KUNGL
TEKNISKA
HÖGSKOLAN

Enterprise Information Systems Management

An Engineering Perspective Focusing on the Aspects of Time and Modifiability

Jonas Andersson

April 2002

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Industrial Information and Control Systems
Department of Electrical Engineering
KTH, Royal Institute of Technology
Stockholm, SWEDEN

Ex.R. 02-04
TRITA-ICS-0203
ISSN 1104-3504
ISRN KTH/ICS/R--02/03--SE

ABSTRACT

Evolution of information systems (ISs) is a multi-facetted issue that over time has proved arduous to manage. On the enterprise level of ISs, an organization's total portfolio of interconnected information systems is considered as one system – an *enterprise information system (EIS)*, consisting of course-grained and heterogeneous components that in themselves may constitute complex ISs. In EISs, considerations concerning legacy systems and commercial-off-the-shelf software (COTS) are pervasive. This doctoral thesis addresses management of EISs in primarily small and medium-sized electric utilities that are active on the reformed Swedish electricity market. The enabling reasons for the choice of electric utilities as unit of analysis are the implications of the recent electricity market reformation, utilities' broad range of interconnected ISs, and small and medium-sized enterprises' sparse resources for strategic management.

This work applies an engineering perspective on EISs management by investigating how description techniques and analysis methods from software architecture may be employed as decision support during planning and implementation of system evolution activities. An enabling motivation for the selection of software architecture as reference discipline for this work is its recent achievements in expressing and analyzing complex software systems consisting of coarse-grained software packages, on the basis of quality attributes. A special emphasis is hereby placed on *modifiability* and the *implication of time*. Presented findings imply that the concepts for architectural description, e.g. *quality attributes*, *architectural taxonomy*, and *architectural integration styles*, combined with *scenario-based architectural analysis*, may successfully contribute to enhance the comprehensiveness of the complex problem domain provided by EIS evolution. The chosen approach promotes structured analysis, as well as stakeholder communication and awareness. This work also suggests adaptations of the investigated architectural concepts to increase their applicability on the enterprise level of ISs in small and medium-sized enterprises.

Keywords: Information technology (IT), Information systems (IS), IS/IT management, Strategic information systems planning, Software architecture, Decision support, Quality attributes, Modifiability, Electric utilities, Deregulation.

ACKNOWLEDGEMENTS

The present thesis summarizes the research work I have carried out since early 1996 at the department of Industrial information and control systems, Royal institute of technology (KTH). The initial problem domain addressed was migration and legacy considerations concerning large-scale information systems, with a special focus on the Swedish power industry. The path from there to now has proved far from straight to explore, and has contained several detours, that in some cases have brought me rather far from the context of this work. However, every part of the tour has generated valuable experiences that have taught me to appreciate the explorations during the fulfillment of this Ph.D. project at least as much as the final goal in itself. Nonetheless, it is satisfactory to discern that the final focal point of this doctoral thesis turned out to be very nearby the initial problem domain addressed.

This thesis could not have been written without the support and encouragement from many people to whom I am deeply indebted. First of all, I would like to thank my supervisor, Professor Torsten Cegrell, who has provided the physical and intellectual environment in which this thesis has matured. I would also like to express my warmest gratitude to Professor Johan Schubert for his persistent encouragement and valuable guidance during the final stages of the authoring of this thesis.

Many thanks also to my present and former colleagues at the department of Industrial information and control systems for making every day enjoyable, and for fruitful discussions on any topic. Especially, I would like to thank Dr. Göran Ericsson, Dr. Magnus Haglind, and Mr. Pontus Johnson for excellent teamwork during various stages of this work, Mr. Mathias Ekstedt, and Mrs. Narcisa Jonsson for valuable inputs and discussions, and Mrs. Judith Westerlund for bringing that extra warm and personal touch to the department.

As this thesis would not had been possible without a close cooperation with companies within the Swedish power industry, I would like to thank these organizations, and the individuals therein who have contributed with their time and their expertise. Especially, I would like to mention Mr. Göran Fremrot with Östkraft, and Mr. Lennart Hansson with Sycon.

Without my personal friends and their encouragement this work would probably not have resulted in a concrete thesis - Many thanks to all of You! A special thanks goes to Mr. Johannes Dellby for friendship also when days were tough. Last, but definitely not least, I would like to thank my family for your understanding and support during these years.

Stockholm, April 2002

Jonas Andersson

ENTERPRISE INFORMATION SYSTEMS MANAGEMENT

AN ENGINEERING PERSPECTIVE FOCUSING ON THE ASPECTS OF TIME AND MODIFIABILITY

Jonas Andersson

LIST OF INCLUDED PAPERS

This thesis includes the following four parts, A to D:

PART A: Andersson J., Johnson P., "IT Infrastructure Architectures for Electric Utilities: A Comparative Analysis of Description Techniques," In: *Proceedings of the 33rd Hawaii International Conference on Systems Sciences (HICSS-33)*, Maui, USA, January 2000.

PART B: Andersson J., Johnson P., "Extending Attribute-Based Architectural Analysis to Enterprise Software Systems," In: *Proceedings of the 3rd Australasian Workshop on Software and System Architectures (AWSA '00)*, Sydney, Australia, November 2000.

PART C: Andersson J., Cegrell T., Cheong K.H., Haglind M., "Strategic Management of Information Technology in Deregulated Electric Utilities: Bridging the Gap Between Theory and Practice," In: *Proceedings of the Portland International Conference on Management of Engineering and Technology (PICMET '01)*, Portland, USA, July 2001.

PART D: Andersson J., Johnson P., "Architectural Integration Styles for Large-Scale Enterprise Software Systems," In: *Proceedings of the 5th IEEE International Enterprise Distributed Object Computing Conference (EDOC '01)*, Seattle, USA, September 2001.

LIST OF RELATED PAPERS AND REPORTS

In addition to the papers incuded in this thesis, the author has also published the following papers and reports on topics related to this work:

Andersson J., "Structured Migration and Reuse in Complex Distribution Management Systems: A Cross Process View," In: *Proceedings of Distribution Automation and Demand Side Management (DA/DSM) Europe 95*, Rome, Italy, November 1995.[1]

Andersson J., "A Strategy for Migration on a Deregulated Energy Market: Case Study Experiences," In: *Proceedings of Distribution Automation and Demand Side Management (DA/DSM) Europe 97*, Amsterdam, the Netherlands, October 1997.[1]

Andersson J., Haglind M., Johansson E., Johansson L., *A State of the Art Study of Commercial Industrial Control Systems - version 2.1*, External Report, Ex.R. 96-11, Industrial Control Systems, Royal Institute of Technology, Stockholm, 1997.[1]

Andersson J., *On IT System Integration – Prospects and Consequences of Energy Market Deregulation*, Licentiate Thesis, Ex.R 97-07, Royal Institute of Technology, Stockholm, 1997.

Andersson J., Cegrell T., Cheong K.H., Haglind M., Johansson E., Johansson L., "IT Strategy for Electric Utilities - From a Paper Tiger to an Effective Management Tool," In: *Proceedings of DistribuTech Europe 98 (DA/DSM)*, London, U.K., October 1998.

Andersson J., Johnson P., "Procurement of Integrated IT Systems for the Deregulated Electric," In: *Proceedings of the International Conference on Electricity Distribution (CIRED '99)*, Nice, France, June 1999.

Andersson J., Silwer M., "Enterprise Software System Infrastructure for Electric Utilities: A Step Towards a Feasible Toolbox of Techniques," *Proceedings of the 2nd Nordic Workshop on Software Architecture (NOSA '99)*, Ronneby, August 1999.

[1] Also included as a part in the author's licentiate thesis.

TABLE OF CONTENT

"Scientists discover the world that exists;
engineers create the world that never was."
Theodore Von Karman

Chapter 1

Introduction

1.1 BACKGROUND TO THE RESEARCH

Evolution of software-intensive systems has proved an arduous problem to manage since the introduction of modern computing. The commonality of software "runaway" projects is revealed by e.g. Eason (1988), Glass (1998), Standish group (2000), and Thorp (1998). Over time, however, the nature of problems related to software evolution has developed in pace with technological achievements in the domain of information technology (IT), and escalating requirements on organizations' competitiveness and overall efficiency; constantly increasing the need for more intricate integration of information systems (ISs). Presently, this trend towards integration implies that most organizations' total portfolio of large-scale interconnected software systems, in this thesis termed *enterprise information systems* (EISs), consists of a significant number of integrated software components of various sizes and technologies, origin from a vast number of software vendors and technology epochs. Component granularity may span from a few lines of code, a class, or a simple service, to complex systems that in themselves constitute large-scale ISs. Also, contributing EISs' overall complexity are considerations regarding legacy and commercial-off-the-shelf software (COTS).

Traditionally, enterprise-wide ISs were developed mainly by a single organization, either in-house by the user organizations, or by contracted software houses. Experiences regarding custom-development of large-scale software systems have over time proved deterrent due to high costs and hazardous implementation. Especially, evolution of these systems have proved problematic, e.g. in terms of software maintenance, and integration with collaborating systems. Many of these

systems are still in operation. Although many of them fulfill business-critical functions, they constitute an increasing problem to their owners due to obsolesce (Bennett 1995; Bennett 2000; Sneed 1995).

An increasing part of new ISs consist entirely, or to large extend, of prefabricated software. The expected promises of COTS include lower development costs, faster implementation, and enhanced integration of ISs. However, COTS brings several potential problems that must be dealt with in order to exploit its advantages. For instance related to that (especially coarse-grained) COTS components may fail to meet specific requirements, and that user organizations' influence on vendors' product lines is limited. Moreover, COTS may require extensive effort for adaptation and integration before it may be put into operation.

The increased focus on COTS also influences customer-supplier relationships as influence on functionality and characteristics of COTS software must be exercised indirectly by user organizations, e.g. by influencing vendors' product lines through user groups or standardization bodies. In addition, considerations concerning third-party COTS components included in vendors' product have become an important issue for user organizations, as they may have a significant impact on EISs from a life cycle perspective. Moreover, software vendor roles are becoming more diversified. Vendors that offer a broad range of products and services tend to be fewer and larger; whereas some IS vendors tend to be more specialized. Other vendors niche themselves as system integrators. This implies changing customer-supplier relationships.

Thus, EISs are *heterogeneous* systems that consist of other systems. In the vein of the growing overall complexity regarding EISs, decisions related to these systems have become increasingly difficult to make. Altogether, the problem domain related to EIS represents a new conceptual echelon; the *enterprise level of ISs*, on which business objectives and organizational constraints are transformed into overarching decisions concerning organizations' use of IT, or the potential of IT is identified and translated into business opportunities.

This doctoral thesis addresses enterprise information systems management (EISM) in primarily small and medium-sized electric utilities that are active on the reformed Swedish electricity market. The enabling reasons for the choice of electric utilities as unit of analysis are the implications of the recent electricity market reformation, utilities' broad range of interconnected ISs, and small and medium-sized enterprises' sparse resources for strategic management. This work further applies an engineering perspective on EISM by investigating how software architecture description and

analysis may be employed to provide decision support during evolution of EISs (Bass et al. 1998; Kazman et al. 1998).

One of the most written causes of EIS investment failure is that too much attention is placed on technology itself, rather than its characteristics or quality attributes, and its links with organizational factors (Luftman (ed.) 1996). Therefore, an enabling motivation for the selection of software architecture as reference discipline for this work is its recent achievements in expressing and analyzing complex software systems consisting of coarse-grained software packages, on the basis of quality attributes (Medvidovic and Taylor 1998). Quality attributes[2] may except for expressing software qualities, also communicate organizational parameters (Bass et al. 1998), such as risk, opportunities, awareness, control, and competence. To focus the scope of this work towards evolution of EISs, the quality attribute *modifiability* together with temporal considerations, i.e. *the aspect of time*, have been selected for further scrutiny.

1.2 RESEARCH RATIONALE

As pointed out above, this work applies an engineering perspective on EISM by investigating how software architecture description and analysis may be employed to provide decision support during evolution of EISs (Bass et al. 1998; Kazman et al. 1998). The theoretical approach of this project is multi-disciplinary in the sense that it addresses previous research contributions in the field of software engineering and aims to relate this to EISM. The target readers of this thesis are both EIS practitioners, and other researchers who wish to further the findings presented in this work. Below, the various elements in the research design are described. Also, a delimitation of the scope for this work is presented together with some clarifying definitions on the terminology used throughout this thesis.

1.2.1 RESEARCH QUESTIONS AND HYPOTHESES

To guide this work, a number of research questions were formulated. Their overall purpose was to guide the collection of data and the formulation of hypothesis. The first question aims to provide an in-depth understanding of the characteristics of EISs in small and medium-sized electric utilities, whereas the two concluding questions address the impact of software architecture description and analysis as

[2] In information systems literature also referred to as systemic competencies (Hendersen and Venkatraman 1996).

decision support for EIS evolution. The presented empirical findings according to the first question are partly based on results from the author's licentiate thesis (Andersson 1997b). The research questions are formulated as follows:

- What is the present state-of-the-practice regarding management of EIS evolution in small and medium-sized electric utilities?
- What impact will the introduction of software architecture description and analysis have as a tool for strategic decision support during evolution of EIS?
- How may software architecture description and analysis be adapted to better aid strategic decisions during evolution of EISs?

From these research questions, more confined questions have been formulated within each field study. Except for providing a delimitation of the scope of this thesis, the research questions also are intended to validate the following hypothesis:

Present state-of-the-art regarding EISM provides inadequate decision support for small and medium-sized electric utilities during EIS evolution to promote effective analysis, stakeholder communication, and stakeholder awareness. One prominent reason for this state of affairs is the prevalent lack of means for conceptualizing the problem domain and to make explicit organizational, computational, and temporal dependencies in terms of trade-offs between qualitative parameters.

1.2.2 RESEARCH DESIGN

The research design (see Figure 1) essentially consists of three parallel tracks encompassing: (1) *research methodology*, (2) *field studies*, and (3) *theoretical studies*. The expected advantage of the concurrent approach has been to attain mutual support between the activities by allowing advancements gained in each track to also influence the progress of the others. In addition, the project can be divided into two main phases, where Phase 1 primarily focuses on identifying feasible research questions, and to gain initial insight in previous work in the area, whereas Phase 2 primarily focuses on testing and to furthering applicable theory.

The theoretical domains that formed the basis for the research during Phase 1 were prevalent contributions on middleware and requirements engineering that were deemed relevant from a user organization perspective. Also, literature on project management contributed to the theoretical frameworks applied during Phase 1. The case studies during Phase 1 were organized as a multiple exploratory case study

(field study Alpha) with the purpose to obtain in-depth domain knowledge regarding small and medium-sized electric utilities. In particular, their efforts to provide EISM were investigated. As a part of the field study, the companies' portfolio of present, and planned IS projects were documented. The author's licentiate thesis is partly based on results gained during field study Alpha (Andersson 1997b). Furthermore, a part of the investigation that focused on IS/IT strategies was later refined and expanded by Cheong (1999) and Haglind (2002). As the research approach applied during Phase 1 was *descriptive* and *interpretive*, research methodology according to Robson (1992), Walsham (1993), and Yin (1994) was applied.

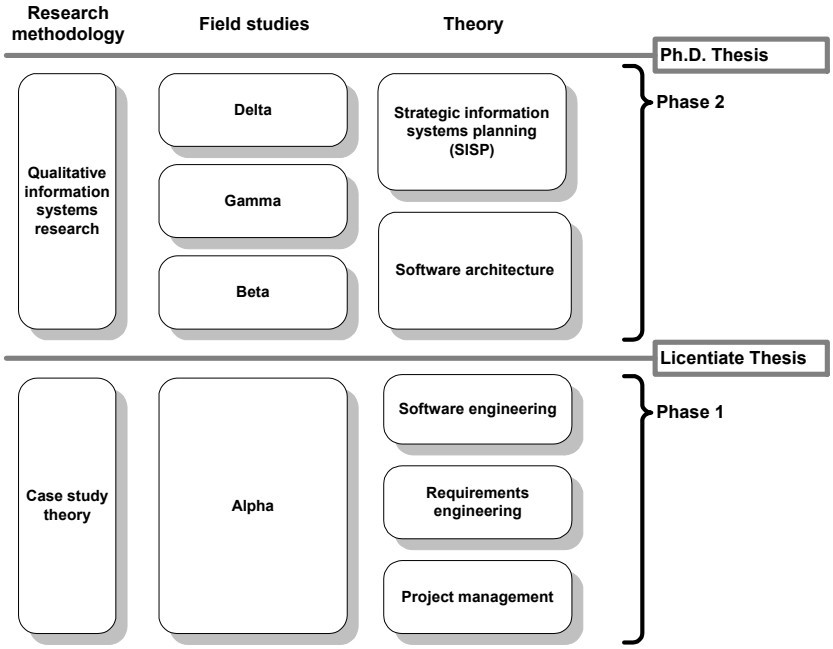

Figure 1. Research design.

Whereas the research carried out during Phase 1 essentially contributed to the formulation of the research questions presented in Chapter 1.2.1, the field studies carried out during Phase 2, more specifically sought to provide information on each of them. The field studies were augmented with further studies of methodology for IS research (Galliers 1992) focusing on *qualitative research* (Myers 1997) and *action research* (Baskerville 1999; Baskerville and Wood-Harper 1996; Dick 1999). During

Phase 2, three major field studies were carried out: Beta, Gamma, and Delta. Field study Beta investigated an acquisition of a settlement system for the deregulated electricity market. Since the IS at stake had to be integrated with several existing ISs, the case could be characterized as a modification of an existing EIS. Especially organizational qualities such as *trust*, *distribution of risk*, *responsibility*, and their implications for the choice of vendor and technical solution were investigated. Field studies Gamma and Delta address one organization's effort to acquire a business system, and may thus be considered as an attempt to create a total EIS. Whilst the first of the two field studies constituted an explanatory case study carried out in retrospect, the latter field study was a participatory case study using action research as research method.

The main motivation behind the selection of qualitative research methods according to and is its abilities to provide rigor in real-world research (Galliers 1992; Myers 1999). The applied research methodology is further presented in Chapter 6, and a summary of the field studies Alpha, Beta, Gamma, and Delta is provided in Chapter 7.

1.2.3 DELIMITATION OF SCOPE AND KEY ASSUMPTIONS

The primary unit of analysis in this work is small and medium-sized[3] electric utilities[4] active within the reformed Swedish electricity market. Both distribution network operators, that are still monopolistic, and electricity retailers have been investigated. In some cases also IS vendors have been included in the data collection. A motivation to the selection of electric utilities as the unit of analysis is given in Chapter 3. Two key assumptions regarding the area of EISM have been guided this work: Firstly, the existence of a gap between theory and practice concerning EISM, and secondly that it is a gap (or lack of alignment) among different disciplines attempting to address evolution of EISs. Furthermore, this thesis applies a user organization perspective of EISM.

The title of this thesis comprises the words "engineering perspective." As the word engineering is intensively debated in new disciplines, such as software engineering, a clarification of the author's interpretation of "good" engineering is provided; an engineering perspective indicates the intention to take the responsibility of the final

[3] The European community defines small and medium sized enterprises (SMEs) as companies with less than 250 employees, a turn-over lower than 40 million ECU, and which are owned for less than 25% by non-SMEs, except banks or venture capital companies.
[4] With exception for the organizations investigated in field study Beta, which were subsidiaries of a major Swedish electric utility.

product of an effort in the sense that important design decisions, and trade-offs, are explicitly identified and dealt with in a structured way. Hence, both technical and organizational parameters should be made explicit and mitigated. Moreover, entrustment of decisions should only be made to stakeholders that have the means, mandate, and the incentives to successfully realize them. Below, primarily some further delimitation and major assumptions concerning the scope of the doctoral thesis are described and justified:

The enterprise level of information systems. An important delimitation for this work is its focus on the enterprise system level of ISs. Perhaps the most pertinent difference between separate ISs and the enterprise level of ISs is in the size and the heterogeneity of the components. EISs are mainly constructed by the integration of complete ISs as components, magnifying both size and complexity of the resulting system. Also, a prominent characteristic for EISs seems to be their heterogeneity concerning connectors; components in EISs are generally not initially designed with the intention to be integrated. As the total life cycle of EISs has proved to be comparably long, sometimes stretching decades, overall heterogeneity is increased by the incorporation of several technology generations, e.g. in terms of operating systems, middleware technologies, and design principles. Another distinguishing aspect of the enterprise level of ISs is that management of evolution efforts commonly comprises multiple projects. The characteristics of EISs, and their components and connectors are further discussed in Chapter 3.

The user organization perspective. This thesis' organizational perspective is the one of an organization that needs ISs in order to support its business operations. I.e., ability as regards planning and implementation of EISs may be a matter of survival for the organization, but is, unlike software vendors, nonetheless not user organizations' core business. Despite this difference, it is pointed out that user organizations' situation share many similarities with vendors' regarding design and implementation of large-scale ISs.

Commercial-off-the-shelf-software (COTS). In this work, an ambition to maximize the use of COTS in EISs is assumed. In the past, EISs were usually custom-developed for their organizations. This situation had the obvious advantage of providing a great deal of latitude to develop software that corresponded with predefined requirements. Over time, however, custom development of large-scale ISs has frequently proved costly, resource demanding, and hazardous (Glass 1998; Standish group 2000). In an attempt to circumvent these problems, most organizations presently strive to construct their EISs out of prefabricated software to as a large degree as possible. Ideally, the use of COTS implies the construction of

EISs by integration of prefabricated software components by only limited adjustments, typically by setting certain parameters. However, COTS may require extensive effort to adapt and to integrate (cf. e.g. Wallnau et al. 2002). Decisions concerning EISs must therefore delicately mitigate expected benefits of using COTS against the effort needed to integrate and adapt the COTS into the target EIS. Also, employment of COTS has an impact on the process of designing an EIS. Whereas custom-developed EISs may be designed in a comparably *continuous* design space, the design space for EISs primarily consisting of COTS becomes rather *discrete* to its nature (cf. Lane 1990). See Chapter 3.4 for a discussion on COTS and its implications to EISs and EISM.

The aspect of time. In separate IS projects, activities are often carefully scheduled into a time-plan with well-defined milestones, goals are generally well defined, and temporal constraints are commonly made explicit by concepts such as projects' critical path. Still, project management is far from uncomplicated, and failed projects are common. In EISM, even more complicated chains of actions must be coped with, as EIS evolution commonly comprises multiple projects carried though during an extensive period of time. Temporal constraints and dependencies constitute a first-class issue; issues such as selection, prioritization, coordination, and resource allocation between ongoing or potential IS projects must be delicately dealt with. Moreover, delimiting the length of IS projects is a vital factor for mitigating risk and promoting feedback of experiences to subsequent projects.

Another important time-related aspect is the one of increasing disorder, "entropy," in ISs left unattended (Bennett 1995; Brooks 1995; Parnas 1994; Sneed 1995). Reasons for this increasing disorder include (1) decreasing awareness and knowledge of ISs as staff are replaced and things are forgotten, (2) actions taken in ISs without considering their side-effects to other parts of the systems, e.g. inconsistencies of data and functionality and deteriorating technical uniformity, and (3) changing technical and organizational systemic context (e.g. due to company mergers and acquisitions, or new operating systems and middleware). Note that the issue of software obsolesce (Sneed 1995) is present both in separate EIS components, and in the EIS as a whole. To conclude, this work assumes that temporal constraints constitute a first-class consideration when planning and implementing EISs. The impact of time in EISM is further discussed in Chapter 3.2.

1.2.4 DEFINITIONS

ISs and IT constitute a vast field that lacks an unambiguously terminology, and even less a taxonomy that serves the purpose of describing ISs distinctly enough to

effectively promote structured analysis, or stakeholder awareness and communication. Hence, key terminology must be clarified in each context applied. As several of the enumerated terms defined below are subject for vivid discussions, it is stressed that the definitions given here are not intended as additional firewood to these discussions. Instead, they should be regarded as clarifications of some fundamental terms within the context of this thesis. See Chapter 4 for further definitions concerning terms and concepts related to software architecture, and discussions on these terms.

Enterprise information system (EIS). An organization's total portfolio of interconnected ISs considered as one system (cf. Chapter 3).

Management and decision support. An important part of this thesis' contribution is the application of software architecture description and analysis as a framework for EISM. Partly in line with the definition of frameworks given in Cheong (1999), a *framework* is a collection of concepts, methods, theories, principles, and ideas that provide guiding principles and directions for *decisions*. These decisions may be a part of a *planning process* or provide guidance with regard to *real actions*, i.e. no distinction is made between decisions regarding planning and implementation of EISs these two issues from an EISM perspective are intrinsically coupled. A *decision* is defined as a choice made by some entity of an action from some set of alternative actions. The entity, i.e. the decision maker, may, in the context of this work, be either an individual or a group. A "good" decision identifies an alternative that the decision maker *believes* will prove at least as *good* as other alternative actions (Doyle and Thomason 1999).

Information technology (IT) and information systems (ISs). As a result of the widespread use of computing and communication technology, especially the recent growth of the Internet, the word IT has become a rather amorphous term that encompasses a lot of issues that in one sense or another have some connection with computer technology. IEEE Computer Society's task force on IT for business applications (TFIT), has formulated a working definition of IT: "*Information Technology is that set of technology components and operation procedures that support a business or organization in managing information so that it can meet its mission. IT embodies the hardware, software, algorithms, databases, tactics, and man-machine interfaces used to create, capture, organize, modify, store, protect, access, and distribute information for ultimate use by people.*" Earl (1989) states that "*IT comprises computing, telecommunications and automation technologies*" from a *technical* standpoint, but also may be regarded as an *activity* that "*comprises all the supply, development and use activities in which an organization has to be involved if it wishes to exploit these technologies to its advantage*," and a philosophy representing the continuation

of the "*aims, means and responsibilities*" typical in organizations during the first 30 years of computing. According to Frenzel (1996), "*information technology is the term that describes the disciplines encompassing computer systems, telecommunication networks, and multimedia applications.*"

IS as a discipline descends from the study of (primarily administrative) computing as an instrument for organizational problem solving (Lyytinen 1987), and has evolved over time into a broad research field that attempts at put IT in its context. Thus, IS comprises a wide array of non-technical factors such as management, organization, economy, legal aspects, and human factors.

Although IT may be considered as the technology domain within the broader field of ISs, there is much confusion around these terms and their linkage. Not uncommonly, definitions are avoided, or they are used in conjunction with each other, (IS/IT), when a broader area of computing and computing related topics is addressed. From the perspective of this work it is sufficient to regard IS/IT as a timeless and summarizing term for the domain related to the computing of information for whatever reason, and of which this thesis address a part, namely decision support for EIS evolution.

Enterprise information system architecture (EISA). The term EISA is based on the analogy to traditional software architecture. Hence, the term will be used to denote the structured description of the collection of software-based systems (components) supporting the operations of an enterprise as a set of components and connectors with assigned properties. It is, however, pointed out that there is no consensus on the definition or even the choice of the term EISA (DISA 1996; The Open Group 1999; Zachman 1987). According to (Bass et al. 1998), software architecture can be defined as:

"The software architecture of a program or computing system is the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships among them."

However, to make model descriptions sufficiently rich to promote stakeholder awareness and communication, as well as structured analysis, the author also includes important related information, the *rationale* of the architecture (Perry and Wolf 1992), such as the motivation for previous design decisions. Other content may be descriptions of *migration paths* to capture the time dimension of EIS evolution. In that sense, this thesis adheres to Maier's rule-of-thumb regarding system architecture, based on the role of architects, namely that a system

architecture is the work result produced by the architect to help its client to make decisions regarding the system (Maier and Rechtin 2000):

"An architecture is the set of information that defines a systems value, cost, and risk sufficiently for the purposes of the systems sponsor."

Quality Attributes. The externally visible properties of a system, according to the definition on software architecture provided by Bass et al. (1998) above, are generally expressed as qualities of the system. Qualities of a system are above its functionality, that in fact may be expressed as a quality attribute. Software qualities may be divided into three categories: factors that may be directly measured by observing a system (e.g. performance, security, availability, and functionality), factors that can only be measured indirectly (e.g. modifiability, reusability, and testability), and finally business qualities that influence and are influenced by the first two categories of quality attribute (e.g. trust, risk, awareness, and ability) (Bass et al. 1998; Heineman and Councill 2001; McCall 1977).

This work focuses on *modifiability* of EISs (in some literature also referred to as maintainability); a quality attribute that on the enterprise system level grasps several important considerations for the planning and design of EISs. Modifiability is further discussed in Chapter 3.2, and a background to quality attributes in general is given in Chapter 4.2.2.

1.3 RELATED WORKS

Evolution of ISs is a multi-facetted issue. Although, no other works have been found with this thesis' combined emphasis on decision support for EISM, software architecture description and analysis, small and medium-sized organizations, and a user-organization perspective, much work is at hand on each enumerated aspect. As the complexity of EISs continuously tends to grow, there is a general strive among both practitioners and academia to provide better means for EISM, in order to catch up and, if possible, gain ground concerning decision support for EIS evolution.

Essentially, previous work related to this thesis may be found in three adjacent domains: *software engineering, (management) information systems*, and *systems engineering*. Some literature on *strategic management* and *project/program management* also addresses issues that are related to this work. In addition, literature written by *practitioners* with the purpose to share and in some cases attempting to codify experiences provides useful empirical information.

Within the domain of IS, there are many written sources focus on the issue of IT in organizational contexts (cf. e.g. Lyytinen 1987; Walsham 1993; Ward and Griffiths 1996). Several textbooks address EISs in relation to business process re-engineering (Davenport 1993; Davenport 2000; Earl 1989; Hammer and Champy 1993). Others directly address the decision process for EISM by discussing (strategic) management of IS/IT (Earl 1993; Frenzel 1996; Lederer and Gardiner 1992; Lederer and Salmela 1996; Mintzberg et al. 1998). Moreover, the close relation between, and the need for aligning IS/IT with, business objectives and business requirements are addressed in literature on strategic *alignment* (cf. Hendersen and Venkatraman 1996; Luftman (ed.) 1996).

The reference discipline for this work is manly software architecture. This domain that originally was oriented towards the investigation of, from this work's perspective, lower system levels, has gradually shifted its focus to embraces larger and more complex systems consisting of coarse-grained components, such as EISs. Contributions in the vein of software architecture description that are related to this work include the introductions of views and viewpoints (Kruschten 1995), architectural styles and patterns (Buschmann et al. 1996; Fowler 1997; Garlan and Shaw 1996; Gamma et al. 1998; Grand 2002; Schmidt et al. 2000), and the employment of the unified modeling language (UML) for architectural description (Kobryn 1998; Medvidovic et al. 2002). Also, several architectural frameworks for description of EISA have been suggested (cf. e.g. DISA 1996; The Open Group 1999; Zachman 1987).

Researchers at Software Engineering Institute (SEI), Carnegie Mellon University, have carried out much work in the integration of COTS intensive systems, architectural analysis, and quality attributes (Bass et al. 1998; Kazman et al. 1994; Kazman et al. 1998; Meyers and Oberndorf 2001; Wallnau et al. 2002). Their contributions on scenario-based architectural analysis have particularly served as an important prerequisite for this work (cf. Bass et al. 1998; Kazman et al. 1994; Kazman et al. 1998). Scenario-based architectural analysis is also addressed by e.g. Bosch and Molin (1999), and Lassing et al. (2002). In addition, several recent theses address architectural integration and evolution of heterogeneous software systems from a variety of perspectives, and on various system levels (cf. e.g. Abd-Allah 1996; Bengtsson 2002; De Line 1999; Dellarocas 1996; Gacek 1998; Häggander 2001; Mattson 2000; Ockerbloom 1998). Also, systems engineering literature addresses architecture and architectural design of large-scale systems (Maier and Rechtin 2000). A comprehensive introduction to systems engineering is given in e.g. Blanchard (1991).

Finally, several standards that relates to this work are at hand, e.g. IEEE 1471-2000 (2000): recommended practice for architectural description, and ISO/IEC 15288 (2001) for life cycle management of hierarchically composed system structures. It is stressed that the application of these standards not in any way are in opposition to this work. Conversely, they may be applied to provide consistent terminology or to align processes in collaborating organizations.

1.4 MAIN CONTRIBUTION OF THIS THESIS

Undoubtedly, EISM in general will remain as a first-class issue for all organizations that operates EISs. As the impact of IT strengthens, these issues will become ever more complicated whilst complexity of EISs continues to grow in terms of heterogeneity, size, and the number of stakeholders affected. Presently, both practitioners and academia strive to provide better means for EISM, in order to catch up and, if possible, gain ground in relation to the growing problem domain in terms of enhanced methods for planning and implementation of EISs.

In view of these attempts, the main contribution of this thesis is that through an inter-disciplinary study increases the knowledge on how software architecture description and analysis may be extended to provide decision support for EIS evolution. In more detail, the contribution of this work is divided into three major parts, I, II, and III:

I. A state-of-the-practice description of the EISs and their technological and organizational context in small and medium-sized electric utilities. To provide a basis for the theory building and testing in the present work, an elaborate set of case studies (Alpha, Beta, and Gamma) has been carried out. The findings from these studies have contributed to establish a state-of-the-practice description of the shortcomings in prevalent practice of EISM, especially concerning issues related to long-term evolution of EISs (cf. Part C, and the summaries of the field studies Alpha, Beta, and Gamma in Chapter 7). Furthermore, the characteristics of the components in EISs in small and medium-sized electric utilities have been compiled. The results reveal heterogeneous EISs consisting of a mix of coarse-grained COTS and legacy components and middleware, which principally was not designed for interaction. Here, the descriptive and interpretive rendering of EISs as presented in Parts A to D, in itself provides an improved understanding for the context in which decisions related to evolution of EISs are made. The compilation of the characteristics of EISs and its implications concerning long-term modifiability is presented in Chapter 3.

II. Evaluation and adaptation of architectural analysis for its capabilities as decision support for EIS evolution. Based on its abilities to make explicit trade-offs between system qualities, and thereby mitigate risks and opportunities on various planning horizons, scenario-based architectural analysis has been employed as a process for EISM. In Chapter 4, a process for scenario-based architectural analysis and its capabilities to provide decision support for evolution of EISs is further discussed. The proposed process is adapted and tested for the enterprise system level in Part B, and further developed as a part of a novel approach for strategic ISs planning in Part C. A summary of the proposed approach is presented in Chapter 5. In short, the proposed framework: (1) is *implementation-centric* in the sense that it stresses the importance of bridging the gap between planning and implementation, (2) uses *structured description techniques* and quality attributes in order to promote awareness and communication between stakeholders, and to provides rationale for qualitative analysis of different alternatives, (3) recognizes and supports *strategic alignment* between business strategies, processes, and actions concerning the EIS, and (4) accentuates the continuous and iterative *assessment* and *prioritization* of modifications to the total EIS, by adding, changing, or replacing *components*.

III. Evaluation and adaptation of architectural description techniques for its capabilities as decision support for EIS evolution. To bring structure to the analysis process addressed above, several concepts from software engineering (primarily software architecture) have been investigated for their capabilities of conceptualizing the problem domain formed by long-term evolution of EISs. In particular, *quality attributes* (see Chapter 4.2.2, and Parts B and D), *views* and *viewpoints* (see Chapter 4.4.1), *architectural integration styles* (see Chapter 4.4.2 and Part D), and *notations for architectural description* (see Chapter 4.4.3 and Part A) have been evaluated and partly adapted for use in EISA description and analysis. Leading presumptions in the evaluation have been the intuitive comprehensiveness of the concepts as stakeholders in user organizations are not trained software engineers, and the concepts' abilities to express important and problematic situations concerning future evolution of EISs. Especially, their capabilities of expressing temporal considerations and dependencies have been scrutinized.

To conclude, it is the author's wish that the increased knowledge provided by this thesis, will contribute to enhance EISM practice in small and medium-sized electric utilities, and will encourage to further research in this vein.

1.5 OUTLINE OF THE THESIS

This doctoral thesis consists of an Introduction and summary and four published papers. The published papers are enclosed in the thesis as Parts A to D; the Introduction and summary is further divided into ten chapters. Chapter 1 provides an overall background to the research topic this thesis, presenting research questions, hypothesis, and the overall research design. Also, the contribution of this work is presented together with references to related works. In Chapter 2, a motivation of the selection of electric utilities as the unit of analysis is given, together with a brief introduction to the reformed Swedish electricity market. Also, the functional areas of ISs within distribution network operators and electricity retailers are described. Thereafter, in Chapter 3, the features of EISs are explained. In particular, considerations regarding system evolution (primarily concerning modifiability and time) are discussed in more detail.

In Chapter 4, the application of architectural analysis and description as means for decision support in evolution of EISs, are discussed. Also, references to the more elaborate renderings on each addressed topic in Parts A to D are provided. Chapter 5 exemplifies the application of software architecture description and analysis in EISM by suggesting a novel framework for strategic ISs planning in small and medium-sized enterprises. Applied research methodology is presented in Chapter 6, together with some brief summaries of field studies Alpha, Beta, Gamma, and Delta in Chapter 7, and a summary of included parts A to D in Chapter 8. Chapter 9 summarizes the most important findings of this doctoral thesis and emphasizes some implications of these. The Introduction and summary concludes with references in Chapter 10.

Chapter 2

Information systems and electric utilities

2.1 ELECTRIC UTILITIES AS THE UNIT OF ANALYSIS IN INFORMATION SYSTEMS RESEARCH

There are several reasons behind the selection of Swedish electric utilities and electricity retailers as the unit of analysis for this work: (1) electric utilities operate a broad spectrum of ISs, thus offering a rich empirical basis for the study these companies' attempts to manage this plethora of interconnected ISs, (2) the electricity market reformation process that was put into operation on January 1, 1996, has forced these companies to virtually simultaneously undergo radical changes as to organization and ISs, (3) the Swedish power industry has a long history of university cooperation that is helpful for researchers who wish to perform participatory research of contemporary phenomenon, such as IS evolution, and (4) as a consequence of (1) and (2), these companies have a real need to enhance and adapt their capabilities concerning EISM, hence providing means for research settings that provide mutual benefit for both researchers and the investigated organization.

In the remainder of this chapter, a brief summary of the market conditions on the reformed Swedish electricity market is given (STEM 2000; STEM 2001; SvK 2001) in Section 2.2, and the different types of ISs used by electric utilities are described in Section 2.3. The chapter is intended to provide a background to the more specific

issues concerning EISs discussed in Chapter 3, and the evaluation of architectural description and analysis as a means for decision support provided in Chapter 4.

2.2 ELECTRICITY MARKET REFORMATION

Swedish utilities have been operating on a stable market for a long time. This is partly due to well-defined market rules, and partly due to a stable technical process. However, the reformation of the Swedish electricity market that was put into operation January 1, 1996, has led to new conditions for the utilities, e.g. in terms of the division of the energy trading and network owning utilities into two legally separated organizations, one *distribution network operator* and one *electricity retailer*. After the deregulation, the Swedish electricity market consists of several independent actors according to Figure 2.

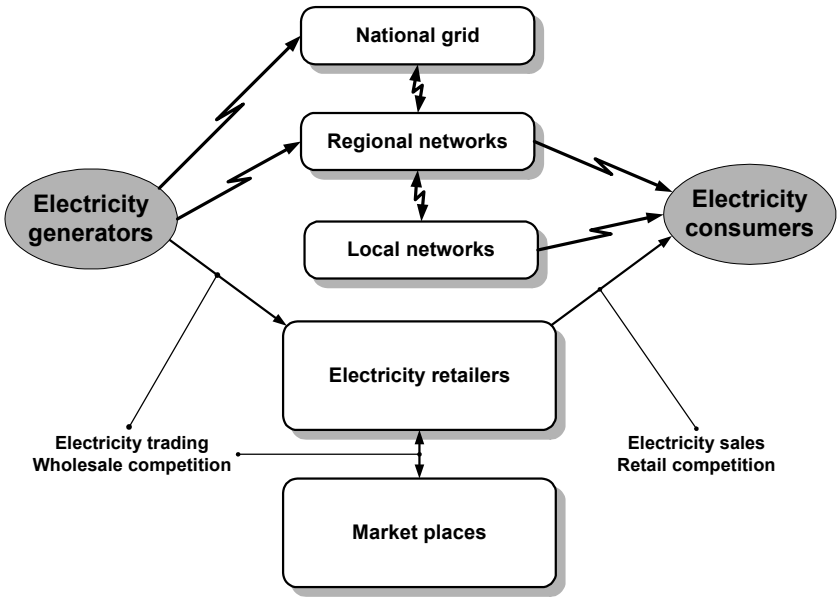

Figure 2. Physical flow of electricity and the relationships between actors on the reformed Swedish electricity market (SvK 2001).

Electricity generators generate the electricity and feed it into the grid. In Sweden approximately half of the generated electricity consists of nuclear power and the other half of hydropower. A minor share, approximately 15%, is produced by other energy sources such as thermal power and wind power.

The *owners* of the national, regional, and distribution networks are responsible for transmitting the electricity from generators to consumers. The *Swedish national grid* authority owns and operates the *national grid* and has the overall responsibility for the Swedish power system, i.e. to ensure total reliability and availability, and to ensure that domestic generation and import in any given moment corresponds to electricity consumption and export. To assume the latter role, Swedish national grid cooperates with several electricity retailers (who, in this role, are termed *balance providers*), which by concluding agreements accept the financial responsibility for ensuring a power system in balance, either by planning its own generation or by trading with other balance providers on the electricity market places. *Regional networks* transport electricity from the national grid to local networks and some large industrial electricity consumers. The *local networks* distribute the electricity to the majority of the consumers, e.g. households and businesses. All network operations are a regulated monopoly, and the tariffs and other conditions are supervised by the *Swedish National Energy Administration*.

Electricity retailers buy electricity directly from generators, or else through the Nordic power exchange (NordPool), and sell the electric energy to consumers. Most retailers were formed by the separation of electricity retailing departments from the rest of the business operations in existing electric utilities at the time for the reformation. Some of these "original" electricity retailers have been sold, have merged with other electricity retailers, or, in some cases, have been sold to other companies that wish to niche themselves on the electricity market. Presently, most of the electricity trading is done by bilateral agreements directly between generators and electricity retailers. However, an increasing part of electricity trading takes place on organized market places, e.g. NordPool. NordPool is divided into the *spot market*, in which electricity is traded in hourly contracts for physical delivery within the next 24-hour period, and the *forward market* that is a pure financial market for price assurance and risk handling. The main benefit of trading on the exchange is that transaction costs are lower than those for bilateral trade agreements.

Consumers, ranging from industries to households, must as a result of the electricity market reform have two separate contracts in order to consume electric energy. To buy electricity, the consumer must have a contract with an electricity retailer, and to connect to a distribution network the consumer must have an agreement with a distribution utility. To determine consumers' electricity consumption, larger customers have electricity meters that automatically report actual electricity consumption on an hourly basis. The consumption of domestic customers is

estimated by profile settlement based on load curves for different categories of customers.

2.3 INFORMATION SYSTEMS WITHIN ELECTRIC UTILITIES

Electric utilities are industrial companies with a broad mixture of ISs. Traditionally, these companies have a rather technology driven management tradition, and were early adopters of advances in software and computer science. Even small and medium-sized electric utilities rely extensively on computerized tools for their daily operations. As an example, a mid-sized electric utility could operate more than 100 interconnected ISs, acquired, integrated, and maintained over a long period of time (Andersson, 1997a).

Electricity market reformation has imposed demands for new and changed functionality e.g. regarding *network balance settlement*, *reporting of network availability*, and *information separation in the divided electric utilities*. Moreover, the electricity retailers, now subject to the free market forces, find themselves in an acute need of ISs support for their entire business process. As utilities, due to changed market conditions, are going through a surge of mergers and acquisitions, the corresponding EIS (cf. the definition given in Chapter 1) must be harmonized with the newly formed organizations in order to achieve alignment between business goals, work processes, and supporting ISs (cf. Luftman (ed.) 1996). Except for adapting the organizations to changing market conditions, utilities have a major need for renewing and to further integrate the utilities present portfolio of interconnected ISs, mainly due to the increased financial strain provided by both increased competition and deregulatory stipulations. For example, of the few small and medium sized electric utilities under study in field studies Alpha, Beta, and Gamma, only one had invested in any solution for *enterprise application integration* (EAI). Most implemented integration solutions were commonly simplistic based on flat file transfer, or desktop integration. All implemented component interfaces were considered as proprietary, except for these related to the exchange of metering data with external actors such as Swedish National Grid, which is based on EDIel[5].

[5] A standardized message format for exchange of electricity market related information, e.g. meter data, and updated customer information. The message format is based on United Nation's EDIFACT (Electronic Data Interchange for Administration, Commerce and Transport) standard, exchanged by the X.400 protocol for electronic mail.

2.3.1 DISTRIBUTION NETWORK OPERATORS

A distribution network operator (cf. Section 2.2) operates ISs to ensure the delivery of electricity to the customers within its area of concession. Below, the major categories of ISs in small and medium-sized electric utilities are briefly described (Andersson 1997a; Andersson et al. 1998; Cegrell 1986; Cheong 1997; Engelken 1999; Persson 1998). It is, however, stressed that the emphasis and priorities considering ISs may vary significantly between different utilities.

Administrative systems incorporate functions for e.g. accounting, financial reporting, and the (financial) asset management, and payroll management. Note that asset management is usually divided into two separate parts: one financial and one technical that commonly reside in different ISs. An emerging type of ISs in electric utilities is the Enterprise Resource Planning (ERP) system. These systems are intended to replace several existing stovepipe[6] administrative applications and support business processes and hence provide both vertical and horizontal integration in organizations. Commonly, ERP systems consist of packaged modules for various purposes that are designed for (proprietary) integration. Several of the larger Swedish electric utilities are in the process of introducing such systems, commonly starting with modules such as finance, accounting, (financial) asset management, and payroll management.

Real-time systems provide real-time information and the infrastructure to control the network remotely. The core of the functionality for distribution automation (DA) is commonly a real-time SCADA[7] system. Except for traditional SCADA functionality, e.g. data collection, state supervision, and switch orders, systems for DA may also include more advanced functions such as volt/var and feeder optimization, and load management. For the collecting meter data, especially for customers not comprised by profile settlement, automated meter reading (AMR) systems collect meter data on an hourly basis.

Geographical information systems (GIS). To manage their considerable volumes of spatial bound data digitally, and to support other ISs with as-designed and as-built models of the distribution network, utilities implement GISs. In addition, these systems also provide a common easy-to-use interface (including Web access) towards other ISs, e.g. for planning and engineering of the network.

---

[6] Stovepipe application is a popular name on ISs that address and solve narrow problems within a part of an organization, e.g. a department (Linchicum 2000).
[7] SCADA, Supervisory control and data acquisition.

Work management system (WMS). WMSs manages the life cycle and the flow of work orders from job initiation to job design, from resource management to processing of the job closing information. Modern job design typically involves engineering design and cost estimates, requiring graphical access and editing of the facility data in the utility office or in the field. Resource management involves the tracking of crews, vehicles, and materials through the life cycle of each job. Job closing involves posing the data to support management reporting, accounting, payroll, etc.

Planning, engineering, and documentation. ISs for planning, engineering, and documentation ensure that the network is engineered and built with adequate capacity and flexibility to reliably and economically deliver electricity from generators to the customers. In this category (technical) asset management and maintenance systems may also be included.

Meter data management and settlement. As the deregulatory framework has separated the roles of retailer and network owner, a data intensive settlement process has been introduced to match electricity consumption (e.g. meter data) towards generated electric energy and financial contracts. In order to fulfill deregulatory demands and to keep down operational costs, utilities strive to automate this process as far as possible by integration of settlement systems, data collection systems, and systems for the reporting of meter data and contractual changes.

Customer information systems (CIS). CIS in electric utilities gathers information that can be related to customers, i.e. contact and contractual information, and the personal ledger, e.g. including coming and future bills. Billing is commonly an integral part of these systems. In modern CIS, a high degree of automation, and flexible multi-channels payment, e.g. by postal giro service, autogiro, and the Internet, is sought. As distribution network operators may also be responsible for non-electrical services such as district heating, water supply, and garbage collection, these services are commonly handled by the same CIS in order to obtain operational synergy effects.

Distribution management systems (DMS). There is some confusion which functions to include in the DMS category of ISs and which to leave out (Cheong 1997; Engelken et al., 1999). Commonly DMSs consist of integrated ISs from the Real-time systems, GIS, WMS, planning, engineering, and documentation categories above.

2.3.2 ELECTRICITY RETAILERS

Electric energy is a volatile product that is sold with small margins. In contrast to distribution network operators whose overall purpose of investment in IS/IT is to enhance efficiency and cut costs, electricity retailers' ISs must provide efficient support for buying and selling electric energy on a competitive market. Although many retailers retain parts of their ISs in common with the mother companies (typically customer ISs), their requirements concerning ISs are different compared to the monopolistically operated distribution network operators. Below, the most important types of ISs operated by electricity retailers are outlined.

Customer information systems (CIS). In addition to providing efficiency, electricity retailers' CISs are moving from a product-centric approach towards a more customer-centric one, in order to satisfy prevalent customers, attract new ones, and provide additional and bundled offers in conjunction with the marketing of energy, e.g. insurances and telephony. As electricity contracts are commonly complex, CISs must be able to represent these business rules to and use the information for e.g. billing. Except for allowing payment through different channels, the CIS must support multi-channel communication with customers, e.g. by the Internet, email, and mail.

Sales and marketing. As electricity contracts may be complex, especially concerning industrial customers, systems that support sales personnel with, e.g. contract management are needed, together with ISs for preparing, executing, and following-up offers to the mass market. Especially, an enhanced level of automation is sought for in order to reduce labor costs, e.g. by scanning customer responses for automatic processing. Moreover, all sales and marketing operations must be closely coordinated with the trading operations within the retailer; an extensive on-line reporting is commonly strived for.

Trading and risk management systems. Except for attracting large volumes of customers, an important part of the electricity retailers' competitiveness lies in their ability to balance customer demand and available generation capacity with a reasonable level of financial risk. Thus, apart from ISs for trading, systems for risk management support an extensive data collection, and analysis of this data, including e.g. (estimated) sold electricity, and weather forecasts.

Chapter 3

The enterprise level of information systems

3.1 INTRODUCTION

From the perspective of this work, the enterprise level of ISs is a conceptual construct aimed to provide separation of concerns between issues related to separate ISs (i.e. systems, normally constructed by a single software vendor organization, which in themselves may consist of integrated software components), and issues that have a more far-reaching impact on enterprises' total portfolio of interconnected ISs and thereby their businesses as a whole. This work applies a user organization perspective on IS (cf. Chapter 1.2.3), i.e. the one of an organization that needs ISs in order to support its business operations according to its business goals and other business or organizational constraints. A user organization may fulfill these needs by acquiring ISs, developing these systems in-house, or by acquiring necessary services.

ISs and middleware may at the enterprise system level be described as architectural elements, e.g. components, connectors, ports, and roles (Garlan et al. 1997a). Note that in the following rendering, components are regarded from a user organization perspective, i.e. reflecting the software packages in which user organizations wish to manage their software, rather than vendors' product lines.

In this chapter, the implications of modifiability and time are discussed. Thereafter, the main characteristics of EIS are described. Especially, considerations concerning legacy IS, COTS and IS integration on the enterprise level of IS are discussed. Let it

be stressed that many of the characteristics and problems presented in this work are shared with problems related to single IS projects or the development of vendors' product lines.

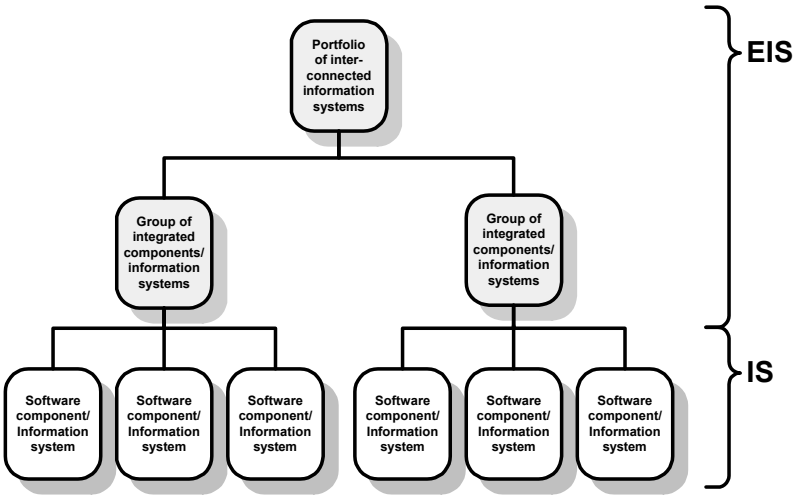

Figure 3. A schematic example of system levels. The shaded boxes denote the enterprise system level.

3.2 MODIFIABILITY AND TIME IN ENTERPRISE INFORMATION SYSTEMS

A consequence of the increased integration of ISs combined with concerns regarding legacy ISs, e.g. undocumented built-in business rules or low quality business critical legacy data, changes to EISs may rarely be commenced from a clean slate. As a result, most actions taken on EISs have the nature of *gradual evolution* of the total portfolio of interconnected ISs, thus emphasizing the importance of considerations about long-term modifiability on the enterprise level, and its implications regarding time. It is further stressed that a distinct separation must be made between system qualities, in this case modifiability, addressing the total EIS, i.e. the enterprise level of IS, and quality attributes addressing individual components.

3.2.1 MODIFIABILITY

According to Bass et al. (1998), modifiability may be the quality attribute that is most closely related to a system's architecture, as modifiability largely is a function of the locality of (potential) changes, e.g. based on the assumption that a widespread change in a system is more costly than changes just applied to a few components. In software engineering literature, *modifiability* and *maintainability* are interchangeably used to denote a system's ability to be modified. Although, some authors make a distinction between them related to whether the purpose of the modification is to correct a perceived "bug" in the system, or to change functionality and/or qualities of the system in order to satisfy new system requirements.

Definitions given of modifiability are abundant (cf. e.g. Bass et al. 2000; Bengtsson 2002; Boehm et al. 1978; McCall 1977; Oskarsson 1982). Some examples of definitions intended for vendor systems are given here, and based on these, a definition deemed sufficient for the present rendering is provided. McCall (1977) and Boehm (1978) formulated some early, partly contradicting definitions on *maintainability* of software, whereas Bass et al. (2000) provide a more recent:

"Maintainability is the effort required to locate and fix an error in an operational program" (McCall 1977).

"A software product possesses the characteristic maintainability to the extent that it facilitates updating to satisfy requirements. A maintainable software product is one which is understandable, testable, and easy to modify" (Boehm et al. 1978)

"Modifiability is the ability of a system to be changed after it has been deployed." (Bass et al. 2000)

Notably, these definitions are primarily intended for vendor systems. A definition for EIS modifiability, however, will be semantically similar to the ones given above but must be distinctly separated from modifiability of its components, i.e. commonly ISs. Based on the definitions above, the following definition, sufficient for the context of this work, is given:

"Modifiability of an EIS, is the ease with which functions and other qualities may be modified in response to changes in its requirements or its context."

3.2.2 THE PURPOSES OF INFORMATION SYSTEM EVOLUTION

Modifiability is related to other quality attributes on different system levels. Bass et al. (1998) and Oskarsson (1982) present four principal aspects of system

evolution that may be used to relate modifiability on the enterprise level of ISs to other quality attributes. Depending on the nature of the modification, our *understanding* of the components and connectors affected by the modification, and our ability to *validate* the result of the actions taken, different quality attributes may be affected (Boehm et al. 1978):

Extending or changing system capabilities. This aspect of modifiability aims to add, correct, or enhance existing functionality and/or qualities. If component replacement is selected, actions regarding this aspect of modifiability may address e.g. integrability or upgradability. Actions may affect quality attributes such as *scalability*, and *distributability*, if issues as to capacity or spatial division of the system are desired.

Deleting unwanted capabilities. Removing "leftover" parts of software systems in a controlled way, constitutes a far more difficult task than introducing new ones, as components may have undocumented dependencies with other components that could cause side effects when a component is removed. For this reason *separability* is addressed.

Adapting the system to new or changing operating environments. Changing the operational environment of ISs, e.g. concerning middleware or operating system requires *portability* of components and connectors in the IS. Also, see the implication of time described below.

Cleaning up and restructuring the system. To keep *complexity* and *heterogeneity* at a manageable level, restructuring of components and/or connectors may be justified. In EISs this may be exemplified with the introduction of a common middleware solution, to which components (if necessary) are encapsulated (wrapped) to provide compatible interfaces with the new middleware.

3.2.3 THE ASPECT OF TIME

Software engineering is a project-oriented domain. Accordingly, most software processes are aimed at the timely delivery of a predefined objective, and the process of getting there, let it be sequentially or iterative (Boehm 1978). Thus, the aspect of time is primarily dealt with within the boundaries provided by single projects. EISs on the other hand, are systems that are operable in a highly time-dependant context that in length exceed ISs projects and system life cycles.

Overall and long-term evolution of an EIS does not necessarily have a single, clearly defined deliverable or finite time horizon (Pellegrinelli 1997), although intermediate

deliverables may be well defined and under harsh timely constraints. Therefore, the modus operandi for EISM evolution resembles the one for *programs*[8] rather than the one for *projects*.

According to Bass (1998), modifiability is largely a function of locality of changes to the system. However, within the context of this work, it is advocated that time is a likewise important parameter for modifiability when expressing qualities regarding system modifiability, as temporal constraints may impose implications concerning future freedom of action, risks, and cost. Examples of temporal constraints are decided planning horizons, or when to initiate, change, halt, or terminate evolution actions, and in which order these actions should be taken.

To provide taxonomy for planning horizons, these are often expressed qualitatively to allow for time-differentiated architectural "snapshots" of the EIS. Frequently used terms are *legacy*, *baseline*, *intermediate*, and *target* architectures. Dependencies between these snapshot architectures are summarized as a description of the *migration path* that describes actions taken to incrementally *migrate* the EIS from one migration phase to the next.

Temporal dependencies are imposed for several reasons: (1) the system itself is constantly changing as components and connectors are upgraded, replaced, or removed, (2) stakeholders' perception of the system is changing as the gap between documentation, awareness, and the real-world system varies, (3) the business and organizational context of the systems is changing together with the future directions of those in any given point of time, (4) the availability of COTS components and connectors is changing in pace with vendors' product lines. Except for these first order dependencies, also higher levels of dependencies may have to be taken into consideration, e.g. the availability and quality of COTS components incorporated in ISs vendors' product lines.

As pointed out by Lehman (1998), ISs are static unless and until humans change them. Flexibility for future changes can only be provided through the introduction of tolerance, responsiveness, and replaceability to explicitly recognized future uncertainty. In addition, software in itself may be considered as a finite and incomplete, and thereby *bounded*, model of its *unbounded* operational context. Hence, there is always a gap between the bounded system and the unbounded application in its unbounded domain. This gap is bridged by assumptions that are embedded in the system in the form of design and implementation decisions. Whereas some of

[8] A group of related projects managed in a coordinated way (Project Management Institute 2000).

these assumptions are made explicitly, many are made implicitly due to contextual factors. But, as a system's operational domain is in a state of constant change, the gap between the system and the domain tends to grow (Lehman 1998). The deviation between the fulfilled functions of software compared to what functions it is required to fulfill, is by Sneed (1995) defined as *obsolescence*.

ISs are widely reported to degenerate during gradual evolution and customization over time (Bennett 1995; Brooks 1995; Parnas 1994; Sneed 1995). According to Perry and Wolf (1992), this may in part be described as an architectural problem in terms of architectural *erosion* and architectural *drift*. Architectural erosion arises when a system's architecture is violated. Conversely, architectural drift occurs as a result of "insensitivity" to the architecture, i.e. the rules implied by the architecture are not clear to those who work with it. Architectural drift obscures the architecture as it results in lack of coherence and clarity of form, and may lead to lesser modifiability (Perry and Wolf 1992) and an increasing risk of architectural erosion (Parnas 1994; Perry and Wolf 1992). Case studies on architectural erosion are presented by e.g. Bosch and van Gurp (2002), and by Jaktman, Leaney, and Liu (1999). A consequence of architectural drift and erosion is *maintenance degradation* that may be considered as a summarizing factor for lack of modifiability due to raising costs for evolution, longer modification cycles, growing impact domains, and increasing level of side effects caused by maintenance actions (Sneed 1995).

A factor that contributes to obsolescence of EISs is that planning and implementation cycles for business operations and ISs may diverge significantly. Planning and implementation of ISs to ensure freedom of action beyond an enterprise's planning horizon for its business operations is difficult and costly. In addition to the often rather lengthy implementation of EISs, the time needed for establishing appropriate objectives and creating stakeholder awareness to such a level that implementation and/or acquisition undertakings may commence must be added.

In literature on strategic planning, obsolescence in EISs is incorporated in the concept of *fit*, used as a measure in the process of coping with turbulent and multi-dimensional environments termed *strategic alignment* (Henderson and Venaktraman 1996; Knoll and Jarvenpaa 1994; Luftman (ed.) 1996). The purpose of alignment is to ensure *fit* between the organizational and technical dimensions considered. *Fit* is broadly defined as, the degree to which the needs, demands, goals, objectives, and/or structure of one dimension are consistent with needs, demands, goals, objectives, and/or structure of another dimension (Nadler and Tushman 1980 in Knoll and Jarvenpaa 1994). Most work on achieving alignment

between organizations and ISs take a strategic management perspective as a starting point. However, as reported by Knoll and Jarvenpaa (1994), relatively little attention is given to the conceptualization of fit from a technology perspective.

3.3 CHARACTERISTICS OF ENTERPRISE INFORMATION SYSTEMS

To describe EISs from the perspective of this work, some notable characteristics of their elements are highlighted below. In order to maintain a consistent terminology in the following rendering, *components* refer to software chunks of *all* sizes and shapes deemed relevant to manage on the enterprise level of user organizations, i.e. ISs are below considered as components in the overall EIS.

Components may be fairly large grained. EISs comprise both fine-grained components (down to source code level) and extremely coarse-grained components (complete single vendor systems). From a user organization perspective, EIS components should reflect the chunks of software that the organization wishes to make explicit in order to maintain a high level of freedom regarding functionality and quality attributes focused upon, e.g. *maintainability*, *upgradability*, or *flexibility*. Thus, components in EISs are often fairly coarse-grained, as they should reflect top management goals, present (legacy) software components, and available COTS software components rather than the IS suppliers' product line architectures.

EISs are normally COTS intensive. Traditionally, organizations developed their business systems as custom-made applications, using only operating systems and some rudimentary third party products such as compilers and database management systems. New technical developments such as the introduction of more extensive *component technology* and *platforms for distributed computing*, brought with it an increased focus on reuse-at-large in terms of COTS software (Meyers and Oberndorf 2001). From an EISM perspective, not only fine-grained software components but also coarse-grained components in terms of complete ISs such as *customer information systems*, *geographical information systems*, or even *enterprise resource planning systems*, may be considered as COTS components.

The supply of COTS components is limited. The number of alternative coarse-grained components is often restricted. Sometimes preferred components do not exist, driving enterprises to either develop these components from scratch, or combining other COTS components to fulfill the requirements of the desired component.

The legacy EIS constitutes the starting point of the system development effort. Enterprises have to take their legacy of software systems under delicate consideration. According to Brodie and Stonebraker (1995), "A legacy IS is any IS that significantly resists modification and evolution." Bennett (1995) pragmatically defines legacy systems as "large software systems that we don't know how to cope with but are vital for our organization."

Traditionally, legacy ISs comprised mainly of non-decomposable software, e.g. custom-developed mainframe-based ISs. However, based on the observation reported by Sneed (1995) that the maintenance implications for many new software technologies are not well understood, problems related to legacy IS are likely to endure. Hence, current state-of-the-art software technology, including ISs based on distributed components from multiple software vendors, integrated by complex middleware such as message brokers may be expected to constitute a far more prominent legacy problem compared to previous legacy systems (Bennett 2000).

A common problem related to legacy software is that business rules may be embedded into the software components and not documented elsewhere (cf. e.g. Bennett 1995). Available interfaces to the legacy software components constitute a key topic and may require considerable effort to elicit. Moreover, not only may the present ISs constitute a significant asset that is costly and time-consuming to replace, but also issues as the organization's ability to cope with system changes without severe disturbances to the business operations, and the often gigantic efforts of cleaning up and converting the enterprise databases, must be addressed.

Components may not be modifiable. Since EISs, to a large extent, consist of COTS software and indecomposable legacy components (Brodie and Stonebraker 1995), changes to the system cannot be made directly on the component, since the source code is normally not available. Therefore, changes to the system must be handled more delicately, either by influencing the software vendor to adapt its COTS product, or by separate changes from the component itself, e.g. by wrapping the component in order to change its external behavior.

Components are normally heterogeneous. On the single system level, components are oftentimes fairly uniform with regard to internal structure (e.g. hardware platform, operating system, programming language, and database management systems) and external interfaces mechanisms (e.g. type of application programmers interfaces and import/export file formats). EISs components cannot be assumed to be constructed in a uniform fashion, as they are commonly

composed to a large extent by the combination of COTS and legacy software components from a wide range of vendors, epochs, and intended purposes.

Connectors are normally heterogeneous. Similar to components, connectors for vendor systems is normally relatively uniform in order to facilitate the commonly rather complex interaction that takes place internally in the system between components. In contrast, connectors in EISs become by nature diverse since their main purpose is to glue together heterogeneous components that commonly are not designed for flexible integration. Moreover, middleware in EISs not only interfaces with other components within the same enterprise, they also provide interfaces to other organizations' ISs bringing even more heterogeneity into the organization's total battery of middleware.

The enterprise software system may contain both data and functional redundancy. As most enterprises aim to maximize the amount of COTS in their EISs, often components available for acquisition not totally correspond to software requirements. Therefore, in order to grasp the bulk of the software requirements, different COTS components may be combined using custom-made software components and connectors such as *wrappers* (Linchicum 2000) or *gateways* (Brodie and Stonebraker 1995). As a result, redundancy in terms of both data and functionality may occur as components' internal content cannot be fully controlled. This, in turn, may lead to considerably concerns as data and functionality also can be *virtually redundant* (i.e. similar but not identical) for instance causing inconsistence in enterprise databases (Andersson et al. 1998; Kohtala and Vaattovaara 1998). See also Parts A, C, and D.

3.4 COTS IN ENTERPRISE INFORMATION SYSTEMS

As pointed out above, EISs are to an increasingly extent constructed through integration of coarse-grained COTS components. However, although generally used, COTS is far from being an unambiguous term. In this work, COTS refers to all software available as commercial products, either by purchase, leasing, or, most commonly, licensing. In literature, terms like *packaged*, *standard* or *shrink-wrapped* software have similar meaning as COTS (Andersson and Nilsson 1996; Carmel 1997; Carmel and Sawyer 1998; Grudin 1991; Klepper and Hartog 1992; Meyers and Oberndorf 2001; Sawyer 2000).

The growing popularity of COTS is based on the leading assumption that combining existing pieces of software into new ISs will enhance productivity, (Meyers and Oberndorf 2001), permit shorter time-to-market, and reduce

implementation cost (Linchicum 2000). From one perspective, building EISs from COTS components, may be seen as the logical continuation of software reuse, extended from a single organization to a larger market, in order to increase the customer base of the reused software, and thereby provide means to enhance revenues and software quality. However, as stressed by Wallnau (2002), the desire to reuse high-quality COTS components is partly obstructed by COTS vendors, who in order to attract customers continuously add new features, that, in turn, affect the quality of COTS components negatively.

As pointed out above, COTS come in many flavors and forms. A fundamental difference can be seen between COTS acquired as industrial products, and COTS acquired as retail products. COTS acquired as industrial products may be categorized using a sliding scale with *COTS-solution* systems on one end of the scale, and *COTS-aggregate* systems on the other (Brownsword and Place 2000). In addition, COTS bought as retail products, as suggested by Brooks (1995), may be classified as *shrink-wrapped* COTS products.

Shrink-wrapped COTS products. Software intended for a mass-market represents a profound change for computer industry, shifting focus from development cost to product quality and costs for integrating, and supporting the product (Brooks 1995). In fact, consumer markets such as entertainment and telecommunication compete with traditional industry as technology drivers. In EISs, shrink-wrapped software may be used "as is" for general purposes, e.g. word processing, or as a basis for fulfilling tailored purposes, e.g. by providing macros or templates for spreadsheet or database applications. Also, a dominating share of e.g. *operating systems*, *middleware*, *database management systems*, *communications software* falls under this category of COTS.

COTS-solution system. These systems, also commonly termed packaged systems or standard systems, are mainly developed by a single vendor. They may be considered as components in EISs, which can be adapted to provide desired functionality and qualities for a specific user organization. The degree of adaptation may differ significantly and include parameterization, data conversion, and even custom-development. COTS-solution systems may, or may not, be upgradeable without further adaptations in the ISs or its systemic context. ERP and SCADA systems are illustrative examples of this type of COTS products.

COTS-aggregate systems are made by integrating mainly COTS components from several vendors, in order to fulfill desired functionality and qualities for a specific purpose. These systems may, in turn, be marketed as a product in a

vendor's product line, or be adapted directly for a specific user organization. Enterprise ISs, as addressed in this work, are COTS aggregate systems unless custom-developed. Especially for domain-specific purposes, the number of alternative coarse-grained components is often restricted, driving enterprises to either develop those components from scratch, and/or combining COTS components to fulfill the requirements. Commonly, the components, or the combination of them, are unprecedented, and thus require substantial resources to integrate and to maintain (Brownsword and Place 2000).

There are several implications for EISs due to an extensive use of COTS. For the user organization, design choices regarding the combination of large-scale COTS components are complex (Heineman and Councill 2001). Requirements must be delicately prioritized and matched toward functionality and qualities of available software packages. The complexity of the problems increases immensely with the number of integrated components. As discussed in Chapter 3.2, taking the aspects of time and long-term modifiability into consideration makes the design problem even more complex. A lot of attention is given to the selection of COTS components in literature (Braun 1999; Kointo 1996; Sawyer 2000), and to the reuse of large-grained COTS components (Braun 1999; Garlan et al. 1994a; Heineman and Councill 2001; Walnau et al. 2002).

In addition there is an organizational aspect regarding COTS as it inflicts customer-supplier relationships. For instance, user organizations' *influence* on, and *dependence* of software vendors may change, when software function and quality of software cannot be controlled directly. Instead, the user-organization must either adapt the COTS component (if possible), with the risk of inhibating smooth upgrades to coming releases, or attempt to influence the vendor to make the desired changes in its product line. The success of the latter approach is naturally dependant on the user organization's status as a customer.

In addition, as software vendors try to use third party COTS components as much as possible in their product lines, acquired ISs may include a significant number of third-party COTS components. These third-party components may have a significant impact on EIS in the long run. Thus, from a user-organization perspective, third party COTS may be of interest in a COTS evaluation and selection process.

3.5 ENTERPRISE APPLICATION INTEGRATION

During the last two decades, integration of heterogeneous components on the enterprise level of ISs has evolved in pace with the growing popularity of distributed systems. However, components in EISs are not usually designed for collaboration, and thus connectors are heterogeneous. This is mainly due to lack of openness, and implications of time, i.e. different life cycles in components and connectors. Although openness is a "buzz word," case studies Alpha, Beta, and Gamma (cf. Chapter 7) unanimously indicate that vendors are reluctant to provide interfaces to other ISs that are not proprietary. Supported (proprietary) interfaces are commonly provided to components with supplementing functionality from either the vendor's own product line, or their partners' (Rahkonen 1996b). Even though more advanced generations of middleware and computing platforms, such as message brokers, have appeared, their life cycles have proved, so far, to be shorter than a significant share of its components, and the time required for introducing them throughout an organization's EIS. Consequently, the present alternative for a user organization that wishes to achieve homogeneous connectors is to reintegrate components during their life cycles.

Thus, compared to single ISs, connectors in EIS are commonly more heterogeneous. Two principles for EIS component integration are integration by *coupling* and integration by *cohesion* (Linchicum 2000; Pressman 2000). Coupling refers to static binding of functions and data; in essence coupling creates one component out of many. Although integration by coupling provides many advantages, it may be less beneficial from a modifiability perspective as changes in one component may require changes in all coupled components as well. Cohesion, on the other hand, is based on the concept of integrating independent components in such as way that changes in one component should not impose any changes in any collaborating component. Cohesive integration enhances modifiability of an EIS as components may be changed, added, or removed without requiring changes to the other components. However, this also increases complexity, as more advanced middleware solutions must be employed. Integration by coupling often offers fastest reward as it may require less implementation effort (in the short run).

Most middleware, migration, and EAI literature agrees on the three-tier model that divides the different integration approaches concerning techniques and technology on the enterprise level into: *user interface level*, *application logic level*, and *data level*integration. Brodie and Stonebraker (1995) classify (legacy) ISs as *decomposable*, *semi-decomposable*, and *non-decomposable*, depending on whether or not collaborating components may be addressed on the data level (decomposable and partly semi-

decomposable) or the user interface level (non-decomposable and partly semi-decomposable).

User interface level. To integrate non-decomposable ISs, typically mainframe-based with obscure and/or undocumented interfaces towards application logic and data, techniques that utilize the user interface as an integration point to component may be employed (Brodie and Stonebraker, 1995). This technique is known as *screen scraping* and basically captures and feeds character streams to and from text-based user interfaces, to create access to data and logic of the interfaced component. This approach is rather primitive and requires understanding of the underlying data storage schema, the application logic, and how the information is presented to the user interface. However, it provides means for black-box modification of components. One enabling technology for user interface level integration is terminal emulators, e.g. 3270 or 5250 terminal emulation software. If graphical user interfaces, using e.g. the Windows win32 environment, are "screen scraped," technologies such as OLE[9] (Chappell 1996) automation may be employed. Screen scraped user interfaces may export API[10]s, or encapsulate methods and data from the user interface into e.g. CORBA[11] (Pritchard 1999), COM[12] (Chappell 1996; Pritchard 1999), C++, or Java objects (Morgenthal 2001). Also, user interfaces published through Web browsers may be used as integration points.

Application logic level. Integration on the application logic level may be carried out in several ways. If no point of integration is at hand, one has to be developed by changing the interior of the components. Components may publish predefined points of integration as APIs, allowing collaborating components to interoperate by procedure calls, or RPCs[13]. COTS applications, e.g. ERP systems or operating systems, commonly provide libraries of APIs. As a result of the increasing focus on application logic level access of functionality and data, there is a growing market for software integration products and standards. This market may be exemplified by enabling technologies and standards like CORBA, COM, and Enterprise JavaBeans (Monson-Haefel 2000) incorporating more and more services. Typical integration solutions are exemplified by remote method invocation, message queuing systems, transaction management systems, message brokers, and adapters, (Linthicum 2000; Ruh, et al. 2001; Thomas 1998).

---

[9] OLE, Object Linking and Embedding.
[10] API, Application Program Interface.
[11] CORBA, Common Object Request Broker Architecture.
[12] COM, Component Object Model.
[13] RPC, Remote Procedure Calls.

Data level. To integrate databases directly, data level integration may be considered. A main motivation for data level integration is to ensure consistency and accessibility in enterprise databases, e.g. by providing a single-point-of-data-entry, eliminating manual updating of the same information in several databases. As an increasing number of ISs decouple their database from the application logic, it is technically rather straightforward to access databases, but successful data level integration requires understanding of database technology, and, in addition, of the information contained in the databases. As database technology is comparably homogeneous, most commercial database management systems provide interfaces, e.g. ODBC[14] or JDBC[15], depending on applicable software technology, through which data may be accessed, commonly by SQL[16] statements. Data formats, that except for the information, also incorporates a structured storage of meta-data, include XML[17] (Dick 2000), EDI[18], and in the electricity industry, EDIel[19]. However, it is stressed that exchange of data relying on flat file transfer with information represented with in-house data formats is a common data level integration solution.

The main reason for the choice of integration level and integration technology is basically the business objectives of the integration effort. In addition, technical considerations that may influence this choice include the availability of COTS connectors, and the quality and documentation-level of the interfaces provided by components.

---

[14] ODBC, Open Database Connectivity.
[15] JDBC, Java Database Connectivity.
[16] SQL, Standardized Query Language.
[17] XML, Extensible Markup Language.
[18] EDI, Electronic Data Interchange.
[19] A standardized message format for exchange of electricity market related information, e.g. meter data, and updated customer information. The message format is based on United nations's EDIFACT (Electronic Data Interchange for Administration, Commerce and Transport) standard, exchanged by the X.400 protocol for electronic mail.

Chapter 4

Software architecture as a tool for decision support

4.1 ENTERPRISE INFORMATION SYSTEMS MANAGEMENT

To further delimit the rather broad issue of EISM, this work focuses on the provision of decision support for evolution of EISs. No separation is made between decision support for planning and implementation of system evolution activities, as planning and implementation of EISs are closely coupled, especially in small and medium-sized organizations. Further, managing evolution of EISs implies dealing with the future, i.e. the relevance of decisions[20] regarding system evolution will be based on our success to forecast the future, based on contemporary information. However, the state-of-the-practice in EISM, as perceived on the basis of literature and accomplished field studies largely relies on ad-hoc heuristics.

To grasp the problem domain concerning EIS evolution in order to provide a rationale for decisions, both the locality of change activities and the time dimension of these activities must be addressed. In addition, the success of evolution efforts will be highly dependant on our ability to capture prevalent heuristics (possibly from other organizations, or even totally disparate domains), and to effectively reuse the experiences encapsulated therein in order to avoid previously faux pas. In this work, the general strategy employed is conceptualization of the problem

[20] Cf. the definition of decisions provided in Chapter 1.

domain with the purpose of making it comprehensible, analyzable, and to dispose of biases, e.g. narrowness caused by our perception of prevalent technology (that may be inapplicable in the time-horizon addressed).

More specifically, the approach for conceptualization and analysis comprises the employment of theory from *software architecture*. An enabling motivation for the selection of software architecture as the reference discipline for this work, is its recent achievements in expressing and analyzing complex and coarse-grained software packages on basis of quality attributes (cf. e.g. Bass et al. 1998; Garlan and Shaw 1996; Heineman and Council 2001). Here, *quality attributes* have an important role as a bridge between technical and organizational considerations.

Based on its abilities to combine modifiability analysis with the mitigation of systemic and organizational qualities, risks, and opportunities, *scenario-based architectural* analysis has been investigated for its capabilities to operationalize problem domains into condensed design alternatives, or lines of action. Especially, its capabilities for providing strategic decision support in small and medium-sized enterprises have been further scrutinized, as this area is considered to be under-developed and under-researched (Levy and Powell 2000). To codify the problem domain and relevant heuristics in order to make them comprehensible, generalizible, and analyzable, architectural concepts such as *quality attributes*, *architectural taxonomy*, *views*, and *scenarios* have been surveyed.

To consider architectural description as a common conceptual model in EISM, is far from being a new idea (cf. e.g. Earl 1989; Eason 1988; Luftman (ed.) 1996; Magoulas and Pessi 1998; Walsham 1993). For instance, Zachman (1987) early advocated "the importance of using some logical construct (or architecture) for defining and controlling the interfaces and the integration of all of the components of the [enterprise information] system." In addition, several frameworks, or meta-architectures, for EISs have been developed (Armour 1999; DISA 1997; The Open Group 1999; Zachman 1987). Several sources also stress the architectural description of the EIS as a dominant part of the organization's IS/IT strategy (Earl 1989; Henderson and Venkatraman 1996; Zachman 1987), and the close relationship between technical and organizational qualities (Luftman (ed.) 1996).

The remainder of this chapter is organized as follows: First, a background to software architecture and its relation to quality attributes is given. Thereafter, software architecture analysis description are introduced together with references to the more elaborate renderings of the applications of these in EISM provided in Parts A to D.

4.2 ENTERPRISE INFORMATION SYSTEM ARCHITECTURE

System architecting stems from the need of bringing together codified *heuristics* and *conceptual models* related to complex artifacts for the purpose of providing means for high-level design decisions. The concept of architecture can be identified in several domains that deal with designing complex systems or artifacts, e.g. buildings or complex industrial systems (Baragry and Reed 1998; Meier and Rechtin 2000; Perry and Wolf 1992; Zachman 1987).

As pointed out by Medvidovic and Taylor (1998), software architecture offers an adequate platform for supporting coarse-grained software evolution. Applying software architecture on the enterprise level of ISs can therefore be seen as a logical prolongation of the trend in recent year's research in software architecture, that follows the general development in software systems development towards integration of rather large granule components (Clements and Northrop 1996). As observed out by Garlan and Shaw (1996), this trend goes as far back as the 1950s when commonly used sequences of machine language were substituted by a single symbol, thus providing some (modest) degree of automation of programming. During the late 1950s and 1960s, high-level languages, such as Fortran, emerged to provide higher levels of abstraction and suggestions on even higher levels of software abstraction emerged in terms of typing of data and modularizations of code (Garlan and Shaw 1996).

An initiating force on the latter subject was Dijkstra, who in 1968 stressed that (as opposed to simply programming in order to produce a correct result) it pays to be concerned with how software is partitioned and structured, by pointing out the elegant conceptual integrity exhibited by such an organization and its corresponding expected gains concerning development and maintenance ease (Dijkstra 1968). Parnas further pressed this line of software modularizations in his seminal paper "On the Criteria To Be Used in Decomposing Systems into Modules" (Parnas 1972), in which he advocates the advantages of modularization by the concept of information hiding, i.e. that every module is characterized by its knowledge of design decisions which it hides from all others, only revealing as little as possible of its inner working by interfaces and definitions. Thus, Parnas demonstrated by *information hiding*, the effectiveness of other relationships between components than the ones provided by considering software modules as sub-program in an execution thread. These ideas were later extended by the concepts of software structure (Parnas 1974), program families (Parnas 1976), module guides

(Clements, Parnas and Weiss 1985), and the proposal of an architectural level of software design (Shaw 1989).

Based on similarities and dissimilarities with other types of architecture, such as systems and building architecture, Perry and Wolf (1992) have heavily influenced the present conception of software architecture as a discipline, by proposing the following threefold definition:

$$\textit{Software Architecture} = \{\textit{Elements, Form, Rationale}\}$$

Further explaining the three items in the triplet, (1) *architectural elements* may be divided into *processing elements*, *data elements*, and *connecting elements*, (2) *form* represents the *properties* (e.g. *functional* aspects and *quality attributes*) and the *relationships* between elements that must be taken into consideration when selecting between design alternatives, i.e. when making *design decisions*, and (3) the *rationale* for design decisions that captures the underpinning motivations for those decisions. Other authors who have elaborated on analogies between software architecture and other types of architecture are Baragry and Reed (1998) and Zachman (1987).

Perry and Wolf further stress three issues that are important from the perspective of this work: Firstly by emphasizing that architecture should place an equal focus on components and their interconnection, secondly by stressing the need of support for design decisions, and thirdly by recognizing the rationale as a part of the architecture. The third issue, however, underlines the difference between architecture as something inherent in systems whether explicitly designed or not, and architecture as something that aims to grasp all relevant motivations for previous design decisions (cf. the definitions given on software architecture in Chapter 1.2.4). On this basis of this, some authors criticize Perry and Wolf's third item because, although relevant, it is not a part of a software system's architecture as it cannot be entirely coded into software and therefore not recovered by the study of the software only (Bass et al. 1998).

This somewhat narrow view of software architecture is not fully sufficient in the context of this work, as dealing with long-term system evolution implies working with yet to exist artifacts, and thus the rationale might constitute the dominant part of the architecture. In this sense, this thesis adheres to Maier's rule-of-thumb regarding system architecture, based on the role of architects, namely that a system architecture is the work result produced by the architect to help his or her client to make decisions regarding the system (Maier and Rechtin 2000):

"An architecture is the set of information that defines a system's value, cost, and risk sufficiently for the purposes of the system's sponsor."

However, it is pointed out that there is no consensus on the definition, or even the choice, of the term EISA (DISA 1996; The Open Group 1999; Zachman 1987). Cf., the definition given on EISA in Chapter 1.

4.2.1 THE PURPOSE OF SOFTWARE ARCHITECTURE

Software architecture has several purposes also valid for EISA. These are summarized as follows by Clements and Northrop (Bass et al. 1998; Clements and Northrop 1996):

Architecture is the vehicle for stakeholder communication. Software architecture represents a common intuitive high-level abstraction of a system that stakeholders, e.g. end-users, business managers, technical staff, and software suppliers may use as a basis for communication, thus increasing awareness and competencies.

Architecture embodies the earliest design decision about a system. Software architecture represents the manifestation of earlier design decisions, i.e. the fundamental rationale of why a system is constructed in a certain way. This brings indispensable information when modifications of the system is considered, e.g. why some design alternatives are abandoned. The formulation of an architectural description is also the earliest point when the system may be analyzed.

Architecture provides a transferable abstraction of a system. Software architecture constitutes a comparably intellectually comprehensible model of the system at stake. If abstracted and properly packaged, this model may be transferred and applied on other systems with similar requirements, thus promoting reuse-at-large.

4.2.2 QUALITY ATTRIBUTES

Software architecture and quality attributes are closely related (Kazman et al. 1994). Thus, much of the research carried out in the software architecture community has focused on adapting definitions of quality attributes for the architectural level of software abstraction, and finding methods for architecture-level attribute-specific analysis and analysis aimed at making explicit trade-offs between them. This work is largely based on the massive effort that, over time, has been directed towards software quality (cf. e.g. Boehm 1978, McCall 1977).

Quality attributes, in general, can be divided into two major categories: those that are directly *observable during execution*, e.g. performance, security, and usability, and those that can be *observed only indirectly*, e.g. modifiability, security, and availability (Pressman 1997). Concerning software architecture, two more categories may be added, *business qualities* and *qualities of the architecture* (Bass et al. 1998).

An outstanding motivation to architectural description and analysis on the enterprise level of ISs is the close relationship between technical and organizational quality attributes, in Bass (1998) referred to as business qualities. The strategic impact of quality attributes is acknowledged in management literature. In fact, one of the most written causes of EIS investment failure is that too much attention is placed on technology itself, rather than on its characteristics, or quality attributes[21], and its links with business qualities such as risk, opportunities, awareness, control, and competence (Hendersen and Venkatraman 1996).

In addition, the architecture itself has qualities, the most pertinent perhaps being *conceptual integrity*, i.e. the underlying theme that unifies all levels and perspectives of the architecture, *correctness*, *completeness*, i.e. that the complete system is addressed in sufficient detail, and *buildability*, i.e. that the resulting system is achievable with conceivable limits concerning organizational, financial and other types of finite resources (Bass et al. 1998).

It is stressed that even though quality attributes commonly are refereed to as non-functional attributes, they are intrinsically related to system functionality. Indeed – functionality may be defined as a quality attribute, i.e. the ability of a system to do the work for which it was intended. Also, as components at the enterprise level of ISs, i.e. IS, in their selves may constitute large, complex, and component-based systems, it is of outmost importance to make explicit which level of the EISs that a certain quality attribute addresses, as the attribute may have significantly different meaning and implications on various system levels.

4.3 ARCHITECTURAL ANALYSIS

Software architecture embodies high-level design decisions regarding system structure, i.e. how the total system is decomposed into components, the relationship between these components, and how the system relates to its context. The structure of the system influences, and is influenced by, imposed and desired quality

[21] In information systems literature also referred to as systemic competencies (Hendersen and Venkatraman 1996).

attributes. Architectural analysis aims to guide these high-level design decisions. Using architectural analysis to provide support for decisions addressing *evolution* of EISs, i.e. modifiability analysis, may be employed in order to either predict future effort to perform system evolution, or to construct and thereafter select among design alternatives that to various degrees support different aspects of modifiability. Plausible approaches to attribute-based analysis, presented in (Bosch and Molin 1999), are *mathematical modeling*, *simulation*, *scenario-based analysis*, and *informal assessments*.

One problem, however, is that modifiability analysis may not be performed independently from other quality attributes, as modifiability addresses future flexibility of different quality attributes, and in addition, has dependencies to more imminent system qualities. Scenario-based architectural analysis has the potential to address several quality attributes. Several methods for scenario-based architectural analysis have been presented during recent years (Bengtsson 2002; Bosch and Molin 1999; Kazman et al. 1998; Lassing et al. 1999).

Moreover, there is an organizational and a social aspect of using an architectural analysis process for decision support in EISM. Therefore, some additional presumptions of a "good" approach for decision support concerning evolution of EISs are provided. The analysis approach must support and acknowledge changing organizational and technical rationales, as well as temporal implications and dependencies between different considered modifications. Further, the approach must be able to generate results despite incomplete and implicit information suitable, at least, for further analysis. Finally, the approach must assume bounded rationality of stakeholders (Simon 1997), e.g. by dealing with growing stakeholder awareness, the elicitation of tacit knowledge, and the promotion of stakeholder communication. This is in line with what Thomason (1998) refers to as *sympathetic planning*, i.e. planning based on collaboration that depends on identifying and adopting preferences and other adequate information about collaborators in order to make decisions that further individual of mutual objectives.

To address these prerequisites, scenario-based architectural trade-off analysis (cf. e.g. Kazman et al. 1998) has been chosen for further scrutiny in this work. The analysis approach has successfully been employed for architectural evaluation of systems both prior to implementation and system during evolution. In addition the analysis approach has been used for evaluation during software acquisition (Bergey et al. 1999).

The fundamental characteristic of scenario-based architectural analysis is the concretization of quality attributes in the form of scenarios, whereas the main intention with such an analysis process is to make explicit and to mitigate risks. As pointed out by Kazman (1999), the method's main benefits are: (1) that it may be carried out early and relatively inexpensively as it assesses architectural artifacts, and (2), it may be performed with a relatively sparse factual basis as it qualitatively focuses on the identification and correlation of trends. As the method is a spiral model (Boehm 1988), its risk-driven approach contributes to gradually enhance stakeholder awareness, competence, and unanimity.

The effectiveness of the qualitative approach to architectural analysis was confirmed during field study Delta (cf. Parts B and C). During this study, it was assumed that the EISs architect considered different future integration scenarios, posing the question: "Which of the present design alternatives will result in the highest modifiability with respect to the defined future scenarios?" It is instructive to note that this question implies no more than an ordinal scale, i.e. it is not necessary to determine an absolute measure of integrability but only a comparative assessment. Moreover, it was found efficient to reflect temporal implications and dependencies by the selection of scenarios (cf. Section 5.2 in Part B).

Below, a short introduction to scenarios is provided. Thereafter, the typical analysis process is briefly outlined together with a discussion on how it may be modified to allow architectural analysis of EISAs.

4.3.1 SCENARIOS

Scenarios, similar to *use-cases* (Jacobson et al. 1992) or *threads* (Maier and Rechtin 2000), describe sequences of systems operations, which represent an important behavior for an actor. Within the context of this work, an actor may be anyone, or anything exercising influence on system evolution, e.g. stakeholders, other organizations, or collaborating ISs. As pointed out by Maier and Rechtin (2000), scenarios commonly describe a single sequence of actions and do not contain branches. Scenarios are widely used throughout all stages of the development process, e.g. as a basis for requirements, architecture, detailed design, and testing (Jacobson 1999). Especially, scenarios elicited during early phases of system development are widely used as a basis for system validation (Jacobson 1992; Sutcliffe et al. 1998). Although, scenarios may be described using notations, e.g. use-case diagrams (Jacobson 1999), they may also be expressively formulated in natural language (Ecklund et al. 1996; Kazman et al. 1999).

More recently, scenarios have been introduced in the software architecture stage of the development process (Bass et al. 1998; Kazman et al. 1994; Kazman et al. 1998). An enabling reason for using scenarios in architectural analysis is that they capture various important or problematic situations in ISs and make their implications on system qualities explicit. In scenario-based architectural analysis, a scenario may be either a *use-case* (Jacobson 1992) – a scenario exemplifying usage of the system, or a *change-case* (Ecklund et al. 1996) – a scenario exemplifying a modification of the system. As the scope of this work is system evolution, the scenarios employed should be regarded as change-cases. In some literature, change cases are referred to as *chance scenarios* (Lassing et al. 2002), or *indirect scenarios* (Bass et al. 1998, Kazman et al. 1999).

In field study Delta, extracted scenarios mainly concerned changing or adding components and connectors in the architecture, as a result of changing requirements or context for the EIS. Also, opportunities to reduce the number of redundant components, and thereby the integration effort was reflected by the scenarios, e.g. components with similar functional content used on different locations and with different adjacent components. One important experience gained, was that most scenarios initially proposed by the stakeholders covered only early phases of the architecture's life cycle. Therefore, in order to avoid that later phases in the architectural life cycle were overlooked, extra attention had to be spent on eliciting scenarios that covered expected future changes in the architecture. This was the primary motivation for the proposal of the *life cycle* viewpoint described in Chapter 4.4.1. Elicitation and prioritization of scenarios proved to be a time-consuming undertaking, as also reported by e.g. Sutcliffe et al. (1998). However, this was partly balanced due to the elicitation process's many positive effects as it helped stakeholders to obtain a more unanimous view of the relevant goals, problems, and their implications, thus promoting stakeholder awareness and communication.

4.3.2 THE ANALYSIS PROCESS

This section describes the steps in the process for scenario-based architectural analysis similar to Bass et al. (1998), Kazman et al. (1994), and Kazman et al. (1998), adapted for the enterprise level of EIS during field study Delta. The method presentation focuses on analysis of modifiability. Let it be stressed here that the analysis method presented below was not completely employed during field study Delta, rather, the proposed adaptation of the method to the enterprise level is one of the results from the study. Although the proposed process model consists of five

sequential steps, the proposed analysis process is, like other methods for architectural analysis, typically a spiral model (Boehm 1988):

Step 1. Definition of scope. Defining the scope for the undertaking includes elicitation of objectives, requirements, constraints, and documentation of existing legacy systems and available COTS components. Stakeholders from different parts of the organization are enlisted in a cross-functional team to ensure effective implementation. During this step, the overall objectives and requirements for the system change are developed. Important activities include the establishment of the boundaries and the characteristics for the modification in order to clarify and limit the problem space that must be coped with. For temporal constraints consideration may determine if the aim of the modification is strategic or tactical, evolutionary or revolutionary, short-termed or long-termed, etc. In addition, a description of the legacy systems is established together with architectural descriptions of relevant available and foreseeable future COTS components and connectors. Also, software vendors' capability to deliver functionality that cannot be acquired off-the-shelf is assessed.

Step 2. Scenario construction. This step aims at visualizing dependencies between business strategy and processes, and the components in the EIS. Also, the scenarios are used to make sure that the planning results are understood early on by making the deliverables visible and tangible. Although scenario construction should start from existing business visions or business processes, the scenarios are also made possible, even suggested, by emerging technologies and legacy systems. Typically, scenarios related to modifiability are extracted as change cases (Ecklund et al. 1996) through changing or adding components in the architecture. Thus, a fundamental concept of scenario-based analysis methods is to operationalize quality attributes as scenarios.

Step 3. Architectural representation. In this step the objective is to propose feasible architectural design alternatives for the EIS. During field study Delta, an approach of expressing different views (cf. Chapter 4.4.1) of design alternatives with architectural integration styles (cf. Part D) was developed. The design alternatives considered are presented in Part B. The basis for the design alternatives was relevant quality attributes, functional requirements, the legacy system, and the relevant COTS components on the market.

Step 4. Architectural analysis. The design alternatives are then reviewed on the basis of the scenarios. In some cases, specific architectural elements are identified as especially important and trade-offs between different quality attributes may have to

be exposed. Here, a key issue is to prioritize available resources for the analysis on issues deemed as relevant, since the analysis effort grows rapidly with the increasing number of design alternatives, scenarios, and components and connectors taken into consideration.

Step 5. Architectural modification. Based on the results from the analysis, the architecture is modified. The resulting architecture is then used as input for further iterations of the process. When it is deemed that no more improvements are feasible, the iterations end. Reasons for adding iteration cycles might be that the architectural description is considered insufficiently detailed, or that the most favorable design alternative constitutes a dead-end, in either a business, organizational, or technical perspective, so that a totally new alternative must be tested and analyzed.

See Part B for a more elaborate discussion on how scenario-based architectural analysis may be employed on the enterprise level of ISs. In Part C, the process for scenario-based architectural analysis is used as a part of a proposed framework for EISM for primarily small and medium-sized organizations, developed during field studies Gamma and Delta (cf. Chapter 7)

4.4 INVESTIGATED ARCHITECTURAL CONCEPTS

When using scenario-based analysis for decision support, the architectural analysis in itself largely is based on sense making and reuse of heuristics. To support the architectural analysis, several architectural concepts have been investigated for their capabilities to provide means for decision support concerning evolution of EISs. In this section, these architectural concepts are introduced together with references to the further presentation of the concepts in Parts A to D.

4.4.1 VIEWS

In parallel to building architecture, an important feature of software architecture is the use of multiple viewpoints[22], or views[23]. An architectural view describes a system with respect of some set of aspects, qualities, or concerns, allowing the description to use the most appropriate notation or description technique for the perspective at stake. Perry and Wolf (1992) suggest the use of views in software architecture to abstract the architectural description from details deemed irrelevant

[22] A *viewpoint* is a template, pattern, or a specification for constructing a view (IEEE 1471-2000 2000).
[23] A *view* is a representation of a system from the perspective of related concerns or issues (IEEE 1471-2000 2000).

on the architectural level of design, thus adapting the architectural description for "various uses and users."

According to Maier and Rechtin (2000), a good set of views should be *complete*, *consistent*, and mostly *independent*. Strictly, *completeness* means that the views together completely define all externally visible properties of the described system. In reality, this is hard to obtain. So, a more pragmatic formulation of completeness would be that the views together cover all *stakeholders' concerns* concerning the modeled system. A set of views is *consistent* if they are abstractions of the same object. As with completeness, means to provide rigorous consistency in architectural description is commonly hard to provide. Ideally, the expressiveness of the view model increases with how *independent*, or "orthogonal," the views are from each other. However, the linkage between views is commonly extensive, and should be made explicit by the viewpoint model.

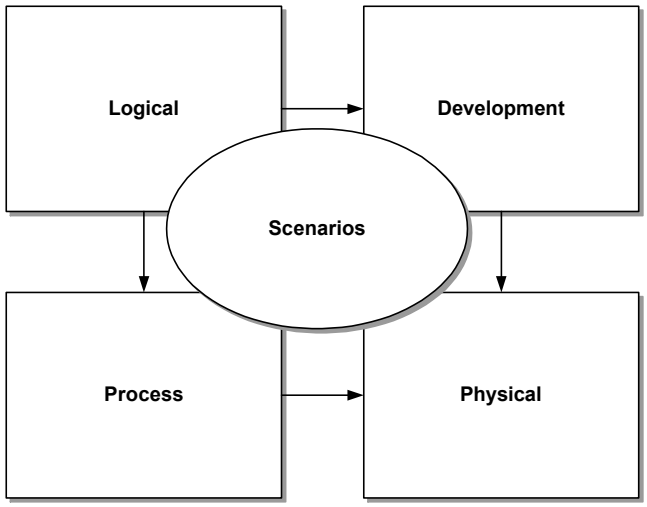

Figure 4. The 4+1 View Model of Architecture (Kruschten 1995)

Several sources describe viewpoints for software architectures on various system levels and for various purposes, e.g. the Zachman framework (Zachman 1987), TOGAF (The Open Group 1999), US Dept. of the Treasury's Enterprise Information Technology Architecture (EITA) (Armour 1999), architectural structures (Bass et al. 1998), and the Rapide language framework (Luckham 1995).

Furthermore, viewpoints for communication in power systems control are provided by Ericsson (1996).

Perhaps the most well known set of viewpoints for software architecture that combines views and scenarios, is the *4+1 model* shown in Figure 4 (Kruschten 1995). The 4+1 architectural view model defines five views (or rather viewpoints): (1) the *logical view* describing the object, component, or data model of the architecture, (2) the *process view* describing concurrency and synchronization aspects of the architecture, (3) the *physical view* describing the mapping of the software onto the hardware and thusly reflecting the distribution of the system, (4) the *development view* describing how the software is organized in its development environment, and (5) the *scenarios* that constitutes a fifth view, intended to illustrate and validate the other four views.

For each view, Perry and Wolf's (1992) threefold definition of software architecture is applied (cf. Chapter 4.2), and applicable UML diagram types are proposed for modeling. The 4+1 view model is rather generic and therefore adaptable also to other notations and tools than the ones originally proposed (Kruschten 1995). Several adaptations of the model have also been formulated for different purposes, using other views to obtain completeness and expressiveness of the architectural description (cf. e.g. Davis and Williams 1997).

During field study Delta, it was discovered that the introduction of views contributed to provide separation of concerns during interaction with different set of stakeholders, and in addition ensured that no significant aspects of the scenarios in the analysis were overlooked. Originally, a subset of 4+1 was considered, due to its explicit relations to scenarios, but in the given context 4+1 turned out to provide inadequate completeness to constitute a proper basis for decision support. In particular, aspects regarding different planning horizons (and their interdependencies), and considerations concerning the provision of COTS components were left without support.

Based on findings gained from field studies Gamma and Delta, an alternative set of viewpoints is therefore proposed, similar to the 4+1 viewpoints, but adapted to incorporate also issues related to COTS and legacy components, and the temporal dependencies between these (see Figure 5). Notably, the viewpoints are developed to address e.g. architectural elements, life cycles, and customer-suppler relationships, from a *user organization perspective*; although the descriptions incorporated in the views may also be used for interaction with stakeholders outside the user organization, such as vendors or business partners.

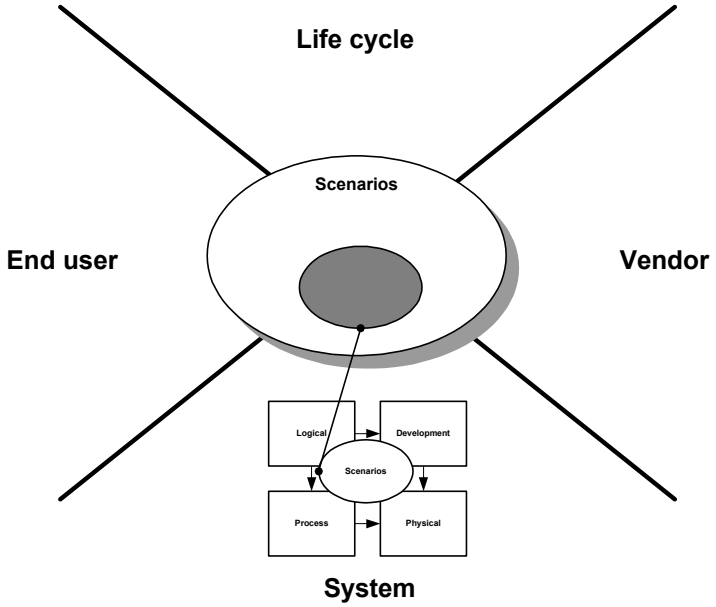

Figure 5. A set of viewpoints adapted for EISM and its relationship to the 4+1 model.

The adapted set of viewpoints comprises: (1) the *end user view* that illustrates user requirements, i.e. the desired target system at a given time if no external constraints need to be taken into consideration (i.e. the initial target architecture), (2) the *life cycle view* that makes explicit temporal dependencies between architectural elements, e.g. considering legacy systems, migration paths, and closure conditions for architectural elements in the EIS, (3) the *vendor view* that captures vendors' abilities to provide desired components, e.g. as COTS components and connectors, and vendors' supply chains of third party components, if these provide implications for the user organization's present or future EIS, (4) the *system view* that includes aspects on the software of the EIS, and (5) *the scenarios* that provide linkage between the views and that contribute to emphasizing important aspects of the problem domain, similar to 4+1.

The proposed viewpoints are not merely a transformation of the 4+1 as they do not replace the 4+1 viewpoints. Rather, they are an extension of the model to address a wider problem domain from a user organization perspective, in contrast to the 4+1 that primarily is created with developing organizations in mind. In fact, the proposed *system* viewpoint may successfully be described using the 4+1 with relevant description techniques for each viewpoint. Thus, the relationship between

the proposed model and the 4+1 may therefore be best described as hierarchical, as depicted in Figure 5, since the proposed set of viewpoints encapsulates 4+1.

The viewpoints for EISM proposed in this section are based on findings from primarily field studies Gamma and Delta (cf. Chapter 7).

4.4.2 STYLES AND PATTERNS

Except for making problem domains more comprehensible, conceptual descriptions, or models, serve the purpose of making explicit similarities with other (already solved) problems, thus enabling reuse of heuristics. One of the primary drivers for architectural design is its ability to codify heuristics as architectural *styles* or *patterns*. Styles and patterns belong to a special class of codified heuristics that prescriptively describe particular choices of form, and their relationship to particular problems (Meier and Rechtin 2000). In scenario-based architectural analysis, styles provide much of the structured rules for the architectural analysis.

The perhaps most interesting part about architectural styles is that they are believed to impact software quality attributes. Certain styles are more appropriate than others for achieving certain quality attributes in a system. It is consequently desirable to classify "good" architectural styles and to employ them when certain qualities are sought. It is, however, far from unproblematic to identify all relevant component constraints and resulting quality attributes of a style.

According to (Alexander et al. 1977), reiterated by (Gamma et al. 1995), a pattern "describes a *problem* that occurs over and over again in our environment, and then describes the core of the *solution* to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice." A set of interrelated patterns forms *a pattern language*. Ideally, one or more of the patterns define the entry point of the pattern language (Schmidt 2000).

Several collections of various types of patterns have been published. Perhaps the most widespread of these is Gamma et al. (1998) on design patterns. Other contributions are Buschman (1996) and Schmidt (2000) on object-oriented patterns for software architecture, Fowler (1997) on reusable object models, and Grand (2002) about design patters for Java. A somewhat controversial type of patterns described is *antipatterns*, i.e. patterns that clarify non-navigable lines of action in management, architecture, and software development (Brown et al. 1998). Although patterns mostly have been applied in architectural design on the single systems level, some attempts to describe more coarse-grained patterns adapted for EAI have been

made, similar to the work presented in Part D of this thesis (Dikel et al. 2001; Linchicum 2000; Lutz 2000).

In software architecture, the notion of *architectural style* is omnipresent. A preliminary definition of architectural style was provided by Perry and Wolf (1992), that in analogy with building architecture described architectural style as a particular codification of design elements and their formal arrangements. Garlan and Shaw (1996) define architectural style as a *vocabulary* of components and connector types, and a set of *constraints* on how they can be combined. Additionally, styles may also encompass semantic models specifying how to determine a system's overall qualities from the properties of its parts. According to Bass et al. (1998), an architectural style is the same thing as a *system pattern* (to be differentiated from *design* and *code patterns*). However, the boundaries between patterns and styles in architectural design are rather vague and subject for a vivid debate.

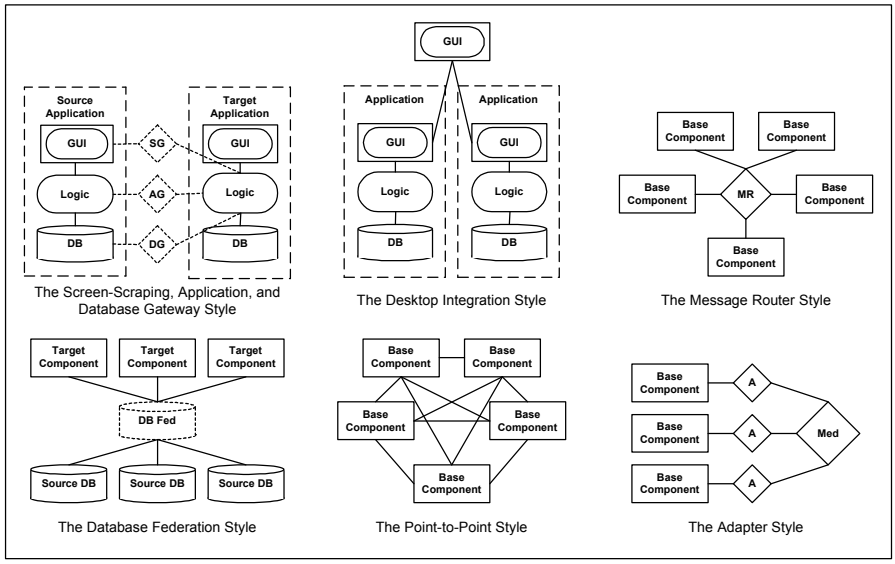

Figure 6. Architectural integration styles (Part D).

In the context of this thesis, an *architectural integration style* denotes a generalization of a historically successful integration solution on the enterprise level of ISs. In other words, applying an architectural style on a certain set of components yields some specific quality attributes. To relate this to Alexander et al. (Alexander 1977) and

Gamma et al. (Gamma et al. 1995), the *problem* is described as a set of components and a set of desired quality attributes while the *solution* is the style definition. This is not a new definition of styles but a reformulation adapted for architectural integration styles for enterprise software systems. The architectural integration styles depicted in Figure 6 were employed during the final iteration of field study Delta (cf. Chapter 4.4).

Constraints on components and connectors. In the proposed style description, the definition is thus expressed as one or several constraints on *components*, *connectors*, *semantics* or *topology*. These defining constraints may imply other constraints (e.g., a constraint on a connector often implies constraints on component interfaces). Constraints on architectural elements are either part of, or results of, the style definition. However, it makes sense to express these constraints separately and in more detail in the case of EISs, as components normally are unmodifiable. Perhaps the most obvious component constraint relates to the component types allowed by the architectural style (stand-alone components, database components, etc.). There are also constraints related to the things that one component needs to know or manage about a collaborating component (DeLine 1999; Gacek 1998; Garlan et al. 1994a; Ockerbloom 1998).

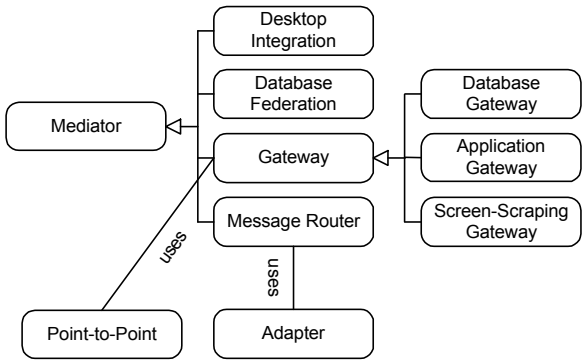

Figure 7. "Uses" and sub-style relations between presented architectural integration styles.

Organizing styles. Styles are related to each other in several different ways, and a clear organization of proposed styles may help both in the use, and in the proposition, of new styles. The two most important relations are the *sub-style* and *uses* relations (cf. Figure 7). A sub-style is constructed by adding constraints to its parent style (or meta-style). A uses relation indicates that one style typically uses

another style. Another way of relating styles is by the type of architectural constraints that they impose. On the topmost level, constraints maybe divided into component type, connector, topological, and semantic constraints. A third set of dimensions for relating styles are the quality attributes that they supposedly support. Particularly interesting to note is that the quality attributes of the styles are most easily expressed using a comparative approach, since it is difficult to e.g. express the scalability of one style without referring to another style.

See Part D for a more elaborative description of how architectural integration styles may be applied in EIS evolution. The case study presented therein constitutes a part of field study Delta (cf. Chapter 7).

4.4.3 DESCRIBING ARCHITECTURAL STRUCTURE

Based on the presented characteristics of the enterprise level of ISs presented in Chapter 3, architectural description of EISs essentially contains the same elements as single systems, i.e. components and connectors, and element interfaces (Garlan and Allen 1994a). However, the nature of these elements differs from single systems (cf. Chapter 3.3). Primarily they are from an architectural perspective, more coarse-grained and heterogeneous compared to elements of single systems.

In systems mainly constructed by the integration of prefabricated components and connectors, there is a need to visualizing and modeling the structure of the system in order to make explicit various design alternatives. Architectural models provide much relevant information. E.g., the decomposition of an EIS into components provides a real measure of complexity, and manifests the separation of concerns between issues dealt with on an enterprise level and issues delegated to separate implementation projects. Note that this separation of concerns may be carried out deliberately for strategic reasons, or may be imposed due to organizational and/or technical constraints.

Much work has been carried out in the vein of developing the formal underpinnings for software architecture. Several Architectural Description Languages (ADL)s such as Acme (Garlan et al. 1997a), Adage (Coglianese and Szymanski 1993), Aesop (Garlan et al. 1994b), C2 (Medvidovic et al. 1996), Darwin (Magee et al. 1995), Rapide (Luckham et al. 1995), SADL (Moriconi et al. 1995), UniCon (Shaw et al. 1995), Meta-H (Binns and Vestal 1993), Z (Spivey 1992), and Wright (Garlan et al. 1994a) have been developed and tested.

As a result of the attempt to formalize software architecture, several taxonomies for architectural connection have been developed. One such taxonomy for architectural

structure is provided by Garlan et al. (1997a) as a basis for their ADL, Acme. They define *components* as the primary computational elements and data stores of a system. *Connectors* represent interaction among components, and a *system* denotes the boundary of an architectural description by representing the configuration of components and connectors. In addition both components and connectors are provided with interfaces. Components' interfaces are defined as a set of ports, where each port describes "a point of interaction" between the component and its collaborators. Connector interfaces are defined as *roles* that define interaction with its collaborating components.

One of the driving forces behind ADLs is their expected capability to forecast potential misfits among architectural elements, popularly termed *architectural mismatch* (Garlan et al. 1994a). Several attempts have been made to identify and analyze architectural mismatches out of formal architectural descriptions (cf. e.g. Abd-Allah 1996; DeLine 1999). One example of an architectural mismatch in EISs, is problems concerning redundant functionality and data, as described in Chapter 3.3. Architectural mismatches impose a strategic impact on EISM as it dictates the ease by which components may be replaced in the future.

However, the Achilles heel of formal ADLs is the trade-off between generality of language and its support of formal analysis. As formalism constrains expressiveness, they are commonly employed in relatively narrow domains (Johnson 2002). Therefore, to provide an alternative to formal ADLs, several attempts have been made to employ description and modeling techniques for software design for modeling of software architecture. As a part of this work, the notations from object-oriented design have been evaluated in order to determine their capabilities for architectural description on the enterprise level of ISs (see Part A). On account of its widespread use, its recent standardization, and its broad scope, the Unified Modeling Language (UML) has been selected as the representative notation for the object-oriented paradigm. According to Booch et al. (1999), Kobryn (1998), Medvidovic et al. (2002), and Störrle (1999), a system's architecture may be modeled using UML.

Since an architectural description language for EISs architecture must be easy to use, only a relatively simplistic subset of the great variety of alternatives to express the structure and behavior of a system has been considered for the modeling of EIS architectural structure. A drawback of the UML notation has been found to be the implicit assumption of a common connector. None of the evaluated notations fully support the architectural concept of component connectors, which proves vital for modeling the heterogeneous nature of the user organizations' EIS. However,

aspects of packaging mismatch as to data and functionality has been successfully visualized, e.g. by using UML class diagrams.

See Part A for the complete evaluation of object-oriented design notations regarding their capabilities for architectural description on the enterprise level of ISs.

Chapter 5

Towards a novel approach for enterprise information systems management

5.1 INTRODUCTION

To exemplify how the architectural concepts investigated in Chapter 4 may be employed as a part of a framework for strategic ISs planning (SISP), an alternative SISP approach for the utility under study in field study Delta was developed. The purpose of the alternative approach was to provide a framework for decision support a prestudy, preceding procurement of several new components to the utility's EIS.

5.2 KEY CHARACTERISTICS OF THE PROPOSED FRAMEWORK

Based on experiences from field study Gamma, four key-areas were given certain attention during the development of the alternative SISP approach. These are further elaborated below:

Implementation-centric. The approach is implementation-centric in its characteristics as it stresses the importance of bridging the gap between planning and implementation. In small and medium-sized enterprises where resources are scarce, it is important that the available resources are applied in an effective manner,

and therefore top-management efforts should not be directed to create a strategy, leaving implementation to other parts of the organization. Furthermore, the whole approach is complemented by a strong focus on teamwork, as user participation helps to ensure fast and effective implementation.

The EISA as the common conceptual model. The approach is based on a framework in which the adapted analysis process, described in Chapter 4.3.2, is a central part. Here, scenarios, and design alternatives expressed in terms of architectural integration styles, provide a conceptual model of an organization's ISs without focusing on specific software technologies and functionality. The purpose of the approach is to help management making necessary trade-offs between the qualities made explicit during the analysis, and thus to mitigate risks, cost, and opportunities.

Alignment. In the proposed approach, business strategies, processes, and actions concerning the EIS are defined and aligned concurrently within one process. In order to stress the importance of alignment (Henderson and Venkatraman 1996), the framework adapts Walsham's three themes, or perspectives, for ISs strategy (Walsham 1993), according to Figure 8. First, the business context provides the understanding of the business environment in which the small and medium-sized enterprise operates with a particular focus on the market, the relationship with customers and suppliers (business strategy and objectives). The second perspective is business process. This focuses on understanding the work processes to appreciate whether information flows inhibit business activities, and also to identify changes that might be made as a result of the introduction of new EIS components. Finally, the strategic content embodies the vision for change and the practicality of its introduction given organizational circumstances.

Assessment and prioritization of EISA components. In order to provide a proper input to the analysis, desired and existing EIS is expressed in terms of architectural elements. An organization's present legacy of ISs comprises a huge amount of business critical information. In order to preserve the value of this investment, the legacy architecture must be monitored and assessed in order to identify and prioritize changes in components and connectors. Also, since the use of COTS components delimits the flexibility of the overall system, the identification and assessment of suitable COTS components must be carried out. As quality and functionality of components is often hard to grasp, the capability of the vendor must also be assessed. The monitoring of present and conceivable components are of outmost importance since changed requirements to those must either be processed as bespoke adaptations risking the COTS status of the component, or as

change requests to the COTS product vendor. To summarize, the assessment of the components and the connectors facilitates trade-off analysis between different implementation alternatives, hence providing input for the planning process.

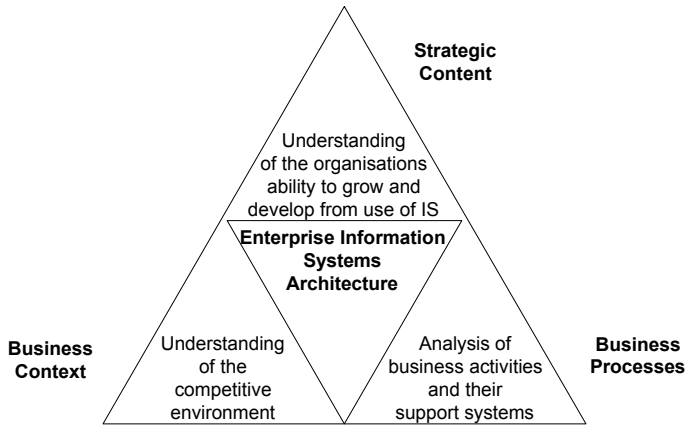

Figure 8. The role of EISs architecture in EISM.

5.3 LESSONS LEARNED

As described previously, there are several factors that collectively influence the way in which an EIS should be designed. Here, the iterative, evolutionary, and interactive characteristics of a spiral process model (Boehm 1988) was found to be highly effective in supporting the gradually increasing awareness of issues concerning legacy EIS components, available COTS components, and business processes during any SISP activities, as well as efficient in managing the ever-changing technical and business requirements that an EIS ultimately must comply with. Thus, the approach allowed the planning team to cope with effects of the environmental turbulence through sense-making and interpretations behavior. Here, the characteristics of the planning process allowed incorporation of good ideas that bubbled up from the operational levels as well as ideas suggested by vendors and other partners. In the process, scenarios were used to describe problematic and/or important situations in the EIS design. Here, the scenarios were found to facilitate effective communication between stakeholders. Moreover, temporal dependencies between different future evolution activities could be made explicit by the use of scenarios.

Although some scenarios started with the business vision, the developed architectural alternatives were made possible, even suggested, by technological advances, especially in terms of COTS middleware solutions. In the analysis of the different design alternatives, the conceptual modeling was found to effectively help management to make necessary trade-offs concerning the EISA, including translation and interpretation between strategic and operational business requirements and ISs requirements, as well as between identified technology opportunities and risks, and business opportunities and risks.

As often referred to as a success factor in IS/IT projects (Standish group 2000), the involvement of top-management in the studied utility proved to be of paramount importance for the process. In contrast to large organizations, it is often easier for management in small and medium-sized enterprises to grasp the total problem domain without loosing too much of the often important details. Thus, similar to Earl, this thesis suggests that senior management involvement could be more valuable in the project level rather than on the strategy formulation level in these companies (Earl 1993).

As a tool for decision support concerning EISM, scenario-based analysis techniques provide a more iterative, goal-oriented, and evolutionary process of organizational learning, compared to the conventional top-down SISP processes, i.e. an approach that corresponds to what Earl refers to as the organizational approach (Earl 1989). It is stressed, based on the results from field study Delta, that additional effects of applying this analysis scheme, other than just providing a more elaborate basis for design decisions, consist of improved stakeholder communication and increased awareness and competencies due to increased stakeholder participation in the planning process.

Chapter 6

Research methodology

6.1 INTRODUCTION

To gain in-depth understanding of the qualitative technical and organizational parameters related to major modifications to EISs, a qualitative research approach has been deployed. Data collection and theory testing have been performed in four studies, three case studies, and one study based on action research. A summary of these field studies is given in Chapter 3. Relating to Figure 9, the cycle of building, testing, and extending theory correspond with the right outlined process.

From the perspective of this work, the primary advantages with the choice of a case study and action research in favor of a quantitative approach are its ability to capture "reality" in greater detail and to allow for more variables than is possible when using quantitative approaches as suggested by e.g. (Galliers 1992). This is further accentuated by the fact that the contribution of this thesis is mainly in the vein of improved engineering practice.

In short, the foundation for this doctoral thesis in terms of employed research methods is: (1) its employment of *qualitative methods*, i.e. *case study methodology* and *action research*, (2) combined with an *interdisciplinary* approach, as theory from several research domains are addressed, and finally its nature of a (3) contribution to applied *engineering science*. The motivation for these enumerated aspects are further elaborated on below.

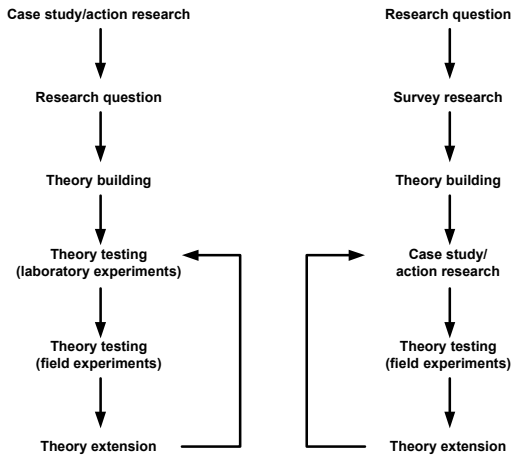

Figure 9. The use of alternative IS research approaches in the process of theory building, testing, and extension (Galliers and Land 1988 recited in Galliers 1992; Jarvenpaa 1988).

6.2 INFORMATION SYSTEM RESEARCH

In the realm of ISs, there has been a general shift in ISs research away from technological to managerial and organizational issues. As presented by Land in Galliers (1992), it can be argued that ISs make sense only in the context of the purpose to which they are employed. Since this implies that the units of analysis commonly become complex, such as large-scale software systems in an organizational context, limitations of an exclusive use of quantitative research methods in ISs research become explicit.

During the last decade, there has been an increased interest and acknowledgement of post-positivist and qualitative methods in ISs research. The earlier unconditional criticism of non-quantitative research methods as less scientific has been lively debated, based on the assumption that the only valid approach to increased human knowledge is the empirical-analytical method (Bleicher 1982). Two major limitations with, for instance, laboratory and field experiments in IS research are given in (Galliers and Land 1987): "(1) There are only a limited number of factors that can be studied under laboratory conditions, and it is difficult to reproduce a "real-world" environment in these circumstances[. Therefore, (2) the] need to apply

values to variables often leads to the elimination of factors that, although they may have relevance, are difficult to value; thus applying to them zero value – which is probably the value they do not have!"

Consequently, many authors (cf. Galliers 1992; Walsham 1993) argue for a post-positivist approach that stresses the importance of *methodological pluralism*, i.e. we cannot assume that there is just one correct research method, but many possible alternatives. Over the last decade, *qualitative* methods, originally developed for the social sciences community to enable researchers to study social and cultural phenomenon, have gained ground in ISs research.

Qualitative research methods are designed to help researchers understand people and the social and cultural context within which they live (Myers 1997). Examples of qualitative research methods are *case study research*, *action research*, and *ethnography*. According to Finkelstein (2000), this shift is also notable in the domain of *software engineering* where there is an increased acceptance of methodological diversity, including qualitative research methods, to make a case for real-world research. Also, software engineering research is becoming increasingly aware of the interplay between software, hardware, and organizational systems, and may thus be argued to attempt to bridge the gap to the fields of *systems engineering* and *ISs*.

In addition, as pointed out by Galliers (1992), ISs research is multi-disciplinary. Foundations for its study are to be found in several "reference disciplines," ranging from philosophy and social sciences to mathematics and engineering sciences. Thus, it is debatable if ISs research may be considered as a homogenous research field working in a well-defined research paradigm as suggested by Kuhn (1970).

Moreover, as this thesis attempts to transfer and adapt taxonomy, description techniques, and analysis models from the technology-oriented field of software architecture into the more organizational dependant area of ISs, it can be considered as an *interdisciplinary* contribution. Although much qualitative research is carried out with its philosophical base in *hermeneutics,* it is stressed that not all qualitative research belongs to the interpretive tradition as described in (Walsham 1993), e.g. hermeneutics or post-positivism. On the contrary, it may be argued that the choice of a qualitative research method (such as the case study method) is independent of the underlying philosophical assumptions adopted. The underlying research epistemology, (Orlikowski and Baroudi 1991, recited in Myers 1997) implies a threefold classification that divides qualitative research into the *positivist*, *interpretive*, and *critical* categories.

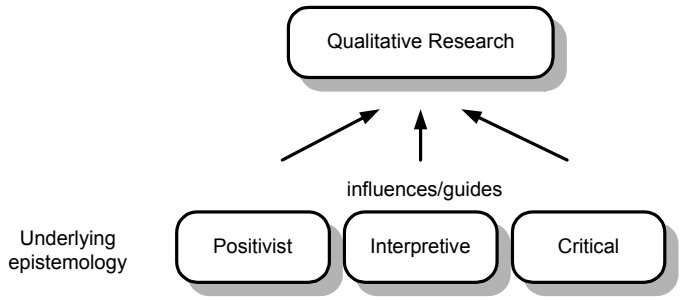

Figure 10. Underlying philosophical assumptions for qualitative research (Myers 1997).

For this thesis work, the relevant philosophical approaches to quantitative research are the positivist (Yin 1994) and the interpretative (Myers 1997; Walsham 1993). Positivist studies generally attempt to test theory, in an attempt to increase the predictive understanding of phenomena, i.e. positivists generally assume that reality is objectively given and can be described by measurable properties that are independent of the observer/researcher and his or her instrument. Interpretive researchers start out with the assumption that access to reality (given or socially constructed) is only through social constructions such as language, consciousness, and shared meanings (Myers 1997).

6.3 PRACTITIONERS' ROLE IN INFORMATION SYSTEMS RESEARCH

According to IS researchers of the post-positivist stance, the study of "real-world" problems should include studies in the "real-world." There are several reasons for this position. For instance, there exists a gap between what industry expects from academic research within the fields of ISs and software engineering, and what is actually produced by this research. This gap is due to both misplaced expectations by industrialists as well as inapplicable research results in academia. By allowing academic research within real-world ISs, much can be gained from an academic perspective. Increased understanding of the complex phenomena that takes place when new ISs are planned, developed, and introduced in their organizational contexts is perhaps the most prominent advantage.

Collaborative research is one navigable approach to bridge this gap, e.g. by means of the case study or action research approaches described below. Except for the

academic advantages including enhanced *relevance* as to addressed research issues, and potential increased *generalization* of results obtained by research strategies such as case studies or action research studies, collaborative research is an effective way to *communicate academic knowledge to industry*. Practitioners' role in the development of engineering disciplines, such as software architecture, is addressed in Garlan and Shaw's (1996) codification cycle (see Figure 11). According to this explanation model, based on the early experiences from the domain of software engineering, "good" engineering practice may emerge from commercial practice by exploiting the results of a companion science.

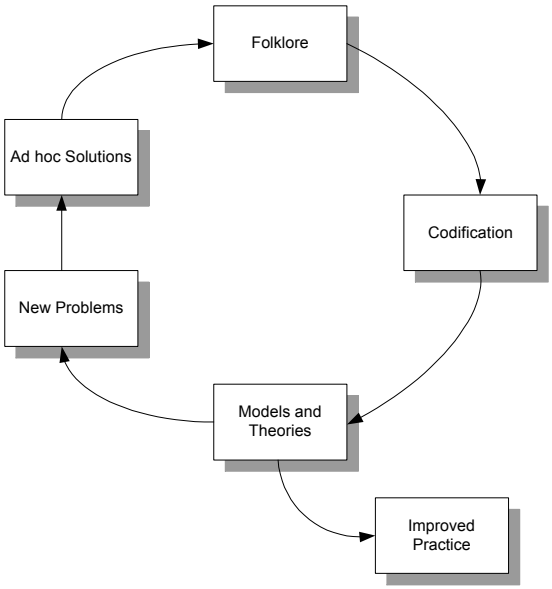

Figure 11. Garlan and Shaw's (1996) codification cycle for science and engineering.

In short, the model can be described as follows. Prevalent problems are at first solved in an ad-hoc manner by practitioners. As working solutions on problems are distinguished, they are informally exchanged among practitioners as "domain folklore." This common knowledge becomes more and more systematic, and is eventually codified, e.g. as written heuristics and rules of procedure. In the context of EISs, practitioners have produced several textbooks under nametags such as *enterprise application integration* (Linthicum 2000), and earlier *legacy system migration* (Brodie and Stonebraker 1995), which attempts to codify engineering experience.

When codified knowledge becomes solid enough, it can be used by scientists to improve models and theories, which, in turn, may contribute to improve engineering practice. To close the loop, the progress in engineering practice will allow us to contemplate more challenging problems that in turn will provide new ad-hoc solutions, and so on.

6.4 CASE STUDIES

Regarding qualitative research within ISs, case studies have gained ground as a valid research strategy over the last decade. The case study methodology applied in this doctoral thesis project has its foundation in the fundamental work made by case study proponents as Robson (1993) and Yin (1994), and the work by Walsham (1993) to enhance case study methodology as a tool for ISs research. To provide a basis for further adaptations of qualitative research to empirical domain of this doctoral thesis project, the author has partly relied on results presented in previous works (Cheong 1999; Johansson et al. 1997; Rahkonen 1996a).

According to Yin (1994), case studies constitute a rewarding research strategy when "how" of "why" questions are being posed to gain understanding of some (rather) contemporary phenomenon with some real-world context. Such case studies are generally termed *explanatory*, and are augmented with two other categories, *exploratory* and *descriptive* case studies.

Explanatory case studies. Many view the explanatory type of case studies as the strongest form of case study research. It is argued that only when the effect on the outcome of a phenomenon of one parameter is explained in detail, knowledge has been gained. The aim of the explanatory type is therefore to *explain causality of events* in an attempt to gain knowledge of a phenomenon. In this thesis, field study Gamma is carried out as an explanatory case study.

Exploratory case studies. The main application of exploratory case studies is basically to develop appropriate hypotheses for further study. Hence, case studies of this type focus on *finding new information* on the topic of the research question. For instance, when starting afresh with research in a domain, it would be appropriate to explore the state-of-the-practice in that field (cf. field study Alpha).

Descriptive case studies The ambition in a *descriptive* case study is somewhat higher than in an exploratory study. Working with previously collected data, or new data, the researcher describes studied phenomena or events within a particular domain. The ambition is not to explain the interconnection of phenomena and

parameters but instead to *describe which parameters affect certain phenomena* within the studied subject, which is the case in field study Beta.

Although case study methodology presents notable strengths in describing real-world phenomenon and/or events, it also, like all research methods, has some potential weaknesses that must be dealt with (Yin 1994). In particular, common themes of criticism concerning case studies as a valid research approach are: (1) its *potential lack of rigor* and (2) its *weaker basis for scientific generalization*. Issues concerning (1) are separately addressed below in Section 6.6. An explanation of *generalization*, or *external validity* regarding (2) is given below:

Generalization. Critics of the case study approach often fail to appreciate the power of *replication logic* as opposed to that of *sampling logic* when implying that case studies are inappropriate for generalization. *Literal replication* refers to case studies' ability to predict the same result for a number of cases, whereas *theoretical replication* aims at predicting contrasting results but for predictable reasons. To refute criticism according to (2) above, Yin (1994) stresses that "the case study, like the experiment, does not represent a "sample," and the investigator's goal is to expand and generalize theory (analytic generalization) and not to enumerate frequencies (statistical generalization)." Hence, the employment of multiple case studies aims at increasing robustness in theory building and testing in the same way as multiple (controlled laboratory or field) experiments.

In the greater perspective, the issue of generalizing results from case studies to theory should be contemplated in the light of *methodical pluralism* as a preferred approach to theory building and testing in ISs research (Galliers 1992), by Yin (1994) referred to as *Level Two [theoretical] interference*. For the role of case studies related to theory building and testing in ISs research, see Figure 9.

In this context of this thesis, field studies Alpha, Beta, and Gamma are based on case study theory, whereas field study Alpha can be described as an exploratory studies, mainly aimed at further developing research questions and hypotheses and obtaining appropriate contextual knowledge. Beta and Gamma are explanatory case studies, aimed at answering questions related to modifiability of ISs at the enterprise level.

6.5 ACTION RESEARCH

Action research, which has been performed in field study Delta, could according to Baskerville and Wood-Harper (1996) be seen as "an interventionist approach to the

acquisition of scientific knowledge that has sound foundations in the post-positivist tradition." Although it was the author's first encounter with this research strategy, it was found adequate for its intended purpose, to test and further the novel approach for ISs planning and implantation of EISs presented in Chapter 5. Four commonly referred to characteristics of action research are according to Baskerville (1999): (1) an orientation towards action and change, (2) a focus on problems, (3) an "organic" process involving systematic and sometimes iterative stages, and (4) collaboration among participants.

In ISs research, its primary use is applied research settings in which there is an attempt to obtain results of practical value to groups with whom the researcher(s) is allied, at the same time adding to theoretical knowledge. (Galliers 1992). A guiding example of the application of action research that has influenced ISs research is Checkland's (1986) soft systems methodology that attempts to link action research and systems development.

An outstanding feature of action research e.g. unlike case study research, is that, the researcher explicitly interacts with the unit of analysis as a part of the research setting. As one of several methods in the post-positivist ISs research stance, the method has some decided strengths. Notably, the practical as well as the theoretical outcomes of action research are emancipatory (Baskerville 1999; Galliers 1992). The author would, for example, stress action research as a strategy for applied science that has the potential to considerably speed up Shaw's codification cycle for engineering and science (Garlan and Shaw 1996) compared to other qualitative research strategies. Moreover, the method aims to make the bias of the researcher explicit (Galliers 1992).

Action research suffers from similar weaknesses as case study research but additionally places a considerable responsibility on the researcher to maintain a holistic and objective position when objectives and results are at odds with other groupings in the organization under investigation (Galliers 1992). The ethics of the particular field study are a key issue, as action research, despite its clear differences sometimes may be taken for consulting (Baskerville 1999; Galliers 1992).

The process for action research has evolved over the last three decades. The most prevalent description is Susman's (1983) five-phase cyclical process summarized in (Baskerville 1996). Referring to Figure 12, the five phases in this model comprises *diagnosing*, *action planning*, *action taking*, *evaluation*, and *specifying learning*. In addition to the five phases, a client system infrastructure (also referred to as or a research environment) must be established.

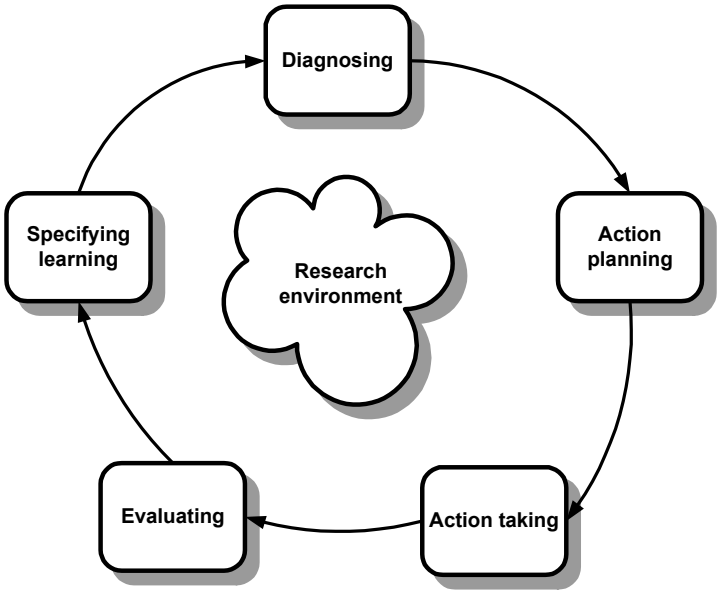

Figure 12. The action research cycle. Adapted from Baskerville (1999).

Research environment. In action research, the research environment, or the *client-system infrastructure*, includes the specification and the agreement that directs the research. Considerations found in the agreements may include boundaries of the research domain, and the entry and the exit of the scientist(s). A key aspect of the infrastructure is the collaborative nature of the undertaking, where the researcher(s) work closely with the practitioners in the investigated organization.

Diagnosing. In the diagnosing phase, the underlying causes for the organization's desire to change is developed into an working hypothesis, taking the theoretical assumptions and the nature of the organization, and its problems, into consideration. It is stressed that the approach for this stage should be holistic, and that reduction and simplification should be avoided.

Action planning. To establish a target for the change and an approach to reach it, an action plan is developed. In the planning process, researchers and practitioners collaborate to find actions that are intended to relieve or improve the problems addressed in the study. Another important prerequisite that directs the action planning is the theoretical framework.

Action taking. The researchers and the practitioner then collaborate in implementing the active intervention in the client's organization. To implement the actions that are to take place, several forms of intervention strategies may be adopted, such as *directive* (in which the research(ers) direct the change), non-directive (i.e. the change sought indirectly).

Evaluation. In order to determine the actions' theoretical effects, and these effects' contribution to solving problems(s) addressed in the study, the researchers and the practitioners particularly evaluate the result of the implemented actions. The evaluation must question critically if the undertaken action(s) were the sole (or a prominent) contributing cause to the success or failure. The outcome of this phase should also include an improved theoretical framework for additional research cycles.

Specifying learning. Although described as a formal phase in the action research cycle, the codification of the knowledge attained during the action research is better described as an ongoing process. According to Baskerville (1999), the knowledge gained in the action research can be directed to three audiences: (1) the collaborating organization that may benefit from the "restructuring of organizational norms to reflect the new knowledge gained by the organization during the research," also referred to as *double-loop learning* (Argyis and Schön 1978), (2) the researcher, as the additional knowledge gained may provide a rationale for forthcoming action research interventions, and (3) the scientific community, as the success or failure of the theoretical framework may provide important knowledge to the research community faced with future research setting.

6.6 RESEARCH QUALITY

The measure of quality, in research processes can be divided into the measures of *reliability* and *validity*. The issue of validity may be further divided into *construction*, *internal*, and *external* validity (Yin 1994).

Reliability. Issues of reliability deal with the (literal) replicability of the research results. The objective of achieving a high degree of reliability is to ensure that another investigator, using the same set of collected data, comes to the same conclusions. Studies that have high reliability thus run less risk of containing bias and errors. Achieving high reliability is done through careful documentation of collected data and performed analysis. When performing case studies, Yin (1994) suggests the use of a *case study protocol* to ensure structured and complete documentation of the case, which may serve as a repository of collected data. Raw

data as well as analyzed and refined results of performed surveys, interviews, and experiments may thereby be stored in a uniform way, accessible to other researchers for further study and scrutiny.

Validity. When research results are qualitative, such as for case studies, the issue of avoiding bias and subjective reasoning, *validity*, is of course of the highest importance. The division of validity into three separate measures suggested by (Yin 1994) provides taxonomy for validity in qualitative research. *Construct validity* deals with establishing correct measures for the concepts being studied. When striving for high construct validity, the researcher should clearly define which aspects of a specific phenomenon are to be studied, and also establish measures that clearly reflect the studied aspects. To verify that the collected data is valid, the approach of triangulation may be used. In the field studies presented in Chapter 3, (data) triangulation has been attained within each case study by using multiple sources of evidence including documents and focused interviews, combined with open-ended interviews and participant observations. *Internal validity* (applicable for explanatory case studies only) addresses the degree to which casual relationships may be established in a research study. The task of achieving internal validity is best tackled by selecting a limited scope of study, thus reducing the amount of parameters necessary to consider. The last measure of validity, that of *external validity* is of concern when the results of the study are to be generalized to a greater environment than the one immediately studied (cf. the discussion on generalization below). Achieving a high degree of external validity is based on having control of external parameters and their effect on the outcome of an event.

6.7 ETHICAL CONSIDERATIONS

When carrying out qualitative research in close corporation with practitioners, ethics is a key issue. Unavoidably, investigating planning and implementation of EISs in their organizational context entails dealing with sensitive information from both vendors and user organizations. In all field studies included in this thesis, there have been a clear agreement between the author and the other involved parties that certain information should not be made available to a third party, without the consent of the information providers. Therefore, a large part of the research protocols (e.g. documented interview and documents) from the field studies are confidential, i.e. only results published in papers are open to the public.

Consequently, the identity of involved companies, projects, and products etc. has been excluded in the rendering in Chapter 3, and Parts B, C, and D. Unfortunately,

this confidential nature of the field studies obstructs to some degree the external visibility of the research. Moreover, in spite of its clear differences (Baskerville and Wood-Harper 1996) critics sometimes take participatory research for consulting. Undoubtedly, a great responsibility is placed on the researcher who must be aware of "political" issues in the client organization (Avison et al. 1999; Galliers 1992). However, it is the author's experience that these issues, in practice, are manageable in the same way as the issue of confidentiality, by the employment of a mutually acceptable framework for cooperation. Also, to avoid potential misunderstandings; interviewees and other key informants have reviewed results from the field studies before including them in the case study protocol.

Chapter 7

Summary of field studies

7.1 INTRODUCTION

As previously explained in Chapter 1, this thesis relies on empirical data collected during a number of field studies (case studies and action research studies) carried out in mainly small and medium-sized electric utilities[24]. The academic contribution of each activity concerning theory building and testing is mainly addressed in Parts A to D, and summarized in Chapter 4 and 5 of this *Introduction and summary*. However, since the field studies of editorial reasons are rather briefly discussed in the included papers, this chapter aims to describe the research setting for each field study, together with a brief summary of the "case." The relationships between the field studies and Parts A to D, are visualized in Figure 13.

To characterize each field study, field study Alpha was a rather broad exploratory multiple case study which main purpose was to seek relevant research questions and to gain domain knowledge. Field study Beta had the nature of an in-depth descriptive case study comprising the acquisition of a settlement system that was highly integrated with the organization's present ISs. In this study, both technical and organizational aspects were investigated, based on the system's selected architecture and the relationship between the buyer and the vendor. To employ a more holistic perspective on EISM, field study Gamma was initiated.

[24] According to the current definition of *Small and Medium-sized Enterprises* (SMEs) provided by the European Community, companies with less than 250 employees, a turn-over lower than 40 million ECU, and which are owned for less than 25% by non-SMEs, except banks or venture capital companies.

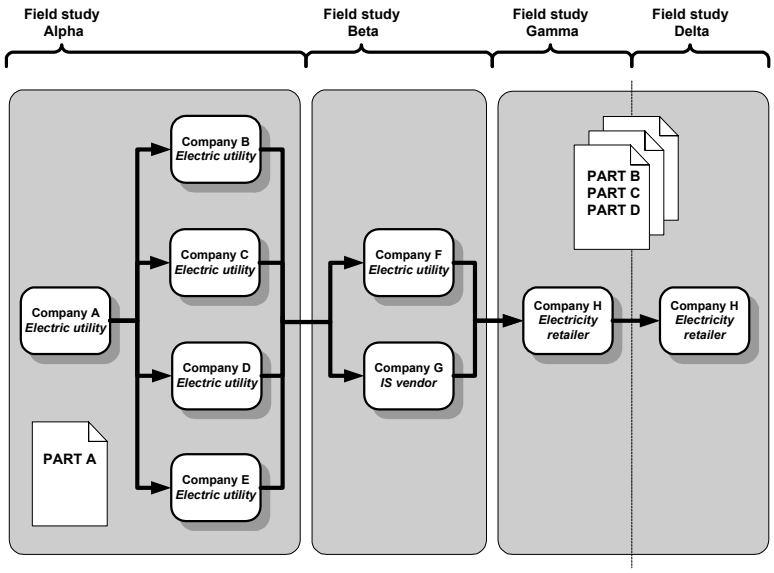

Figure 13. An overview of the field studies and their relations to Parts A, B, C, and D.

This study was carried through as an explanatory case study aiming to seek the causes for a (according to the client organization) failed pre-study, preceding the acquisition of a new business system. As a part of the study, the legacy architecture of the present EISs was audited and later used as input for the theory building presented in Parts B, C, and D. Another purpose of field study Gamma was to prepare for field study Delta that was an action research study aimed to test and further parts of the theory based on findings from field studies Alpha, Beta, and Gamma.

Parts A, B and D were written in very close cooperation with Pontus Johnson; the specific individual contributions are therefore difficult to separate. However, the work of the present author is concerned with the concept of evolution of ISs, while Pontus Johnson's work (Johnson 2002) considers ISs integration. Further differentiating Pontus Johnson's work from the present is its emphasis on deduction-based (formal) approaches to architectural analysis, whereas the present work explicitly focuses on the aspects of modifiability and temporal constraints. Part C was written in cooperation with Magnus Haglind and the field studies (Gamma and Delta, cf. Chapter 7) were performed jointly with him. However, in the theory building and analysis, Magnus Haglind's focus is on strategic planning

(Haglind 2002), while the present author investigates architectural description and analysis for its applicability as a tool for decision support in EISM.

7.2 FIELD STUDY ALPHA: AN EXPLORATORY CASE STUDY

This field study is the result of five case studies, which were carried out by the author in cooperation with four colleagues during a time period of two years. The field study has the nature of a multiple exploratory case study (Yin 1994) involving five electric utilities. Initially, a smaller investigation, that could be characterized as a pilot case study, was carried out to increase the overall rigor and relevance of the four concluding case studies included in the field study. Excluding the pilot case study, data collection was mainly carried out by eleven open-ended interviews.

At the time of the case studies, 1997 to 1998, all investigated electric utilities were in the process of restructuring themselves to better match the conditions of the reformation process on the Swedish electricity market, which was initiated on January 1, 1996. Two of the utilities in the study were (at the time still rather independent) subsidiaries of major Swedish Enterprises, whereas the remaining three fell under the definition of small and medium-sized enterprises provided in Chapter 1. The size of the organizations ranged from 30 000 to 100 000, and up to 150 employees.

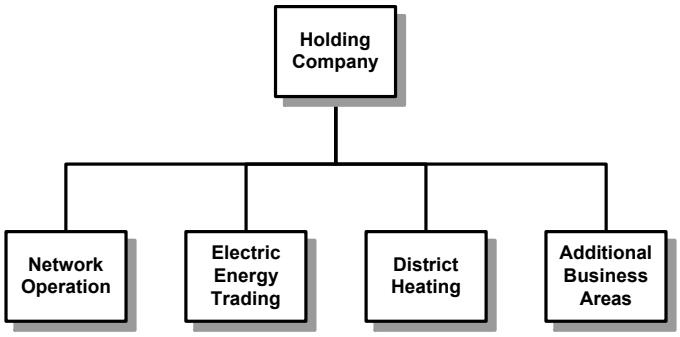

Figure 14. The business areas of a typical electric utility. Partly adapted from Cheong (1999).

Except for providing domain knowledge regarding the organizations' structure, its business activities and processes, and the concerns of the organization, the case

studies had two overall objectives. Firstly, the organizations' effort to provide strategic management on the enterprise level of their ISs was investigated. Ongoing actions such as IS projects in progress and the work in project steering committees etc., were matched against the organizations' strategies for their EISs, either in the form of IS/IT strategy documents, or in the form of statements made in interviews with the management teams for each organization. Secondly, the companies' portfolio of present, and planned ISs and IS projects were documented. The investigation with regards to the IS/IT strategies of electric utilities was later refined and expanded by Cheong (1999) and Haglind (2002).

The case studies included in field study Alpha are also described (from a slightly different perspective) in Andersson et al. (1998), and provided the primary empirical basis for the author's licentiate thesis (Andersson 1997b). From the perspective of this work, the findings from the case studies within field study Alpha have contributed to the formulation of the research question (cf. Chapter 1) and the characteristics if enterprise level of ISs presented in Chapter 3.

7.3 FIELD STUDY BETA: A DESCRIPTIVE CASE STUDY

This field study was performed by the author and a colleague as a descriptive case study of a newly commissioned combined IS for balance and network settlement of electric energy (SvK 1996; SvK 1997), intended for the newly deregulated Swedish electricity market. The case study was carried out in retrospect and covered the complete acquisition project, including initial planning, pre-studies, evaluation of tenders, development, and deployment. Data was collected from several sources of evidence, including 13 focused interviews with key-informants from both the user organization and the supplier, and contractual documents, pre-study reports, the request for tender, status reports (from both the user organization and the supplier), steering committee protocols, and the design specification of the IS. The analysis focused on the relationships between technical and organizational parameters in the project, partly by employing Software Engineering Institute's (SEI's) Software Acquisition Capability Maturity Model (Ferguson 1996) as an analysis framework.

The studied procurement was of interest for several reasons. Firstly, the procurement strategy was well defined in the early phases of the project. Secondly, as a consequence of the procurement strategy the contract only contained technology independent requirements, i.e. functional and qualitative requirements. Thirdly, the project incorporated a strict definition of responsibilities between the buyer and the vendor. Fourth, the buyer and the vendor followed very different modus operandi regarding their cultures for acquisition and development projects

respectively. Moreover, their organizational size and economical strength differed significant between the buyer and the supplier. The buyer was a major Swedish utility, whereas the supplier was a newly formed company for whom the buyer constituted an important future reference. Fifth, both the buyer and the vendor considered the concluded project as successful.

The data collection and analysis performed during this field study is more elaborately presented in Andersson and Johnson (1999). The field study especially contributed to create knowledge concerning the organizational implications of EIS evolution.

7.4 FIELD STUDY GAMMA: AN EXPLANATORY CASE STUDY AND FIELD STUDY DELTA: AN ACTION RESEARCH STUDY

Data collection and analysis of this field study was carried out as an explanatory case study by the author and a colleague. The main issue addressed was to explain why an EISs planning initiative had failed to reach the expectations of the company under study, and based on this explanation propose some characteristics of a process and techniques for planning and implementation of EISs, suitable as work hypothesis (Baskerville and Wood-Harper 1996) for field study Delta. Data collection was accomplished by five open-ended interviews with key-stakeholders within the electricity retailing company and the owner companies, and by document studies of consultant reports, combined with records and notes from meetings. The study was carried out in retrospect in relations to the investigated events.

The enterprise under study was recently formed by a merger of the previously energy sales operations of five municipality-owned electric utilities. Initially, the company was established with only three utilities as owners. One of these owners was considerably bigger than the others, and had thusly a major influence on the strategies and the operations of the firm. A year later, the number of owners was extended when a rather large utility acquired a 25% part of the company. This accelerated the consolidation process, in which the management team had been focused issues such as establishing the organizational structure, management team, and working routines, rather than creating competitive edge. Hence, the management team started to review the strategies in order to prepare for forthcoming changes in the electricity market, e.g. increased competition as a result of changes in the deregulatory framework by the introduction of a new standardized system for settlement based on consumption profiles.

Early in the strategy review, management got several indications that most identified problems were related to shortcomings in the existing EISs to support the business operations sufficiently. Prevalent EISs were a legacy from the owner companies, and these companies were also responsible for the governance of most systems. It became also evident that an important instrument for finalizing the organizational consolidation of the company was a unification of the EISs, as existing legacy EISs consisted of a wide array of legacy systems and components that were poorly fit for their purpose. Employed ISs were found to lack functionality in several critical areas and provided on the other hand redundant functionality and data storage in others.

Moreover, employed ISs were not designed for collaboration. The company experienced problems to merge information for customer support, energy trading, and sales activities, on a daily basis, and virtually lacked means for effective analysis and reporting on the enterprise level. In order to quickly reap some of the benefits of the merger while maintaining business operations, rudimentary scripts and/or spreadsheet programs were used as temporary solutions for the exchange of information between the legacy components and the compilation of this information. This, however, increased overall heterogeneity and complexity unfavorably and was therefore considered as a short termed tactical solution. As a consequence of the low level of integrability, the degree of automation was low which, in turn, demanded considerable manual intervention to provide a reasonable flow of operations.

To initiate the process towards a more appropriate EIS, the management team decided to undertake a systematic a formal planning process. A major management consultancy firm was contracted to apply their methods and frameworks to contribute to the development of a migration strategy for the major part of the legacy EIS. In the process of developing the strategy, a comprehensive set of activities was carried out by the consultants, mainly directed towards the management team. The activities included elicitation of key elicitation of (primarily) functional and (to some extent) qualitative requirements, by interviewing stakeholders in the owner companies to capture their domain knowledge and previous experiences.

As the company at the time was formed only five years earlier neither documented work routines and processes, nor an explicitly formulated business strategy was available to guide the consultants and to delimit the scope of their work. Consequently, the proportions of the applied method grew and become difficult to grasp rather early in the process. Although different design alternatives were presented in the strategy, they failed to fulfill some (partly implicit) expectations of

the company and its managing team. Also, the conclusions presented were found to be on a too abstract level to provide any crisp guidance for coming activities. The management team found it hard to select and to prioritize the most important actions and quality attributes for their specific situation. However, even if the initiative at the time was deemed as a failure, the investigation revealed that it had provided some important effects. As a direct result of the strategy process, the management team increased their awareness of the problems with the present EISs, and the need of explicit business strategies to guide further planning and implementation of EISs.

In Field study Delta, that comprises the same organizations as field study Gamma, the author and a colleague was invited by the organization to participate in the ongoing EISM process as external advisors. Thus, this field study could be considered as action research according to e.g. Baskerville (1999). With the findings from field study Gamma, an initial proposal concerning analysis process based on the scenario-based architectural analysis (cf. e.g. Kazman et al. 1998) and further adapted for EISs in Parts B and C, was proposed as a work hypothesis for the study. The analysis process was thereafter refined during three iterations of the analysis process (and the action research cycle, cf. Chapter 6.5). The architectural analysis focused on modifiability in terms of integrability in various planning horizons. Also, trade-offs against other technical and business quality attributes were investigated. During this field study, a number of architectural integration styles were formulated in order to elucidate different principal design alternatives consequences concerning quality attributes.

To sum up, the results from field studies Gamma and Delta are presented in Chapter 4 and 5 of this thesis. In Part B, the applicability of scenario-based architectural analysis on EISs is scrutinized, and Part C further puts the proposed analysis process into the context of SISP framework. To provide guidance for the attribute-based architectural analysis with respect to design alternatives and their connections to quality attributes, the concept of architectural integration styles is described and discussed in Part D.

.

Chapter 8

Summary of included parts

PART A: IT INFRASTRUCTURE ARCHITECTURES FOR ELECTRIC UTILITIES: A COMPARATIVE ANALYSIS OF DESCRIPTION TECHNIQUES

J. Andersson, P. Johnson

Proceedings of the 33rd Hawaii International Conference on Systems Sciences (HICSS-33), Maui, USA, January 2000.

The escalating development of IT enables utilities to reorganize or migrate from their existing disparate software systems towards an integrated EIS[25] that embraces the total organization. Integration of software systems is expected to increase competitiveness and to cut costs. However, since utilities' present EISs are heterogeneous as to type and technical platform, overlapping with regard to both data and functionality, and relying on ad-hoc low-level middleware, integration and EISM often turn out to be hazardous.

This paper presents a comparative analysis of architectural modeling capabilities of established notations used in structured, information engineering and object-oriented design methods. In the paper, the notations are applied to typical configurations found in electric utilities' EISs. The ambition is to show that

[25] In the paper termed enterprise software system infrastructure (ESSI).

techniques originally intended for design of vendor systems also may be employed to modeling heterogeneous EISs, from a user organizations perspective. Further, the total architectural description of an EIS should provide an overall view of the present and future target architecture that can be used as a basis for strategic and tactical decisions regarding the utility's EIS. Examples of such decisions are the establishment of boundaries between EIS components and the identification of possible simplifications in the structure of the EIS.

The results presented in the paper indicate that a structured approach that makes use of well-defined notations for architectural modeling, may be used to visualize technical risks and opportunities in an organization's current and future EIS, and furthermore may improve communication between stakeholders involved in EIS evolution, e.g. top-management, end-users, system administrators, system owners, and vendors. Furthermore, aspects of overlapping and shared data and functionality may be visualized successfully using object-oriented notations, such as UML[26] class diagrams. The problems that are put forward in the paper have been identified during exploratory case studies of EISM in four Scandinavian electric utilities[27].

[26] UML, Unified Modeling Language.
[27] Companies B, C, D, and E in field study Alpha.

PART B: EXTENDING ATTRIBUTE-BASED ARCHITECTURAL ANALYSIS TO ENTERPRISE SOFTWARE SYSTEMS.

J. Andersson, P. Johnson

Proceedings of the 3rd Australasian Workshop on Software and System Architectures (AWSA '00), Sydney, Australia, November 2000.

As the size and complexity of EISs grow and the dependency on IS/IT increases, assurance of satisfactory quality attributes such as reliability, availability, and integrability becomes an important issue. This paper is based on the observation that the properties of systems developed by single vendor organizations differ significantly from the interconnected collections of ISs operated by user organizations.

One approach to the assessment of quality attributes of software systems is architectural analysis. While existing methods of architectural analysis are primarily aimed at analysis of vendor-developed software systems, this paper presents a method for analysis of EISs. The paper explores the relation between traditional software systems and EISs from a user organization perspective; describing how these two system levels differ. The consequences on the architectural analysis effort are considered in an investigation[28] of an extensive software system modification at an electricity retailing company active on the deregulated Swedish electricity market. The paper reveals that architectural analysis is meaningful on the enterprise level, but analysis methods from the traditional architecture discipline need to be significantly modified to be useful.

[28] Company H in field study Delta.

PART C: STRATEGIC MANAGEMENT OF INFORMATION TECHNOLOGY IN DEREGULATED ELECTRIC UTILITIES: BRIDGING THE GAP BETWEEN THEORY AND PRACTICE

J. Andersson, T. Cegrell, K.H. Cheong, M. Haglind

Proceedings of the Portland International Conference on Management of Engineering and Technology (PICMET '01), Portland, USA, July 2001.

The rapid advancement in IT is transforming business organizations worldwide. Many organizations are making huge investments in IT in order to retain and/or to advance their positions in increasingly competitive and global markets. However, many research studies have shown that only few investments have actually succeeded to bring about the intended benefits. The purpose of this paper is to highlight critical issues in achieving alignment between business and IS/IT strategies in small and medium-sized electric utilities. Further, the results presented in the paper indicates that defining an IS/IT strategy and an EISA is not primary a rational, analytical, and certainly not only a technical activity. Rather, potential success is highly dependent on the organization's ability to ensure communication between involved stakeholders, and ability to manage a process that supports the gradually increasing awareness of issues concerning legacy EIS components, available COTS components, and business processes.

To circumvent some of the identified obstacles, the paper proposes a novel, alternative approach for EISM in small and medium sized organizations, which is less comprehensive and more flexible than most present methodologies for strategic IS planning, thus attempting to bridge the gap between theory and practice. The approach is implementation-centric and based on a framework in which scenario-based architectural analysis is used as a process for EISM. Furthermore, the proposed planning process is iterative and the rationale for this is the need to support the gradually increasing awareness, as well as to tackle the ever-changing technical and business requirements that an EIS ultimately must comply with.

PART D: ARCHITECTURAL INTEGRATION STYLES FOR LARGE-SCALE ENTERPRISE SOFTWARE SYSTEMS

J. Andersson, P. Johnson

Proceedings of the 5th IEEE International Enterprise Distributed Object Computing Conference (EDOC '01), Seattle, USA, September 2001.

A predominant problem in management of EIS evolution is integration on the enterprise level of ISs. Despite the considerable efforts spent on the development of new standards and technologies for software interoperation, the integration of ISs that originally were not designed to interact with each other is a major undertaking, requiring in-depth knowledge of existing systems, incorporation of integration products, and development and/or parameterization of various kinds of adapters and gateways.

This paper presents the concept of architectural integration styles, i.e. architectural styles describing software structures of integration solutions for enterprise software systems. The paper further proposes an approach for selection of styles based on the characteristics of the existing software applications and the desired quality attributes of the integrated system. A number of architectural integration styles for enterprise systems are presented in the paper, and a case study of the style selection process applied to a mid-sized Swedish electricity retailer[29] is described.

[29] Company H in field study Delta.

Chapter 9

Concluding remarks

9.1 SUMMARY OF RESULTS

Evolution of ISs is a multi-facetted issue that over time has proved arduous to manage. On the enterprise level of ISs, an organization's total portfolio of interconnected ISs is considered as one system – an *Enterprise Information Systems (EIS)*, consisting of course-grained and heterogeneous components that in themselves may constitute complex ISs. In EISs, considerations concerning legacy systems and COTS are pervasive. A prominent motivation for considering the enterprise level of ISs as a separate conceptual echelon, is the need for providing separation of concern between project-oriented management of single ISs, and the more holistic, and strategic, management of EISs.

This work applies an engineering perspective on EISM by investigating how description techniques and analysis methods from software architecture may be employed as decision support during planning and implementation of system evolution activities. An enabling motivation for the selection of software architecture as reference discipline for this work is its recent achievements in expressing and analyzing complex software systems consisting of coarse-grained software packages, on the basis of quality attributes (cf. e.g. Bass et al. 1998; Garlan et al. 1994; Heineman and Council 2001). A special emphasis is hereby placed on the quality attribute *modifiability* and the implication of *time*.

Here, EISM of primarily small and medium-sized electric utilities, active on the Swedish deregulated electricity market, has been scrutinized from a *user organization perspective*. The enabling reasons for the choice of electric utilities as unit of analysis, are the implications of the recent electricity market reformation (e.g. deregulatory

demands on reporting of meter and customer data, a more competitive environment resulting in a strive for effectiveness, and company mergers and acquisitions), utilities' broad range of interconnected ISs, and small and medium-sized electric enterprises' sparse resources for strategic management.

Several prominent features of EISs that have an impact on system evolution have been identified: (1) components may be fairly course-grained (complete ISs), (2) EISs are normally constructed with COTS components, (3) the supply of COTS components is limited, (4) the legacy IS constitutes the starting point of the system development effort, (5) components may not be modifiable, (6) components cannot be assumed to be constructed in a uniform fashion, (7) connectors are normally heterogeneous, and (8) the EIS may contain both data and functional redundancy.

The resulting problem domain consequently constitutes a design space that is *discrete* and *time* dependent in nature. This implies that as requirements dictated by the organization and its context must be delicately mitigated with the present (legacy) EIS and the availability of COTS components and connectors. Complex temporal dependencies occur, as all enumerated factors are mutually dependent on each other, but yet in a state of constant change.

As a part of this work, a process for scenario-based architectural analysis has been tested as a strategic decision support tool during a pre-study, preceding the acquisition of several new components to an utility's EIS. The analysis process was augmented with set of viewpoints, and in the final iteration of the process, the *architectural integration styles* were applied in order to make explicit principal designs and their connection to quality attributes. The findings imply that the concepts of architectural description, e.g. *quality attributes*, *architectural ontology*, *views*, and *scenarios*, combined with scenario-based architectural analysis, may successfully be utilized on the enterprise level of ISs. Four viewpoints were defined and employed as a part of the study in order to ensure some degree of completeness in the analyzed architectural description, in line with other architectural viewpoint models (Kruschten 1995; Maier & Rechtin 2000).

Here, applying the iterative, evolutionary, and interactive characteristics of a spiral process model for architectural analysis (Boehm 1988) were found to be highly effective in supporting the gradually increasing awareness among stakeholders. In the analysis of the proposed design alternatives, the conceptual modeling was found to effectively help top-management to mitigate risks and opportunities, and furthermore to help them make necessary trade-offs concerning identified quality attributes and business requirements. Furthermore the approach was found to

promote structured analysis of proposed design alternatives, as well as enhance stakeholder communication and awareness.

As often referred to as a success factor in ISs management (Standish group 2000), involvement of the top-management in the studied utility proved to be of vital importance for the process. In contrast to large organizations, it is often easier for management in small and medium-sized enterprises to grasp the total problem domain without loosing too much of the often important details. Thus, similar to Earl (1989), this thesis suggests that senior management involvement could be more valuable in the project level rather than on the strategy formulation level in these companies.

The chosen approach furthermore proved efficient to reduce the initial problem domain (that by the client organization was perceived as rather endless) into a relatively manageable design space. After the first iteration of the analysis process, four feasible design alternatives were condensed, based on constrains concerning business objectives, legacy considerations, and available COTS components and connectors. Although the volume of the analysis task was found to increase dramatically with the number of views, scenarios, time horizons, and design alternatives taken into consideration, the discrete nature of the design space contributed to keep down the number of components and thereby the effort required for the analysis. A lesson learned is also that the volume of the analysis task in itself served as a useful indicator for decisions as it provided a tangible measure of complexity affecting e.g. the selection of granularity of components, and make-versus-buy decisions.

Another consequence of an increased use of coarse-grained COTS components and connectors is that customer-supplier relationships are changing. Thus, user-organizations may have to influence their software suppliers *indirectly* (possibly in collaboration with other customers), e.g. through user-associations or standardization bodies. Also, third-party software has become significantly interesting for user-organizations, as its content and quality may have a significant impact on the long-term modifiability of EISs in particular. Consequently, user organizations that wish to influence software vendors' product lines must act proactively and with a planning horizon that may considerably exceed the one achievable within the frame of a single IS acquisition project.

Concisely, employing architectural description and analysis for decision support in EISM does not provide a cookbook for which decision to make in every situation, as because the context for each decision is unique. However, architectural

description and analysis provide an approach for conceptualizing the design space into conceivable lines of action, and to make explicit relevant trade-offs between qualities of the system and its context, in order to mitigate risks and opportunities. Also, architectural concepts, such as *views* and *styles*, may help to ensure that no important aspects are overlooked, and to promote reuse of previous experiences. It is also stressed that architectural description and analysis does not exclude other approaches for EISM. Conversely, the approach provides a framework in which other techniques for more detailed conceptualizing and analysis, of e.g. specific quality attributes may be employed.

9.2 FURTHER WORKS

As research processes tend to create more questions than answers, it may be concluded that an abundance of further work remains within the area of EISM. Contemplating this work in retrospective, there are several logic proceedings for research in this area. Firstly, as the techniques and methods applied have been adapted *during* the concluding field studies of this work, a full-scale test of the total approach for using architectural concepts for EISM in small and medium-sized electric utilities would further strengthen the results presented in this thesis. Secondly, this work focuses on evolution of EISs and thereby on *modifiability*. Other quality attributes have consequently been regarded *relatively* modifiability. A conceivable future research direction is therefore to execute a similar investigation focusing on other types of quality attributes that require alternative analysis approaches. Thirdly, the generalization of presented findings is strictly confined to the domain of small and medium-sized electric utilities. To increase the generalization of this work, large organization and/or enterprises in other business domains should also be taken into consideration. In addition to these suggestions for further research, it is stressed that work which aims to package experiences concerning EISM for more general reuse should be encouraged, e.g. by means of cataloguing architectural integration styles.

Chapter 10

References

Abd-Allah A.A. (1996), *Composing Heterogeneous Software Architectures*, Ph.D. Thesis, University of Southern California, August 1996.

Abowd G., Allen R., Garlan D. (1993), "Using Style to Understand Descriptions of Software Architecture," *ACM Software Engineering Notes*, pp. 9-20, December 1993.

Alexander C., Ishikawa S., Silverstein M., Jacobson M., Fiksdahl-King I., Angel S. (1977), *A Pattern Language*, Oxford University Press, 1977.

Allison G.T. (1971), *Essence of Decision: Explaining the Cuban Missile Crisis*, Boston: Little, Brown, 1971.

Andersson J. (1997a), "A Strategy for Migration on a Deregulated Energy Market – Case Study Experiences", In: *Proceedings of DA/DSM DistribuTech Europe 97*, Amsterdam, The Netherlands, October 1997.

Andersson J. (1997b), *On IT System Integration – Prospects and Consequences of Energy Market Deregulation*, Licentiate Thesis, Royal Institute of Technology, Stockholm, Sweden, 1997.

Andersson J., Cegrell T., Cheong K.H. Haglind M., Johansson E., Johansson L. (1998), "IT Strategy for Electric Utilities - From a Paper Tiger to an Effective Management Tool," In: *Proceedings of DA/DSM DistribuTech Europe 98*, London, U.K., October 1998.

Andersson J., Johnson P. (1999), "Procurement of Integrated IT Systems for the Deregulated Electric," In: *Proceedings of the International Conference on Electricity Distribution* (CIRED '99,), Nice, France, June 1999.

Andersson R., Nilsson A. (1996), *The standard application package market—an industry in transition?*, In: *Advancing Your Business: People and Information Systems in Concert* (Lundeberg M., Sundgren B. (eds.), EFI: Stockholm School of Economics, Sweden, 1996.

Argyis C., Schön D.(1978), *Organizational Learning: A Theory of Action Perspective*, Addison-Wesley, Reading, 1978.

Armour F., Kaisler S., Simon Y (1999), "A Big-Picture Look at Enterprise Architectures", *IT Professional*, pp. 35-42, January-February 1999.

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

Baragry J., Reed K. (1998), "Why is it so hard to define software architecture?," In: *Proceedings of Asia Pacific Software Engineering Conference*, pp. 28-36, 1998.

Basili V, Yakimovich, D., Bieman J., "Software Architecture Classification for Estimating the Cost of COTS Integration," In: *Proceedings of the 11th International Conference on Software Engineering, ICSE '94*, 1994.

Baskerville R.L. (1999), "Investigating Information Systems Research with Action Research," *Communications of the Association for Information Systems*, 2(19), October 1999.

Baskerville R.L., Wood-Harper A.T. (1996), "A Critical perspective on action research as a method for information systems research", *Journal of Information Technology*, 11(4), pp. 235-246, 1996.

Bass L., Clements P., Kazman R. (1998), *Software Architecture in Practice*, Addison-Wesley, 1998.

Bass L., Klein M., Bachmann F. (2000), *Quality Attribute Design Primitives*, Technical Note CMU/SEI-2000-TN-017, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, USA, December 2000.

Bengtsson P.O. (2002), *Architecture-Level Modifiability Analysis*, Doctoral Thesis, Blekinge Institute of Technology, 2002.

Bennet K. (1995), "Legacy Systems: Coping with success", *IEEE Software*, 12(1), pp. 19-23, January 1995.

Bennet K., Rajlich V. (2000), "Software Maintenance and Evolution: A Roadmap," In: Finkelstein A. (ed.), "The Future of Software Engineering," *Proceedings of the 22nd International Conference on Software Engineering*, ICSE 22, ACM Press, 2000.

Bergey J.K., Fissher M.J., Jones L.G., Kazman R. (1999), *Software Architecture Evaluation with ATAM in the DoD System Acqusition Context*, Technical Note CMU/SEI-99-TN-012, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, USA, September 1999.

Binns P., Vestal S. (1993), "Formal real-time architecture specification and analysis," In: *Proceedings of the 10th IEEE Workshop on Real-Time Operating Systems and Software,* May 1993.

Blanchard B.S. (1991), *System Engineering Management*, John Wiley & Sons, New York, 1991.

Bleicher J. (1982), *The Hermeneutic Imagination: Outline of a Positive Critique of Scientism and Sociology*, Routledge and Kegan Paul, London, 1982.

Boehm B.W., Brown J.R., Kaspar H., Lipow M., McLeod G., Merritt M. (1978), *Characteristics of Software Quality*, North-Holland Publishing Company, Amsterdam, 1978.

Booch G., Jacobson I., Rumbaugh J. (1999), *The Unified Modeling Language User Gudie*, Addison-Wesley, USA 1999.

Bosch J., Molin P. (1999), "Software Architecture Design: Evaluation and Transformation", In: *Proceedings of the IEEE Conference and Workshop on Engineering of Computer-Based Systems*, pp. 4-10, 1999.

Bosch J., van Gurp G. (2002), "Design Erosion: Problems & Causes", *Journal of Systems & Software*, 61(2), pp. 105-119, Elsevier, March 2002.

Braun C.L. (1999), "A lifecycle process for the effective reuse of commercial off-the-shelf (COTS) software", In: *Proceedings of the 5th symposium on Software reusability*, pp. 29-36, Los Angeles, USA, 1999.

Brodie M.L., Stonebraker M. (1995), *Migrating Legacy Systems: Gateways*, Interfaces, and the Incremental Approach, Morgan Kaufmann Publishers, 1995.

Brooks, F. P. (1995), *The Mythical Man-Month*, The 20th anniversary ed., Addison-Wesley, 1995.

Brown J.B., Malveau R.C., McCormick H.W., Mowbray T.J.(1998), *Anti Patterns: Refactoring Software, Architectures, and Projects in Crisis*, John Wiley & Sons, 1998.

Brownsword L., Place P. (2000), *Lessons Learned Applying commercial Off-the-Shelf Products*, Technical Note CMU/SEI-99-TN-015, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, USA, June 2000.

Buschmann F., Meunier R., Rohnert H., Sommerlad P., Stahl M (1996), *Pattern-Oriented Software Architecture: A system of Patterns*, John Wiley & Sons, 1996.

Carmel E. (1997), "American hegemony in packaged software trade and the culture of software", *The Information Society* 13(1), pp. 125-142, 1997.

Carmel E., Sawyer S. (1997), "Packages software Development Teams: What Makes Them Different?," *Inforamtion Technology and People*, 11(1), pp. 7-19, 1997.

Cegrell T. (1986), *Power System Control - Technology*, Prentice Hall International, U.K., 1986.

Cegrell T. (1997), "Integrated Information and Control Systems," In: *Proceedings of DA/DSM DistribuTech Europe 97, Amsterdam*, The Netherlands, October 1997.

Cegrell T. (1998), "Systems Specification and Requirements Engineering," Published in the technical report: *The ISES Project. Enersearch*, Malmö, Sweden, 1998.

Cegrell T., Andersson J., Cheong K-H., Haglind M., Johansson E., Johansson L. (1998), "IT strategy for electric utilities – from a paper tiger to an effective management tool", In: *Proceedings of DA/DSM DistribuTECH Europe '98 Conference*, London, U.K., October 1998.

Cegrell T., Sandberg U. (1994), *Industriella Styrsystem*, SIFU Förlag, Sweden, 1994.

Chappell D. (1996), U*nderstanding ActiveX and OLE: A Guide for Developers and Managers*, Microsoft Press, 1996.

Checkland P. (1986), *Systems Thinking, Systems Practice*, John Wiley & Sons, 1986.

Cheong K.H. (1997), *Distribution Automation: Cost-Effective Introduction Strategies*, Licentiate Thesis, Dept. of Industrial Control Systems, Royal Institute of Technology, Stockholm, Sweden, October 1997.

Cheong K.H. (1999), *IT strategy for Electric Utilities – A Framework towards Effectiveness*, Ph. D. Thesis, Dept. of Industrial Control Systems, Royal Institute of Technology, Stockholm, Sweden, October 1999.

Ciborra C.U., de Profundis (1997), "Deconstructing the Concept of Strategic Alignment," *Scandinavian Journal of Information Systems*, 9(1), 1997.

Clements P., Northrop L. (1996), *Software Architecture: An Executive Overview*, Technical Report, CMU/SEI-96-TR-003, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, USA, 1996.

Clements P., Parnas D., Weiss D. (1985) "The Modular Structure of Complex Systems," *IEEE Transactions on Software Engineering*, 11(1), pp. 259-266, 1985.

Coglianese L., Szymanski R. (1993), "DSSA-ADAGE: An Environment for Architecture-based Avionics Development," In: *Proceedings of AGARD93,* May 1993.

Davenport T. (1993), *Process Innovation: Reengineering Work through Information Technology*, Harward Business School Press, 1993.

Davenport T. (2000), *Mission Critical: Realizing the Promise of Enterprise Systems*, Harward Business School Press, Boston, USA.

Davis M.J., Williams R.B. (1997), "Software Architecture Characterization", In: *Proceedings of the 1997 Symposium on software reusability*, ACM SIGSOFT Software Engineering Notes, 22(3), pp. 30-38, Boston, USA, May 1997.

DeLine R. (1999), *Resolving Packaging Mismatch*, Ph.D. Thesis, Carnegie Mellon University, USA, 1999.

Dellarocas C. (1996), *A Coordination Perspective on Software Architecture: Towards a Design Handbook for Integrating Software Components*, Ph.D. Thesis, Massachusetts Institute of Technology Center for Coordination Science, 1996.

Dick B. (1999), *What is action research?,* Available on line at: http://www.scu.edu.au/schools/gcm/ar/whatisar.html, 1999.

Dick K. (2000), *XML: A Manager's Guide*, Addison-Wesley, 2000.

Dijkstra E.W. (1968), "The Structure of the 'T.H.E.' Multiprogramming System," *Communications of the ACM*, 11(5), pp. 453-457, 1968.

Dikel D.M., Kane D., Wilson J.R. (2001), *Software Architecture: Organizational Principles and Patterns*, Prentice Hall, New York, 2001.

DISA (1996), *Department of Defence Technical Architecture Framework for Information Management*, Vol 1-8, Defence Information Systems Agency, United States Department of Defence, Available on-line: http//www-library.itsi.disa.mil/tafim.html, 1996.

Doyle J., Thomason R. (1999), "Background to qualitative decision theory," *AI magazine,* 20(2), pp. 55-68, Summer 1999.

Earl M.J. (1989), *Management Strategies for Information Technology*, Prentice Hall, 1989.

Earl M.J. (1993), "Experiences in strategic information systems planning," *MIS Quarterly*, pp. 1-24, March 1993.

Eason K. (1988), *Information Technology and Organizational Change*, Taylor & Francis Publication, U.K., 1988.

Ecklund E.F., Delcambre L.M.L., Freiling M.J.(1996), "Change cases: use cases that identify future requirements," *In: Proceedings of the eleventh annual conference on Object-oriented programming systems, languages, and applications, ACM SIGPLAN Notices*, 31(10), pp. 342 - 358, October 1996.

Engelken L., Gay A., Tram H. (1999), "Development of an information technology strategy and architecture for energy delivery utility mergers", In: *Proceedings of IEEE Transmission and Distribution Conference*, 1999.

Ericsson G. (1996), On Communication in Power System Control, Ph. D. Thesis, Dept. of Industrial Control Systems, Royal Institute of Technology, Stockholm, Sweden, August 1996.

Ferguson J. (1996), *Software Acquisition Capability Maturity Model (SA-CMM) Version 1.0*1, Technical Report CMU/SEI-96-TR-020, Software Engineering Institute, Carnegie Mellon University, Pittsburgh 1996.

Finkelstein A. (ed.) (2000), "The Future of Software Engineering," In: *Proceedings of the 22nd International Conference on Software Engineering*, ICSE 22, ACM Press, 2000.

Fowler M. (1997), *Analysis Patterns: Reusable Object Models*, Addison-Wesley, 1997.

Frenzel C.W. (1996), *Management of Information Technology*, 2nd ed., CTI, Cambridge, 1996.

Gacek C. (1998), *Detecting Architectural Mismatch During System Composition*, Ph.D. Thesis, University of Southern California, 1998.

Galliers R.D, Land F.F (1987), "Choosing Appropriate Information Systems Research Methodologies", *Communications of the ACM*, 30(11), pp. 900-902, 1987.

Galliers R.D. (ed.) (1992), *Information Systems Research: Issues, methods, and practical guidelines*, Blackwell Scientific Publications, 1992.

Galliers R.D., Land F.F. (1988), "The Importance of Laboratory Experimentation in IS Research: A Response," *Communications of the ACM*, 31(12), pp. 1504-1505, 1988.

Gamma E., Helm R., Johnson R., Vlissides J. (1998), *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1998.

Garlan D, Allen D. (1994a), "Formalizing Architectural Connection," In: *Proceedings of the 16th International Conference on Software Engineering*, ICSE '94, Sorrento, Italy, 1994.

Garlan D. (2000), "Software Architecture: A Roadmap," In: Finkelstein A. (ed.), "The Future of Software Engineering," *Proceedings of the 22nd International Conference on Software Engineering*, ICSE 22, ACM Press, 2000.

Garlan D. Allen R (1997b), "A formal basis for architectural connection," *ACM Transactions on Software Engineering and Methodology,* July 1997.

Garlan D. Monroe R., Wile D. (1997a), "ACME: An Architecture Description Interchange Language," In: *Proceedings of CASCON'97*, 1997.

Garlan D., Allen R., Ockerbloom J. (1994a), "Exploiting style in architectural design environments," In: *Proceedings of SIGSOFT94: The second ACM SIGSOFT Symposium on the Foundations of Software Engineering,* pp. 170-185. ACM Press, December 1994.

Garlan D., Allen R., Ockerbloom J. (1994b), "Architectural Mismatch: Why Reuse is so Hard," *IEEE Software*, 1994.

Garlan D., Shaw M. (1996), *Software Architecture – Perspectives on an Emerging Discipline*, Prentice Hall, 1996.

Glass R.L. (1998), *Software Runaways: Lessons Learned from Massive Software Project Failures*, Prentice Hall, 1998.

Grand M. (2002), *Java Enterprise Design Patterns*, John Wiley & Sons, 2002.

Grudin J. (1991), *Interactive systems: Bridging the gap between developers and users*, IEEE Computer 24(5), pp. 59-69, 1991.

Häggander D.(2001), *Software Design Conflicts*, Ph.D. Thesis, Blekinge Institute of Technology, 2001.

Haglind M. (2002), *Information Systems Planning in the Deregulated Electric Power Industry: Addressing Small and Medium Sized Enterprises*, Ph. D. Thesis, Dept. of Industrial Information and Control Systems, Royal Institute of Technology, Stockholm, Sweden, February 2002.

Hammer M, Champy J (1993), *Reengineering the corporation: A manifesto for business revolution*, Harper Business, New York, 1994.

Heineman G.T., Councill W.T. (eds.) (2001), *Component-Based Software Engineering: Putting the Pieces Together*, Addison-Wesley, 2001.

Hendersen J.C., Venkatraman N. (1996), "Aligning Business and IT Strategies," In: Luftman J.N. (ed.), *Competing in the Information Age - Strategic Alignment in Practice*, Oxford University Press, 1996.

IEEE (2000), *IEEE 1471-2000: Recommended Practice for Architectural Description*, IEEE Standard, IEEE Architecture Working Group, 2000.

ISO/IEC 15288 (2001), *ISO/IEC 15288: Life Cycle Management: System Life Cycle Processes*, Final Draft International Standard (FDIS), 2001.

Jacobson I., Christerson M., Jonsson P., Övergaard G. (1992), *Object Oriented Software Engineering – A Use Case Driven Approach*, Addison-Wesley Publishing Company, USA, 1992.

Jacobson I., Rumbaugh J., Booch G. (1999), *Unified Software Development Process*, Addison-Wesley Publishing Company, USA, 1999.

Jaktman C.B., Leaney J., Liu M. (1999), "Structural Analysis of the Software Architecture: A Maintenance Assessment Case Study," In: *Proceedings of the 1st Working IFIP Conference on Software Architecture*, WICSA1, San Antonio, USA, February 1999.

Jarvenpaa S. (1988), "The Importance of Laboratory Experimentation in IS Research," *Communications of the ACM*, 31(12), pp. 1502-1504, 1988.

Johansson L., Andersson J., Bäcklund M., Daugulis A., Cheong K.H., Haglind M., Johansson E., Silwer M. (1997), "Case Study Research in Large System Engineering Projects," In: *Proceedings of DA/DSM Europe 97*, Amsterdam, the Netherlands, 1997.

Johnson P. (2002), *Enterprise Software System Integration: An Architectural Perspective*, Ph. D. Thesis, Dept. of Industrial Information and Control Systems, Royal Institute of Technology, Stockholm, Sweden, April 2002.

Kazman R., Barbacci M., Klein M., Jeromy Carriere S., Woods S.G (1999), "Experience with performing architecture tradeoff analysis," In: *Proceedings of the 1999 International Conference on Software Engineering*, pp. 54-63, Los Angeles, USA, May 1999.

Kazman R., Bass L. (1994), *Toward Deriving Software Architectures from Quality Attributes*, Technical Report CMU/SEI-94-TR-10, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, USA, 1994.

Kazman R., Klein M., Barbacci M., Longstaff T., Lipson H., Carriere J. (1998), "The Architecture Tradeoff Analysis Method," In: *Proceedings of the Fourth IEEE International Conference on Engineering of Complex Computer Systems*, pp. 68 –78, 1998.

Klepper P., Hartog C. (1992), "Trends in use and management of application package software," *Information Resources Management Journal*, 5(4), pp. 33-37, 1992.

Knoll K.K., Jarvenpaa S.L. (1994), "Information technology alignment or "fit" in highly turbulent environments," In: *Proceedings of the 1994 computer personnel research conference on Reinventing Information Systems*, Alexandria, USA, April 1994.

Kobryn C. (1998), "Modeling Enterprise Software Architectures Using UML," In: *Proceedings of Second International Workshop on Enterprise Distributed Object Computing*, EDOC '98, pp. 25-34, 1998.

Kohtala J., Vaattovaara M. (1998), "Experiences about computer applications for distribution management, In: *Proceedings of DA/DSM DistribuTECH Europe '98 Conference*, London, U.K., October 1998.

Kontio J. (1996), "A case study in applying a systematic method for COTS selection," In: *Proceedings of the 18th International Conference on Software Engineering*, IEEE Computer Society Press.

Kruchten P. (1995), "Architectural Blueprints: The "4+1" View Model of Software Architecture," *IEEE Software*, November, 1995.

Kuhn T. (1970), *The Structure of Scientific Revolution*, 2nd ed., University of Chicago Press, 1970.

Lane T.G. (1990), *A Design Space and Design Rules for User Interface Software Architecture*, Technical Report CMU/SEI-90-TR-22, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, USA, 1990.

Lassing N., Bengtsson P.O., van Vliet H., Bosch J. (2002), "Experiences with ALMA: Architecture –Level Modifiability Analysis," *The Journal of Systems and Software*, No. 61, pp. 47-57, 2002.

Lassing N., Rijsenrij D., van Vliet H. (1999), "Towards a Broader View on Software Architecture Analysis of Flexibility," In: *Proceedings of 6th Asia Pacific Software Engineering Conference*, pp. 238-245, 1999.

Lederer A.L., Gardiner V. (1992), "The process of strategic information planning", *Journal of Strategic Information Systems,* 1(2), pp. 76-83, 1992.

Lederer A.L., Salmela H. (1996), "Toward a Theory of Strategic Information Systems Planning," *Journal of Strategic Information Systems*, 5(3), pp. 237-253, 1996.

Lehman M.M. (1998), "Software's future: managing evolution," *IEEE Software* , 15(1), pp. 40-44, January-February 1998.

Levy, M., Powell P. (2000), "Information Systems Strategy for Small and Medium sized Enterprises: An Organizational Perspective", *Journal of Strategic Information Systems*, No. 9, pp. 63-84, 2000.

Linthicum D. (2000), *Enterprise Application Integration*, Addison-Wesley, 2000.

Luckham D.C., Kenney J.J., Augustin L.M., Vera J., Bryan D., Mann W. (1995), "Specification and analysis of system architecture using Rapide," *IEEE Transactions on Software Engineering*, 21(4), pp. 336-354, April 1995.

Luftman J.N. (ed.) (1996), *Competing in the Information Age - Strategic Alignment in Practice*, Oxford University Press, 1996.

Lutz J. (2000), "EAI Architecture Patterns," *EAI Journal*, March, 2000.

Magee J., Dulay N., Eisenbach S., Kramer J. (1995), "Specifying distributed software architectures," In: *Proceedings of the Fifth European Software Engineering Conference,* ESEC95, September 1995.

Magoulas T., Pessi K. (1998), *Strategisk IT management*, Ph.D. Thesis, Dept. for Informatics, University of Gothenburg, Gothenburg, Sweden, March 1998.

Maier M.W., Rechtin E.R. (2000), *The Art of Systems Architecting*, 2nd edition, CRC Press, 2000.

Mattson M. (2000), *Evolution and Composition of Object-Oriented Frameworks*, Doctoral Thesis, Blekinge Institute of Technology, 2000.

Mayer Sasson A. (1993): "Open Systems Procurement: A migration strategy," *IEEE Transactions on Power Systems*, 8(2), pp. 515-521, May 1993.

McCall J.A., Richards P.K., Walters G.F. (1977), *Factors in Software Quality*, Three volumes, US Rome Air Development Center Reports NTIS AD/A-049 014, 015, 055, November 1977.

Medvidovic N, Taylor R.N. (1998), "Separating fact from fiction in software architecture," In: *Proceedings of the 3rd international workshop on Software architecture*, pp. 105-108, Orlando, USA, November 1998.

Medvidovic N., Oreizy P., Robbins J.E., Taylor R.N. (1996) Using object-oriented typing to support architectural design in the C2 style. In: *SIGSOFT96: Proceedings of the 4 th ACM Symposium on the Foundations of Software Engineering,* ACM Press, October 1996.

Medvidovic N., Rosenblum D.S., Redmiles D.F., Robbins J.E. (2002), "Modeling software architectures in the Unified Modeling Language," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 11(1), pp. 2-57, January 2002

Meyers B.C., Oberndorf P (2001), *Managing Software Acquisition: Open Systems and COTS Products*, Addison-Wesley, 2001.

Mintzberg H., Ahlstrand B., Lampel J. (1998), *Strategy Safari – A Guided Tour Trough the Wilds of Strategic Management*, The Free Press, New York, 1998.

Monson-Haefel R. (2000), *Enterprise JavaBeans*, 2nd Ed., O'Reilly & Associates, 2000.

Morgenthal J.P. (2001), *Enterprise Application Integration with XML and Java*, Prentice Hall, 2001.

Moriconi M., Qian X., Riemenschneider R. (1995), "Correct architecture refinement," *IEEE Transactions on Software Engineering, Special Issue on Software Architecture,* 21(4), pp. 356-372, April 1995.

Myers M.D. (1997), "Qualitative Research in Information Systems," *MIS Quarterly*, 21(2), pp. 241-242., June 1997, Available on-line: http://www.misq.org/misqd961/isworld/.

Nadler D. and Tushman M.L. (1980), "A Congruence Model for Diagnosing Organizational Behavior," in Miles R., *Resource Book in Macro organizational Behavior*, pp. 30-49, Goodyear, Santa Clara 1980.

Ockerbloom J. (1998), *Mediating Among Diverse Data Formats*, Ph.D. Thesis, Carnegie Mellon University, 1998.

Orlikowski W., Baroudi J. (1991), "Studying information technology in organizations: research approaches and assumptions," *Information Systems Research*, 2(1), pp. 1-28, 1991.

Oskarsson Ö. (1982), *Mechanisms of Modifiability*, Ph.D. Thesis, Software Systems Research Center, Linköping University, Linköping, Sweden, May 1982.

Parnas D.L. (1972), "On the Criteria To Be Used in Decomposing Systems into Modules," *Communications of the ACM*, 15(12), pp. 1053-1058, 1972.

Parnas D.L. (1974), "On a 'Buzzword': Hierarchical Structure," In: *Proceedings of IFIP Congress 74*, pp 336-339, North Holland Publishing Company, 1974.

Parnas D.L. (1976), "On the Design and Development of Program Families", *IEEE Transactions on Software Engineering*, 2(1), pp. 1-9, 1976.

Parnas D.L. (1994), "Software Aging," In: *Proceedings of the 16th International Conference on Software Engineering*, ICSE '94, pp. 279-287, Sorrento, Italy, 1994.

Pellegrinelli S. (1997), "Programme management: organising project-based change," *International Journal of Project Management*, 15(3), pp. 141-149, June 1997.

Perry D.E., Wolf A.L. (1992), "Foundations for the study of Software Architecture," *Software Engineering Notes*, ACM Press 17(4), New York, pp. 40-52, 1992.

Persson M. (1998), *IT-användning i elbolag*, Utlandsrapport Sveriges Tekniska Attachéer, 1998.

Pressman R.S. (1997), *Software Engineering: A Practitioner's Approach*, 4th Ed., McGraw-Hill, USA, 1997.

Pritchard J. (1999), *COM and CORBA Side by Side: Architectures, Strategies, and Implementations*, Addison-Wesley, 1999.

Project Management Institute (2000), *A Guide to the Project Management Body of Knowledge* (PMBOK® Guide), Project Management Institute, USA, 2000.

Rahkonen T. (1996a), *Case Study Theory Applied in a Technical Research Project*, External Report, Ex.R. 96-02, Department of Industrial Control Systems, Royal Institute of Technology, Stockholm, Sweden, 1996.

Rahkonen T. (1996b), *User Strategies for Open Industrial IT Systems*, Ph.D. Thesis, Dept. of Industrial Control Systems, Royal Institute of Technology, Stockholm, Sweden, 1996.

Robson C. (1993), *Real World Research: A Resource for Social Scientists and Practitioner Researchers*, Blackwell Publishers Ltd, Oxford, 1993.

Ruh W., Maginnis F., Brown J. (2001), *Enterprise Application Integration: A Wiley Tech Brief*, John Wiley & Sons, 2001.

Sawyer S. (2000), "Packaged Software: Implications of the differences from custom approaches to software development," *European Journal of Information Systems*, no. 9, pp.47-58, 2000.

Schmidt D.C., Stal M., Rohnert H., Buschmann F. (2000), *Pattern-Oriented Software Architecture: Patterns for Concurrent and Networked Objects*, John Wiley & Sons, 2000.

Shaw M. (1989), "Larger scale systems require higher-level abstractions," In: *Proceedings of the 5th international workshop on Software specification and design*, ACM SIGSOFT Software Engineering Notes, 14(3), pp. 143-146, April 1989.

Shaw M., DeLine R., Klein D.V., Ross T.L., Young D.M., Zelesnick G. (1995), "Abstractions for software architecture and tools to support them," *IEEE Trans on Software Engineering,* 21(4), pp. 314-335, April 1995.

Simon H.A. (1997), *Administrative behaviour*, 4th edition, The Free Press, New York, 1997.

Sneed H. (1995), "Planning the re-engineering of legacy systems," *IEEE Software*, 12(1), pp.24-34, 1995.

Spivey J. (1992), *The Z Notation*, Prentice Hall, 1992.

Standish group (2000), *Perils in Gas & Electricity*, Available on-line: http://standishgroup.com/, 2000.

STEM (2000), *Electricity Market 2000*, Statens Energimyndighet (Swedish National Energy Administration), available at http://www.stem.se, 2000.

STEM (2001), *Electricity Market 2001*, Statens Energimyndighet (Swedish National Energy Administration), available at http://www.stem.se, 2001.

Störrle H. (1999), "Architectural Modeling with the Unified Modelling Language," In: *Proceedings of the 2nd Nordic Workshop on Software Architecture*, NOSA '99, Ronneby, August 1999.

Susman G. (1983), "Action Research: a sociotechnical systems perspective," In: Morgan G. (ed.), *Beyond Method: Strategies for Social Research*, pp. 95-113, Sage, Newbury Park, 1983.

Sutcliffe A.G., Maiden, N.A.M., Minocha S., Manuel D. (1998), "Supporting scenario-based requirements engineering," *IEEE Transactions on Software Engineering*, 24(12), pp. 1072-1088, December 1998.

SvK (1996), *Svenska kraftnäts balanstjänst, Handbok, andra reviderade utgåvan*, Svenska Kraftnät, 1997.

SvK (1997), *Avräkningshandbok för elbranschen: Kapitel 1 och 2, Utgåva p1A*, Svenska Kraftnät, 1997.

SvK (2001), *The Swedish Electricity Market and the Role of Svenska Kraftnät*, SvK - Svenska Kraftnät (Swedish Grid), available at http://www.svk.se, 2002.

The Open Group (1999), *The Open Group Architectural Framework*, Available on-line: http://www.opengroup.org/togaf/, 1999.

Thomas A. (1998), *Selecting Enterprise JavaBeans Technology*, Patricia Seybold Group, 1998.

Thomason R.H. (1998), "Qualitative Decision Theory and Interactive Problem Solving (extended abstract)," In: Haddawy P., and Hanks S. (eds.), *Working Notes of the American Association for Artificial Intelligence Spring Symposium on Interactive and Mixed Initiative Decision-Theoretic Systems*, pp. 197-113, Menlo Park, USA, 1998.

Thorp J. (1998), *The Information Paradox: Realizing the Business Benefits of Information Technology*, McGraw-Hill Publication, 1998.

Wallnau K.C., Hissam S.A., Seacord R.C. (2002), *Building Systems from Commercial Components*, Addison-Wesley, 2002.

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

Ward, Griffiths, *Strategic Planning for Information Systems*, 2nd ed., John Wiley & Sons, U.K., 1996.

Witt B.I., Baker T.F., Merritt E.W. (1994), *Software Architecture and Design – Principles, Models, and Methods*, Van Nostrand Reinhold, 1994.

Yin R. (1994), *Case Study Research: Design and Methods*, 2nd Ed., SAGE Publications, Thousand Oaks, California, USA 1994.

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