Lappeenrannan teknillinen yliopisto
Lappeenranta University of Technology

Päivi Ovaska

STUDIES ON COORDINATION OF SYSTEMS DEVELOPMENT PROCESS

Thesis for the degree of Doctor of Science (Technology) to be presented with due permission for public examination and criticism in the Auditorium 1383 at Lappeenranta University of Technology, Lappeenranta, Finland, on the 12th August, 2005, at noon.

Acta Universitatis

Lappeenrantaensis

214

Supervisors	Professor Matti Rossi Department of Management Information Systems Science Helsinki School of Economics Finland
	Professor Jouni Lampinen Department of Information Technology Lappeenranta University of Technology Finland
Reviewers	Professor Jan Pries-Heje Institute of Design and Use of IT IT University of Copenhagen Denmark
	Professor Eija Karsten Department of Information Technology University of Turku Finland
Opponents	Professor Joe Nandhakumar School of Management University of Bath United Kingdom
	Professor Jan Pries-Heje Institute of Design and Use of IT IT University of Copenhagen Denmark

ISBN 952-214-063-5

ISBN 952-214-064-3 (PDF)

ISSN 1456-4491

Lappeenrannan teknillinen yliopisto

Digipaino 2005

Abstract

Päivi Ovaska
Studies on coordination of systems development process
Lappeenranta, 2005
83 p.

Acta Universitatis Lappeenrantaensis 214

Diss. Lappeenranta University of Technology

ISBN 952-214-063-5, ISBN 952-214-064-3 (PDF), ISSN 1456-4491

This thesis examines coordination of systems development process in a contemporary software producing organization. The thesis consists of a series of empirical studies in which the actions, conceptions and artifacts of practitioners are analyzed using a theory-building case study research approach. The three phases of the thesis provide empirical observations on different aspects of systems development. In the first phase is examined the role of architecture in coordination and cost estimation in multi-site environment. The second phase involves two studies on the evolving requirement understanding process and how to measure this process. The third phase summarizes the first two phases and concentrates on the role of methods and how practitioners work with them.

All the phases provide evidence that current systems development method approaches are too naïve in looking at the complexity of the real world. In practice, development is influenced by opportunity and other contingent factors. The systems development process is not coordinated using phases and tasks defined in methods providing universal mechanism for managing this process like most of the method approaches assume. Instead, the studies suggest that managing systems development process happens through coordinating development activities using methods as tools.

These studies contribute to the systems development methods by emphasizing the support of communication and collaboration between systems development participants. Methods should not describe the development activities and phases in a detail level, but should include the higher level guidance for practitioners on how to act in different systems development environments.

Keywords: information systems development, software architecture, requirement elicitation, case study, grounded theory

UDC 004.414 : 004.2

Acknowledgements

Carrying out this research has been an interesting journey for me. It has offered me the excitement of searching, enjoyment in finding and understanding. I have learned a lot about research work and systems development. Now, when this journey is almost finished, I feel that I have learned more that I could originally expect. But I would never been able to go on with this journey alone. Many people and organizations have helped me to focus and to see things differently. I will try to express my gratitude to most of the people and organizations that helped make this thesis happen.

During the research process, a number of publications were written and published in various international forums. This would not have been possible had I not received guidance from my supervisor Matti Rossi in the preparation of the publications. It was from him that I learned that there are a number of ways a paper can be "shot down". Thank You Matti, for encouraging me to go on despite of all the difficulties especially in the beginning of the publishing process. Emeritus professor Pertti Järvinen deserves special thanks for sending me "just the right" papers. I would also like to thank my co-writers, Matti Rossi, Kari Smolander, Pentti Marttiin and Alexandre Bern for their help. Especially with the journal articles their help in writing and focusing was crucial.

The work of the external reviewers of this thesis, professors Eija Karsten and Jan Pries-Heje is gratefully acknowledged.

I wish to express my sincerest gratitude to Kari Smolander, Uolevi Nikula and Hanna-Kaisa Lammi, friends of mine and colleagues following a similar path to the dissertation with whom I had the opportunity of sharing the joys of doing research and university life. Kari's expertise also provided me with a platform for thoughts in this thesis.

I also would like to thank my former employer, TeliaSonera Corporation and all my colleagues working there at that time. I wish to address my thanks especially to Ari Tolonen, Juha Marjeta, Marketta Gland, Mika Kataikko, Ari-Pekka Mattila, Esa Nyman and Marja-Leena Heikkinen.

As with any scientific research, the role of empirical data is invaluable. I am therefore forever grateful to the company I received the data from and the personnel whom I had contact with. Since the anonymity is to be preserved, no names are cited here. Without these people, however, the claims made in thesis could not have been empirically explored. I want to thank my daughter, Katariina, for transcripting the interviews into written text.

I am grateful about the support provided by Lappeenranta University of Technology and especially my supervisor Jouni Lampinen, Pekka Toivanen, Kirsimarja Blomqvist and Jarmo Partanen. I also appreciate Tiina Kauranen and Riikka Ketonen for their professional help in editing the language of this thesis.

The financial support provided by South Carelia Polytechnic and especially by Antti Lehmusvaara and Mikko Huhtanen is highly appreciated.

The financial support by the The Foundation of William and Ester Otsakorpi (William ja Ester Otsakorven säätiö) and The Research Foundation of Lappeenranta University of Technology (Lappeenrannan teknillisen korkeakoulun tukisäätiö) is also gratefully acknowledged.

I see this thesis as a learning process continuing my whole lifetime, not only as a couple of years' study process. Already my mother Sirkka and my father Yrjö gave me good groundings to my life. Also my brother Matti has influenced my life immensely. I wish that You, my dear mother, could share this moment with me.

Finally, I would like to express my gratitude to my dearest ones, my husband Harri and my children Valtteri, Katariina and Simo, for supporting and helping me. You are my source of energy and meaning of my life.

Suuret kiitokset isälleni Yrjölle ja Airalle sekä appivanhemmilleni Liisalle ja Väinölle saamastani tuesta ja avusta.

Lappeenranta, June 2005

Päivi Ovaska

List of publications

I. Ovaska, P., M. Rossi, P. Marttiin (2004): "Architecture as a Coordination Tool in Multi-site Software Development", in *Software Process Improvement and Practice*, 8(4), pp. 233-248, John Wiley & Sons Ltd.

II. Ovaska, P., A. Bern (2004):"Architecture as a Predictor of System Size – A Metaphor from Construction Projects", in *Proceedings of the 16th International Conference on Advanced Information Systems Engineering* (CAISE '04 Forum), Riga, Latvia, June 7-11, Riga Technical University, pp. 193-203.

III. Ovaska, P., M. Rossi, K. Smolander (accepted): "Filtering, Negotiating and Shifting in the Understanding of Information Systems Requirements", *Scandinavian Journal of Information Systems*, IRIS Association.

IV. Ovaska, P. (2004):"Measuring Requirement Evolution – A Case Study in the E-commerce Domain", in *Proceeding of the 6th International Conference on Enterprise Information Systems* (ICEIS(3)), Porto, Portugal, April 14-17, INSTICC, pp. 669-673.

V. Ovaska, P. (forthcoming in 2005): "Working with Methods: Observations on the Role Methods in Systems Development", in *Information Systems Development: Advances in Theory, Practice and Education, ISD 2004*, edited by O. Vasilecas, A. Caplinskas, W. Wojtkowski, W.G. Wojtkowski, J. Zupancic, Springer, pp. 185-197.

Symbols and abbreviations

ANSI/SPARC	used to refer to the three-level database systems architecture, literally ANSI/Systems Planning and Requirement committee
CASE	Computer Aided Software Engineering
CMM	Capability Maturity Model
COM	Component Object Model
CORBA	Common Object Request Brokering Architecture
CSCW	Computer Supported Cooperative Work
EJB	Enterprise Java Beans
ER	Entity Relationships
FP	Function Point
GT	Grounded Theory
HCI	Human-Computer Interaction
IS	Information Systems
ISD	Information Systems Development
LOC	Lines of Codes
OMT	Object Management Technique
RAD	Rapid Application Development
SDLC	Systems Development Life Cycle
SE	Software Engineering
SMS	Short Message Service
SPI	Software Process Improvement

SPICE	Software Process Improvement and Capability Determination
SSM	Soft Systems Methodology
UML	Unified Modeling Language
XML	Extensible Markup Language
XSLT	Extensible Stylesheet Language Transformation

Contents

1 Introduction

1.1 Motivation

Information systems play an ever-increasing role in today's society and industry. We regularly use many kinds of information systems, such as electronic mail and the Internet, mobile phones, cars controlled by computer and digital TV sets. "Information systems has also become an essential part of the modern organizations that enables new products and services and innovative ways to deliver products and services faster" (Weill and Broadbent 1998).

These information systems, such as mailing or security management systems, are planned, designed and implemented by people (systems developers). The competence needed of systems developers is diverse: they need to understand the requirements of the problem domain (such as security control) at some level, know the possibilities that different implementation technologies (such as mobile networks or the Internet) can offer them and to communicate their ideas and solution proposals to a large set of people, even between different languages. This is a different situation than in the 1950's and early 1960's when the systems were more monolithic and systems developers were thus able to master their development work in smaller groups.

During thirty years of history, planning, designing and implementing, the information systems have faced a great deal of productivity problems. These productivity problems include slipping schedules and cost overruns (Boehm 1987; Brooks 1995), and low systems quality with increased maintenance costs (Boehm 1987). According to Standish Group's 1998 survey, only 26% of systems development projects were delivered on time, on budget, and with promised functionality, wasting billions of dollars annually; 46% were completed over budget and behind schedule, with fever functions and features than originally specified (Keil and Robey 2001). Another study reported that between 30-40% of all systems development projects went over budget and exceeded their time schedule (Keil, Mann et al. 2000). Many researchers have even gone as far as to speak of a "software crisis" or a "system crisis" (Brooks 1987). *Software crisis* is characterized by projects that are "always over budget, behind schedule, and unreliable" (Glass 1994). These kind of problems forced information systems (IS) and software engineering (SE) communities to direct their efforts towards improvement of software quality and productivity.

In response to these productivity and quality problems, researchers have developed different methodical approaches. Methods are seen important elements in both disciplines and method approaches are based on the strong belief that methods provide universal consistent mechanism for achieving the management and control of the systems development process. This belief is criticized by (Nandhakumar and Avison 1999; Truex, Baskerville et al. 2001). We can classify these methods approaches into three categories based on how their view of the reasons and solutions for the software crisis: system modeling and architecture approaches, process approaches, and socio-technical approaches. Next, we will briefly look at how each of these approaches has dealt with the software crisis.

Traditionally, the software crisis has been considered a *consequence of the increased technical complexity of the systems*. For example, Dijkstra wrote already in 1972: "The major course of a software crisis is that the machines have become several orders of magnitude more powerful! To put it quite bluntly: as long as there were no machines, programming was no problem at all. When we had a few weak computers, programming became a mild problem, and now that we have gigantic computers, programming has become an equally gigantic problem "(Dijkstra 1972). Dijkstra along with many other researchers considered the system as a technical system (Hirschheim, Klein et al. 1995). Creating conceptual, more abstract models of the system has been one approach to decrease the complexity of systems. Traditional structured methods such as Yourdon's Structured Analysis (Yourdon 1989) as well as object-oriented methods such as OMT (Rumbaugh, Blaha et al. 1990) are trying to resolve the software crisis by modelling the system. The basic principles for architecting software-based systems is the approach that use these system models (Smolander 2003). The concepts of the structured approach, such as information hiding (Parnas 1972), functional decomposition, cohesion and coupling (DeMarco 1978), are the basic principles of these *system modeling and architecture approaches*.

Especially, the software engineering (SE) community holds the view that the *quality of products depends on the quality of the production process* (Georgiadou 2003). This process encompasses both technical and managerial concerns, including technical development, project management, data and configuration management, and quality assurance and control. Suggested benefits of process structuring include improved productivity through standardisation, and easier division of labour by breaking processes down into tasks (Dowson 1993). Besides, this approach is based on a strong belief that a well defined and documented systems development process can lay foundation for improving long term productivity. The earlier life-cycle model (Walters, Broady et al. 1994) approaches have developed such models as the waterfall, prototyping and iterative models (Boehm 1988). These models define the order of software development activities in software production, beginning with a statement of the software requirements and ending with a description of its operation and maintenance (Blom 1994). The later approach, software process improvement (SPI) approach, such as CMM (Paulk, Curtis et al. 1993; Herbsleb, Zubrow et al. 1997; Kuilboer and Ashrafi 2000), sees that explicit and coherent processes and their improvement increase the effectiveness and predictability of the software projects and software products' quality. This approach has gained lately much attention especially in practical systems development organizations (Glass 1999).

More recently, in the information systems (IS) field, the system has been seen more as a social system that is technically implemented (Hirschheim, Klein et al. 1995). Already the classical view of user participation in ETHICS (Mumford 1983) and Soft System Methodology (Checkland 1981; Checkland and Scholes 1990) considered both social and technical aspects of systems development. The other views, such as Scandinavian trade unionist approach (Nygaard and Bergo 1974) focuses primarily on the structure and power of economic power relationships in systems development. These approaches, which we can call *socio-technical approaches*, are supported by the observations that many of the problems encountered during system development are not technical, but more related to insufficient user involvement, evolving requirements or coordination of development work. Those challenges deal more with social interaction and communication (Hirschheim, Klein et al. 1995) emphasizing *productivity problems as a consequence of the growing social complexity of the systems*. These approaches are mostly identified by the academic community and are not much recognized by practitioners (Lyytinen 1987b).

Paradoxically, despite the efforts devoted to methodical approaches, systems have continued to fail (Georgiadou 2003). The status of methods as a whole has been described as a "method jungle", as "an unorganized collection of methods more or less similar to each other" (Jayaratna 1994). Methods are not fully accepted among practitioners, and there is evidence of significant problems with the use of these methods (Russo, Wynekoop et al. 1995; Wynekoop and Russo 1995; Fitzgerald 1998; Iivari and Maansaari 1998; Nandhakumar and Avison 1999; Fitzgerald 2000). The development of new methods has tended to be technology-driven, being often influenced by the introduction of improved techniques and software tools (Nandhakumar and Avison 1999). Only a few studies concentrate on the process of systems development in their organizational context (Curtis, Krasner et al. 1988; Orlikowski 1993; Nandhakumar and Avison 1999; Beynon-Davies and Williams 2003). Systematic surveys of the existing literature in both IS (Wynekoop and Russo 1997) and SE (Glass, Vessey et al. 2002) fields revealed that most of the research papers in these fields consist of normative research in which concept development is not based on empirical grounding or theoretical analysis, but merely upon the author's opinions. Many researchers (Curtis, Krasner et al. 1988; Orlikowski 1993; Fitzgerald 1996; Introna and Whitley 1997; Glass, Vessey et al. 2002) call for more empirical studies in order to understand how information systems are developed in today's organizations and how well methods are used before proposing improvements or new methods.

1.2 About these studies

As stated above, there are too few empirical studies that concentrate on the process of systems development in the organizational context. Therefore, this thesis aims to fill this gap by further clarifying how systems development process is managed in practice. This objective is reached by conducting a series of empirical studies of two systems development projects in a contemporary organization that competes in the information technology business. We study the early systems development, which we consider to be most important phases in solving the software crisis: architecture design and requirement elicitation. In these studies the actions, conceptions and artifacts of practitioners are interpreted and analyzed using theory-building case study approach. The objective for this research is twofold: 1) to understand how practitioners manage the systems development process and 2) to make a contribution to the theory and practice of systems development.

The studies of this thesis highlight the problems in the current approaches to systems development, which still largely assume that systems development is effectively managed by system developers and managers, and methods provide universal mechanism for achieving this management. Instead, our studies suggest that there is no universal method for managing the systems development process, but development is coordinated through the methods appropriate to the situation. In this development process, methods play an important role as tools, which guide the practitioners to understand the final system in different situations.

Using the metaphor 'Trukese navigator' we can describe systems development in general as a process of Trukese navigation to the island in the following way:

"The Trukese navigator begins his navigation with an objective to proceed to a particular island without a reference to any explicit principles and without any planning, unless the intention to proceed to a particular island can be considered a plan. He sets off toward the objective and responds to the conditions as they arise in an ad hoc fashion. He utilizes information provided by the wind, the waves, the tide and current, the fauna, the stars, the clouds, the sound of the water on the side of the boat, and he steers accordingly. His effort is directed to doing whatever is necessary to reach the objective." (Suchman 1987).

The rest of this thesis is divided into two parts, an introduction and an appendix including five scientific publications. The first part (introduction) is composed of six chapters. Chapter 2 defines the context of these studies. Chapter 3 describes the research area, problem and research methodology. In Chapter 4, the publications included in the appendix are summarized. Chapter 5 discusses the research results along with their implications for research, practice and systems development methods. Finally, Chapter 6 of the introduction summarizes the whole thesis, lists its contributions, identifies its limitations and suggests topics for further research. The appendix is composed of five publications that have gone through a scientific referee process. These publications present the results of this thesis in detail.

2 Managing systems development process

The aim of this chapter is to give a general orientation to the subject of this thesis by describing the terminology and concepts around managing systems development process. This chapter includes also the description of related research. The chapter ends with a summary of the scope of this thesis.

2.1 Information systems development methods

In this thesis, we study the information systems development process and its coordination in practice. We first define *Information Systems Development (ISD),* as being *a change process taken with respect to an object system in an environment by a development group using tools and an organized collection of methods to produce a target system* (Lyytinen 1987a; Lyytinen 1987b). In the following figure (Figure 1) the elements of this environment are presented.

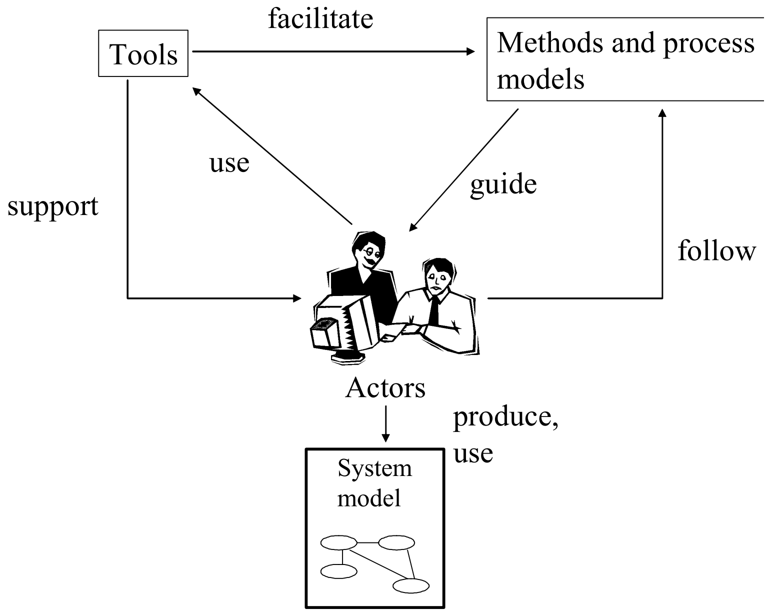


Figure 1. Systems development environment. Adapted from (Marttiin 1998)

Systems development environment is a composition of different factors: actors, methods, processes and tools, system models, and the relationships of these. These factors should determine the methods, process models and supporting tools used in systems development (Marttiin 1998). In this environment, the set of *contingency factors* or *contingencies* (Kast and Resenzweig 1974; van Slooten and Schoonhoven 1996) influence the composition of actors, methods and supporting tools. These contingency factors include economic resources (time and money available), organization and stakeholder related factors (e.g. managerial commitment, clarity of goals, importance of system, and knowledge, experience, and resistance of users), and project and user related factors (e.g. project size, skill, dependencies with other projects, and developers' familiarity with the application area, methods and tools). We can say that each development process is unique. Methods as situation based artifacts are discussed in (Sol 1983; Avison and Fitzgerald 1988; Curtis, Krasner et al. 1988; Kumar and Welke 1992). Their message is that there is no best way to develop a system, and no method is suitable for all development situations.

Lyytinen's definition emphasizes the process that takes place in an organization. We can notice the situation dependency of the systems development process when looking at the details of the definition. *The change process can be described as an event, where the development groups acts on, formulates and alters current object systems guided by multiple objectives* (Hirschheim, Klein et al. 1995). *An object system consists of phenomena,* *which the development group is observing*. Object systems are perceptions of the target of change, which may vary among the members of the development group (Hirschheim, Klein et al. 1995). We emphasize also the latter part of Lyytinen's definition, which states that *development group uses methods and tools to describe the system.*

It is generally believed that the power of methods comes from Descartes, who proposed that truth is more a matter of the proper method than genial insight or divine inspiration (Hirschheim, Klein et al. 1995). Influenced by Descartes, the concept of method entered mathematics and natural sciences. As these sciences have defined what counts as knowledge in the Western world, the concept of method has deeply influenced policies and practices in industrial societies and the managing of technical or social change (MacKenzie and Wajsman 1999). Therefore, the majority of disciplined approaches to systems development follow some methodical guidelines.

Systems development methods have been developed primarily to formalize the systems development activities and to allow effective sharing of information through standardized notations and models of views of systems (Rossi 1998). The simplest way to see methods is "*the way of doing things*" (Angell and Straub 1993), which is a popular view among system development practitioners (Wynekoop and Russo 1995). In these studies, we define methods as *"a predefined and organized collection of techniques and a set of rules which state by whom, in what order, and in what way the techniques are used*" (Smolander, Tahvanainen et al. 1990; Tolvanen 1998). There exist many other, quite similar, definitions of methods. The most widely used method definitions are exemplified in the following table (Table 1).

Table 1. Examples of widely used definitions of systems development method

Source	*Method definition*
Hirschheim et al. (Hirschheim, Klein et al. 1995)	An organized collection of concepts, methods, beliefs, values and normative principles supported by material resources
Avison and Fitzgerald (Avison and Fitzgerald 1995)	A collection of procedures, techniques, tools and documentation aids which will help the systems developer in their efforts to implement a new information system. A method will consist of phases, themselves consisting of subphases, which will guide the systems developers in their choice of the techniques that might be appropriate at each stage of the project and also help them plan, manage, control and evaluate information systems project. It is usually based on some 'philosophical' view.
Wynekoop and Russo (Wynekoop and Russo 1995)	A systematic approach to conducting at least one complete phase (e.g. requirement analysis, design) of systems development, consisting of a set of guidelines, activities, techniques and tools, based on a particular philosophy of systems development and the target system
Rumbaugh (Rumbaugh 1995)	1) a set of fundamental modeling concepts to capture semantic knowledge about a problem and its solution 2) a set of views and notions for presenting the underlying modeling information 3) a step-by-step iterative process for constructing models and implementation of them 4) a collection of hints and rules of thumb for performing development

Almost all the methods include a process aspect describing tasks for actors to carry out. Many methods include *process models* describing sequences and concurrency of tasks related to the production of *system models* and documents (like in (Yourdon 1989)).

Two terms frequently mentioned in connection with methods are techniques and tools. A technique is *a way to accomplish the desired state of affairs by a series of steps*

(Welke 1983). We can use modeling techniques, such as Data Flow Diagramming (DeMarco 1978) to support systems analysis and design, brainstorming techniques in co-operation and reviewing techniques to gather information. Description languages, such as UML (Jacobson, Booch et al. 1999), make the conceptualization of the object system shareable among the members of the development group. Tools, such as CASE (Bubenko, Langefors et al. 1971; Orlikowski 1993) and CSCW (Lee and Malone 1990; Grudin 1994a; Grudin 1994b), are used to support methods and development work (Wijers 1991). These automated tools provide support for data analysis, design and implementation phases in systems development as well as for managing the systems development (Nandhakumar and Avison 1999).

2.2 Process models

The process model aspects of the method can be distinguished based on several criteria, but most often they include modeling related processes (way of working) and management related processes (way of controlling) (Olle, Hagelstein et al. 1991). The former describes how the method produces results, the outcomes of the method use, and the latter how the project is planned, organized and managed.

Process models have been used as management tools in systems development since the early days of information systems (IS) and software engineering (SE) fields. Early models were called life cycle models (SDLC) until the term software process was introduced and gained popularity. For example, in the year 1987, the whole conference of ICSE (International Conference on Software Engineering) was dedicated to the theme of software process (ICSE 1987). Lifecycle models present the ideal structure of systems development process focusing on sequences between the activities (e.g. requirement elicitation, analysis, architecture design, detailed design, testing, maintenance), deliverables (specification or document, such as requirement specification) produced in these activities and possible iterations between them (Bubenko 1986). The oldest model, waterfall, models the systems development as a definite set of steps (stages or levels), all the deliverables from one stage being passed to the next level just as water cascades from one level to the next in a natural waterfall without backflow (Georgiadou 2003). As an answer to the pitfalls of the waterfall model, there have been number of improvement to it towards more iterative development, like prototyping (Kautz, Kuhlenkamp et al. 1992) and spiral model (Boehm 1988). Waterfall and spiral models are shown in the following figure (Figure 2).

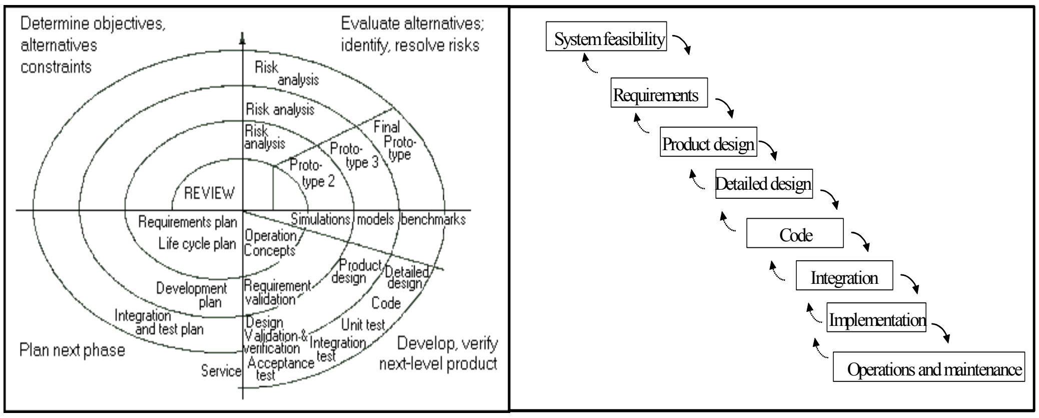


Figure 2. Waterfall (right) and spiral (left) models of the software process. Adapted from (Boehm 1988)

Software process approaches can be classified into software process improvement (SPI) approaches. SPI approaches aim at increasing the maturity and quality of software processes in organizations (Humphrey 1989; Zahran 1998). Kontio (Kontio 1995) points out that the basic philosophy behind these approaches is similar. They are based on Crosby's maturity concept (Crosby 1979) for assessing the maturity of quality management of an organization. SPI and maturity models are founded in the following principles:

1) the product quality is dependent on the process quality
2) the maturity of the process can be characterized by analyzing the existence of a set of practices, and
3) there is an optimal, universal order for implementing these features and compliance to this order determines the maturity of an organization

The most widely known and used maturity model is the Capability Maturity Model (CMM) (Humphrey 1989; Paulk, Curtis et al. 1993) developed by the Software Engineering Institute (SEI). The CMM describes 5 levels of maturity (initial, repeatable, defined, measurable, and optimizing) which indicate process capability. Other SPI models includes ISO 9001 (ISO 1991), Bootstrap (Koch 1993) and Spice (Dorling 1993). The main purpose of SPI approaches is to change the way practitioners develop software emphasizing problem solving, knowledge creation, participation, integrated leadership and continuous improvement in an organization (Mathiassen, Pries-Heje et al. 2002).

The investment in process improvement has been reported to give significant business benefits to organizations, such as improved product quality, reduced time to market, enhanced productivity (Zahran 1998), increased organizational flexibility, along with employee satisfaction (Yamamura 1999). Glass (1999) has evaluated financial gains of different software technologies – structured techniques, fourth generation languages (4GL), computer aided software engineering (CASE), formal methods, the cleanroom method along with object orientation and process models – concluding that the process model data reflects a consistent pattern of improvement gained. Most of these reports originate in the US and there is no empirical evidence on their applicability elsewhere (Abrahamsson 2002). Although many SPI approaches are generally known in Europe, they are not widely used (Kautz and Larsen 2000; Abrahamsson 2002).

2.3 System models

Basically, modeling grew out of techniques of data organization and file design that led to the development of database technology around the mid-1960s (Hirschheim, Klein et al. 1995). We define modeling generally as "*the act of purposely abstracting a model from a part of the universe*" and a model "*as a purposely abstracted and unambiguous conception of a domain*" (Proper, Verrijn-Stuart et al. 2005). By system modeling we mean *abstracting a model from a part of the information system*, and system model as *an abstracted conception of a information system domain*. System modeling is connected with data abstraction from programming languages (Brodie 1979), evolution of knowledge representation and artificial intelligence (Senko 1975). The best-known system modeling approaches are entity-relationships (ER) modeling (Chen 1976) and object-oriented modeling (Booch 1986; Rumbaugh, Blaha et al. 1990; Jacobson, Booch et al. 1999). The late 1970s and 1980s saw a dramatic growth in the number of system models. Semantic models, interpretive predicate logic models and dynamic models were proposed (Hirschheim, Klein et al. 1995). During the 1990's the architecture modeling and architecture descriptions have gained attention in modeling large and complex systems (Zachman 1987; Bernus, Mertins et al. 1998; Boar 1999; Bosch, Gentleman et al. 2002).

During system modeling a perceived object system is defined using a *conceptual structure*. A conceptual structure includes "*a set of concepts and relationships forming a vocabulary to define system models*" (Neches, Fikes et al. 1991). Modeling techniques are based on a subset of the conceptual structure, and thus modeling may stress data, process, behavior, organization or problems aspects of a system (Olle, Sol et al. 1982). The conceptual structure applied in system modeling is typically represented by semiformal graphical notations (e.g. data flow diagrams (Gane and Sarson 1979)), text (e.g. root definitions (Checkland 1981)) or formal languages (e.g. Z (Spivey 1988)).

These system models have one main purpose: to specify the system as early in the development process as possible for the purpose of better understanding the system, validating the requirements and verifying the design and implementation (Bubenko 1986). In this way the system models have a connection between requirement specification and architecture description, which form the system models of the requirements and architecture of the system.

Requirement specification contains a system model of the real world and specifies its problems and requirements. Requirement elicitation is a process, which deals with detecting and representing requirements of the system. It is a part of the requirement development activities in requirement engineering. Requirement specification has been considered an important part of the requirement elicitation process. It serves as a means of communication between the user and the systems developer and represents in a systematic fashion the current state of the real world, its problems and its future requirements (Pohl 1994; Kotonya and Sommerville 1998).

By architecture in this thesis we mean "*the structure of the components of a system, their interrelationships and principles and governing their design and evolution over time*" (Garlan 2001). The use of architecture descriptions and architecture modeling provides a number of important benefits, such as acting as communication and negotiation vehicle between stakeholders, and capturing design decisions and the global structure of the system (Bass, Clements et al. 1998; IEEE 2000).

2.4 Actors and coordination issues

In most cases, systems need to be developed in teams. As Winograd and Flores (1986) observed, collaboration and communication exist in all human action except for the simplest tasks. Teamwork requires tight coordination among all efforts involved in the development process. Kraut and Streeter (1995) argued that coordination becomes much more difficult as project size and complexity increases. Empirical studies show that goup work causes difficulties in projects; for example, communication bottlenecks and breakdowns are common in systems development projects (Curtis, Krasner et al. 1988).

Most systems development methods and process models only give an overview of human interactions. They characterize possible roles for actors to participate in development (Kruchten 2000). Some methods address communication problems far more deeply based on either a political conflict – co-operation game, or a continuous exchange of arguments (Lyytinen 1987b). Communication difficulties between users and developers have been tackled especially in the UTOPIA project (Bodker, Ehn et al. 1987) and ETHICS (Mumford 1983).

The terms communication, collaboration, control and coordination in managing systems development are sometimes mixed. Some definitions separate them, like (Henderson and Cooprider 1994; Vessey and Sravanapudi 1995). Yang (Yang 1995) defines coordination as ordering of activities in the process, collaboration as management of shared data, and communication as an exchange of information between users. Sabherwal (Sabherwal 2003) defines control as improving performance relative to a certain goal when the goals of activities differ from those of the larger overall entity. We use in this thesis the definition of coordination that combine communication, control, collaboration and coordination. We define coordination using the definition by Malone and Crowston (1994): *Coordination is management of interdependencies between activities*. Here, activities can be activities or objects; everything that has dependencies requires coordination (Malone and Crowston 1994). Malone and Crowston (Malone and Crowston 1994) describe four basic coordination processes, which include collaboration and coordination; and support processes including communication and control. A more detailed description of these basic coordination and support processes is given in the publication I of the appendix.

2.5 Related research

"*Reality is the murder of a beautiful theory by a gang of ugly facts* " (Glass 1996; Glass 1997)

As observed in this thesis, the systems development methods used by practitioners have a set of common concepts originating in the structured techniques of the 1960s and 1970s (Fitzgerald 2000). These concepts include life cycle thinking (Curtis, Krasner et al. 1988), design strategies such as functional decomposition (Gane and Sarson 1979) and information hiding (Parnas 1972). Some studies argue that even object-orientation can be traced back to this period and the definition of the Simula programming language (Nygaard and Dahl 1966; Fitzgerald 2000). In the organization of these studies, they used object-oriented method (Booch, Rumbaugh et al. 1998) and combined it with waterfall process model. Recently, agile methods in general and Extreme Programming (XP) in particular have gained a strong following among system development practitioners. Agile methods have been developed by practitioners as an alternative for the complicated traditional methods. Agile methods emphasize communication, development speed, lighter documentation and team effectiveness (Cockburn 2001; Abrahamsson, Warsta et al. 2003). However, no empirical evidence of their practicability or effectiveness exists (Herbsleb, Zubrow et al. 1997; Abrahamsson, Warsta et al. 2003). Although the research community has developed advanced socio-technical approaches to meet the social and human problems in systems development, these are not much used by practitioners (Lyytinen 1987b).

The use of methods in practice is reported limited (Russo, Wynekoop et al. 1995; Wynekoop and Russo 1995; Iivari and Maansaari 1998; Nandhakumar and Avison 1999; Fitzgerald 2000), a phenomenon discovered also in this thesis. Some system developers state not using any method whereas others argue that they tend to use parts of methods rather than following all the steps required by a particular method (Fitzgerald 1996). Previous studies have reported method usage rates from 62 to 87 percent (Russo, Wynekoop et al. 1995; Fitzgerald 1998). These studies assume that if a method is not followed in detail, it is not used at all (Fitzgerald 1998). This does not mean that the method does not have any influence on the development.

Distinctions between levels of method use are important, especially the borders between systematic, ad-hoc, and no use of methods. What does it actually mean when practitioners say that they follow some method? For example how fully should method use be defined and documented, how completely should a method be followed, and how widely spread should the use of an obligatory method be in an organization before we can claim that methods are actually used. Fitzgerald (Fitzgerald 1996) suggests a distinction between formalized and non-formalized methods: a formalized method denotes a commercial or a documented method, and a non-formalized one a non-commercial or an undefined method. By considering only the use of formalized methods the rate of method use drops considerably: from 40 to 20 percent (Fitzgerald 1996). A field study by Smolander et. al (Smolander, Tahvanainen et al. 1990) partly confirms these findings by observing that the methods applied are mostly a collection of loosely coupled informal techniques. Instead, our studies suggest that methods are used as tools that are picked up and chosen according to the situation. These kinds of methods use can be categorized as ad-hoc use of methods.

Some studies suggest that systems development in practice is not done according to any formalized method, but rather by providing learning and guidelines for development participants on how to organize their work (Baskerville, Travis et al. 1992; Mathiassen, Munk-Madsen et al. 1996). This claim supports our findings. Cockburn argues that methods reside in the tacit understanding held between participants, and in their habits of conversation (Cockburn 2001). Nandakumar and Avison (Nandhakumar and Avison 1999) go as far as to say that methods are "treated primarily as a necessary fiction to present an image of control or to provide a symbolic status in today's organization". In contrast, Undhelkas and Mandapur (1995) propose the metaphor of a "road map" for development methods, suggesting that methods may not be able to recognize all situational factors and are more useful for a "foreigner" than for a "seasoned" practitioner.

The following reasons have been reported for the limited use of methods:

- methods do not address the most troublesome aspects of development, especially thin spread of application domain knowledge, conflicting requirements and breakdowns of the communication process (Curtis, Krasner et al. 1988),
- methods do not deal with distributed development (Bubenko 1986)
- methods are too mechanistic and detailed (Nandhakumar and Avison 1999),
- methods do not take into account different development situations (Fitzgerald 1998),
- methods do not consider individual creativity and intuition (Fitzgerald 1998),
- methods do not respect learning over time (Fitzgerald 1998),
- by using methods one might lose sight of the fact that the real objective is the development of the actual system, not the methods (Fitzgerald 1998).

The surveys indicate that local methods are more popular than their commercial counterparts (Russo, Wynekoop et al. 1995). Surveys suggest that from 40 to 60 percent of organizations using methods have developed the methods in-house (Hardy, Thompson et al. 1995; Russo, Wynekoop et al. 1995). Therefore, although organizations develop their own methods, the methods need to be adapted to different use situations in the same way as third-party methods. One of the better known examples of this kind of in-house method is Nokia's OMT++, which has been enhanced from OMT to be used for designing network management systems for mobile phones (Aalto and Jaaksi 1994). The organization described in this thesis had also developed its own mixture of "textbook" methods, which were aimed to be adapted to every project. However, the organization's methods did not guide the adaptation at all. Unfortunately, there is no evidence of whether these in-house methods are used more successfully than textbook methods (Tolvanen 1998). Studies of in-house method development indicate that the selection of methods, their development and their introduction seems to be done in an ad-hoc manner without any control over the adaptations (Smolander, Tahvanainen et al. 1990; Russo, Wynekoop et al. 1995), a finding that is also supported by our studies.

Our studies support Introna and Whitley's (Introna and Whitley 1997) arguments of the use of methods that emerges as a part of our understanding and involvement in the problem situation, and not merely because of the required steps of the methods. Prototyping (Kautz, Kuhlenkamp et al. 1992), for instance, was developed for a situation where the users' information need cannot be sufficiently specified. Various researchers have gone even so far as to develop special contingency frameworks

(Naumann, Davis et al. 1980) determining the relationship between situational factors and the best fitting development strategies.

2.6 Summary

In this chapter, the scope of this thesis and the general orientation to the managing and coordinating systems development process was presented. The chapter started with the description of the information systems development based on Lyytinen's framework, proceeding further to the different methodical approaches to systems development. These methodical approaches were mostly based on the modelling concept: modelling systems development process or object system. We defined coordination with Malone and Crowston's coordination theory, which emphasizes coordination, collaboration, communication and control in managing systems development process in modern development environments.

As a summary, based on the literature and our findings, it can be argued that there is a clear difference between development goals in industry and those in academic research. While industry is looking for practical and simple methods to be able to coordinate and guide the development work, research in academic environments emphasizes new and better multi-stage, heavy and more or less linear universal methods.

3 Research Goal and Methodology

During the research, the shaping of the research goals and problems has been an inseparable part of my learning process. The research started with the analysis of how architecture affects a development project in a software producing company, but continued with the study of the social and technical complexities of systems development. At the end of the process the research turned into a study of how practitioners work with systems development methods. The phenomena have revealed themselves gradually through data, literature, and discussions with industrial practitioners and university researchers. In this chapter, I shall explain how this process has shaped the goals, problems and methodology during the research. I will start by describing how the research problem has been shaped during the process. After that, the research method and the process of the research will be explained.

3.1 The research problem and its shaping

This research project included three phases: studies on how architecture affects a multi-site development project, studies on how requirements were shaped and interpreted during the systems development and how this process is to be estimated, and a study on how practitioners work with systems development methods. Next I will briefly explain how these three phases shaped the research problem:

1. Studies on how architecture affects multi-site coordination of development

   The objective of this phase was to clarify the systems development problems related to software architecture and investigate how practitioners cope with these problems in systems development. This phase consisted of two parts: a qualitative study about social complexities and a quantitative study about technical complexities. During the analysis, the problems analyzed in the qualitative study evolved more to coordination and communication problems for which architecture provided a tool. In the quantitative study, the understanding of the architecture as a size predictor in the project cost estimation got its basic shape. In the same time the analysis method changed from an evolutionary algorithm approach (Zelinka and Lampinen 1999) to a traditional statistical analysis (Lawson and Hansen 1974). This happened during the discussions with university researchers, who were experts in statistical analysis. They advised me to use the traditional and simpler way to analyze the correlation between architecture metrics and development effort for our purposes.

2. Studies of the requirement understanding process

   In the beginning of phase two, I was trying to find coordination problems or problems related to software architecture, but I observed that the problems were more related to requirement understanding and organizational conflicts. This observation shaped my research problem towards the interpretation of the requirement understanding process and how this could be measured to get better estimates of the project timetable along with the architecture measures from the phase one.

   At the end of this phase, the observations so far suggested that methods in the organization played an important role in the case study projects. This led me to shape the study towards the interpretation of the role of methods and their use in the studied organization.

3. Study on how practitioners work with systems development methods

   In the beginning of this phase I started to compare the results of phase one and phase two according to their similarities and differences (cross-case analysis). During this analysis, it appeared that the coordination and the requirements understanding in the projects were the result of using and adapting methods based on the practitioner's background, experience and the development situation at hand.

Although Eisenhardt suggests that early identification of the research question is important, it is equally important to recognize that it may shift during the research process (Eisenhardt 1989). After the constantly varying learning process during the research, the final research questions can be formulated as follows:

How is systems development process managed in practice?

Based on the use of methods and their support for the management of the systems development in our case projects, the research question can be decomposed into the following subquestions:

1) *How do practitioners use methods?*

2) *Do methods give support to systems development practitioners?*

3) *Do theories support practice in systems development?*

In the beginning of my research, I regarded the information (architecture and requirements of the system) as a given input to the systems development and considered software engineers as passive recipients and technical developers of this information. During the research process I observed that software developers are, in practice, more knowledge workers creating new information and knowledge, both of which are increasingly important resources to the modern organizations (Schultze 2000). The knowledge is created through communication, social interaction and negotiation between development participants.

Each of the publications included in this thesis approaches these questions from a distinctive point of view as described later in Chapter 4.

3.2 The selection of research methods

"The proper place to study elephants is the jungle, not the zoo" (McLean 1973)

According to Järvinen's classification (Järvinen 2000a; Järvinen 2000b) of research approaches this research can be categorized into empirical theory-building approach. Järvinen (Järvinen 2000a) defines research approach as a *"set of research methods that can be applied to the similar research objects and research questions"*. According to Järvinen, theory-building approach includes, among others, case study and grounded theory methods. One futher method, action research (Avison, Lau et al. 1999; Baskerville and Pries-Heje 1999), could also be a possible method for these studies. According to action research method, the researcher can be a participant in the systems development in an organization, simultaneously evaluating the use and role of methods. Action research generates a theory that is grounded in action (Susman and Evered 1978). Action produces knowledge to guide practice, which is achieved by modifying the reality as part of the research process itself. However, action research method was not appropriate for this thesis due to the situation the organization faced at the time the research study started.

My background as an engineer and my deep familiarity with the research context (Nandhakumar and Jones 1997) clearly had an influence on the selection of research methods. I had worked in the case study company for five years before starting the research study. When I started my dissertation, I took a leave of absence from the company, which enabled me to get some distance to the projects and data (Nandhakumar and Jones 1997). The main idea in selecting the research methods was that I wanted to look at the data from different perspectives, and perhaps in phases one and two be more convinced of information accuracy also discussed in (Yin 1994). I chose the quantitative and qualitative perspectives. Factors mentioned above favored the case study approach (Eisenhardt 1989; Yin 1994). However, we used grounded theory method for analyzing the qualitative data in the phases one and two.

Before going further, a brief description of the case study approach and grounded theory is necessary. A brief description of the quantitative method used in these studies is also given.

A case study is a research approach, which focuses on understanding the dynamics present within single settings (Eisenhardt 1989). Bembasat, Goldstein and Mead (1987) give the following definition of case study research:

"A case study examines a phenomenon in its natural setting, employing multiple methods of data collection to gather information from one or a few entities (people, groups, or organizations)"

Case studies can involve either single or multiple cases, and numerous levels of analysis (Yin 1994). Case studies typically combine data collection methods such as archives, interviews, questionnaires and observations. Evidence may be qualitative, quantitative, or both (Eisenhardt 1989; Yin 1994). Finally, case studies can be used to accomplish various aims: to provide a description (Kidder 1982; Eisenhardt 1989), test a theory (Pinfield 1986; Eisenhardt 1989) or generate a theory (eg. (Gersick 1988; Eisenhardt 1989). Theory-building case study research can use a priori constructs to help shape the initial design of the theory-building process (Eisenhardt 1989). However, Eisenhardt makes a distinction between within-case analysis and cross-case analysis, which is a specific feature of the theory-building case study research approach (Eisenhardt 1989). Within-case study analysis typically involves detailed case study write-ups for each site. The cross-case analysis compares the data with different techniques across cases, thus improving the researcher's ability to process information in novel ways (Eisenhardt 1989).

Grounded theory is a research method developed originally for social sciences by Glaser and Strauss in the 1960s (Glaser and Strauss 1967). It was later developed further and reinterpreted by the original authors (Strauss and Corbin 1990) and others (e.g (Eisenhardt 1989; Locke 2003)). The basic tenet of this approach is that a theory must emerge from data, or in other words, a theory must be grounded in data. Hence the method is more inductive than deductive. As defined by two of its major proponents (Strauss and Corbin 1990), *"the grounded theory is a qualitative research method that uses a systematic set of procedures to develop an inductively derived grounded theory about a phenomenon"* (p. 24). The intent is to develop an account of a phenomenon that identifies the major constructs or categories in grounded theory terms, their relationships, and the context and process, thus providing a theory of the phenomenon that is much more than a descriptive account.

Grounded theory requires that theory is emergent from data, but does not see these as being separate. Data collection, analysis and theory formulation are regarded as reciprocally related, and the approach incorporates explicit procedures to guide them. Research questions are open and general rather than specific hypotheses, and the emergent theory should account for a phenomenon which is relevant and problematic for those involved. Analysis involves three processes from which sampling procedures are derived and which may overlap: *open coding*, where data is broken up to identify relevant categories; *axial coding*, where categories are refined, developed and related; and *selective coding*, where the "core category", or central category that ties all other categories in the theory together, is identified and related to other categories (Glaser and Strauss 1967). Data collection is guided by *theoretical sampling*, or sampling on the basis of theoretically relevant constructs. Two key procedures, asking questions and making comparisons, which Glaser and Strauss call constant comparison (Glaser and Strauss 1967), are specifically detailed to inform and guide analysis and to aid theorizing. Other procedures, such as memo writing and the use of diagrams, are also incorporated as essential parts of the analysis, as are procedures for identifying and incorporating the interaction and process. The need for a high level of theoretical sensitivity on the part of the researcher is explicitly promoted. The method of the grounded theory is iterative, requiring a steady movement between concept and data, as well as comparative, requiring a constant comparison across types of evidence to control the conceptual level and the scope of the emerging theory (Locke 2003).

Case study research is the most common empirical approach used in studying information systems (Orlikowski and Baroudi 1991; Alavi and Carlson 1992). Most famous examples of cases studies can be exemplified as (Markus 1983; Curtis, Krasner et al. 1988; Broadbent and Weill 1993; Earl 1993; Orlikowski 2002). The grounded theory approach is applied in a number of information systems studies, such as those by Orlikowski (1993), Scott (1998), Calloway (1991), Priese-Heje et. al. (Pries-Heje, Baskerville et al. 2004) and software engineering studies (Coplien and Devos 1999; Purao, Rossi et al. 2002).

The quantitative analysis method was chosen based on the study question and chosen data, as recommended in (Chelimsky 1992). The research aim was a correlation analysis and the method was a simple linear regression model. We used this simple linear regression model to calculate the correlation between the metrics of the system and the development effort. The other purpose of the quantitative analysis was to demonstrate the use of metrics in project timetable estimation. In this method, it is assumed that the correlation is linear between metrics, and the systems development effort is linear. We chose this linear model because of the small sample of data and also to demonstrate how prediction can happen with this kind of simple model. In reality, the systems development is not linear, and the effort estimation should happen with non-linear methods, such as in studies of Venkatachalam and Pedrycz (1993), Peters et al. (1999). We could get sufficiently reliable results for the correlation analysis although for the effort estimation this analysis was only the first attempt to estimate the project timetable and effort. In phase one, we complemented the quantitative analysis with metaphorical analysis (Lakoff and Johson 1980; Frost and Morgan 1983; Schultze and Orlikowski 2001) to better understand the studied phenomenon.

3.3 Research process

In this chapter, a detailed description of the three phases of the research process is given. First, the preparations made for the studies are described. Then, the processes of data collection, data analysis and hypothesis shaping are characterized. In the end, we discuss the finishing and reporting procedure of the research studies.

3.3.1 Preparing for the studies

The beginning of theory-building studies includes an initial definition of the research question, a specification of a priori constructs, a selection of cases and crafting instruments and protocols (Eisenhardt 1989). The shaping of the research question is already discussed in Chapter 3.2, but the others need more clarification in the following.

Specification of a priori constructs can help shape theory-building research (Eisenhardt 1989). This is also identified later in the grounded theory approach as a form of seed category (Miles and Huberman 1984). In phase one, a notion of the common object from Malone and Crowston's coordination theory (Malone and Crowston 1990; Malone and Crowston 1994) was used as a starting point to interpret the coordination in the project. In phase two, the concept of a technology frame of reference (Orlikowski and Gash 1994) was used to interpret the requirement understanding in the project. Quantitative studies in phases one and two included hypotheses testing studies, and the interpretations from the qualitative studies (in phase on coordination problems and in phase two requirement evolution) were used as a priori constructs.

The selection of cases relied on the theoretical sampling principle (Glaser and Strauss 1967), in which cases are chosen as extreme situations and polar types in which the process of interest is "transparently observable". The sampling plan of the current study was designed to be built around projects displaying problems in systems development, big problems that caused delays to the project's timetable. Within these projects in the studied organization, we chose projects of polar types: one project had problems inside the project, the other problems with the customer; one was smaller and the other one bigger; they both produced service platforms for different business areas. The analysis revealed that the projects had even more different features, such as the orientation, attitudes and experience of the participants, and the communication between participants that extended the emergent theory (Eisenhardt 1989). To facilitate iteration and comparison, which is an inevitable feature of the grounded theory method (Locke 2003), these two projects were analyzed one by one, a strategy also adopted by (Orlikowski 1993).

Crafting instruments and protocols. A combination of qualitative and quantitative data collection methods were used in both case projects, which is typical for theory-building researchers (Eisenhardt 1989). The relationship between qualitative and quantitative data was two-way: the qualitative data was used for understanding the metrics and their relationships in quantitative analysis and quantitative data was useful for the visualization of the phenomena found in the quantitative data. Mintzberg describes their relationships in the following way: "We uncover all kinds of relationships in our hard data, but it is only through the use of this soft data that we are able to explain them" (Mintzberg 1979). In both quantitative studies, the multiple investigators were used in the analysis, but also in the interpretation of the results. They often had complementary insights and different perspectives also on the qualitative studies that gave novel insights into the data, and they also enhance the confidence in the findings (Eisenhardt 1989).

3.3.2 Data collection

During the studies, most of the data was collected from project extensive documentation based on the dynamic process of data collection (Glaser and Strauss 1967), where samples were extended and focused according to the emerging needs of the theoretical sampling. In both case projects, the project documentation data was complemented with interviews among project participants. Table 2 shows the data available from both the projects.

The interviews were all tape-recorded and completely transcribed. The length of the interviews varied from half an hour (focused interviews) to two hours (group interview). Several hundreds of pages of project documentation, the transcribed interviews and 170 000 lines of source codes were analysed during the studies.

In phase one, a group interview was carried out in the form of a project kick-out meeting. In the meeting, the project participants discussed what happened in the project, what was good and bad, what the major problems were and how they would do things differently next time. The comment 'a lot of wasted work' from one designer in this kick-out meeting characterizes the feelings of the participants: a lot of useless documentation, like use case descriptions and architecture and module design documentation that were produced in the project. They could not keep all these documents up-to-date and use them in the project. Also, the excessively 'heavy' process model and the lack of iterations were clearly mentioned as drawbacks of the project in the group interview.

During phase two, focused interviews were conducted to identify the reasons for changes in the requirements of the project. The project participants were asked to reflect on the project's history by showing the analysis and implementation models of the system and to describe their understanding of what happened in the project between the requirement analysis and implementation phases of the project.

The data for the quantitative statistical analysis in both phases one and two was collected from the architecture and component design specifications, source code, project management database and bills from subcontractors. In the project management database, the data included the time spent on each task by the project participants. These tasks were divided according to phases used in projects. In the cases where foreign consultants were involved in the development work, the development effort data was taken from the subcontractors' bills.

3.3.3 Data analysis

In these studies, we used Eisenhardt's principle of within-case and cross-case analysis to interpret the findings in different phases of this thesis (Eisenhardt 1989). In the within-case analysis "the overall idea is to become intimately familiar with each case as a standalone entity" allowing unique patterns to emerge before trying to generalize patterns across cases (Eisenhardt 1989). In the first two phases of the research the qualitative data analysis was based on the grounded theory (Glaser and Strauss 1967; Strauss and Corbin 1990). In those phases, quantitative data analysis with a simple linear regression method (Lawson 1995) was carried out. I conducted the qualitative analysis alone in all the phases, but, in phases one and two, two other researchers did the quantitative data analysis and provided complementary insights (Eisenhardt 1989) into the qualitative analysis in our discussions. As outsiders of the qualitative data collection, they were able to look at the data more with greater objectivity (Nandhakumar and Jones 1997) than I was, which facilitated a more reliable data analysis as a whole.

In the first and second phase of the thesis, the qualitative data analysis started quite early on in the studies, right after the data on the project meeting minutes was collected. In the first phase, data collection continued with design specifications simultaneously with the analysis of the data on the project meeting minutes. In phase two, the collection of requirement specification data started while the analysis of the data on the project meeting minutes was still being performed. The quantitative data collection started as soon as some initial findings of the qualitative data analysis were made. Such overlap of data collection and analysis is strongly recommended by (Eisenhardt 1989) and (Strauss and Corbin 1990). Within qualitative data analysis in both phases, the coding procedure included three phases: open coding, axial, coding and selective coding, which proceeded iteratively through the analysis process (Glaser and Strauss 1967). To help manage the quite extensive amount of information and the analysis process, the Atlas.ti (Scientific Software 2001) tool was used. It helped in the analysis process, for example in the retrieval of categories, memo making and recording of semantic relationships.

The quantitative data analysis was hypotheses testing in nature and naturally used a priori constructs. Both hypotheses were based on the initial findings of the corresponding qualitative studies. To the statistical data analysis we used the Matlab Optimization Toolbox (MathWorks Inc. 2003).

In the following we describe the data analysis process in the three phases of the thesis.

Table 2. Data available from the case projects

Data/Document/Artifact DS project	*Data/Document/Artifact EC project*
15 Progress Report (from Project Manager)	15 Progress Reports (from Project Manager)
Project management Software: Plan vs. Actual costs	Project management Software (Niku Workbench): Plan vs. Actual costs, development effort
11 Project Steering Group Meeting Minutes	16 Project Steering Group Meeting Minutes
46 Project Group Meeting Minutes	26 Project Group Meeting Minutes
Project Plan	Project Plans
Functional Specifications	Requirement Specification document
Requirement Catalogue	Project Quality Criteria document
Risk analysis document	Architecture Specification
Project Quality Criteria document	26 Module Specifications
Architecture Descriptions	22 Tool subsystem UI specifications
Module Specifications	Kick-off presentation, Steering Group kick-out meeting minutes
Group Interview with project participants	Focused interviews of three BU members (development manager, team leader, designer)
Group interview slides	Focused interviews of four IDU members (steering group representative, project architect, 2 project designers)
Source code (Lines of Codes) 138 000 LOC	Source code (Lines of Codes) 32 000 LOC

Phase one

An overview of the qualitative research process has been given in the Figure 3 of Publication I. As we can see from this figure, the first phase of the data analysis was open coding. *Open coding* started with the identification of problems and deviations related to coordination and software architecture in the project progress, using mainly project meeting minutes and the group interview. We further used architecture and design specifications to help pinpoint the problems. We observed in total 329 deviations and problems related to software architecture and coordination.

In the *axial coding* phase, we used a notion of common object as a seed category (Miles and Huberman 1984) based on Malone and Crowston's coordination theory (Malone and Crowston 1990; Malone and Crowston 1994) to help in the interpretation of coordination problems in the project. We identified two types of common objects from the study. The following example (Figure 3) shows a translated excerpt of a transcript after axial coding. This example shows how common objects were interpreted from the project material.

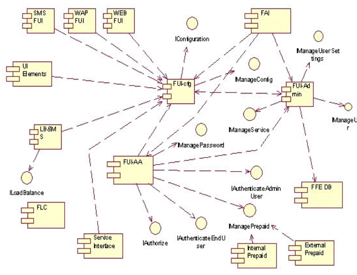
”On the Server side, we gained experience from new technologies, like XML, XSL and CORBA”	Technical orientation of the Server Common object: development activity
”We had a lot of problems with Client and Server synchronization. The Client was first and the Server was behind, it should be wice versa ” ”The Client-Server interface was dependent on core resources”	Interface and interdependence problems Common object: component
”AA and MS&LB need communication with architect and designer”	Communication problems Common object: development activity
 SMS FUI WAP FUI WEB FUI FAI IConfiguration IManageUserSettings UI Elements FUI-cfg IManageConfig FUI-Admin IManageUser LB&MS IManageService IManagePassword FUI-AA ILoadBalance IAuthenticateAdminUser FFE DB FLC IManagePrepaid Service Interface IAuthorize IAuthenticateEndUser Internal Prepaid External Prepaid	Interface problems Common object: component
”The second actual buiold was made 24th August, FFE not ready for testing”	Assembly order problems Common object: development activity

Figure 3. Translated excerpt of a transcript after axial coding in the phase one

The interpretations that produced the categories on the right side of the example (Figure 3) required much knowledge about the organization, its way of documenting and the language used. For example, the UML diagram in the middle describing the components and their interfaces was interpreted as an interface problem having many interdependencies between components.

The analysis also included memoing, where hypotheses and important general observations from the data were recorded (Strauss and Corbin 1990). The following figure (Figure 4) shows an example of an early memo describing the observations on the interdependencies between system components.

```
MEMO: Core architecture description: how CORE communicate with
FFE->60:2 (1 Quotation) (Super, 11.07.02 10:32:31)
  P60: Core_Components_1.JPG
       173 - 363
No codes
No memos
Type: Commentary

   description of Server services to describe the
   interface between Server and Client is described in
   Tech_Spec_ Core services document. This was
   ready in Server DP3 phase, Client DP3 was month
   before, so Client implementation was going on one
   month before they new what kind of services
   Server is offering them.
```

Figure 4. Memo describing the observations on architecture as coordination tool in phase one

In the *selective coding* phase, the identification of common objects helped in finding the interdependencies between activities that caused coordination problems in the project. We identified three interdependencies between components and three interdependencies between development activities. After that, we made the cross-case analysis (Eisenhardt 1989) to determine the differences between same-site and multi-site coordination in these interdependencies.

In the quantitative analysis, we used metaphorical analysis (Lakoff and Johson 1980; Frost and Morgan 1983; Schultze and Orlikowski 2001) to help understand the architecture of the system. According to our metaphor of architecture drawing, we identified three categories that described the architecture of our case study system best. Within these categories, we attempted to select the properties, which were simple and could be calculated based on project design specifications. We chose the simple linear prediction model to analyze the correlation between architecture properties and systems development effort. From these seven property values for six components, a total of 42 property values were calculated. From these property values we calculated coefficient values using Matlab Optimization toolbox. In the end of this quantitative process, we calculated the model errors to determine the quality of our cost effort estimation model and analyzed the results based on coefficient values.

Phase two

An overview of the qualitative research process is given in Figure 2 of Publication III. As the figure indicates, open coding was the first phase in the data analysis. *Open coding* started with the identification of problems and deviations in the project progress. During the development, these were issues that were brought to the project meetings for discussion and decision-making. The steering and project group meeting minutes were the main sources for problem and change identification. We observed in total 150 problems and deviations related to project progress. Most of the concerns brought to the steering group were related to the other subsystem and its requirements. Also the system development styles and strategies caused concerns.

To better understand the requirements of the system, we investigated in more detail the requirements specification document. We were able to extract only four initial requirements that were related to the other subsystem. Our analysis continued as we used three conceptual models of both subsystems developed in the qualitative study.

Through these models, we were able to grasp how the subsystems evolved through different phases of systems development. The content of these conceptual models suggested to us that the other subsystem's requirements changed considerably during the process. This led us into investigating further why this subsystem's requirements changed so much, while the other subsystem's requirements remained stable.

To answer this question, we made focused interviews among the project participants to identify the reasons for these changes. Project participants were asked to reflect on the project's history by showing the analysis and implementation models of the system and to describe their understanding of what happened in the project between the requirement analysis and implementation phases. Four of the interviewed project participants were from the development side and three from the business side. Business side representatives were also asked about the competences and processes of development side during the time the project was running and how these competences and processes had evolved after that. The interviews were audio taped and fully transcribed to preserve all the details.

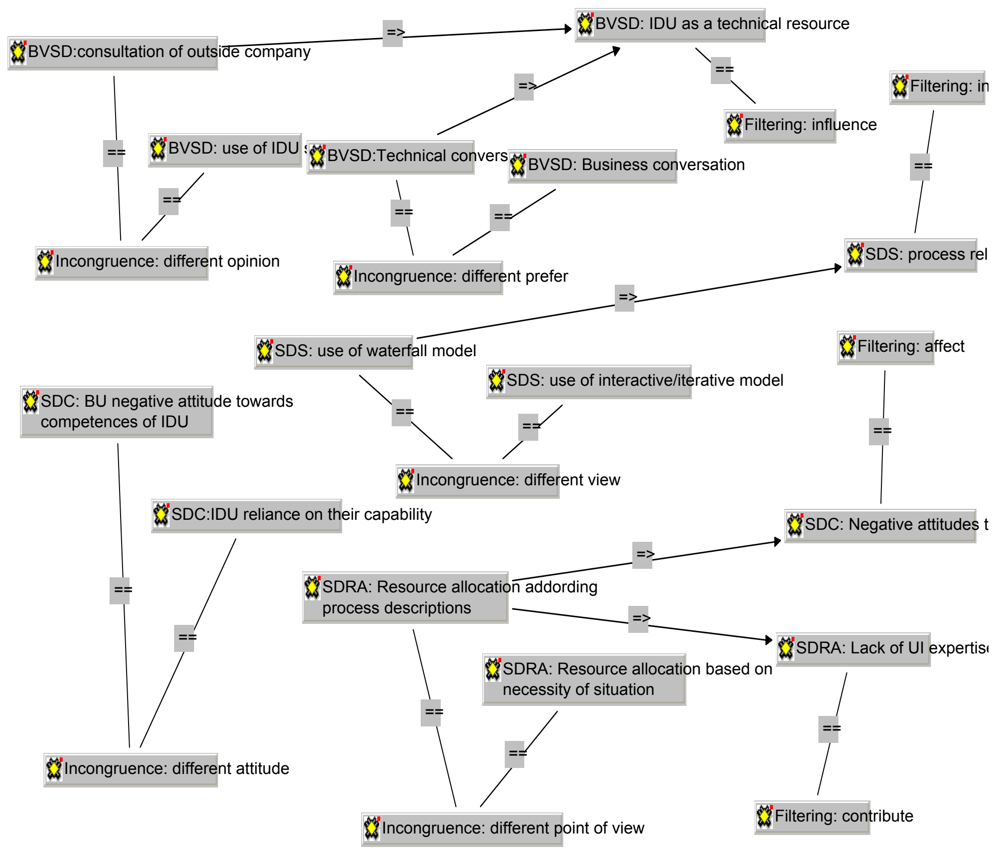
Based on the interviews, we observed that requirements did not change that much during the project, but the understanding of them changed. The open coding proceeded in parallel, treating each interview as confirmation or further development of results from earlier findings. During this process, the categories developed gradually. First, we identified quite concrete concerns of system development. In further analysis, we found more subtle contextual attitudes and expectations about how systems development should be performed. These attitudes and expectations were so strongly visible in the data that we could interpret them as technology frames (Orlikowski and Gash 1994). In further analysis, we used technology frames as a priori construct or lenses to the data. This further analysis consisted of group analysis, cross-group analysis and re-examination of the categories. This iterative examination of the data yielded five categories of technology frames, which were used as a basis for the next phase, axial coding.

During the *axial coding*, we identified four processes of stereotypical "tensions" between these attitudes and expectations, which affected how project participants emphasized these frames of understanding in different phases of the project. Figure 5 shows a translated excerpt of a transcript after axial coding.

"…In the beginning of the project, we did not understand how big the other subsystem was. The customer gave us few sketches of user interfaces and we thought that it was like these… We did not want to involve with UI things, we concentrated on platform …We left these user interfaces to the UI designer, who came into the project during the design phase according to our process model… When Ann and UI designer came to project, we started to understand the tool functionality better"	System development capability: Designers did not want to involve UI things Frameshift: more UI capable resource to the project Origins: processes
„…We did not know very much about designing UI. We had some experience of the UI courses in the university and we felt that it was time consuming and boring…"	System development capability: No capability of UI development Origins: technical education
"…Nobody wanted to be involved in the UI thing... So we left it out in the requirement gathering and concentrated on the platform only ..."	System development capability: Concentration on the platform Filtering: Lack of UI expertise Origins: company's history
"…We left these user interfaces to the UI designer, who came into the project in the design phase according to our process model…"	System development strategy: Use of strict process model Filtering: Inhibited the understanding of requirements Origins: processes

Figure 5. An example of a transcript after axial coding using Atlas.ti

In the *selective coding* phase, the causal relationships between categories were recorded with Atlas.ti's semantic network capability. Figure 6 shows an example of such a network diagram. In the figure, the boxes represent categories, the arrows the interpreted causalities between them, and the lines simple associations between the categories. The abbreviations BVSD (business value of system development), SPS (system development strategy), SDC (system development capability) and SDRA (system development resource allocation) correspond to the categories of frames of understanding.



BVSD:consultation of outside company
=>
BVSD: IDU as a technical resource
==
=>
Filtering: influence
Filtering: in
==
BVSD: use of IDU
BVSD:Technical convers
BVSD: Business conversation
==
==
==
Incongruence: different opinion
Incongruence: different prefer
SDS: process rel
=>
SDS: use of waterfall model
Filtering: affect
SDS: use of interactive/iterative model
SDC: BU negative attitude towards competences of IDU
==
==
==
Incongruence: different view
SDC:IDU reliance on their capability
SDC: Negative attitudes t
=>
SDRA: Resource allocation addording process descriptions
==
=>
SDRA: Lack of UI expertis
==
SDRA: Resource allocation based on necessity of situation
==
==
==
Incongruence: different attitude
Filtering: contribute
Incongruence: different point of view

Figure 6. An example of semantic network diagram using Atlas.ti

Based on project material, interviews and analysis we formulated the project narrative to trace how the participants' attitudes and expectation influenced the systems development process. In the end of the research process, the project narrative was sent by email to the project manager to get her opinion on the correspondence of the narrative with the reality. She adjusted some details, which did not affect the main findings.

In the quantitative analysis, we formulated the metric describing the identified requirement evolution in the project. The metric was quite simple: it calculated the concepts found during the analysis and implementation models of the system. These analysis and implementation models were formed based on the project specifications.

In the statistical analysis, we used the same simple prediction model as in phase one of the study. The other metrics needed were chosen based on simplicity and wide usage. Using this prediction model, we calculated the correlation between metrics chosen and the development effort. In the end of the process, we formulated the model errors to determinine the reliability of our prediction model and analyzed the results.

<u>Phase three</u>

In phase three, we used cross-case analysis to interpret the final results in this thesis (Eisenhardt 1989). We searched for cross-cased patterns to compare the multi-site and same-site development by listing their similarities and differences (Eisenhardt 1989). We selected pairs of cases and listed similarities and differences between each pair. In this phase, the number of cases was actually three because one of the case projects consisted of two subprojects. The cross-category table produced in this process is shown in Table 3.

Table 3. Cross-category table between projects in the studies

Category	*Case Project 1 subproject A*	*Case Project 1 subproject B*	*Case Project 2*
Adaptation to method	yes	no	yes
Problems	minor technical inside project	inside project, coordination and architecture understanding	conflicts with client, requirement understanding
Timetable estimation	no major problems	problems	problems
Orientation and attitudes	Architect's technical orientation	Architect's business orientation	Designer's positive attitude towards UI and windows
Experience	Experienced project group	Unexperienced project group	Unexperienced project group
Communication	good	bad	in the beginning bad, later good
Understanding the system	fairly good	not very good	in the beginning no, learning to understand
Understanding the development situation	yes	no	in the beginning no later yes
Method purpose	communication and coordination	method use failed	communication and learning
The most meaning metric (relative to development effort)	Coupling (NIC)	-	Requirements Creep

3.3.4 Shaping the hypothesis

From the within-case analysis, the cross-case analysis and overall impressions, tentative tenses and concepts and their relationships begin to emerge, which is called hypothesis shaping (Eisenhardt 1989). The idea is that researchers constantly compare emergent theory and "raw" data – iterating towards a theory with closely fit data (Eisenhardt 1989).

In the hypothesis shaping, we used the semantic network diagram capability of Atlas.ti software (Scientific Software 2001). This semantic network is shown in Figure 7 below. This network shows the relationships between the core categories used to interpret the role of methods.

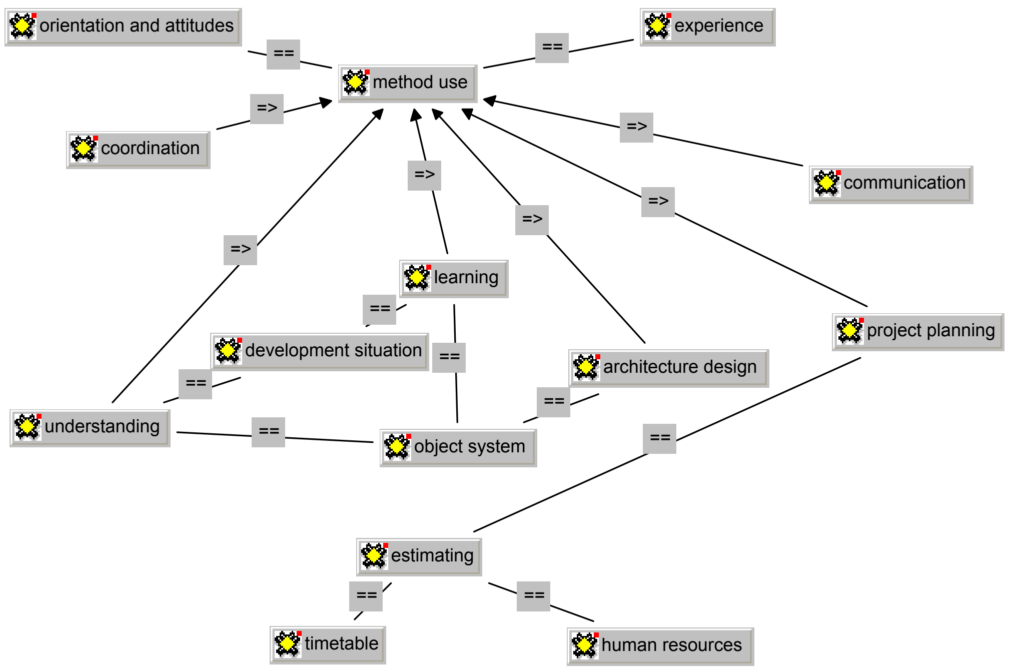


Figure 7. Semantic network of the role of methods produced with Atlas.ti

3.3.5 Finishing and reporting the studies

Eisenhardt (Eisenhardt 1989) distinguishes the phase "enfolding literature". By this phase Eisenhardt means the comparison of the findings with similar and conflicting literature. The aim of this phase is to raise confidence, creative thinking, and the validity, generalizability and conceptual level of the findings. Yin (Yin 1994) refers to this as "analytic generalization" to distinguish it from the more typical statistical generalization that generalizes from a sample to a population. In phase one, the main comparisons were done with Malone and Crowston's coordination theory (Malone and Crowston 1990; Malone and Crowston 1994)(Publication I) and cost estimation literature (Putnam 1978; Albrecht 1979; Boehm 1981; Verner and Tate 1992)(Publication II). The results in both studies were related to existing literature. The comparisons of phase two (Publication III and Publication IV) were made with traditional requirement engineering approaches (Pohl 1994; Kotonya and Sommerville 1998; Jarke, Rolland et al. 1999; Wiegers 1999) and existing socio-technical approaches to requirement elicitation (van Lamsweerde 2000; Bergman, King et al. 2002; Davidson 2002; Tomayko 2002), especially the concept of a technological frame (Orlikowski and Gash 1994; Davidson 2002). Both these provided conflicting and similar concepts and patterns, which both provided an alternative and more creative view to our findings. In phase three, the findings were compared to a few empirical studies of the role of methods in systems development (Russo, Wynekoop et al. 1995; Unhelkas and Mandapur 1995; Fitzgerald 1998; Iivari and Maansaari 1998; Nandhakumar and Avison 1999; Fitzgerald 2000).

The reporting of this thesis included five publications. Reports from the first phase of the thesis described the role of architecture as a coordination tool in multi-site development (Publication I) and as a predictor of system size (Publication II). The second phase of the thesis includes reports of the requirement understanding process (Publication III) and of measuring the requirement understanding process (Publication IV). Phase three was a summary of previous reports concentrating on the role of methods in systems development (Publication V).

Nearing the end of the studies using the grounded theory, the researcher should always ask the question "Are we reaching theoretical saturation in our study?" Theoretical saturation simply means the point at which incremental learning is minimal because the researchers are observing phenomena seen before (Glaser and Strauss 1967). Quite often the practical reality poses some restrictions, such as time and money (Eisenhardt 1989). Or as Locke states: "the practical reality is that as researchers we will have to decide on and articulate the story our data makes it possible to tell. My own experience is that after a time, analysts find that the conceptual categories we have in process are developed at the point where they are able to account pretty much for our data, and we come clear about our story" (Locke 2003). In my studies, the arguments against reaching the theoretical saturation were that after phase two the interviews with practitioners in the organization gave me the feeling that the systems development process in the organization was sufficiently studied, and adding more cases from this particular organization would not add much substance to the findings. At the end of phase three, I observed that the cross-case analysis was quite suitable for our data and the story behind the role of methods was quite clear. Extending the studies to include other organizations would give more substance and diverging findings, but this is a topic for further studies. Perhaps based on this argumentation, I can say that this thesis has reached its theoretical saturation.

3.4 Summary

This chapter described how the research problem has been shaped during the process, explained the research methods and explicated their selection.

The research consisted of three phases following an empirical theory-building case study approach. The whole research process included within- and cross-case analysis. Phase one (within-case analysis) was the case study of the architecture and coordination. Phases consisted of quantitative and qualitative studies reported in Publications I and II. Phase two (within-case analysis) consisted of quantitative and qualitative studies on requirement evolution and understanding, reported in Publication III and Publication IV. In phase three of this thesis we interpret the final result using cross-case analysis of phases one and two. The findings of this phase are reported in Publication V.

Within each of these phases, we used different research methods. In within-case analysis, we used grounded theory method to interpret the qualitative finding and linear regression analysis for interpretation of quantitative findings. In phase three, the final results were interpreted using cross-case analysis. Phase one consisted of one case with two subcases, phase two of one case and phase three of all these cases. In phases one and two we used a priori constructs. In all phases we used literature analysis to compare the findings with similar or conflicting literature. Table 4 summarizes the essential methodical details and the publications including the findings.

Table 4. Summary of methods used in the studies

Research process	*Phase one*	*Phase two*	*Phase three*
Research approach	Within case analysis	Within case analysis	Cross-case analysis
A priori constructs	The notion of common object Hypothesis that coupling and cohesion has impact on development effort	The concept of technology frame of reference Hypothesis that requirements creep correlates with development effort	
Case selection	One case with two subcases	One case	Three complementary cases
Data collection	Project documentation, group interview	Project documentation, focused interviews	Data from previous phases
Data analysis	Grounded theory using Atlas.ti, memoing, cross-category tables Linear regression model, function optimization using Matlab, metaphorical analysis	Grounded theory using Atlas.ti, memoing, semantic networks Linear regression model, function optimization using Matlab	Cross-category analysis, cross-category tables, semantic networks
Enfolding literature (comparison of the findings with similar or conflicting literature)	Coordination theory and cost estimation literature	Traditional requirement engineering and socio-technical requirement engineering	Empirical studies of methods use
Reporting	Publication I, Publication II	Publication III, Publication IV	Publication V

4 Summary of the Publications

In this chapter, I will present short summaries of the publications that form the main contribution to this thesis. The results of this research are presented in detail in the appendix consisting of five publications. These publications have been published separately in scientific conferences (Publications II, IV and V) and in scientific journals (Publications I and III). The summaries are presented in a uniform manner, where first the research objectives and methods are mentioned, then the results are overviewed, and the relation to the whole thesis is summarized. Before the summaries, I will try to sketch my contribution and relation to other researchers in the joint publications (Publication I, Publication II and Publication III).

The first article describes the role of architecture in coordinating distributed development work. The second article presents the early ideas of architecture possibilities to predict the system size and estimate systems development effort. In the third article, the nature of the process of requirement understanding is characterized. The fourth article expands the second article by presenting early ideas of how the process of requirement understanding can be measured quantitatively and used for estimating the project timetable. The fifth article summarizes the first four articles by broadening the earlier findings with the systems development method use.

4.1 About the joint publications

Publications I, II and III are joint publications written together with other researchers. Publication I was written with two other researchers, Matti Rossi and Pentti Marttiin. They both brought a valuable contribution to the writing process. I completed the analysis work and two earlier versions of the paper for the conferences on my own.

Pentti Marttiin elaborated the paper with more thorough literature analysis of coordination and multi-site development, which also helped me see the results more clearly. Matti Rossi helped with focusing the study and formulating the paper into an acceptable form. Publication II was the result of work with another researcher, Alexandre Bern, who wrote his Master's thesis on this subject. Together we created the metrics describing the architecture of the system during intensive conversations, interaction and cooperative work. Alexandre Bern completed the actual calculations with the data, I did the metaphorical analysis, and together we interpreted the results and wrote the publications. Publication III was written with two other researchers, Matti Rossi and Kari Smolander. My task as main author was to write the first version of the paper. Matti Rossi contributed to the analysis of requirements engineering approaches and Kari Smolander to the interpretation of the research results. Although I completed the analysis work alone, Matti Rossi and Kari Smolander helped me with focusing the results.

4.2 Publication I: Architecture as a Coordination Tool in Multi-Site Software Development

This paper appeared in the scientific journal named Software Process: Improvement and Practice., 8(4), pp. 233-248, John Wiley & Sons Ltd.

4.2.1 Research objectives and methods

This first publication aimed at understanding the role of architecture in multi-site software development. The study was performed in a project that developed a directory service platform for international markets. The development took place in three different sites in Finland with foreign designers also participating in the project. The project was divided into two subprojects to facilitate easier management. The division was carried out on the basis of the architecture and technology: one subsystem had a highly distributed, component-based architecture and the other was a centralized subsystem, which handled authentication, authorization and user interfaces. The functionality of the services required the subsystems to communicate only through an easily extensible and configurable interface.

The situation and the goals in the project were topical: multi-site coordination, which has become more common in the changing global environments (Castells 1996; Finholt, Rocco et al. 1998), and information systems architecture (Zachman 1987; Karimi 1988) that was considered a tool of managing the changes and uncertainties in technology and business (Weill and Broadbent 1998). The focus of this study was twofold: First, we studied the coordination problems in the project using grounded theory (Glaser and Strauss 1967), identifying categories of processes that explained most of these problems. Second, we used these categories to identify differences between same-site and multi-site situation (Eisenhardt 1989). The actual interpretation of results was based on the comparison of Malone and Crowston's coordination theory (Malone and Crowston 1990; Malone and Crowston 1994).

4.2.2 Results

We observed that it was not enough to coordinate only the activities and subsystems (Curtis, Krasner et al. 1988; Kraut and Streeter 1995; Grinter, Herbsleb et al. 1999; Herbsleb and Grinter 1999; Carmel and Agarwal 2001) in the multi-site environment, but it was also important to coordinate the interdependencies between the activities to achieve a working system (Malone and Crowston 1990). The main mechanism for coordination was the architecture of the system that described the subsystems or components and their interfaces (Garlan and Perry 1995). This kind of shifting of the coordination interests from activities to their interdependencies required software architects and designers to have a common understanding of the architecture. This common understanding also required that the chief architect was capable of maintaining the integrity of the architecture and communicating it (Smolander 2002). Without this common understanding of architecture, the participants would have to communicate the interfaces, which is naturally more difficult and time consuming across distributed sites with cultural differences and long geographical distances. When both communication and common understanding of the architecture failed, it caused software integration problems (Herbsleb and Grinter 1999).

4.2.3 Relation to the whole

This study formed the basis of my notion of the role of methods. In this study, the practitioners tried to follow the method phases strictly, but were not able to as the methods did not guide the coordination work. The methods were used mainly for communication purposes. The role of methods was mainly to support learning the development situation at hand.

4.3 Publication II: Architecture as a Predictor of System Size - A Metaphor from Construction Projects

This paper appeared in the Proceedings of the 16th International Conference on Advanced Information Systems Engineering, the CAISE' 04 Forum, pp. 193-203. The earlier version of this paper was published in the Proceedings of the ACM ESEC/FSE International Workshop on Intelligent Technologies for Software Engineering WITSE03, Helsinki, Finland, in September 2003. Here I will present the later version.

4.3.1 Research objectives and methods

This publication aimed at providing a different, quantitative view of the problems observed in the first publication (Publication I). The objective of the publication was twofold: 1) to create a metric suite to describe the architecture based on empirical data from the directory service platform project and 2) to analyze the correlation between the metrics and the actual development effort of each component. For the correlation analysis we used an applied mathematical principle, namely the function optimization method named non-negative least square problem analysis (Lawson 1995). By this, we were aiming to get the first empirical evidence of the usability of the architecture as a predictor of system size to be used for estimating the development effort in the project (Putnam 1978; Albrecht 1979; Boehm 1981; Verner and Tate 1992; Boehm and Fairley 2000). The metaphorical analysis was used mainly for understanding the role of architecture as a size predictor (Schultze and Orlikowski 2001) and to make better sense of the new situation (Frost and Morgan 1983; Schultze and Orlikowski 2001).

4.3.2 Results

The findings of this study suggest that architecture as the size predictor of the system considers the size of the system more widely than current LOC (Boehm 1981; DeMarco 1982; Jones 1986; Emrick 1987) or FP (Boehm 1981; Putnam and Myers 1992; Symons 1998; Boehm, Clark et al. 2000) approaches. This result can be interpreted to suggest that more reliable estimates can be produced for the project effort and timetable if we use architecture as size predictor. In the present project, the cost estimation methods were based mostly on 'comparison to similar, past projects based on personal memory' (Heemstra 1992). This proved successful for the same-site subproject with a more familiar situation, but not for the multi-site subproject.

4.3.3 Relation to the whole

The two studies reported in Publication I and Publication II supported each other in the interpretation of the analysis results. The central notion of 'Different interpretations of the software architecture by project designers' and Figure 4 (Publication I) was revealed in this study, just as a concrete example. The studies provided each other with complementary views for coordination, views, which were important for understanding the phenomena. This study contributed remarkably to the understanding of the role of methods as a tool for estimating the project timetable.

4.4 Publication III: Filtering, Negotiating and Shifting in the Understanding of Information Systems Requirements

This paper was accepted into the Scandinavian Journal of Information Systems. An earlier version of this paper was published in the proceedings of the Scandinavian Conference of Information Systems, IRIS27. Here I will present a later version.

4.4.1 Research objectives and methods

In this publication, we studied a large electronic commerce platform development project in order to examine the requirements understanding process by applying the grounded theory research method (Glaser and Strauss 1967). The publication formed the main contribution to two RE approaches in the literature: the traditional requirements engineering approach that considers requirements understanding as systematic techniques of requirement elicitation, documentation and management (Pohl 1994; Kotonya and Sommerville 1998) and the resent socio-technical approaches which see requirement elicitation as a political process (Bergman, King et al. 2002) or as a socio-cognitive problem solving process (Davidson 2002). We approached the subject technology frames (Orlikowski and Gash 1994) as an a priori construct and claim to provide a new interpretation of how technology is collectively interpreted in organizations.

4.4.2 Results

We observed that requirement understanding can be described as a social and organizational process of filtering, negotiating and shifting. In the early phases of the project, there was incongruence in the content of these frames between project participants. This incongruence redirected the participants' attention away from the information and caused feedback aimed at filtering the understanding of project requirements. The participants' ability to resolve this incongruence in the later phases of the project redirected their focus towards relevant information and led to a shift in the understanding of requirements. The shift in the system context happened through iteration between the solution and problem spaces when the participants increased their understanding of the systems requirements iteratively. The project highlights problems in current approaches to requirement elicitation and systems development in general, which still largely assume that projects proceed with distinct phases and more or less in a waterfall fashion from a vague understanding of the idea of the system into a concrete system, which satisfies the originally found requirements. Instead, we observed that the process of requirement understanding was filtering, negotiating and shifting different attitudes and expectations about systems development through the entire project. This process can be described as an ad-hoc and iterative process, where the software requirements unfold during social interaction, communication and negotiation between parties.

4.4.3 Relation to the whole

This publication was perhaps the most important publication contributing mainly to the interpretation of the role of methods and of systems development in general. The guiding role of methods was clearly apparent in this project and the change process was highly conflict resolving and iterative which required mutual understanding and learning from the participants. The participants perceived the object system as a problem space system (Purao, Rossi et al. 2002; Smolander 2003) trying to understand the requirements of the system. It could be observed that the role of methods was to aid learning and understanding the system and its requirements.

4.5 Publication IV: Measuring Requirement Evolution - A Case Study in the E-commerce Domain

This paper appeared in the Proceedings of the Sixth International Conference in Enterprise Information Systems, ICEIS 2004, pp. 669-673.

4.5.1 Research objectives and methods

The purpose of the publication was to attempt to understand the requirement understanding process in an e-commerce project (Publication III) from the quantitative point of view. We developed a measurement for the emergent requirement evolution process and analyzed it's applicability for estimating the project effort and timetable. For the analysis, we used correlation analysis and the non-negative least square problem solving method (Lawson 1995). We calculated other measurements from the system, like cohesion and coupling, and analyzed the correlation between these measurements along with the requirements creep and development effort. From these results, we interpreted the applicability of the requirements creep metrics for estimating project costs and timetable.

4.5.2 Results

The developed quantitative measurement of requirement evolution, namely the requirements creep, included provisions for the emergence of new requirements during systems development, which cannot be anticipated in the requirement elicitation and analysis phase. The existing quantitative approaches for measuring requirement evolution assume that all the requirements exist and can be seen in the requirement specification document (Andersson and Felici 2001; Andersson and Felici 2002). This requirement creep measurement had such a strong impact on the development effort in our correlation analysis that the other measurements were negligible. Based on these results, we suggest that measuring the requirement creep as an emergent process of requirement understanding could be a helpful in estimating the project effort and timetable. However, this would require an appropriate prediction model that also considers other factors of systems development, for example the scalability of the system and the capability of the project personnel.

4.5.3 Relation to the whole

This publication was used for interpreting the requirement understanding process, along with Publication III. They complement one another by considering the process from different perspectives, qualitative and quantitative, providing two valuable perspectives on the same phenomenon. The three conceptual models formed for the calculation of the requirement creep measurement (Figures 4, 5 and 7 in Publication III) were used to identify the emergent requirement understanding process, just one concrete example of the interaction of these two studies. This publication deepened the understanding of the importance of method support for the project timetable estimation. In this project, the organization's methods did not directly support the estimation, and the organization used the timebox method (Martin 1991; McConnell 1996). In other words, the customer gave the development time, and the project had to fit as many features as possible inside this given timebox. The lack of a proper method strengthened our understanding and interpretation of the role of methods as a tool for estimating project duration.

4.6 Publication V: Working with Methods - Observations on the Role of Systems Development Methods

This paper has been accepted to be published in Information Systems Development: Advances in Theory, Practice and Education, ISD 2004, edited by O. Vasilecas, A. Caplinskas, W. Wojtkowski, W.G. Wojtkowski, J. Zupancic , Springer, pp. 185-197.

4.6.1 Research Objectives

The aim of this paper was to provide a summary of the earlier studies, provide a view of working with systems development methods and contribute to the empirical research on the role of systems development methods (Nandhakumar and Avison 1999).

4.6.2 Results

The observations of our studies suggest that working with methods is more about social interaction and mutual understanding between project participants than progressing according to preset milestones and strictly following the prescribed phases of the method. In our case studies, successful use of systems development methods required good communication between project participants who then adopted the method to the development situation at hand. Without this communication the methods could not be used. When communication was successful, methods were used for learning to understand the system and its requirements. Without a common understanding of the system between project participants, the estimation of the project timetable and resources became unrealistic and caused schedule overruns and wrong timing of resources.

4.6.3 Relation to the whole

This summary paper formed the basis for the interpretation of the role of methods by describing how practitioners work with methods.

5 Discussion and implications

5.1 Discussion

"There is no single development, in either technology or management technique, which by itself promises even one order-of-magnitude improvement within a decade in productivity, in reliability, in simplicity" (Brooks 1995)

Brooks (1987) locates in the 1950s a particular shift from writing software metaphor to building software metaphor. Before 1950s, information systems were implemented without the use of explicit methods (Walters, Broady et al. 1994). Computers were mainframes with simple terminals, systems development was monolithic and architectures were simple. System developers' practices were impacted by writing software metaphor (Brooks 1987; Bryant 2000). Hirschheim, Klein and Lyytinen (1995) call this the pre-method era. In the late 1950s and early 1960s, when the first methodical approach, the systems development lifecycle, was introduced, it was impacted by the construction metaphor from engineering discipline (Bryant 2000). The aim of this approach was to achieve a better manageability of the systems development process. Since then, the construction metaphor has dominated research and practice. Methodical approaches based on this metaphor and used by practitioners expect that methods need to be followed strictly using predefined phases and milestones in order to be able to manage the systems development process. These approaches are based on the assumption of a universal mechanism for achieving management and control.

Current methodical approaches mainly suppose that system developer using methods is like European navigator, an example given in Lucy Suchman's book (Suchman 1987) discussing the notion of situated action and its role in interaction between human and machine interaction (HCI) in the following way:

"The European navigator begins with a plan – a course – which he has charted according to certain universal principles, and he carries out his voyage by relating his every move to the plan. His effort throughout his voyage is directed to remaining on course. If unexpected events occur, he must first alter the plan, and then respond accordingly. Once the European navigator has developed his operating plan and has available the appropriate technical resources, the implementation and monitoring of his navigation can be accomplished with a minimum of thought. He has simply to perform almost mechanically the steps dictated by his training and by his initial planning synthesis."(Suchman 1987)

On the contrary, the observations made during these studies suggest that method use can be slightly different. Both projects studied in this thesis did not seem to follow the process of taking up a method and using it as such to solve the problems projects faced. These studies indicate that methods are not used for managing the development process by progressing according to predefined milestones and strictly following method stages. Rather, methods were used as tools for coordinating systems development process whenever it was possible. Studies in phase one suggest that the method for designing software architecture is the main tool for coordination and software cost estimation. Studies in phase two implicate that social and organizational requirements elicitation methods are tools for coordinating the systems development process and estimating project's duration and effort.

In these studies, practitioners seemed to choose the method based on the situation. To them the question was when and how they should use the methods. Practitioners tried to strictly follow the method phases, but they could not do it; they could use them more as guidelines for development and learn development practices through them. Methods, which only describe a series of development tasks provided little help in analyzing how requirements should be negotiated, how multi-site development should be coordinated and how these and other factors contributed to the project's timetable and costs. During the development, the participants' general attitudes towards methods changed and they started to regard methods as a set of tools. They picked and chose the parts of methods they could use. Methods seemed to provide the participants with tools for understanding the final system in different situations rather than making them strictly follow some universal method.

A practitioner working with methods this way can be described by the story of the Trukese navigator, who navigated to the island in the following way:

"The Trukese navigator started his navigation with an objective rather than a plan. He sets off toward the objective and responds to conditions as they arise in an ad hoc fashion. He utilizes information provided by the wind, the waves, the tide and current, the fauna, the stars, the clouds, the sound of the water on the side of the boat, and he steers accordingly. His effort is directed to doing whatever is necessary to reach the objective. If asked, he can point to his objective at any moment, but he cannot describe his course". (Suchman 1987)

Suchman (1987) discusses the relation between the plan and the situated action. She advocates a view where action is always situated, i.e. dependent on and a result of the specific circumstances of a situation. In the same way as our studies, her argumentation denies the role of the general method as a way to improve practice. Additionally, Schön (1983) argues that our common understanding of rationality has little to do with rationality in practice. The "technical rationality", the dominant understanding of rationality in most of the systems development methods, is not well suited for practitioners who operate in a complex, dynamic and messy world.

Clearly, the Trukese navigator is involved in instrumental action in getting from one island to another, and just as clearly the European navigator relies upon his chart regardless of his degree of expertise. Trukese navigator uses wind, stars and clouds as tools to coordinate his navigation to the island, while European navigator relies on planned phases in his navigation. The use of methods observed in our studies is more like the Trukese than the European navigation inferring that there can be different ways of acting depending on the nature of the activity and contingencies. In our studies, the methods mainly described the course of development, but guided neither the situation nor the understanding of the system and its requirements. Whenever a project could not adapt to the situation and use a combination of these factors, it failed to reach an acceptable objective - a system that meets customer requirements.

The situation and contingencies in the projects studied were contradictory: the object systems were modern enterprise architecture for international markets, business area were new at least partly and the systems development methods were extremely traditional. In the project studied in phase one, the participants were distributed into three different sites and the work was to be coordinated according to the architecture. On the contrary, project participants in phase two had different opinions on business values, development strategies and resource politics. The requirements in both projects were demanding, including extensibility, high configurability and distribution. The methods in the organization did not support architecture design and coordination in a multi-site development environment and supposed that requirements exist and are clear, ready just to be gathered into a requirement specification document. The company's process model based on the traditional waterfall life cycle model required that requirements were 'frozen' after requirement elicitation and documentation and the design was based on this view of requirements. If there were changes to the requirements in the later phases, the changes went through the change management procedure in a attempt to manage systems development process in this way. The architecture methods were based purely on UML descriptions which describe the interfaces mainly at the technical level, like 'TCP/IP' and 'XML'. Projects did not have I skills and past experiences from the same kinds of projects and situations.

Nevertheless, the studies in this thesis do not eliminate the role of methods, but rather highlight their role as tools and guidelines that the developers can draw from in diverging situations. Methods play a role as tools for communication, providing the developers with a common language and bridging developers at different sites. When one writes that "we'll do that during the DP2", as was the case in current studies, the two developers understand what is expected in the DP2 as well as when it will arrive. Methods also play a role as tools for collaboration in the form of requirements and architecture. Methods support controlling by providing tools for estimation of software timetable and cost. However, methods do not determine the outcome of events very strongly, but rather remain at a fairly rough and tacit level. In this way, our findings support the notion of software development as knowledge work, in which methods form subjective know-how that allows people to act (Schultze 2000).

Based on these studies, we can claim that the productivity and quality problems described in Chapter 1 are not solved by trying to manage and control the systems development process with universal methods. By accepting the reality that systems development can be a natural outcome of social interaction and communication between development participants that can only be coordinated by methods, we have a chance to solve at least a part of the software crisis. Highlighting understanding the requirements and the architecture of a system as deeply as possible, and treating each project as a unique situation, we can help practitioners improve their development productivity. As a summary, by adopting a less naïve view on methods, we can solve at least part of the software crisis. As Introna and Whitely (Introna and Whitley 1997) states "The question should not to be to get the method; it should be how to help the developers to improvise in the situation in order to get the job done".

5.2 Implications for research

As illustrated in this thesis, the whole systems development is slightly different than many approaches and currently available methods assume. Current approaches still largely assume that projects proceed through distinct phases and more or less in a waterfall fashion from an understanding of the idea of the system to a concrete system, which satisfies the originally found requirements. Most methods developers see methodical development as following predefined method phases strictly to achieve the quality of the final product. Socio-technical approaches like (Mumford and Weir 1979; Walsham 1993) Soft System Methodology (SSM) (Checkland 1981; Checkland and Scholes 1990) and Professional Work Practices methods (Hirschheim, Klein et al. 1995) provide a wider view of systems development than traditional functional approaches. They do not see a single reality, only different perceptions of it and they perceive of systems requirements as socially constructed, emphasizing the early activities of systems development. Our studies have consequences beyond this view. They suggest that systems development is a coordination process, where the understanding of the system unfolds during social interaction, communication and negotiation between project participants during the whole project lifetime. In our case studies, traditional methods did not guide the development group in new development environments or in new and uncertain business areas, also highlighted by Curtis, Krasner et al. (1988), Krasner et al. ,Baskerville, Travis et al. (1992) and Fitzgerald (1998). Methods were used as a tool for coordination among systems development participants, not as a list of phases to be adhered to in detail. Systems development is not so much a rational process but depends on the circumstances and is more an ad hoc improvisation process towards a working system. Methods serve as tools for systems development but do not determine its course. The conflict between contemporary organizational needs and traditional systems development methods used were obvious, highlighted also in Baskerville, Travis et al. (1992).

As metaphors have a powerful impact on people's practices (Brooks 1987), especially regarding software, the studies of this thesis suggest that the building software metaphor should be shifted towards coordinating software metaphor. The coordinating software metaphor should emphasize the coordination, collaboration and communication aspects of systems development. This kind of view of software has important implications for systems development research.

Research, according to the findings of this thesis, should focus on systems development approaches based on the situations of the development projects and how to guide the practitioners in these situations. Systems development research should examine systems development consisting of both technical and organizational aspects and being a heterogeneous organizational process which continues through the whole project lifetime instead of examining general universal approaches. When this heterogeneous organizational process is understood more deeply, it is possible to develop methods to guide the practitioners through this process. Resent research into social and organizational processes in requirement elicitation by (Bergman, King et al. 2002; Davidson 2002) appears particularly promising in this regard.

5.3 Implications for practice

Recent findings showing that practitioners are not using methods as defined by their originators, but rather are tailoring them, provide interesting evidence that practice may be beginning to move in the direction of situation specific methods, without waiting for research to show the way. However, our studies suggest that this is not done in a conscious way and practitioners have no guidance for this tailoring process. For that reason, the core challenge in organizational systems development seems to be to find an appropriate way to use methods and to decide how to use them in the diverging situations the projects face. It is crucial to understand the role and the use of methods inside the organization. For example, the following questions are worth clarifying: Are methods used for determining, regulating, supporting the action (Iivari and Maansaari 1998), or to remind about the action (Ehn and Kyng 1987)? Do they serve as a model of the ideal process highlighting the documentation (Parnas and Clements 1986) or as a vehicle of learning (Checkland and Scholes 1990)? Our studies highlight the guiding role of methods: they provide practitioners with tools to coordinate the systems development process. In our case study projects, methods were not used to determine the course of development, but more to provide guidance and support in understanding the system and its requirements. When an organization explicitly understands the role of their methods, it is easier to find or develop an appropriate method and also guide its adaptation in different situations.

In practice, organizations should improve both their informal and formal communications practices to make knowledge sharing in the projects more effective. Organizations should pay special attention to developing team building, communication and negotiation. In a multi-site environment, the informal communication capabilities can be improved by, for example, more effective use of electronic communication channels and CSCW (Computer Supported Cooperative Work) technology (Lee and Malone 1990; Grudin 1994a; Grudin 1994b). Multi-site coordination also requires a chief architect, who is capable of maintaining the integrity of the architecture and communicating it to gain a common understanding of the architecture among project stakeholders.

For the understanding the development situations and the developed systems itself, software organizations should use metaphors (Kendall and Kendall 1993; Schultze and Orlikowski 2001) and narratives (Davidson 1997). In our study, we used a metaphorical analysis to help understand the size of the system. The use of metaphors can be worth exploring to enable their use also in requirement elicitation. New metaphors can help see the requirements of the system from a new point of view (as for instance the first time you see yourself on video). A metaphor is nothing more or less than a model, playing the same role in our understanding as a map that helps us find our way through an unknown landscape. They can be useful in visualizing the invisible future and making abstract matters more concrete. The use of narratives in an organization to describe the success and failures of systems development can support and help systems developers to understand the problems they face in practical situations and to find ways to cope with these problems. Further, narratives can provide some expert knowledge for inexperienced systems developers.

An important challenge in organizational systems development is recognizing and explicitly acknowledging conflicting assumptions and expectations among stakeholders inhibiting the understanding of requirements. It is especially important to recognize assumptions that are not directly related to the system and its context, but more to organizational issues. It is also important to understand that changes in the understanding of some requirements during the project could have a far reaching ripple effect on other requirements and the project itself. Instead of trying to identify all the requirements in advance, requirement engineers should identify obstacles and emerging opportunities of requirement understanding and improvise on, or around, them (Nandhakumar, Rossi et al. 2003). Our studies also highlight the problems of having too narrow a scope of requirement gathering or interpretation, which can lead to the omission of key information by the developers.

5.4 Implications for systems development methods

As illustrated in this thesis, the profile of the development environment is different from that which prevailed when many of the currently available commercial methods were first proposed some 25 to 30 years ago. Thus, there is a need to put great effort into updating our understanding of the use of development methods and concentrate on developing methods more suited to the needs of the current practical development climate. Current methods approaches believe that it is possible to understand and solve a typical system development problem by using methods. Using methods as tools does not mean anything if their usage and the situations in which they are used are not understood. This means that it is important to understand the situation before using a method, not vice versa as most of the method approaches currently assume. This view highlights the methods' role in understanding the situation, also stated by Introna and Whitley (1997). This has several implications for future development methods, which have an important role in guiding the practitioners to understand the situations in the project. According to the studies of this thesis, four issues, in particular, must be addressed in methods in order for them to guide practitioners in different situations. In the following, these issues are summarized based on the publications.

First, systems development methods should support coordination in different environments. Especially, coordination and communication are the most important aspects in multi-site development environment with geographically and culturally dispersed teams. The methods should support communication, not be mere tools and techniques for communication, but also themselves provide a common language for the stakeholders. Curtis, Krasner and Iscoe discuss methods "serving as a boundary object to which several stakeholders associate their particular meanings" (Curtis, Krasner et al. 1988). In other words, methods provide a way and a language for stakeholders to communicate and understand each other. Our studies emphasize the architecture and architecture design as a multi-site coordination tool. Communication requires methods that support informal communication. The use of architecture as a coordination tool requires the use of multiple viewpoints or representations in architecture design, a chief architect to communicate and coordinate architectural structures, and an interface design activity early enough in the architecture design phase.

Second, the requirement elicitation method should support and help the practitioner in recognizing and explicitly acknowledging the conflicting assumptions and expectation among systems development project participants. The concept of technology frame could be a useful tool that methods support and guide when identifying these organizational obstacles, assumptions and expectations. Further, the method should guide the practitioner in the use of metaphors (Lakoff and Johson 1980; Frost and Morgan 1983; Schultze and Orlikowski 2001) to increase the understanding of systems requirement. New metaphors can help see the requirements of the system from a new point of view (similarly to when you see yourself on video for the first time). A metaphor is nothing more than a model, playing the same role in our understanding as a map that helps us find our way through an unknown landscape. They can be useful in visualizing the invisible future and making abstract matters more concrete.

Third, the project management methods should include guidelines for project's cost estimation. Our studies highlight the use of the architecture and requirement understanding process as a basis for cost estimation. These include the collection of history information to help the estimation. In order to get more reliable estimates of the project timetable and costs, the methods should guide and assist the practitioners in better understanding the complexity of both the system and the systems development. In our study, we used metaphorical analysis to help understand the size of the system. The use of metaphors in cost estimation (Kendall and Kendall 1993; Schultze and Orlikowski 2001) can be worth exploring. The cost estimation model should also take into account the non-linear nature of systems development, i.e the non-linear effect of changes in the systems size to the development effort. More empirical research should be conducted to achieve better estimation methods appropriate for practical organizations in uncertain and turbulent business environments. These methods should be sufficiently simple and take into account both the increasing complexity of the systems and their development.

Fourth, as our studies indicate, the development lifecycle model and the development strategy are situation dependent. Neither waterfall model nor prototyping and iterative development are suitable for every situation. Rather, a strategy based on the situation may be more appropriate. Our studies contradict the methodical steps and their granularity. Rather than prescribing detailed of steps which developers are expected to follow, the methods' focus should be on higher level goals and deliverables. The precise manner in which these are actually achieved should be left to practitioner. However, practitioners need some guidelines as how to achieve these goals in different situations. As our studies indicate, the questions 'how' and 'when' are especially interesting, also the question 'why' would help the practitioner. Furthermore, methods should guide in the use of narratives (Davidson 1997) to describe the successes and failures of systems development. The use of narratives can support and help systems developers to understand the problems they face in practical situations and how to cope with these problems.

6 Conclusions

In this Chapter, I will present the summary and the main contribution of this thesis. In the end of this Chapter, I will discuss the limitations of these studies and possible future research topics.

6.1 Summary and contributions

This thesis has described a company competing in the information technology field following two of its projects with a series of in-depth case studies. For the analysis of the phenomena we used the theory-building case study approach with qualitative and quantitative methods. The three phases of the thesis have provided empirical observations about different aspects of systems development. In the first phase of the thesis, we examined the role of architecture in coordination and cost estimation in a multi-site software development from quantitative and qualitative viewpoints. The second phase involved two studies, one qualitative and the other quantitative, on the evolving requirement understanding process and the measurement of this process. The third phase was a study based on the first two studies on the role of methods and how practitioners work with them.

The answer to the research question presented in Chapter 3.1 is explained here as a course of research phases.

The first phase of the thesis provided a view of coordination in a multi-site development environment and of the way methods supported the participants in this work. We observed that systems development process is coordinated using architecture as a tool. The systems development participants would have needed methods, which support multi-site coordination, but the methods provided them no guidance. Our studies imply that, in a multi-site environment, it is not enough to coordinate only development activities like current approaches assume, but it is also important to coordinate the interdependencies between activities.

In the second phase, we studied the requirement understanding process. The studies suggest that practitioners use the filtering, negotiating and shifting process of requirement understanding as a tool for coordinating systems development process. In this process, different attitudes and expectations about systems development changed through the whole project lifetime. The project participants would have needed methods to support them in resolving conflicts between participants and to help them understand the requirements of the final system, but the methods were not able to guide them in these problems. These studies highlight the problems in current approaches to systems development, which still largely imply that projects proceed with distinct phases and more or less in a waterfall fashion. These approaches are mainly based on the assumption that systems development proceed from a vague understanding of the idea of the system to a concrete system, which satisfies the originally found requirements.

The third phase of the thesis includes an observation that working with methods is social interaction and mutual understanding between project participants in using the methods based on the development situation at hand. It is not so much progressing according to milestones and strictly following the method's phases like most of the methods developers assume. Methods serve as tools for systems development but do not determine its course.

The main contribution of this thesis is that systems development in a modern market environment is much more complex and more driven by opportunity than acknowledged by current development methods. Systems development is not coordinated using activities and phases defined in methods. Instead, it is coordinated using methods as tools that help practitioners achieve a common goal. Now, methods should not describe the development activities and phases in a detail level, but should include higher level guidance for practitioners on how to act in different situations. Based on this thesis, the systems developer can be described as a knowledge worker creating new information and knowledge in the organization.

6.2 Limitations of these studies

These studies have obvious weaknesses, as all studies do. The quantitative research method has sought its form during the thesis. As we explained in Chapter 3.2, we first used an evolutionary algorithmic approach and then turned to statistical analysis. The main purpose of the quantitative analysis was to calculate the correlation between the developed metrics and the development effort, and our statistical analysis fits this purpose well. But it is too simplistic for cost estimation purposes, as noted in both

Publication II and Publication IV. Fortunately, this was only my secondary goal in the quantitative analysis.

In general, combining quantitative and qualitative methods was not easy, mainly because of the lack of the empirical examples in the literature. Some literature exists under the subject of triangulation (Gable 1994; Markus 1994; Mingers 2001), but they deal mainly with the analysis of surveys quantitatively and qualitatively forming a different situation than in this thesis.

A critical issue for researchers concerns the generalizability of the results of their work, and Yin (Yin 1994) notes that this issue is often raised with respect to case studies. Different arguments for the generalizability of case study research have been given (Eisenhardt 1989; Dutton and Dukerich 1991; Yin 1994; Walsham 1995), also discussed in Chapter 3.3.5. It is argued that in case study research, the identified concepts and categories are compared to theoretical concepts and patterns, unlike in statistical generalization from a sample to a population. In phase one of the thesis, the identified concepts were compared to Malone and Crowston's theoretical concepts of coordination (Malone and Crowston 1990; Malone and Crowston 1994), and in phase two to existing socio-technical approaches in requirement elicitation (van Lamsweerde 2000; Bergman, King et al. 2002; Davidson 2002; Tomayko 2002), especially the concept of a technological frame (Orlikowski and Gash 1994; Davidson 2002). In phase three, the findings were compared to some empirical studies of the role of methods in systems development (Russo, Wynekoop et al. 1995; Unhelkas and Mandapur 1995; Fitzgerald 1998; Iivari and Maansaari 1998; Nandhakumar and Avison 1999; Fitzgerald 2000). Still, due to the nature of this thesis, in which the understanding of method use was interpreted on the basis of separate phenomena found in one organization, the generalization of the use of methods may be limited. Therefore, the understanding gained in these studies provides a basis for understanding similar phenomena in the same settings rather than enabling the understanding of phenomena in other contexts.

6.3 Future research

As mentioned earlier, there is a lack of empirical research on the role of methods in practical organizational context. It is not useful to study the use or non-use of methods, nor whether the methods used are text-book or in-house ones. This kind of knowledge does not improve our understanding of how practitioners work with methods. Neither is it worthwhile to develop new methods if they are not based on any empirical grounding. These studies described the use of methods in one organizational context and in one business area. In other contexts and in more mature business areas, the findings could differ. When enough knowledge of the role of methods is acquired, the developed methods could then be experimented with and developed further, for example using action research as research method.

References

Aalto, J.-M. and A. Jaaksi (1994). "Object-Oriented Development of Interactive Systems with OMT++". *Proceedings of the Technology of Object-Oriented Languages and Systems (TOOLS 14)*, Santa Barbara, CA, August, Prentice-Hall**:** 205-218.

Abrahamsson, P., J. Warsta, M. T. Siponen and J. Ronkainen (2003). "New Directions on Agile Methods: A Comparative Analysis". *Proceedings of the 25 th International Conference on Software Engineering*, Portland, Oregon, May 23-28**:** 244.

Alavi, M. and P. Carlson (1992). "A review of MIS research and diciplinary development", *Journal of Management Information Systems,* **8**(4): 45-62.

Albrecht, A. J. (1979). "Measuring application development productivity". *Proceedings of the Joint SHARE/GUIDE/IBM Appl. Development Symposium*, Monterey, CA, October**:** 83-92.

Andersson, S. and M. Felici (2001). "Requirements Evolution: From Process to Product Oriented Management". *Proceedings of the 3rd International Conference on Product Focused Software Process Improvement*, Kaiserslautern, Germany, September 10-13, Springer Verlag**:** 27-41.

Andersson, S. and M. Felici (2002). "Quantitative Aspects of Requirement Evolution". *Proceedings of the 26th Annual International Conference on Computer Software and Applications Conference (COMPSAC 2002)*, Oxford, England, August 26-29, IEEE Society**:** 27-32.

Angell and Straub (1993). "Though this be madness, yet there is method in't", *Journal of Strategic Information Systems,* **2**(1): 5-14.

Avison, D. and B. Fitzgerald (1988). *Information Systems Development Methodologies: Techniques and Tools*, Oxford, Blackwell.

Avison, D. and B. Fitzgerald (1995). *Information Systems Development: Methodologies, Techniques and Tools, 2 nd edition*, New York, McGraw-Hill.

Avison, D., F. Lau, M. D. Myers and P. A. Nielson (1999). "Action Research", *Communications of the ACM,* **42**(1): 94-97.

Baskerville, R., J. Travis and D. P. Truex (1992). "Systems without method: The impact of new technologies on information systems development projects", in *Transactions on the impact of computer supported technologies in information systems development*. J.I.DeGross (eds.)**:** 241-260.

Baskerville, R. L. and J. Pries-Heje (1999). "Grounded action research: a method for understanding IT in practice", *Accounting, Management and Information Technology,* **9**(1): 1-23.

Benbasat, I., D. Goldstein and M. Mead (1987). "The case study research strategy in studies of information systems", *MIS Quarterly,* **11**(3): 369-386.

Bergman, M., L. King and K. Lyytinen (2002). "Large Scale Requirements Analysis Revisited: The need for Understanding the Political Ecology of Requirements Engineering", *Requirements Engineering,* **7**(3): 152-171.

Beynon-Davies, P. and M. Williams (2003). "The diffusion of information systems development methods", *The Journal of Strategic Information Systems,* **12**(1): 29-46.

Bodker, S., P. Ehn, J. Kammersgaard, M. Gustafsson and L. Johansson (1987). "An UTOPIAN experience: on design of poweful computer-based tools for skilled graphic workers", in *Computers and Democracy: A Scandinavian Challenge*. G. Bjerkenes, P. Ehn and M. Kyng (eds.). Aldershot, Avebury**:** 251-278.

Boehm, B. (1981). *Software Engineering Economics*, New Jersey, Prentice Hall.

Boehm, B. (1987). "Improving Software Productivity", *IEEE Computer,* **20**(8): 43-58.

Boehm, B. (1988). "A Spiral Model of Software Development and Enhancement", *IEEE Computer,* **21**(5): 61-72.

Boehm, B., B. Clark, E. Horowitz, R. Madachy, D. Reifel, B. K. Clark, B. Steece, A. W. Brown, S. Chulani and C. Abts (2000). *Software Cost Estimation with COCOMO II*, New Jersey, Prentice Hall.

Boehm, B. and R. E. Fairley (2000). "Software Cost Estimation Perspectives", *IEEE Software,* **17**(6): 22-26.

Booch, G., J. Rumbaugh and I. Jacobson (1998). *The Unified Modelling Language User Guide*, Massachusetts, Addison-Wesley.

Broadbent, M. and P. Weill (1993). "Improving business and information strategy alignment: Learning from the banking industry", *IBM Systems Journal,* **32**(1): 162-179.

Brooks, F. P. J. (1987). "No Silver Bullet: Essence and Accidents of Software Engineering", *IEEE Computer,* **20**(4): 10-19.

Brooks, F. P. J. (1995). *The Mythical Man-Month - 20th Anniversary Edition*, Boston, Addison-Wesley.

Bubenko, J. A., jr., B. Langefors and A. Solvberg (1971). *Computer-Aided Informations Systems Analysis and Design*, Lund, Studentlitteratur.

Calloway, L. J. and G. Ariav (1991). "Developing and Using Qualitative Methodology to Study Relationships among Designers and Tools", in *Information Systems Research:Contemporary Approaches and Emergent Traditions*. H. E. Nissen, H. Klein and R. Hirschheim (eds.). Amsterdam, North-Holland**:** 175-193.

Carmel, E. and R. Agarwal (2001). "Tactical Approaches for Alleviating Distance in Global Software Development", *IEEE Software,* **18**(2): 22-29.

Castells, M. (1996). *The Rise of the Networked Society*, Malden, MA, USA, Blackwell Publishers.

Checkland, P. (1981). *Systems Thinking, Systems Practice*, Chichester, UK, John Wiley & Sons.

Checkland, P. and J. Scholes (1990). *Soft System Methodology in Action*, Chichester, UK, John Wiley & Sons.

Chelimsky, E. (1992). Quantitative Data Analysis: An Introduction, United States General Abbounting Offices, http://www.gao.gov/special.pubs/pe10111.pdf, accessed May 6, 2005.

Cockburn, A. (2001). *Agile Software Development*, Boston, Addison-Wesley.

Coplien, J. O. and M. Devos (1999). "Architecture as Metaphor". *Proceedings of the World Multiconference on Systemics,Cybernetics and Informatics*, Orlando, Florida, July 23-26, Institute of Informatics and Systemics**:** 737-742.

Curtis, B., H. Krasner and N. Iscoe (1988). "A Field Study of the Software Design Process for Large Systems", *Communications of the ACM,* **31**(11): 1268-1287.

Davidson, E. J. (1997). "Examining Project History Narratives: An Analytic Approach", in *Information Systems and Qualitative Research*. A. S. Lee, J. Liebenau and D. J.I (eds.). London, Chapman and Hall**:** 123-148.

Davidson, E. J. (2002). "Technology Frames and Framing: A Socio-Cognitive Investigation of Requirement Determination", *MIS Quarterly,* **26**(4): 123-148.

DeMarco, T. (1978). *Structured Analysis and System Specification*, New Jersey, Prentice Hall.

DeMarco, T. (1982). *Controlling Software Projects: Management, Measurement and Estimation*, New Jersey, Prentice Hall.

Dijkstra, E. (1972). "The Humble Programmer", *Communications of the ACM,* **15**(10): 859-866.

Dowson, M. (1993). "Software Process Themes and Issues". *Proceedings of the Second International Conference on the Software Process*, Berlin, Germany, February 25-26**:** 54-62.

Earl, M. J. (1993). "Experiences in Strategic Information Systems Planning", *MIS Quarterly,* **17**(1): 1-24.

Ehn, P. and M. Kyng (1987). "The Collective Resource Approach to Systems Design", in *Computers and Democracy*. G. Bjerkenes, P. Ehn and M. Kyng (eds.). Aldershot, Avebury**:** 17-57.

Eisenhardt, K. M. (1989). "Building Theories from Case Study Research", *Academy of Management Review,* **14**(4): 532-550.

Emrick, R. D. (1987). "In search of a better metric for measuring productivity of application development". *Proceedings of the International Function Point Users Group Conference (IFPUG)*, San Antonio, Texas.

Finholt, T. A., E. Rocco, D. Bree, N. Jain and J. D. Herbsleb (1998). "NotMeeting: A field trial of NetMeeting in a geographically distributed organization", *SIGGROUP Bulletin,* **20**(1): 66-69.

Fitzgerald, B. (1998). "An empirical investigation into the adoption of systems development methodologies", *Information & Management,* **34**(6): 317-328.

Fitzgerald, B. (2000). "Systems development methodologies: the problem of tenses", *Information Technology & People,* **13**(3): 174-185.

Frost, P. J. and G. Morgan (1983). "Symbols of sensemaking: the real-ization of the framework", in *Organizational symbolism*. L. R. Pondy, P. J. Frost, G. Morgan and T. C. Dandrike (eds.). New York, Wiley**:** 419-437.

Gane, C. P. and T. Sarson (1979). *Structured Systems Analysis: Tools and Techniques*, New York, Prentice Hall.

Garlan, D. and D. Perry (1995). "Introduction to the Special Issue on Software Architecture", *IEEE Transaction on Software Engineering,* **21**(4): 269-274.

Georgiadou, E. (2003). "Software Process and Product Improvement: A Historical Perspective", *Cybernetics and Systems Analysis,* **39**(1): 125-142.

Gersick, C. (1988). "Time and transition in work teams: toward a new model of group development", *Academy of Management Journal,* **31**(1): 9-41.

Glaser, B. and A. L. Strauss (1967). *The Discovery of Grounded Theory: Strategies for Qualitative Research*, Chicago, Adline.

Glass, R. (1994). "The Software Research Crisis", *IEEE Software,* **11**(6): 42-47.

Glass, R. (1999). "The realities of software technology payoffs", *Communications of the ACM,* **42**(2): 74-79.

Glass, R., I. Vessey and V. Ramesh (2002). "Research in software engineering: an analysis of the literature", *Information and Software Technology,* **44**(8): 491-506.

Grinter, R. E., J. D. Herbsleb and D. E. Perry (1999). "The Geography of Coordination: Dealing with Distance in R&D Work". *Proceedings of the International Conference on Supporting Group Work (GROUP'99)*, Phoenix, Arizona, November 14-17, 1999, ACM Press**:** 306-315.

Grudin, J. (1994a). "Groupware and Social Dynamics: Eight Challenges for Developers", *Communications of the ACM,* **37**(1): 93-105.

Grudin, J. (1994b). "Computer-Supported Cooperative Work: History and Focus", *IEEE Computer,* **27**: 19-26.

Hardy, C., J. Thompson and H. Edwards (1995). "The use, limitations and customization of structured development methods in United Kingdom", *Information and Software Technology,* **37**(9): 467-477.

Heemstra, F. J. (1992). "Software cost estimation", *Information and Software Technology,* **34**(10): 379-382.

Henderson, J. C. and J. G. Cooprider (1994). "Dimensions of IS Planning and Design Aids: A Functional Model of CASE TEchnology", in *IT and the Corporation of the 1990's: Research studies*. T. Allen and M. Scott-Morton (eds.). New York, Oxford University Press**:** 221-248.

Herbsleb, J., D. Zubrow, D. Goldenson, W. Hayes and M. C. Paulk (1997). "Capability maturity model and the software quality", *Communications of the ACM,* **40**(6): 30-40.

Herbsleb, J. D. and R. E. Grinter (1999). "Architectures, Coordination, and Distance: Conway's Law and Beyond", *IEEE Software,* **16**(5): 63-70.

Hirschheim, R., H. Klein and K. Lyytinen (1995). *Information Systems Development and Data Modeling: Conceptual and Philosophical Foundations*, Cambridge, Cambridge University Press.

Iivari, J. and J. Maansaari (1998). "The usage of systems development methods: are we stuck to old practices?" *Information and Software Technology,* **40**(9): 501-510.

Introna, L. and E. Whitley (1997). "Against method-ism:exploring the limits of method", *Logistics Information Management,* **10**(5): 235-245.

Jarke, M., C. Rolland, A. Sutcliffe and R. Dömges (1999). *The Nature of Requirements Engineering*, Aachen, Shaker Verlag GmbH.

Jayaratna, N. (1994). *Understanding and Evaluating Methodologies:NIMSAD a Systematic Framework*, New York, McGraw-Hill.

Jones, C. (1986). *Programmer productivity*, New York, McGraw-Hill.

Järvinen, P. H. (2000a). "Research Question Guiding Selection of an Appropriate Research Method". *Proceedings of the European Conference on Information Systems*, Vienna, July 3-5, Vienna University of Economics and Business Administration**:** 124-131.

Järvinen, P. H. (2000b). "On a variety of research output types". *Proceedings of the Scandinavian Conference on Information Systems (IRIS23): Doing IT Together*, Uddevalla, Sweden, August 20, Laboratorium for Interaction, University of Trollhättan**:** 251-256.

Karimi, J. (1988). "Stratetic Planning for Information Systems: Requirements and Information Engineering Methods", *Journal of Management Information Systems,* **4**(4): 5-24.

Kast, F. E. and J. E. Resenzweig (1974). *Organization and Management: A Systems Approach. Second Edition*, Tokyo, McGraw-Hill.

Kautz, R., K. Kuhlenkamp and H. Zullighoven (1992). *Prototyping: An Approach to Evolutionary Systems Development*, London, Springer Verlag.

Keil, M., J. Mann and A. Rai (2000). "Why Software Projects Escalate: An Empirical Analysis and Test of Four Theoretical Models", *MIS Quarterly,* **24**(4): 631-664.

Keil, M. and D. Robey (2001). "Blowing the Whistle on Troubled Software Projects", *Communications of the ACM,* **44**(4): 87-93.

Kendall, J. E. and K. E. Kendall (1993). "Metaphors and Methodologies: Living beyond the Systems Machine", *MIS Quarterly,* **17**(2): 149-171.

Kidder, T. (1982). *Soul of a new machine*, New York, Avon.

Kotonya, G. I. and Sommerville (1998). *Requirement Engineering*, New York, John Wiley & Sons.

Kraut, R. E. and L. A. Streeter (1995). "Coordination in Software Development", *Communications of the ACM,* **38**(3): 69-81.

Kruchten, P. (2000). *The Rational Unified Process: An Introduction*, New York, Addison-Wesley.

Kuilboer, J. P. and N. Ashrafi (2000). "Software process and product improvement: an empirical assessment", *Information and Software Technology,* **42**(1): 27-34.

Kumar, K. and R. Welke (1992). "Methodology Engineering: A Proposal for Situation-specific Methodology Engineering", in *Challenges and Strategies for Research in Systems Development*. W. W. Cotterman and J. A. Senn (eds.). Chichester, Wiley**:** 257-269.

Lakoff, G. and M. Johson (1980). *Metaphors We Live By*, Chicago, The University of Chicago Press.

Lawson, C. L. (1995). *Solving Least Squares Problems*, Society of Industrial and Applied Mathematics.

Lawson, C. L. and R. Hansen (1974). *Solving Least Squares Problems*, New Jersey, Prentice Hall.

Lee, J. and T. W. Malone (1990). "Partially shared views: A scheme for communicating among groups that use different type hierarchies", *ACM Transactions on Information Systems,* **8**(1): 1-26.

Locke, K. (2003). *Grounded Theory in Management Research*, London, Sage.

Lyytinen, K. (1987a). "A taxonomic perspective of information systems development: theoretical constructs and recommendations", in *Critical issues in information systems research*. R. J. Boland and R. Hirschheim (eds.), John Wiley & Sons**:** 3-41.

Lyytinen, K. (1987b). "Different Perspectives on Information Systems: Problems and Solutions", *ACM Computing Surveys,* **19**(1): 5-46.

MacKenzie, D. and J. Wajsman (1999). *The Social Shaping of Technology,* Phidalelphia, USA, Open University Press.

Malone, T. W. and K. Crowston (1990). "What is Coordination Theory and how can it help design cooperative work systems?" *Proceedings of the Conference on Computer Supported Cooperative Work (CSCW '90),* Los Angeles, October 7-10, 1990, ACM Press**:** 357-370.

Malone, T. W. and K. Crowston (1994). "The Interdisciplinary Study of Coordination", *ACM Computing Surveys,* **26**(1): 87-110.

Markus, M. L. (1983). "Power, Politics and MIS Implementation", *Communications of the ACM,* **26**(6): 430-444.

Martin, J. (1991). *Rapid Application Development*, New York, Macmillan Publishing.

Marttiin, P. (1998). *Customisable Process Modelling Support and Tool for Design Environment*. Department of Computer Science, University of Jyväskylä, Ph.D.Dissertation.

Mathiassen, L., P. A. Munk-Madsen, P. A. Nielsen and J. Stage (1996). "Method Engineering: Who's the Customer?" in *Method Engineering. Principles of Method Construction and Tool Support*. S. Brinkkemper, K. Lyytinen and R. Welke (eds.). London, Chapman & Hall.

MathWorks Inc. (2003). Matlab, http://www.mathworks.com, accessed May 5, 2005.

McConnell, S. (1996). *Rapid Development*, Microsoft Press.

McLean, E. (1973). "Comment on Empirical studies of management information systems", *Data Base,* **4**(4): 181.

Miles, M. B. and A. M. Huberman (1984). *Qualitative Data Analysis: A Sourcebook of a New Methods*, Beverly Hills, Sage.

Mintzberg, H. (1979). "An emerging strategy of "direct" research", *Administrative Science Quarterly,* **24**(4): 582-589.

Mumford, E. (1983). *Designing Human Systems-the ETHICS Method,* Manchester, Manchester Business School.

Mumford, E. and M. Weir (1979). *Computer Systems in Work Design: The ETHIC Method,* New York, Wiley.

Nandhakumar, J. and D. Avison (1999). "The fiction of methodology development: a field study of information systems development", *Information Technology & People,* **12**(2): 176-191.

Nandhakumar, J. and M. Jones (1997). "Too close for comfort? Distance and engagement in interpretive information systems research", *Information Systems Journal,* **7**(2): 109-131.

Nandhakumar, J., M. Rossi and J. Talvinen (2003). "Planning for "drift": Implementation process of enterprise resource planning systems". *Proceedings of the 36 th Hawaii International Conference on Systems Sciences (HICSS),* Big Island, HI, USA, 7-10 January, IEEE Computer Society**:** 241-250.

Naumann, J. D., G. B. Davis and J. D. McKeen (1980). "Determining Information Requirements: A Contingency Method of the Selection of a Requirements Assurance Strategy", *Journal of Systems and Software,* **1**(4): 273-281.

Nygaard, C. and O.-J. Dahl (1966). "Simula: an ALGOL-based simulation language", *Communications of the ACM,* **9**(9): 671-678.

Nygaard, K. and O. T. Bergo (1974). "The Trade Unions-New Users of Research", *Personell Review,* **1**(1): 5-10.

Orlikowski, W. J. (1993). "Case Tools as Organizational Change: Investigating Incremental and Radical Changes in Systems Development", *MIS Quarterly,* **17**(3): 309-340.

Orlikowski, W. J. (2002). "Knowing in Practice: Enacting a Collective Capability in Distributed Organizing", *Organization Science,* **13**(3): 249-273.

Orlikowski, W. J. and J. J. Baroudi (1991). "Studying Information Technology in Organizations: Research Approaches and Assumptions", *Information Systems Research,* **2**(1): 1-28.

Orlikowski, W. J. and D. C. Gash (1994). "Technological Frames: Making Sense of Information Technology in Organizations", *ACM Transactions on Information Systems,* **12**(2): 174-207.

Parnas, D. L. (1972). "On the Criteria To Be Used in Decomposing Systems into Modules", *Communications of the ACM,* **15**(5): 330-336.

Parnas, D. L. and P. C. Clements (1986). "A rational design process: how and why to fake it?" *IEEE Transactions on Software Engineering,* **2**(2): 251-257.

Paulk, M. C., B. Curtis, M. B. Chrissis and C. V. Weber (1993). "The Capability Maturity Model for Software: A Tutorial", *IEEE Software,* **12**(1): 74-83.

Pedrycz, W., J. F. Peters and S. Ramanna (1999). "Fuzzy Set Approach to Cost Estimation of Software Project". *Proceedings of the IEEE Canadian Conference on Electrical and Computer Engineering (CCECE 99)*, Edmonton, Canada, May 9-12**:** 1068-1073.

Pinfield, L. (1986). "A field evaluation of perspectives on organizational decision making", *Administrative Science Quarterly,* **31**(3): 365-388.

Pohl, k. (1994). "Three Dimensions of Requirements Engineering: framework and its application", *Information Systems,* **19**(3): 243-258.

Pries-Heje, J., R. Baskerville, L. Levine and B. Ramesh (2004). "The High Speed Balancing Game: How Software Companies Cope with Internet Speed", *Scandinavian Journal of Information Systems,* **16**(11): 11-54.

Purao, S., M. Rossi and A. Bush (2002). "Towards an Understanding of the Use of Problem and Design Spaces during Object-Oriented System Development", *Information and Organization,* **12**(4): 249-281.

Putnam, L. H. (1978). "A General Empirical Solution to the Macro Software Sizing and Estimation Problem", *IEEE Transaction on Software Engineering,* **4**(4): 345-361.

Putnam, L. H. and W. Myers (1992). *Measures for Excellence*, Yourdon Press.

Rossi, M. (1998). *Advanced Computer Support for Method Engineering*. Department of Computer Science, University of Jyväskylä, Ph.D.Dissertation.

Rumbaugh, J. (1995). "What is a method?" *Journal of Object-Oriented Programming,* **8**(6): 10-16.

Rumbaugh, J., M. R. Blaha, W. Lorensen, F. Eddy and W. Premerlani (1990). *Object-Oriented Modeling and Design*, New Jersey, Prentice Hall.

Russo, N., J. Wynekoop and D. Waltz (1995). "The Use and Adaptation of System Development Methodologies", in *Managing Information & Communications in a Changing Global Environment: Proceedings of the Information Resources Management Association International Conference*. M. Krosrowpour (eds.). Atlanta, Idea Group Publishing**:** 162.

Sabherwal, R. (2003). "The Evolution of Coordination in Oursourced Software Development Projects: a Comparison of Client and Vendor Perspectives", *Information and Organization,* **13**(3): 153-202.

Schultze, U. (2000). "A Confessional Account of an Ethnography About Knowledge Work", *MIS Quarterly,* **24**(1): 3-41.

Schultze, U. and W. J. Orlikowski (2001). "Metaphors of virtuality: shaping an emergent reality", *Information and Organization,* **11**(1): 45-77.

Scientific Software (2001). Atlas.ti-The Knowledge Workbench, http://www.atlasti.de/, accessed May 5, 2005.

Scott, J. (1998). "Organizational Knowledge and The Intranet", *Decision Support Systems,* **23**(1): 3-17.

Smolander, K. (2002). "Four Metaphors of Architecture in Software Organizations: Finding out The Meaning of Architecture in Practice". *Proceedings of the*

International Symposium on Empirical Software Engineering (ISESE 2002), Nara, Japan, October 3-4**:** 211-221.

Smolander, K. (2003). *On the Role of Architecture in Systems Development*. Department of Information Technology, Lappeenranta University of Technology, Ph.D.Dissertation.

Smolander, K., V.-P. Tahvanainen and K. Lyytinen (1990). "How to Combine Tools and Methods in Practice - a Field Study". *Proceedings of the Second Nordic Conference CAISE '90*, Stockholm, Sweden, May 8-10, Lecture Notes in Computer Science**:** 195-211.

Sol, H. G. (1983). "A Feature Analysis of Information Systms Design Methodologies: Mehodological Considerations", in *Information Systems Design Methodologies: A Feature Analysis*. T. W. Olle, H. G. Sol and C. J. Tully (eds.). Amsterdam, Elsevier Science Publishers**:** 1-7.

Strauss, A. and J. Corbin (1990). *Basics of Qualitative Research: Grounded Theory Procedures and Applications*, Newbury Park, CA, Sage.

Suchman, L. (1987). *Plans and Situated Action*, Cambridge, Cambridge University Press.

Susman, G. I. and R. D. Evered (1978). "An Assessment of the Scientific Merits of Action Research", *Administrative Science Quarterly,* **23**(4): 582-603.

Symons, C. R. (1998). "Function Point Analysis: Difficulties and Improvements", *IEEE Transaction on Software Engineering,* **14**(1): 2-11.

Tolvanen, J.-P. (1998). *Incremental Method Engineering with Modeling Tools - Theoretical Principles and Empirical Evidence*. Department of Computer Science and Information Systems, University of Jyväskylä, Ph.D.Dissertation.

Tomayko, J. E. (2002). Engineering of Unstable Requirements Using Agile Methods, http://www-di.inf.puc-rio.br/~julio/tcre-site/p1.pdf, accessed April 4, 2005.

Truex, D. P., R. Baskerville and J. Travis (2001). "Amethodical systems development: The deferred meaning of systems development methods", *Accounting, Management and Information Technologies,* **10**(1): 53-79.

Unhelkas, B. and G. Mandapur (1995). "Practical aspects of using methodology: A road map approach", *Report of Object Analysis and Design,* **2**(2): 34-36,54.

Walsham, G. (1993). *Interpreting Information Systems in Organizations*, Chister, John Wiley & Sons.

Walters, S. A., J. E. Broady and R. J. Hartley (1994). "A Review of Information Systems Development Methodologies", *Library Management,* **15**(6): 5-19.

van Lamsweerde, A. (2000). "Requirements engineering in the year 00: A research perspective". *Proceedings of the International Conference on Software Engineering (ICSE 2000)*, Limerick, Ireland**:** 5-19.

van Slooten, K. and B. Schoonhoven (1996). "Contingent information systems development", *Journal of Systems and Software,* **33**(2): 153-161.

Weill, P. and M. Broadbent (1998). *Leveraging the new Infrastructure*, Boston, Harvard Business School Press.

Venkatachalam, A. R. (1993). "Software Cost Estimation Using Artificial Neural Networks". *Proceedings of the International Joint Conference on Neural Networks, IJCNN 93*, Nagoya, Japan, October 25-29**:** 987-999.

Verner, J. M. and G. A. Tate (1992). "A software size model", *IEEE Transaction on Software Engineering,* **18**(4): 265-278.

Vessey, I. and A. P. Sravanapudi (1995). "CASE Tools as Collaborative Support Technologies", *Communications of the ACM,* **38**(1): 83-94.

Wiegers, K. E. (1999). *Software Requirements*, Microsoft Press.

Wijers, G. (1991). *Modeling Support in Information Systems Development*, Amsterdam, Thesis Publishers.

Winograd, T. and F. Flores (1986). *Understanding computers and cognition*, Norwood, NJ, Ablex.

Wynekoop, J. and N. Russo (1995). "Systems development methodologies: unanswered questions", *Journal of Information Technology,* **10**(2): 65-73.

Wynekoop, J. and N. Russo (1997). "Studying system development methodologies: an examination of research methods", *Information Systems Journal,* **7**(1): 47-65.

Yang, Y. (1995). "Coordination for process support is not enough!" *Proceedings of the 4 th European Workshop on Software Process Technology*, Noooedwijkerhout, Holland, April 3-5, Lecture Notes in Computer Science**:** 205-208.

Yin, R. K. (1994). *Case Study Research: Design and Methods (2nd ed.)*, Newbury Park, Sage.

Yourdon, E. (1989). *Modern Structured Analysis*, London, Prentice Hall.

Zachman, J. A. (1987). "A Framework for Information Systems Architecture", *IBM Systems Journal,* **26**(3): 276-292.

Zelinka, I. and J. Lampinen (1999). "DELA: An Evolutionary Learning Algorithms for Neural Networks". *Proceedings of the 5th International Conference on Soft Computing*, Brno, Czech Republic, June 1-4**:** 410-414.

Appendix I: Publications

Publication I

Architecture as a Coordination Tool in Multi-site Software Development

Ovaska, P., M. Rossi, P. Marttiin (2004): "Architecture as a Coordination Tool in Multi-site Software Development", in *Software Process: Improvement and Practice*, 8(4), pp. 233-248, John Wiley & Sons Ltd.

Publication II

Architecture as a Predictor of System Size - A Metaphor from Construction Projects

Ovaska, P., A. Bern (2004):"Architecture as a Predictor of System Size – A Metaphor from Construction Projects", *Proceedings of the 16th International Conference on Advanced Information Systems Engineering* (CAISE '04 Forum), Riga, Latvia, June 7-11, Riga Technical University, pp. 193-203.

Publication III

Filtering, Negotiating and Shifting in the Understanding of Information Systems Requirements

Ovaska, P., M. Rossi, K. Smolander (accepted): "Filtering, Negotiating and Shifting in the Understanding of Information Systems Requirements", *Scandinavian Journal of Information Systems*, IRIS Association.

Publication IV

Measuring Requirement Evolution – A Case Study in the E-commerce Domain

Ovaska, P. (2004):"Measuring Requirement Evolution – A Case Study in the E-commerce Domain", *Proceedings of the 6th International Conference on Enterprise Information Systems*, Porto, Portugal, April 14-17, INSTICC, pp. 669-673.

Publication V

Working with Methods: Observation on the Role of Methods in Systems Development

Ovaska, P. (forthcoming in 2005): "Working with Methods: Observations on the Role Methods in Systems Development", *Information Systems Development: Advances in Theory, Practice and Education,* edited by O. Vasilecas, A. Caplinskas, W. Wojtkowski, W.G. Wojtkowski, J. Zupancic, Springer, pp. 185-197.