1 Requirements Engineering: Setting the Context

Aybüke Aurum and Claes Wohlin

Abstract: This chapter presents a brief overview of requirements engineering and provides an introduction to some of the critical aspects in the field. This includes offering and understanding of the different levels of requirements involved in requirements engineering; namely organizational, product and project level requirements, and illustrating the role of different stakeholders in requirements engineering. The chapter also aims to demonstrate how the three parts of this book are interrelated.

Keywords: Requirements management, business requirements, product requirements, project requirements, stakeholders, requirements taxonomy

1.1 Introduction

The objective of this chapter is twofold. First, it aims to provide a brief introduction to requirements engineering and secondly it aims to set a common context for the other chapters of the book. This introductory chapter is provided to set the stage for the remaining chapters and nightlight to some of the important areas covered by this book. The remaining chapters require a basic understanding of requirements engineering to benefit from the deeper insights provided in each of the other chapters. These chapters are divided into three parts, each with a different focus as shown in the table of Contents and described briefly in the Preface.

Requirements engineering is accepted as one of the most crucial stages in software design and development as it addresses the critical problem of designing the right software for the customer. Requirements engineering is increasingly becoming a set of processes that operates on different levels, including organizational, product and project levels. Furthermore, it is a continuous process on organizational and product levels and limited process in time on the project level. However, most requirements engineering research to date is devoted to handling requirements on the project level, making this the main focus of this chapter. The different levels are revisited in Section 1.4. Requirements engineering on the project level is the process by which the requirements for a software project are gathered, documented and managed throughout the software development lifecycle.

The development of a software requirements specification is widely recognized as the bases of system functionality. Software requirements are the critical determinants of software quality, given empirical studies showing that errors in requirements are the most numerous in the software life-cycle and also the most expensive and time-consuming to correct. According to the Standish group report in 1995 [10] 52.7% of projects cost (named as challenged projects), 189% of their original budget estimates, and only a disappointing 42% of the original features of challenged projects were implemented. The study demonstrates that 16.1% of all US software projects are developed on-schedule, on-budget and with all originally planned features, while 31.1% of projects are terminated before completion. It was also observed that the average project is delivered at approximately three times the budget and in three times the scheduled time.

Such poor figures lead to questioning the causes of these deficiencies. Often these problems are a result of inadequate requirements [25]. According to a survey conducted with 350 organizations in the USA (with over 8000 projects), one third of the projects were never completed and one half succeeded only partially. About half of the managers that were interviewed identified poor requirements as a major source of problems, along with other factors such as low user involvement and unclear objectives. Similarly, according to another survey which was conducted with 3800 organizations over 17 countries in Europe, most problems are in the area of requirements specifications (50%) and requirements management (50%) [18]. In 1999, the Standish group report [11] revealed three of the top ten reasons for "challenged" projects and project failure were lack of user involvement, unstable requirements and poor project management. In a 2001 report, while user involvement was no longer a key concern, unstable requirements and poor project management remained amongst the primary reasons for project failure [12].

In a more recent survey of twelve UK companies requirements problems accounted for 48% of all software problems [20]. In one of the case studies, Tveito and Hasvold [38] observed that there was a huge gap between the day to day operations of a hospital and software developers' domain knowledge of these operations, though every year healthcare organizations spend large amounts of money and resource on IT systems. Tveito and Hasvold argue that this gap is due to insufficient requirements gathering and misunderstanding requirements due to the lack of domain knowledge.

These facts and figures only depict the sad reality of "software depression". Furthermore, the cost of repairing requirements-related problems dramatically increases as the software development process progresses. A study by Boehm and Papaccio [6] revealed that it costs US$1 to locate and fix an error in the requirements definition stage, US$5 in the design phase, US$10in the coding phase, $20US during unit testing, and up to $200 US after system delivery. It is therefore evident that the RE process has important ramifications for the overall success of a software project. Although the above example dates back just over 15 years, the ratio remains the same today.

Requirements engineering is concerned with the identification of goals to be achieved by a proposed system, the operation and conversion of these goals into services and constraints, as well as the assignment of responsibilities for the resulting requirements to agents such as humans, devices and software. Requirements engineering has now moved from being the first phase in the software development lifecycle to a key activity that spans across the entire software development lifecycle in many organizations. New products or new releases of products are entering the market or delivered to customers at an increasingly fast pace. In order to improve requirements engineering processes, current practices in the real world need to be examined. Understanding and modeling current requirements engineering processes is an important step towards improving requirements engineering practices and therefore increasing the success of software projects [31].

Researchers agree that the requirements engineering process should consist of structured and repeatable activities where both engineering and management aspects are properly handled [39]. Unfortunately, there is a lack of agreement regarding the appropriate requirements engineering process models to use across different industries, as the selection of available models spans from activity-based process models to decision-oriented paradigms, each with their own subset of model structures.

The objective of this chapter is to provide the context in which the other chapters of this book operate. As briefly mentioned above, this context includes an understanding of the different process levels involved in requirements engineering. Moreover, the different stakeholders and their respective roles in requirements engineering must be understood. The activities involved in the processes are presented at a high level providing the reader insight into the work being performed as part of requirements engineering. This chapter provides a brief introduction to some fundamental building blocks of requirements engineering to allow the reader reap the full benefit and a clear understanding of the other chapters.

The chapter is outlined as follows. Section 1.2 provides an introductory background to the area of requirements engineering. This is followed by a brief discussion about the roles of stakeholders in Section 1.3. In Section 1.4, different levels of requirements are presented. The management of requirements is discussed in Section 1.5 and Section 1.6 explores the future of the area. Finally, empirical evidence is touched upon in section 1.7 and some conclusions are presented in Section 1.8.

1.2 Background

This objective of this section is to present background information on requirements engineering.

1.2.1 What is a Requirement?

All projects begin with a statement of requirements. Requirements are descriptions of how a software product should perform. A *requirement* typically refers to some aspect of a new or enhanced product or service. The widely cited IEEE 610.12-1990 standard [24] defines a requirement as:

(1) "A condition or capability needed by a user to solve a problem or achieve an objective,

(2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents,

A documented representation of a condition or capability as in (1) or (2)".

Therefore, requirements include not only user needs but also those arising from general organizational, government and industry standards. Clearly, a requirement is a collection of needs arising from the user and various other stakeholders (general organization, community, government bodies and industry standards), all of which must be met. Ideally, requirements are independent of design, showing "what" the system should do, rather than "how" it should be done. However, this is not always possible in practice That is, the meanings of "what" and "how" differ from person to person [15].

Requirements can be classified in many ways, as illustrated in Table 1.1. While the literature draws a distinction between different types of requirements, in practice it is not always easy to identify such differences [3]. For example, a user requirement concerned with security may be classified as a non-functional requirement. However, during implementation other requirements may evolve, which are distinguishably functional such as user authorization [37]. More examples on this issue can be found in Chapter 6.

Table 1.1 Types of Requirements

Requirements Classification
• *Functional requirements* -- what the system will do • *Non-functional requirements* -- constraints on the types of solutions that will meet the functional requirements e.g. accuracy, performance, security and modifiability
• *Goal level requirements* -- related to business goals, • *Domain level requirements* -- related to problem area, • *Product level requirements* -- related to the product, • *Design level requirements* -- what to build
• *Primary requirements* -- elicited from stakeholders • *Derived requirements* -- derived from primary requirements
Others classifications e.g. • *Business requirements* versus. *technical requirements*, • *Product requirements* versus. *process requirements* -- i.e. business needs versus how people will interact with the system • *Role based requirements* e.g. client requirements, user requirements, IT requirements and security requirements

Having understood the basics of what constitutes a requirement, the next step is to elaborate on the process used to manage and engineer requirements.

1.2.2 Requirements Engineering Process

Requirements engineering refers to all life-cycle activities related to requirements. This includes mainly gathering, documenting and managing requirements. With the growing awareness of the significance of requirements in the software process, requirements engineering increasingly becomes an area of focus in software engineering research.

Common requirements engineering activities are elicitation, interpretation and structuring (analysis and documentation), negotiation, verification and validation, change management and requirements tracing. There are several process models available to describe the requirements engineering process. The requirements engineering process is often depicted in different forms, including a linear model, incremental model, non-linear model and spiral models. Kotonya and Sommerville [25] suggest a conceptual linear requirements engineering process model, which indicates iterations between activities. On the other hand, Macaulay [30] provides a purely linear requirements engineering process model that does not indicate the overlapping or iteration of activities, suggested by the Kotonya and Sommerville [25] model. While some researchers tend to portray the requirements engineering process as a linear model, non-linear models have also been suggested. Loucopoulos and Karakostas [27] depict the requirements engineering process as iterative and cyclical in nature. Alternatively, the spiral model represents a sequence of activities being performed in iterations, resulting in gradual progression requirements engineering model [5]. However, it has implications on the requirements engineering process model. A spiral approach would require requirements to be handled in each round in the spiral model, which is similar to the ideas presented by Kotonya and Sommerville [25]. They provide a second requirements engineering process model, which depicts the same requirements engineering activities as in their linear model only occurring in a spiral representation. The activities from the linear process model are repeated in iterations, forming a spiral. At the end of each iteration a decision is made as to whether to accept the requirements document or to perform a further iteration.

Results from studies of the requirements engineering processes in practice have indicated that the systematic and incremental requirements engineering models presented in literature may not necessarily reflect the requirements engineering processes in current practice. Martin et al (2002), who examined the requirements engineering process in a case by case study, found that generally projects were handled by following a linear model, with some iteration of activities. Most of the projects they examined followed a generally linear process until the prototyping phase, which then resulted in an iterative process. Martin et al., [32] indicated that the Loucopoulos and Karakostas [27] model was a good representation of the ad hoc process and the iterative nature of prototyping, but did not show the progression of phases. On the other hand, Nguyen and Swatman [35] found that the requirements engineering process in their case study did not occur in a systematic, smooth and incremental way, rather it was opportunistic, with sporadic simplification and restructuring of the requirements model when it reached points of high complexity. Furthermore, Houdek and Pohl [22] performed a case study in the field but could not produce a monolithic requirements engineering process model of requirements engineering activities, as they were too heavily intertwined and not seen as separate tasks by the participants of the study

Requirements engineering field studies have also gathered conflicting results as to the status of requirements engineering process standards in organizations. This indicates that the area is not fully matured in the sense that there is not one standard process that is universally used and accepted. Instead several different requirements engineering processes have been presented. Kotonya and Sommerville [25] put forward that not many organizations have a standard requirements engineering process definition. Consistent with this, Hofmann and Lehner [21] examined requirements engineering processes of 15 requirements engineering teams in industry and found that most participants saw the requirements engineering as ad hoc, with only some of the projects using an explicitly defined requirements engineering process or customizing a company wide requirements engineering process standard for the project. Furthermore, studies of requirements engineering in web development projects have further confirmed the ad hoc nature of requirements engineering [28]. In contrast to these findings, El Emam and Madhavji [17] concluded that organizations tend to use standard requirements engineering processes, as they are viewed as best practices. Chatzoglou [13] used a three-phased mail-out survey to examine the requirements engineering process in 64 projects to understand the differences between projects with different characteristics. Particular focus was placed on human resources. The main conclusions were that a standard process methodology should be used but should also be tailored to the specific needs of each project. Furthermore resources should be put into the initial iteration of the requirements engineering process.

Since requirements engineering processes are fundamental to the success of software projects, it is therefore no surprise that improving the requirements engineering process can subsequently enhance the chances of developing successful software. Prior to devising strategies for software process improvement, research and analysis of present requirements engineering processes must be undertaken to provide a solid grasp of current requirements engineering practices.

1.3 The Role of Stakeholders in RE

In essence, requirements engineering aims to transform potentially incomplete, inconsistent and conflicting stakeholder goals into a complete set of high quality requirements. Information systems researchers define stakeholders "*…as these participants in the development process together with any other individuals, groups or organizations whose actions can influence or be influenced by the development and use of the system whether directly or indirectly*" [36]. Typical stakeholders are the product managers, various types of users and administrators from the client side, and the software team members from the software development side. This view is somewhat limiting when considering software development for markets. The traditional view of software development, and requirements engineering, is that of bespoke software development. This is the situation when software is developed with a specific customer in mind and when it is often possible to have direct contact with this one user/customer. This situation becomes different when developing software for a market or a set of customers, in particular if all customers are not known at the time of development. This has led to studies of market-driven software development, where one important issue is to identify and handle the different stakeholders under these situations. More information on market driven requirements can be found in Chapter 13.

As software projects are becoming increasingly complex, software developers face the challenge of identifying the goals of stakeholders who come from a diverse range of backgrounds. It might be also very difficult to represent the essential requirements of software in a way which is accessible to all stakeholders, as software effectively is invisible [9]. The importance of stakeholder involvement in requirements engineering activities is widely accepted given that accurate identification of stakeholder needs largely determines the quality of the software product.

One of the major problems in requirements engineering is the management of different types of inconsistencies resulting from requirements elicitation, modeling, specification, and prioritization activities. Inconsistencies become particularly apparent when having different stakeholders and viewpoints, since different stakeholders have different ways of expressing themselves and different opinions as well as priorities. Although some researchers point out that inconsistencies between requirements models may be desirable, as they allow further elicitation (in capturing requirements models) and they recommend tolerating some internal inconsistencies during requirements modeling [23, 33], the success of requirements engineering projects depends on accurate analysis of these perspectives for incompleteness and inconsistencies. Therefore, requirements need to be negotiated and validated before they are documented and developers commit to implementing them.

1.4 Different Levels of Requirements

Effective management of the software product development process contributes to sustainable competitive advantage for software companies. This implies that managers need to consider customers' requirements, and business requirements, as well as the technological opportunities which may be distinct or overlap. It is important to stay on budget, reduce life cycle time and achieve product performance goals, to ensure that the software requirements are aligned with business goals. These challenges are not unique to software development and are in fact typical of complex system products. In the age of the Internet there have been significant changes in business environments creating more complex demands on the technologies that support business information systems.

Consequently understanding, analyzing, modeling and managing requirements have become equally complex. In order to deliver high quality software systems on time and on budget, it is essential to have properly structured and controlled requirements specifications that are understandable, comprehensive and consistent.

Table-1.2: Requirements classification in three levels

	Strategic Management	**Tactical Management**	**Operational Management**
Requirements at organizational level	*business strategy *competitiveness *technology * marketing *economic value of the product	* planned benefits of the product	* tradeoff between technology-push and market-pull
Requirements at product level	* packaging requirements for a specific release * product architectures	* resource management *implementation of a specific release	*change management * accommodating syntactic or semantic changes * requirements volatility
Requirements at project level	*project planning *feasibility study	* project management	*validation in terms of which requirements will go to the next release

The requirements engineering process is one of the main contributors to the success of software projects. This is particularly true in a global competitive market where time-to-market and meeting stakeholder requirements are key success factors. Thus, improving the requirements engineering process can significantly increase the likelihood of software project success. According to Edwards et al., [16] contemporary software design approaches often mix business issues with IT implementation issues to form monolithic systems that are no more responsive to change than their predecessors. IT systems in this industry would therefore need to be dynamic and quickly adaptable to their environments. The current expanded perspective of software products in business has various implications for managing software development processes, i.e. software requirements should not be solely handled in software projects. Based on Anthony's [1] three level managerial decision making model, namely strategic, tactical and operational decisions, Aurum and Wohlin [2] illustrate how to conduct an analysis of the requirements engineering process and its underlying decision-making processes using classical decision making frameworks. In this book, we adopts a similar view, i.e. that the management of software requirements is subject to organization-oriented, product-oriented and process-oriented activities and they need to be managed at strategic, tactical and operational levels. Table 1.2

illustrates classification of software requirements in 3*3 matrixes where each cell includes few examples of requirements activities or decisions. The three levels can be briefly described as follows.

a) **Requirements at the organizational level:** The senior management team of an organization may have strategic objectives and long-term goals in terms of market share and so forth. The goals and strategies at the organizational level will inevitably influence which products that an organization ought to develop. Thus, requirements posed on products must first be evaluated on an organizational level to ensure that the requirements are aligned with the goals and strategies of the organization. One of the main challenges faced when successfully developing software products is that of determining how the end product will support business objectives.

b) **Requirements at the product level.** The requirements of software products must be aligned with the business goals of the software development organization. One of the crucial questions is how to balance customers' concerns with developers' concerns. Goal modeling techniques in requirements engineering serve as a mechanism by which one can link requirements to strategic objectives anchored in the context of the overall business strategy model. The requirements are typically both functional and non-functional requirements. Product management has to ensure that the requirements are aligned with the goals and objectives in terms of the product. This may mean selecting the requirements for the product that are best aligned with the overall goals and strategies of the organization.

c) **Requirements at the project level.** Requirements on the product level must be packaged into parts that go into specific projects or releases of the software. It is important that requirements are prioritized and selected based on their fulfillment of both product and organizational goals and strategies. Requirements may be chosen for implementation based on whether they fulfill the needs of a specific and important customer, or whether they potentially open up a new market segment to the organization. These requirements define the conditions under which the project will be run, including issues related to project planning, risk management, budget and cost**.**

The growth in strategic importance of IT implies that tools, techniques and processes need to be integrated with software system requirements so that they are aligned with the strategic business objectives and business model of the organizations they support. Business change is a part of system development. As systems become more integrated and involve more users from diverse backgrounds, software developers are pressured to understand the implications of their decisions in relation to cost/benefit analysis, particularly during early life cycle activities [8, 19, 26]. System engineering and management literature, in particular risk management literature, stress the importance of project planning effort, schedule planning, cost planning, and risk assessment in product development as being essential in the generation of products that meet customer requirements and align with strategic business goals.

1.5 Requirements Management

The quality of a software product is largely determined by the quality of the development process used to create it. Many projects fail due to mistakes in the elucidation of requirements, while others fail because of the requirements have become outdated by the time the project is delivered [9]. It is also a major challenge to developers to determine which requirements changes will cause a major problem in the project or the product itself [9]. Managing requirements engineering phases is crucial to the successful development of software products. In order to deliver high quality software systems on time and on budget it is essential to have properly structured and controlled requirements specifications that are understandable, comprehensive and consistent.

As mentioned above, it is important to have a good understanding of stakeholder goals and ensure their involvement in the requirements engineering process. The management of requirements involves establishing a shared understanding between the stakeholders and the requirements they have specified for inclusion in the software product. The essential practices of requirements managements are:

- **Requirements elicitation, specification and modeling:** This involves understanding the needs of stakeholders, eliciting requirements, modeling and collecting them in a repository. This is an important stage in software development however, for a variety of reasons, including cognitive, communicative and motivational reasons, the requirements tend to be incomplete and inconsistent. Therefore, there is always room for improvement in these activities.
- **Prioritization:** It is not always easy for developers to decide which requirements are important to customers. This activity assists project managers with resolving conflicts (where customers and developers collaborate on requirements prioritization), plan for staged deliveries, and make necessary trade-off decisions.
- **Requirements dependencies and impact analysis:** It is important to acknowledge that requirements change and that this may significantly impact the software project [14]. Several issues such as recording decisions, understanding the effect of business changes and the use of domain models are yet to be addressed [29].
- **Requirements negotiation:** Requirements engineering is essentially a complex communication and negotiation process involving customers, designers, project managers and maintainers. The people, or stakeholders, involved in the process are responsible for deciding what to do, when to do it, what information is needed, and what tools need to be used [25]. In many situations conflict is inherent in requirements, thus they need to be negotiated between stakeholders. Some tools, such as Win-Win Groupware, have been developed to support stakeholders throughout the negotiation process [7]. The requirements negotiation activity is one of the most crucial activities in software

development as it has a great impact on the final product. In reality, this activity is carried out in parallel with the activities mentioned above and continues until the requirements are implemented. Further information on negotiation can be found in Chapter 7.

- **Quality assurance:** The objective is to ensure that high quality requirements are recorded in the specification document. The purpose of quality assurance is to establish reasonable and realistic levels of confidence when writing and managing requirements. It is important that both customers and developers are involved in quality assurance activities in requirements engineering as they influence the success of project. It is important to stress that quality assurance of requirements is not only an activity in the requirements phase in projects. Quality assurance must be addressed throughout the software life cycle. Requirements should be traced throughout development and the quality assured, for example, through inspections, reviews and testing.

1.6 New Trends and the Next Practice

The technological improvements in the global market are closely related to business environments. New concepts such as enterprise systems, e-business and telecommunications have led to new trends in research for researchers and practitioners. Furthermore, the complexity of working in a distributed and heterogeneous environment is causing profound changes in the skills needed and the technology used to develop and maintain software applications. In this ever-changing business and technology environment, new trends have started emerging and have caused fundamental shifts in software development. In a similar fashion, requirements engineering has begun to evolve from its traditional role, as a mere front-end in the software development life cycle, towards becoming a key focus in the software development process; a process that requires a more precise understanding of the field itself. Today, the definition of what the software development life cycle constitutes is expanding and evolving as new technologies emerge, forcing software developers to scramble to position themselves in a rapidly changing business environment [34].

The requirements engineering process is a decision-rich complex problem solving activity. Decision making and managing the phases of requirements engineering is becoming increasingly crucial to the successful development of software products. The complexity of the activities involved in the requirements engineering process call for the need for organizations to coordinate the decision-making process and increase visibility of the decisions and the roles played with respect to decision-making in requirements engineering more visible. In order to support the requirements engineering process, a better understanding of activities involved in the process itself as well as an appreciation for the decisions made throughout these activities is necessary [2]. In other words, software developers need to have a better understanding of the range of decisions made at the organizational, product and project levels to ensure effective management of the requirements engineering process.

Software developers need a better understanding of what it takes to generate adequate management support and stakeholders' participation in the requirements engineering process. The effective management of the requirements engineering process mandates procedures and tools to support the phases of the requirements engineering process model and also takes into account other issues e.g. social, political and cultural issues. There is a strong need for decision support throughout software development at the organizational, project and product levels. As new software developments approaches are emerging, such as agile methods, trends in business and technology force requirements engineering to expand its role in the software development life cycle.

1.7 Empirical Evidence

Empirical research aims to capture quantitative evidence and compares theory to reality, helping us to draw conclusions and to evaluate new methods and tools. Empirical research is important to the requirements engineering field because the results of such studies both help to characterize the potential problems (regarding requirements at the business, product and project levels) with which the field is concerned and evaluate new techniques in a relevant context. Empirical research provides valuable insight into aspects of requirements engineering. Furthermore, both academics and software practitioners need supporting evidence from case studies, field studies and experiments before adopting new technologies. Collecting empirical evidence from industry is often time consuming and can become very complicated. However, this is necessary to quantify and demonstrate their relative merits to the requirements engineering community.

Depending on the purpose of the evaluation, whether it is techniques, methods or tools, and depending on the conditions for the empirical investigation, the three most common types of quantitative investigations (strategies) are:

- Experiment [40]: Experiments are often highly controlled and hence also occasionally referred to as controlled experiment and often run in a laboratory setting. When experimenting, subjects are assigned to different treatments at random.
- Case study [41]: The case study is normally conducted studying a real project. Case studies are used for monitoring projects, activities or assignments. Data is collected for a specific purpose throughout the study.
- Survey [4]: A survey is often an investigation performed in retrospect, when e.g. a tool or technique, has been in use for some time. The primary means of gathering qualitative or quantitative data are interviews or questionnaires.

1.8 Conclusion

This chapter has two key contributions: (a) from a theoretical point of view, it provides a brief introduction to the area of requirements engineering, and (b) form a practical point of view, it aims to provide the reader with guidelines to some important aspects of requirements engineering that are needed to obtain the full benefit of the other chapters of this book.

There are three parts in this book. Part 1 contains "state-of-the-art" chapters that address the key requirements engineering activities mentioned in Section 1.5, namely requirements elicitation, specification and modeling, prioritization, requirements dependencies, impact analysis, requirements negotiation and quality assurance issues. Part 2 is intended to address new trends in requirements engineering and pinpoints advantages and pitfalls of these trends. Finally, Part 3 contains chapters focusing on empirical evidence from academic research as well industrial case studies.

Reference

1. Anthony RN (1965) Planning and control systems: a framework for analysis. Harvard University, Boston, USA
2. Aurum A, Wohlin C (2003) The fundamental nature of requirements engineering activities as decision making process. Journal on Information and Software Technology, 45(14): 945-954
3. Berry DM, Lawrence B (1998) Requirements engineering. IEEE Software 25(2): 26-29
4. Babbie E (1990) Survey research methods. Wadsworth, ISBN 0-524-12672-3
5. Boehm BW, (1988) A spiral model of software development and enhancement, Computer, May, 21(5): 61-72
6. Boehm, BW, Papaccio, PN (1988) Understanding and controlling software costs. IEEE transactions on software engineering, 14 (10): 1462-1477
7. Boehm BW, Grünbacher P, Brigges RO. (2001) Developing groupware for requirements negotiation: lessons learned. IEEE, Software, May/June, pp. 46-55
8. Boehm, BW, (2003) Value-based software engineering. ACM SIGSOFT, Software engineering notes, March, 28(2): 1-12
9. BSC'04 (2004) The challenges of complex IT projects. The report of a working group from the Royal academy of engineering and the British computer society, ISBN 1-903496-15-2. Access on 20th October 2004. http://www.bcs.org/BCS/News/PositionsAndResponses/Positions/complexity.htm
10. Chaos'94 (1995) The Standish group. Access on 4th October 2004. http://standishgroup.com/sample_research/
11. Chaos'98 (1999) A recipe for success. The Standish group report. Accessed on 4th October 2004 http://www.standishgroup.com/sample_research
12. Chao'01 (2002) Extreme chaos. The Standish group report. Accessed on 4th October 2004 http://www.standishgroup.com/sample_research

13. Chatzoglou PD (1997) Factors affecting completion of the requirements capture stage of projects with different characteristics. Information and Software Technology, 39 (9): 627-640
14. Curtis B, Krasner H, Iscoe N, (1988) A field study of the software design process for large systems. Communications of the ACM 31(11):1268-1287
15. Davis, A. (1990) System testing: Implications of requirements specifications. Information and Software Technology, 32 (6): 407-414
16. Edwards J, Coutts I, McLeod S (2000) Support for system evolution through separating business and technology issues in a banking system. In: Proceedings of international conference on software Maintenance, 11-14 October, pp. 271 -276
17. El Emam K, Madhavji NH (1995) A field study of requirements engineering practices in information systems development. In: Proceedings of 2nd international symposium on requirements engineering, York, England, IEEE CS Press, pp.68-80
18. European Software Institute (1996) European user survey analysis. Report USV_EUR 2.1, ESPITI project, January
19. Faulk SR, Harmon RR, Raffo DM (2000) Value-base software engineering: A value-driven approach to product-line engineering. In: Proceedings of 1st international conference on software product-line engineering, Colorado, August 28, 2000
20. Hall T, Beecham S, Rainer A (2002) Requirements problems in twelve companies: an empirical analysis. IEE proceedings software, 149 (5): 153-160
21. Hofmann HF, Lehner F, (2001) Requirements engineering as a success factor in software projects. IEEE Software, 18 (4): 58-66
22. Houdek F, Pohl K, (2000): Analyzing requirements engineering processes: a case study. In: Proceedings of the 11th international workshop on database and expert systems applications, Greenwich, UK, 6-8 September, pp.983-987.
23. Hunter A, Nuseibeh B, (1997) Analyzing inconsistent specifications. In: Proceedings of 3rd international symposium on requirements engineering, RE'07, Annapolis, Md, pp.78-86
24. IEEE-STD 610.12-1990 - Standard Glossary of Software Engineering Terminology, 1990, Institute of Electrical and Electronics Engineers
25. Kotonya G, Sommerville I, (1998) Requirements engineering – processes and techniques, John Wiley & Sons UK
26. Lauesen, S (2002) Software requirements - styles and techniques, Addison-Wesley, London, UK
27. Loucopoulos P, Karakostas V, (1995): System requirements engineering. McGraw-Hill Book company Europe
28. Lowe D, Eklund J, (2001) Development issues in specification of web systems. In Proceedings of 6th Australian workshop on requirements engineering, 22-23 November, University of New South Wales, Sydney, Australia , pp. 4-13
29. Lubars M, Potts C, Richter C (1993) A review of the state of the practice in requirements modelling. In: Proceedings of the IEEE international symposium on requirements engineering, IEEE Computer Society, San Diego, USA, pp. 2-14
30. Macaulay LA (1996) Requirements engineering. Springer-Verlag, New York, London
31. Madhavji NH, Holtje D, Hong W, Bruckhaus T (1994) Elicit: a method for eliciting process models. In: Proceedings of CAS conference, Toronto, Canada, 31 October – 3 November, pp.11-122

32. Martin S, Aurum A, Jeffery R, Paech B (2002) Requirements engineering process models in practice. In: Proceedings of 7th Australian workshop on requirements engineering, AWRE'02, 2-3 December, Melbourne, pp. 41-47
33. Menzies T, Easterbrook S, Nuseibeh B, Waugh S (1999) An empirical investigation of multiple viewpoint reasoning in requirements engineering. In: Proceedings of IEEE international symposium on requirements engineering, 7-11 June, pp.100 – 109
34. Miller E (2002) For survival, start thinking lifecycle management. Computer-aided engineering, 21 (1): 15-18
35. Nguyen L, Swatman P (2003) Managing the requirements engineering process. Requirements engineering, 8 (1): 55-68
36. Pouloudi A, Whitley EA (1997) Stakeholder identification in inter-organizational systems: Gaining insights for drug use management systems. European journal of information systems, 6: 1-14
37. Sommerville I (2001) Software engineering. Pearson Education Ltd, UK
38. Tveito A, Hasvold P (2002) Requirements in the medical domain: Experiences and prescriptions. IEEE Software, Nov-Dec, pp.66-69
39. van Lamsweerde A., (2000) Requirements engineering in the year 00: a research perspective. In: Proceedings of 22nd International conference on software engineering, pp.5-19
40. Wohlin C, Runeson P, Höst M, Ohlsson MC, Regnell B, Wesslén A (2000) Experimentation in software engineering – An introduction. Kluwer Academic Publishers, Boston, MA, USA
41. Yin RK (1994) Case study research design and methods. Sage Publications, Beverly Hills, California, USA

Author Biography

Aybüke Aurum is a senior lecturer at the School of Information Systems, Technology and Management, University of New South Wales. She received her BSc and MSc in geological engineering, and MEngSc and PhD in computer science. She is the founder and group leader of the requirements engineering Research Group (ReqEng) at the University of New South Wales. She also works as a visiting researcher in National ICT, Australia (NICTA). Dr. Aurum is the editor of "Managing Software Engineering Knowledge" book. Her research interests include management of software development process, software inspection, requirements engineering, decision making and knowledge management in software development. She is on the editorial boards of Requirements Engineering Journal and Asian Academy Journal of Management.

Claes Wohlin is a professor in software engineering at the School of Engineering at Blekinge Institute of Technology in Sweden. He is also pro vice chancellor of the institute. Prior to this, he has held professor chairs in software engineering at Lund University and Linköping University. He has a M.Sc. in Electrical Engineering and a Ph.D. in Communication Systems both from Lund University, and he has five years of industrial experience. Dr. Wohlin is co-editor-in-chief of the journal of Information and Software Technology published by Elsevier. He is on the editorial boards of Empirical Software Engineering: An International Journal, and Software Quality Journal. Dr. Wohlin received the Telenor Nordic research Prize in 2004 for his achievements in software engineering and improvement of software reliability in telecommunications.