46-6.

No 1086 **Ulf Johansson:** Obtaining Accurate and Comprehensible Data Mining Models - An Evolutionary Approach, 2007, ISBN 978-91-85715-34-3.

No 1089 **Traian Pop:** Analysis and Optimisation of Distributed Embedded Systems with Heterogeneous Scheduling Policies, 2007, ISBN 978-91-85715-27-5.

No 1091 **Gustav Nordh:** Complexity Dichotomies for CSP-related Problems, 2007, ISBN 978-91-85715-20-6.

No 1106 **Per Ola Kristensson:** Discrete and Continuous Shape Writing for Text Entry and Control, 2007, ISBN 978-91-85831-77-7.

No 1110 **He Tan:** Aligning Biomedical Ontologies, 2007, ISBN 978-91-85831-56-2.

No 1112 **Jessica Lindblom:** Minding the body - Interacting socially through embodied action, 2007, ISBN 978-91-85831-48-7.

No 1113 **Pontus Wärnestål:** Dialogue Behavior Management in Conversational Recommender Systems, 2007, ISBN 978-91-85831-47-0.

No 1120 **Thomas Gustafsson:** Management of Real-Time Data Consistency and Transient Overloads in Embedded Systems, 2007, ISBN 978-91-85831-33-3.

No 1127 **Alexandru Andrei:** Energy Efficient and Predictable Design of Real-time Embedded Systems, 2007, ISBN 978-91-85831-06-7.

No 1139 **Per Wikberg:** Eliciting Knowledge from Experts in Modeling of Complex Systems: Managing Variation and Interactions, 2007, ISBN 978-91-85895-66-3.

No 1143 **Mehdi Amirijoo:** QoS Control of Real-Time Data Services under Uncertain Workload, 2007, ISBN 978-91-85895-49-6.

No 1150 **Sanny Syberfeldt:** Optimistic Replication with Forward Conflict Resolution in Distributed Real-Time Databases, 2007, ISBN 978-91-85895-27-4.

No 1155 **Beatrice Alenljung:** Envisioning a Future Decision Support System for Requirements Engineering - A Holistic and Human-centred Perspective, 2008, ISBN 978-91-85895-11-3.

No 1156 **Artur Wilk**: Types for XML with Application to Xcerpt, 2008, ISBN 978-91-85895-08-3.

No 1183 **Adrian Pop:** Integrated Model-Driven Development Environments for Equation-Based Object-Oriented Languages, 2008, ISBN 978-91-7393-895-2.

No 1185 **Jörgen Skågeby:** Gifting Technologies - Ethnographic Studies of End-users and Social Media Sharing, 2008, ISBN 978-91-7393-892-1.

No 1187 **Imad-Eldin Ali Abugessaisa:** Analytical tools and information-sharing methods supporting road safety organizations, 2008, ISBN 978-91-7393-887-7.

No 1204 **H. Joe Steinhauer:** A Representation Scheme for Description and Reconstruction of Object Configurations Based on Qualitative Relations, 2008, ISBN 978-91-7393-823-5.

No 1222 **Anders Larsson:** Test Optimization for Core-based System-on-Chip, 2008, ISBN 978-91-7393-768-9.

No 1238 **Andreas Borg:** Processes and Models for Capacity Requirements in Telecommunication Systems, 2009, ISBN 978-91-7393-700-9.

No 1240 **Fredrik Heintz:** DyKnow: A Stream-Based Knowledge Processing Middleware Framework, 2009, ISBN 978-91-7393-696-5.

No 1241 **Birgitta Lindström:** Testability of Dynamic Real-Time Systems, 2009, ISBN 978-91-7393-695-8.

Linköping Studies in Statistics

No 9 **Davood Shahsavani:** Computer Experiments Designed to Explore and Approximate Complex Deterministic Models, 2008, ISBN 978-91-7393-976-8.

No 10 **Karl Wahlin:** Roadmap for Trend Detection and Assessment of Data Quality, 2008, ISBN: 978-91-7393-792-4

Linköping Studies in Information Science

No 1 **Karin Axelsson:** Metodisk systemstrukturering- att skapa samstämmighet mellan informa-tionssystemarkitektur och verksamhet, 1998. ISBN-9172-19-

296-8.

No 2 **Stefan Cronholm:** Metodverktyg och användbarhet - en studie av datorstödd metodbaserad systemutveckling, 1998. ISBN-9172-19-299-2.

No 3 **Anders Avdic:** Användare och utvecklare - om anveckling med kalkylprogram, 1999. ISBN-91-7219-606-8.

No 4 **Owen Eriksson:** Kommunikationskvalitet hos informationssystem och affärsprocesser, 2000. ISBN 91-7219-811-7.

No 5 **Mikael Lind:** Från system till process - kriterier för processbestämning vid verksamhetsanalys, 2001, ISBN 91-7373-067-X

No 6 **Ulf Melin:** Koordination och informationssystem i företag och nätverk, 2002, ISBN 91-7373-278-8.

No 7 **Pär J. Ågerfalk:** Information Systems Actability - Understanding Information Technology as a Tool for Business Action and Communication, 2003, ISBN 91-7373-628-7.

No 8 **Ulf Seigerroth:** Att förstå och förändra systemutvecklingsverksamheter - en taxonomi för metautveckling, 2003, ISBN91-7373-736-4.

No 9 **Karin Hedström:** Spår av datoriseringens värden - Effekter av IT i äldreomsorg, 2004, ISBN 91-7373-963-4.

No 10 **Ewa Braf:** Knowledge Demanded for Action - Studies on Knowledge Mediation in Organisations, 2004, ISBN 91-85295-47-7.

No 11 **Fredrik Karlsson:** Method Configuration - method and computerized tool support, 2005, ISBN 91-85297-48-8.

No 12 **Malin Nordström:** Styrbar systemförvaltning - Att organisera systemförvaltningsverksamhet med hjälp av effektiva förvaltningsobjekt, 2005, ISBN 91-85297-60-7.

No 13 **Stefan Holgersson:** Yrke: POLIS - Yrkeskunskap, motivation, IT-system och andra förutsättningar för polisarbete, 2005, ISBN 91-85299-43-X.

No 14 **Benneth Christiansson, Marie-Therese Christiansson:** Mötet mellan process och komponent - mot ett ramverk för en verksamhetsnära kravspecifikation vid anskaffning av komponentbaserade informationssystem, 2006, ISBN 91-85643-22-X.