Prioritization of Stakeholder Needs in Software Engineering

Understanding and Evaluation

Patrik Berander
This thesis is submitted to the Faculty of Technology at Blekinge Institute of Technology, in partial fulfillment of the requirements for the degree of Licentiate of Technology in Software Engineering.

Contact Information:
Patrik Berander
Department of Systems and Software Engineering
School of Engineering
Blekinge Institute of Technology
PO Box 520
SE-372 25 Ronneby
SWEDEN

E-mail: patrik.berander@bth.se
Web: http://www.ipd.bth.se/pba
Abstract

In everyday life, humans confront situations where different decisions have to be made. Such decisions can be non-trivial even though they often are relatively simple, such as which bus to take or which flavor of a soft drink to buy. When facing decisions of more complex nature, and when more is at stake, they tend to get much harder. It is often possible to deal with such decisions by prioritizing different alternatives to find the most suitable one.

In software engineering, decision-makers are often confronted with situations where complex decisions have to be made, and where the concept of prioritization can be utilized. Traditionally in software engineering, discussions about prioritization have focused on the software product. However, when defining or improving software processes, complex decisions also have to be made. In fact, software products and software processes have many characteristics in common which invite thoughts about using prioritization when developing and evolving software processes as well. The results presented in this thesis indicate that it is possible to share results and knowledge regarding prioritization between the two areas.

In this thesis, the area of prioritization of software products is investigated in detail and a number of studies where prioritizations are performed in both process and product settings are presented. It is shown that it is possible to use prioritization techniques commonly used in product development also when prioritizing improvement issues in a software company. It is also shown that priorities between stakeholders of a software process sometimes differ, just as they do when developing software products. The thesis also presents an experiment where different prioritization techniques are evaluated with regard to ease of use, time consumption, and accuracy. Finally, an investigation of the suitability of students as subjects when evaluating prioritization techniques is presented.
Acknowledgements

First of all, I would like to express my gratitude to my supervisor, Professor Claes Wohlin, for giving me this opportunity, being supportive, and asking all the necessary questions. I would also like to thank my secondary supervisor, Dr. Mikael Svahnberg, for enjoyable discussions about research ideas and for valuable comments on the thesis.

Colleagues at Blekinge Institute of Technology also deserve thanks, not at least the people in the SERL group and in the BESQ project. In particular, I want to thank Per Jönsson for reviewing and discussing papers and this thesis, Lars-Ola Damm for reviews of the early papers during my research and for enjoyable cooperation in courses etc., and Anna Wikström-Ask for being a helping hand during the work with Chapter 6. I would also like to thank Lena Karlsson and Dr. Björn Regnell (Lund University), and Professor Anneliese Andrews (Washington State University) for fruitful cooperation in papers. Thanks also to Daniel Karström for giving comments on the early versions of Chapter 5.

I would also like to take the opportunity to thank all the subjects that have been part of the studies, both from academia and from Ericsson, for giving me results to work with. Ericsson, as my industrial research partner, has given input for studies and provided the necessary material to conduct industrial research studies. In particular, I want to thank Anna Eriksson, Dan-Magnus Svensson, Johan Häggman and Lars Angelin.

Family and friends have been neglected too many times the last years when days have become nights, and nights have become weekends. Especially, my mother deserves thanks for supporting me through the years, and for making me a cover illustration for this thesis. Thanks Malin for supporting me and still being the captain of this voyage, despite my too long working hours. Last but not least, my father, who passed away too early, inspired me (and still do), showed interest, and supported me in what I was doing.

This work was partly funded by the Knowledge Foundation in Sweden under a research grant for the project “Blekinge - Engineering Software Qualities (BESQ)” (http://www.bth.se/besq).
# Table of Contents

## Chapter 1:

### Introduction

1.1 Software Processes and Software Products .......................................................... 1
1.2 Research Methodology ....................................................................................... 3
1.3 Work Process and Outline .................................................................................. 14
1.4 Contribution ....................................................................................................... 20
1.5 Future Work ..................................................................................................... 30
1.6 Summary ......................................................................................................... 34
1.7 References ....................................................................................................... 39

## Chapter 2:

### Requirements Prioritization

2.1 Aspects of Prioritization ..................................................................................... 49
2.2 Prioritization Techniques .................................................................................... 52
2.3 Stakeholders in the Prioritization Process ......................................................... 55
2.4 Using Requirements Prioritization ...................................................................... 60
2.5 An Example of a Requirements Prioritization .................................................... 64
2.6 Summary ......................................................................................................... 74
2.7 References ....................................................................................................... 75

## Chapter 3:

### Identification of Key Factors in Software Process Management – A Case Study

3.1 Method ............................................................................................................ 81
3.2 Result .............................................................................................................. 82
3.3 Combined Analysis .......................................................................................... 86
3.4 Discussion ....................................................................................................... 93
3.5 Summary ....................................................................................................... 97
3.6 References .................................................................................................... 100

## Chapter 4:

### Differences in Views between Development Roles in Software Process Improvement

- A Quantitative Comparison .................................................................................... 105

4.1 Method ........................................................................................................... 107
4.2 Result ............................................................................................................. 112
4.3 Analysis ......................................................................................................... 119
4.4 Discussion ..................................................................................................... 122
4.5 Summary ..................................................................................................... 124
4.6 References ................................................................................................... 125
Chapter 1

Introduction

A researcher is not ‘one who knows the right answers’ but ‘one who is struggling to find out what the right questions might be’.

Estelle M. Phillips and Derek S. Pugh 1994 [68]

In our everyday life, we make decisions about many different things, e.g. when buying a DVD-player, food, a telephone, etc. Often, we are not even conscious of making one. Usually, we do not have more than a couple of choices to consider, such as which brand of mustard to buy, or whether to take this bus or the next one. Even with just a couple of choices, decisions can be difficult to make. When having tens, hundreds or even thousands of alternatives, decision-making becomes much more difficult.

One of the keys to making the right decision is to prioritize between different alternatives. It is often not obvious which choice is better, because several aspects must be taken into consideration. For example, when buying a new car, it is rather easy to make a choice when only considering speed (one only needs to evaluate which car is the fastest). When considering multiple aspects, such as price, safety, comfort, and luggage load, the choice gets much harder. When developing software processes or software products, similar tradeoffs must be made. The functionality that is most important for the customers might not be as important when the cost is high, and a process improvement issue might not be as urgent when the risk associated with it is very high. The most desirable things to develop are those that are most wanted by the stakeholders, least risky, least costly, and so forth. Prioritization helps to cope with all these complex decision problems.
In this thesis, the area of requirements prioritization in software engineering is investigated. Traditionally in software engineering, requirements prioritization concerns prioritizing requirements for a software product. In this thesis, the topic of requirements prioritization is widened and prioritization in relation to software processes is also discussed. In fact, software product development and software process development are rather closely related, as discussed further in Section 1.1. Since prioritization of requirements (regardless of application area) in software engineering traditionally refers to the area of requirements engineering, an active choice was made to use the key word *needs* instead of *requirements* in the title of this thesis. However, the term *requirement* is used more frequently than *need* in this thesis and it should be remembered that this refers to requirements on both processes and products if nothing else is stated.

As stated above, this thesis discusses requirements prioritization taking both process and product development into consideration. In this chapter similarities between the two areas are discussed and some discussions are made of how to take advantage of these similarities. In the subsequent chapters, different research studies are presented, all with their own focus and contributions. However, everyone of these five remaining chapters discusses prioritization in one way or another. Two of these chapters present studies where prioritization has been used in process development and two chapters’ present studies where prioritizations have been evaluated in a product setting. Finally, one chapter presents a literature overview of the current body of knowledge when it comes to prioritization in a product setting.

The remainder of this chapter is structured as follows. In Section 1.1, the activities in software process development and software product development are outlined, and similarities between the two areas are identified. Section 1.2 prepares the reader with the relevant information about the research methodology used in this thesis, and discusses issues to consider when performing research. Section 1.3 describes in which settings the work in this thesis has been conducted and how the different chapters relate to each other. Next, each chapter of the thesis is presented and the contribution of the thesis is discussed in Section 1.4. Section 1.5 outlines possible future work while Section 1.6 summarizes the chapter.
Introduction

1.1 Software Processes and Software Products

Traditionally, when discussing needs and requirements in software engineering, these are generally related to the software product. Nevertheless, needs and requirements are also present in the development of software processes. Already in 1987, Osterweil noted that software processes and software products are similar [63]. Even though people agree with this simile, the similarities between the two are not often discussed and the two are seldom discussed together. One sign of this is that both the requirements engineering and the software process communities have their own conferences, journals and workshops. This thesis tries to align these two areas in general and the matter of prioritization of the needs and requirements on the products and processes in particular.

The remainder of this section focuses on describing how processes (Section 1.1.1) and products (Section 1.1.2) are developed. During this presentation, the two are considered separately and no similarities between the two are pointed out. However, it is probably possible to recognize some similarities when reading the sections. In Section 1.1.3, some reflections are made that point out similarities between process and product development. References to where such similarities have been pointed out and some reflections of similarities when working with requirements are presented. The aim is to introduce the areas at the same time as identifying similar characteristics between process and product development.

1.1.1 Software Processes

Process oriented thinking is something that is natural to mankind since human beings generally want to replace chaos with order [22]. A process could be defined in a variety of ways (several examples can be found in [85]) but one commonly accepted definition is: “A process is what people do, using procedures, methods, tools, and equipment, to transform raw material (inputs) into a product (output) that is of value to customers” [66]. The pieces of information that people generally want to extract from a process are [18]:

- what is going to be done
- who is going to do it
- when and where it will be done
- how and why it will be done
- who is dependent on that it is being done
This way, processes provide a structure and communication channel as help for carrying out the work. They further provide a good foundation for creating an organization that shares the knowledge between the co-workers, which makes the organization less dependent on specific individuals and minimizes the impact of staff turnover [25]. Further, processes help the staff to see the overall direction for the business and the employees get an enhanced understanding for their own roles as well as for others in the organization [77]. Another purpose of processes is to provide a disciplined environment for the work through controlling problems by providing an early understanding of what is wrong, at the time when fixing the problem is most effective [37].

In software engineering, there exist a number of different process models (e.g. eXtreme Programming [3], Rational Unified Process [51]). There also exist quality standards and models that describe what to have (or improve) in an organization (e.g. CMMI [1] and The TickIT Guide [39]). Both these kinds of models and standards have their strengths and weaknesses and they commonly focus on providing an ideal software process or model suitable for many companies. However, many organizations need to define and improve their ways of working [63]. The need for standards focusing on how to define and improve processes within an organization has been recognized [77]. Much literature discusses how to define and how to improve software processes, all with diverse views of how to do it (e.g. [23][55][58]). Nevertheless, some key steps can be identified and the condensed description below is based on these key steps.

1.1.1.1 Choose Process and Plan the Work

The first step is to decide what process(es) should be defined or improved by identifying, prioritizing and choosing among opportunities and problems [55]. The next step is to select a team to drive the work (e.g. sponsors, champions, and change agents) and train them in the activities that they are going to perform [20][35][37][55]. The roles involved in this work have a great responsibility and it is important to choose the right persons for these roles. It is very important to involve all stakeholders (e.g. users and managers) from the beginning and the dedicated persons in the process group must be able to motivate and inspire the stakeholders to get them involved [4].
1.1.1.2 Requirements on the Software Process

When developing software processes within an organization, it is important to describe the right activities, roles, tools etc. in the process. The requirements on the process could be captured in two main ways; descriptively (how the process is currently performed) and prescriptively (how the process should be performed). The descriptive way is also commonly denoted as baselining and is a key activity when starting a process initiative, while the prescriptive way is used when improving an already defined process [5][18][38]. Eliciting information about a process is based on trying to get users’ views of a process by artifact and document analysis, interviews, observations, etc. [5]. Many constraints apply when trying to elicit the requirements of a process, e.g. limited availability of personnel, distributed knowledge, different work languages, collaboration problems, personnel cannot articulate or do not know how the work is performed, etc.[5][24][48].

1.1.1.3 Documenting the Process

An agreed upon and commonly shared process is a major factor of development effectiveness [17] and documenting is important if deviations shall be avoided and if the described process shall be repeated continuously. Several different languages for documenting the process exist (e.g. GRAPPLER, STATEMATE) and the purposes with such are [18]:

- They facilitate human understanding and communication of the process.
- They support process management and continuous process improvement.
- They facilitate automated guidance and execution of resulting process descriptions.

It is important to note that documenting a process is not enough; it must also be followed [66]. The documentation shall be easy to read, and not too formalistic but rather useful and necessary (e.g. [4][16][72]). No matter how good the process elicitation is, people do not consider using it if they do not understand the documentation or if it is too extensive. Furthermore, the level of detail of a process description should be suited to the user and the complexity of the task. A process description could be very flexible when the level of detail is right, but it is still desirable to have different levels of detail described to satisfy the needs of all users [16][18][39][67].
Next, the description should be published, communicated and the users should be trained to use it in the most efficient way [37].

1.1.1.4 Verification and Validation of the Process

When a process is documented, the description must be verified and validated so that it really reflects the users’ requirements on the process description. Personal interpretations by the documenters might, and probably will, cause that the process description does not reflect the elicited process after all. Therefore, a review of the process description must be made. This is preferably made by the future users of the process description (the ones that are supposed to work according to it) [4]. If the reviews shall be efficient, the users only review the parts of the process where they are involved since this will make the reviewers more focused and interested in the result [4]. An additional way of evaluating and verifying processes are to use models like CMMI [1] to determine the quality of a process [64]. The philosophy behind this is that the process is evaluated with the requirements of such models in mind, implying that the process is “tested” by the model. In models of this kind, questionnaires are often used to assess the process, and this questionnaire could then be seen as a test plan [64]. This could be a good way of getting a view from outside the organization, preventing the organization from being too focused on its own environment.

1.1.1.5 Process Maintenance and Process Improvement

One of the most important things to consider in process management is to have a change management process encouraging employees to initiate changes [16][36][72]. Without changes, the processes will not be followed since the fact that as more experience and knowledge is gained, the work methods will change. This way of changing processes could be seen as maintaining the current process by performing small adjustments for suitability and for correctness. However, sometimes more extensive changes have to be made and then an extensive process improvement initiative might be more suitable. The intention with software process improvement is to learn from mistakes and continuously improve. A software process improvement must be accepted throughout the organization and the organizational culture may also need to change [38]. It must be acceptable to fail and fire-fighting rewards should be eliminated and replaced with rewards based on long-term improvements [22][35]. It is also necessary to estimate how much today’s process shortcomings cost (e.g. blown schedules, high product support costs, unsatisfied customers, and personnel turnover [35][56]) and compare these figures to the expected gain of the improvement.
Since Chapters 3 and 4 of this thesis focus on the area of software process improvement, a short introduction of the steps involved in software process improvement is presented below. The launch of a software process improvement initiative is based on that a baseline is defined and the above mentioned activities are performed. Beside these activities, software process improvement includes the following steps:

**Identify Problems with the Baselined Process:** To prevent that the improvements just become (another) shelf ware, it is important to improve those areas that yield the greatest long-term benefit, add most value to the business, make the users’ work easier, and where the most problems are encountered [62][81][83]. Users of the process must be asked about their problems and their solutions to the problems since they know their problems very well and are the best source for finding solutions [37].

**Set Priorities and Goals:** Changes should be made in small steps to keep the organization from being disoriented, and they must be tested and adjusted before widespread implementation [18][35]. By prioritizing several aspects (e.g. severity of problem, cost of fixing), focus could be put on the most pressing needs for the organization [37]. Furthermore, by letting the future users be a part when setting goals, it is more likely that clear and meaningful goals are stated, which increase commitment and acceptance [37][80].

**Develop a Plan to Accomplish the Goals:** The next step is to plan by making risk analysis, evaluate opportunities and problems, specify critical success factors and metrics to measure achievement, attach schedules and resources, etc. [38]. Only one or a few new aspects should be tried at the same time, but besides this/these, standards and regulations must be followed as usual [22]. More extensive process improvements should be tried in pilot projects [35][77]. Future users should be involved, since they will understand and know what to expect from the change [37].

**Implementation of the Improved Process:** If not implementing the improved process and follow the new description, nothing will really change [83]. This step basically includes the same activities as presented in Sections 1.1.1.2 to 1.1.1.4 and the problems with the established baseline, and the solutions of such problems, should be elicited, documented, and verified.
Measure the Result of the Improvement: After implementation, the change should be evaluated by comparing measures from the baseline with measures of the changed work. This is done to evaluate if the change was an improvement or just a change. It takes time to learn new work methods and incorporate them into the organization and it is important to know that even if the short-term results are shaking, the long-term results might be excellent [82]. It is also important to investigate if the process is followed; if not, the measures might be non-accurate [5].

Redo the Cycle: “Process improvement is a journey, not a destination” [83]. When the first improvement cycle is finished, it is time to start planning the next one. Japanese industry has been successful for a long time by seeking opportunities for improvements rather than waiting for problems revealing opportunities [56]. Seeking such opportunities could be conducted either on a periodic basis (e.g. once a month), or as a post-mortem evaluating what worked well and what worked less well in a project [37].

1.1.2 Software Products

A product could be defined as “Anything that can be offered to a market for attention, acquisition, use or consumption that might satisfy a want or need. It includes physical objects, services, persons, places, organizations and ideas” [49]. As can be understood by the definition, the concept of products could refer to both commodities, and services. In addition, products could be a mixture of the two since a product rarely is a pure commodity or a pure service [10][49]. There are well documented differences between the two but the discussions in this thesis do not distinguish between them [53]. When discussing software products, it is common that these are both commodities and services, or a mixture of the two [32]. For example, the software of an ATM machine is a commodity when sold to a bank but offered as a service when the bank customers use the ATM software. The core of the above definition of products is that it should be an individually marketable entity [73]. Hence, to digest the above definition in relation to software products, these products can be defined as: A software product is an individually marketable entity that is the result of a software development process used to create it.

When developing a software product, a variety of different customer situations exist. One of two extremes is that there is only one customer available and this customer orders a bespoke system. The
other extreme is that a system is developed with everyone in the whole world as potential customers (market-driven). A list where the differences between such development efforts are highlighted could be found in [13], and issues related to requirements prioritization can be found in Chapter 2. However, these two situations are the extremes and there exist many situations in between. For example, in the telecommunication industry, it is common that there exist a number of different operators that serve as potential customers. Hence, the supplier does not develop bespoke software, at the same time as they do not develop consumer software on a mass market.

Except for the customer situation, the way of developing software and the organizational structure also depend on other factors. Such factors could for example be organizational goals, company and project size, type of work, or domain [59][67]. Hence, the way of developing software differs very much between different organizations, and it is not possible to find one single way of developing software [2]. The products can for example be developed iteratively, incrementally, evolutionary or in a waterfall manner and activities can be performed in a variety of ways and at different stages in the project [76]. Nevertheless, it is possible to identify fundamental steps common to all different ways of developing software, and a condensed description of these steps are presented below.

### 1.1.2.1 Finding a Business Opportunity and Planning the Work

The first step of a product development initiative is to have something to develop and a business opportunity. In a bespoke situation, this can start with a customer coming to an organization with an idea [27]. In a market-driven situation, new market opportunities can be generated in a variety of ways (e.g. competitor analysis, customer analysis, active search, and brainstorming) [49][53]. When having a business opportunity, the planning for the product development is started. In this early stage (often referred to as feasibility study [76]), key roles (e.g. product manager, project manager, and chief architect) of the project are identified and assigned, and additional resources are estimated. The key roles (together with suitable personnel), analyze the feasibility of the continuation of the project [76].
1.1.2.2 Requirements Engineering

The next step is to collect and define the requirements of the product to understand what customers and users want the system to do [67]. This is a key step when developing a successful product [15][26][67][76] since the cost of change and rectifying a wrong decision increases dramatically during the development cycle [11]. The way of dealing with requirements differ between different kinds of projects and products [2][67] but all known process models involves requirements engineering in one way or another.

In the first activity, requirements must be elicited through interviews, observations, focus groups, etc. [50][52][75]. Here, it is common that stakeholders cannot express what they need, have conflicting viewpoints, specify a solution instead of a demand, have too many requirements considering the time and cost constraints, their demands change over time, and so forth [52]. Thus, the requirements should be analyzed to make sure that the “right” requirements have been elicited. Next, the requirements should be prioritized (see Chapter 2 for more information) and negotiated [50]. When doing this, the aim is to find a set of requirements that the stakeholders are satisfied with [50][76]. The requirements should then be translated into a document that defines the set of requirements [76]. Last, the requirements should be validated to ensure that the requirements have been interpreted correctly [50].

1.1.2.3 Design and Implementation

The next step is to translate the requirements into an executable software system [76]. The first activity in this step is generally to transfer the requirements into a solution, and the description of this solution is called design [67]. The design can be modeled in several notations (e.g. UML, Booch), in many different steps, and on many levels (e.g. architectural design, interface design, component design [76]). The natural continuation is to implement the designed requirements into a system. When implementing the requirements, there exist numerous different languages (e.g. C++, Java), tools (e.g. Visual Studio, JBuilder), and there is no general process to follow but the activities can be performed in many different ways [67][76]. Most software is developed by many people and much cooperation and coordination is necessary. Thus, it is very important that others understand what and why different programmers have done certain things, and how it fits to their work [67].
1.1.2.4 Verification and Validation of the Product

The next step is to make sure that the system conforms to the specification and that the system meets the expectations of the stakeholders [76]. Software systems are very complex in nature both in terms of states and of the number of people involved, and users are not always sure of what they really need. Hence, faults exist not only because of wrongly implemented software, but also because user expectations are not fulfilled [67]. Software systems are built up by several sub-systems that contain modules with procedures and functions, and these could contain many different types of faults (e.g. algorithm faults, performance faults) [67]. Thus, it is necessary to perform the testing activities in stages and in conjunction with the implementation of the system [76]. There exist many different test activities that can be performed in different steps of the testing process, examples of such are unit testing and integration testing [67][76].

1.1.2.5 Maintenance of the Product

Software systems are built to be flexible and it is due to this flexibility software is used in large complex systems; but the flexibility also commonly causes changes in the software [67][76]. Three different types of software maintenance exist to handle such changes: repair faults, adapt for different operating environments, and add or modify the functionality [76]. The reasons for changes differ; it could for example be that the users like to do something in a different way, or that the nature of the system itself changes [67]. Studies have shown that between 50 and 75 percent of the total development effort is spent on maintenance activities, and it is usually cost-effective to invest effort for reducing maintenance costs, i.e. creating systems that are prepared for changes [67][76]. However, these figures are rather old, and are based on bespoke systems [76]. For market driven systems, less focus is put on maintenance since changed functionality is taken care of in new releases of the product [13]. Hence, the figures above are probably not relevant for such systems. In the market-driven situation, maintenance is substituted with development projects and starts over with the steps described in Sections 1.1.2.1 or 1.1.2.2.

1.1.3 Requirements on Products and Processes

As can be seen in the previous sections, there are some similarities between software products and software processes. As stated in Section 1.1, this was recognized as early as 1987 by Osterweil in the paper entitled “Software Processes are Software Too” [63]. In that
paper, it is outlined how a software process can be considered as a software product and even be implemented as such. Even though processes seldom are implemented in the detail discussed by Osterweil (e.g. programming languages with striking similarities to the programming languages of conventional products), the descriptions above (Sections 1.1.1 and 1.1.2) reveals that there still are large similarities between steps conducted when developing a product or process.

Both when developing processes and products, the initiative starts with finding an opportunity and assigning personnel. This is followed by a requirements phase where user requirements are found and outlined. This is continued with a phase where the found requirements are transferred into a representation of the system (i.e. a description, or design and code). Further, this representation is validated to assure that it conforms to the user requirements, and verified and validated to assure these requirements are represented in a correct way. The last phase in both cases is that the needs of both processes and products changes/evolve and this is handled by maintaining and improving the result.

In the follow-up paper of the 1987 paper, written by Osterweil in 1997, he continues to discuss the similarities by the following quote: "Processes and applications are both executed, they both address requirements that need to be understood, both benefit from being modelled by a variety of sorts of models, both must evolve guided by measurement, and so on. Thus it seemed important to suggest that software process technology might not need to be invented from scratch (or reinvented), but that much of it might be borrowed from application software technology." [64].

Even though the way of developing processes and products is similar during the whole development cycle, only one part of the development is covered in this thesis. The part that is discussed in more detail is the part where the process requirements, process improvement issues, or product requirements are elicited and specified (Sections 1.1.1.2 and 1.1.2.2). Osterweil [64] also discusses this area in detail in his second paper on the subject. In this paper, he states that there has been a lack of interest in studying software process requirements and that no parallels have been demonstrated in the research literature while other areas (e.g. coding and evaluation) have received an intense interest in supporting the activities [64]. Today, however, some interest about the requirements phase of software process development have been demonstrated (see Sec-
1. Introduction

The chapter discusses the similarities between requirements of software processes and software products. Even though it might not be as apparent as in other parts of process development, there are many similarities, such as:

- Both functional and non-functional requirements exist [64].
- More careful requirements engineering will result in greater acceptance, less rework, and hence less costs [5][75].
- An owner (e.g., process champion or product manager) that initiate and lead the development should preferably be involved [20][53].
- Dependencies between the different requirements exist [31][50].
- Future users are important to involve [15][77].
- Priorities should be assigned [37][75].

Except for the similarities in the development processes earlier outlined, the list above reveals that there exist many similarities between requirements of software processes and software products. Actually, a presentation of the requirements process as made of the whole development lifecycle (Section 1.1.2) would also result in a process similar to process development (i.e., requirements are elicited, analyzed, prioritized, documented, validated, and they evolve). However, as the similarities between products and processes have been pointed out elsewhere, it was decided to keep with this parallel in this thesis as well. It should also be noted that even though there are similarities between processes and products, the two are not the same. There are probably more differences than similarities between the two (e.g., customer situation), even though the focus here is put on identifying and discussing similarities.

As can be seen in the list above, the last bullet brings up the issue of prioritization as a similarity between product and process development. This thesis focuses on prioritization and attempts to show how these similarities could be used in an efficient way. In this thesis, different cases are presented where prioritization is used in both a process and a product context. Four different studies are presented where prioritizations have been used and two of these focuses on prioritization of process improvement issues and two focuses on prioritization of product requirements.
In the studies presented in this thesis, it is hence shown that prioritization could be used in both areas as a way of finding the right things to develop into a process description or into a product. Furthermore, since it is possible to use prioritization in both areas, it seems inevitable to exchange experiences between the two areas when it comes to requirements prioritization. In such an experience exchange, both areas would receive benefit from sharing the knowledge, i.e. it would be a win-win situation. Today, most research that is performed in the area of prioritization in software engineering is focused on prioritization of product requirements. However, as are shown in this thesis, it is possible to use knowledge from research about prioritization of product requirements in the process area as well. It is also shown that experiences from prioritizations of process improvement issues also successfully could be used when discussing prioritization of product requirements.

The above discussion is further elaborated in the remainder of this thesis, and Section 1.4 gives a more detailed discussion about the contributions. Furthermore, this chapter is followed by five different chapters that ultimately provide the contribution of the thesis. These chapters discuss studies where prioritization techniques have been used in different settings. However, before a further elaboration of these discussions are presented, some information about research methodology relevant for this thesis, about the research settings, and how the work has evolved is presented in the subsequent sections.

### 1.2 Research Methodology

A commonly accepted definition of research and development (R&D) is: "creative work undertaken on a systematic basis in order to increase the stock of knowledge, including knowledge of man, culture and society, and use this stock of knowledge to devise new applications" [61]. There are three different types of activities in relation to this definition: basic research, applied research, and experimental development. While basic research concerns knowledge acquisition without having a particular application in mind, applied research is directed towards a practical aim or objective. Experimental development uses this acquired knowledge from the above mentioned types of research to produce new, and improve already existing, materials, processes, etc. [61].
In 1994, Glass discussed the “software-research crisis” and concluded that the main problems in software engineering are related to that the researchers in software engineering studied problems not relevant for industry [30]. The reason for this was that research conducted was more close to basic research (since no practical aim or objective existed) than applied research and that research issues were not originated or evaluated in industry. The conclusion of this is that applied research in software engineering should be conducted empirically together with industry in settings where ideas can be tried out truly evaluative [30].

When conducting research together with industry, one of the challenges is to be able to say something sensible about a complex, relatively poorly controlled situation that is generally rather “messy” [70]. In industry research, emphasis is on the substantive or practical importance rather than merely on statistically significant findings, and much could be gained from transferring a study to a real environment [70]. This kind of research also implies that the researcher must be more a generalist than a specialist, have well developed social skills, be able to use several different research methods, etc. [70]. However, it still is possible to conduct both laboratory studies (with or without industrial professionals) and field studies (in a “real” environment) [70].

The remainder of this section focuses on describing research methods suitable for the above described way of performing research (empirical research with industrial relevance). The purpose is to provide the reader with the necessary information (in relation to the content of this thesis) about empirical research and research methods and what considerations that should be done in order to obtain reliable results. Empirical research methods, how to collect the data, and how to validate the data are discussed in the subsequent sections. In the first subsection (Section 1.2.1) different empirical research methods are presented. Next, a presentation of qualitative and quantitative data collection is done in Section 1.2.2. Finally, a discussion of what should be considered when performing a study with respect to validity issues is presented in Section 1.2.3.

### 1.2.1 Empirical Research Methods

When conducting empirical studies, a number of different approaches can be used to obtain the results and gather the data. The three most common empirical methods used in software engineering are experiments, case studies, and surveys [47]. However,
other approaches (e.g. action research [21]) could also be used when performing a research study. Experiments, case studies, and surveys are presented below, in decreasing level of formalism [47].

**Experiment:** An experiment is made in order to investigate causal relationships between factors by controlling surrounding variables [21]. This means that a particular event is replicated with some conditional difference to control its outcome. Hence, it could be evaluated if the conditional difference affected the event positively or negatively. Experiments are often conducted in small scale with much control [47].

**Case Study:** A study focusing on a single project that is not replicated is called a case study; in other words, an investigation that is conducted on a particular situation or problem. This can be made directly (e.g. interviews, observation) or indirectly (e.g. studying company reports, documentation) [21]. Case studies are conducted in typical conditions and are hence hard to generalize to every possible situation [47].

**Survey:** Surveys look at many teams and projects at the same time, and are usually undertaken by interviews or questionnaires. Hence, they include larger amount of data than case studies. Therefore, more consideration must be taken to identify samples, select samples, design questionnaires and define interviews [21][47]. Surveys are research in large scale [47].

### 1.2.2 Approaches for Collecting the Data

When collecting the data in either of the research methods explained in Section 1.2.1, some approach must be taken of how to collect the data. The main question is whether to do a qualitative or quantitative investigation. Below, explanations of the two are presented and then how to choose between them.

#### 1.2.2.1 Qualitative Approach

Simply speaking, a qualitative study asks simple straightforward questions while getting complex answers that contain a lot of information. Therefore, when having conducted the investigation, the investigator has a lot of material to analyze and try to find patterns, happenings and views within [78]. When analyzing, the investigator must interpret the answers and try to understand what the respondents really meant [79]. Issues like frame of reference, motive, social processes, and social context affect the results [34]. The data col-
lected in a qualitative study could be captured in several ways, including observations, interviews, and ethnographical studies [57].

1.2.2.2 Quantitative Approach

Simply speaking, a quantitative study introduces numbers into the answers. It must not be just numbers in the general definition but also in a figurative sense where words as “longer”, “more”, and “larger” could be used [79]. The advantage of quantitative studies is that statistical analyses could be used to express the relationship of factors [34]. Therefore, it is important to know the rules for what we can do with the information that is collected in a quantitative study.

Quantitative studies are not able to find unknown information [74] and must therefore be supported by qualitative studies in order to find information that is not known. When the information is found, it is possible to investigate its magnitude through quantitative studies. It is also possible to generalize the results to a larger extent in quantitative studies since more subjects can be reached and more standardization of the questions can be obtained [34]. The data collected in a quantitative study are most often captured by questionnaires or interviews (with large control) [34] but could also be captured through analyzing measurements from projects, and so forth.

1.2.3 Choosing between a Qualitative or Quantitative Approach

When choosing which approach to use, there is no right or wrong answer but rather that the approach used depends upon the purposes of the investigation [79]. Further, the two different techniques are not mutually exclusive; it is possible to do a pre-study by qualitative studies and the main investigation by quantitative studies, or vice versa. In fact, combining them are often to prefer [57][70].

1.2.3.1 Sampling of Respondents (External Validity)

It is important to have the right people to respond to the investigation, a common rule is to have as many as possible to be respondents to achieve as accurate results as possible. Nevertheless, it is often impossible to have all people in a population as respondents,
and sampling of respondents must therefore be made [79]. It is important that this sample corresponds with the whole population the researcher wants to describe [57]. Factors to consider when selecting sample are: How accurate must the investigation be? How many answers are the investigators able to handle? How much effort is reasonable to demand from the respondents (e.g. could a company afford all employees taking time to answer?)? [79]. If the sampling is incorrect, it might not be possible to generalize the findings, which introduces a threat to the external validity [70].

The discussion above is primarily aimed at quantitative investigations; qualitative investigations might even be designed in the opposite way. In qualitative investigations, the strategy could be to get people that are not typical for the event to answer because it could illustrate the different viewpoints that are present in the population [34]. Nevertheless, they must be a part of the population that shall be investigated in order to get viewpoints within that population. It could also be possible to combine qualitative and quantitative methods in order to address another threat to external validity; the threat that there often is a difference between what people say and what people do [70]. By using both kind of methods, such threats could be limited (see further discussion about this in Section 1.2.3.4).

1.2.3.2 Standardization (Reliability)

Standardization refers to the degree of which the questions and situations are the same for the respondents. With low degree of standardization, the questions are adapted to the respondent regarding language, follow-up questions are posed depending on answer, the respondent steers the order of the questions etc. With high degree of standardization, the questions are read in the same order, are formulated the same etc. [79].

The level of standardization differs between different investigations. When extensive answers as in qualitative studies are preferred, the level of standardization might be relatively low. When more specified and quantifiable answers are preferred, a higher level of standardization is to prefer to get as accurate answers as possible [79]. This implies that qualitative studies are characterized by flexibility while quantitative studies are characterized by standardization or structure [34].

Standardization ultimately refers to the reliability of the study, i.e. how consistent the investigation is over time and if the same results are found if the investigation is done more than once [57][70]. For
example, if the questions are straightforward, not contradictive, and performed under the same circumstances, they are more likely to be interpreted the same way by the respondents [79].

1.2.3.3 Internal Validity

Validation of the investigation is necessary in order to know that the investigation can be trusted. The internal validity of an investigation is the degree to which the analyst measures what he or she intends to measure [46][57]. This means that if the questions are incomprehensible, ambiguous and conducted in the wrong time and place, the study would probably be a waste of time and introduce threats to internal validity [70]. For example, if the intention is to measure how often a person reads the newspaper, the answer given in a questionnaire should be based on how many days he reads it and not if he or she reads it often or seldom which involves a great deal of subjectivism by the respondent [79].

1.2.3.4 Validation of Research

When assuring the validity of the research, it is possible to use techniques to validate that the information retrieved is as accurate as possible. Triangulation is possible to use for this purpose and the following four different types of triangulation exist [65]:

Data Source Triangulation: The purpose of this triangulation technique is to validate the investigation by providing a secondary source of information (e.g. support an interview with data from a project). By this, it is possible to find differences between what people say and what they do.

Analyst Triangulation: This technique uses multiple analysts to analyze the obtained data. If just one analyst is used, it is possible that the result is dependent on his or her interpretation of the material. With more analysts involved, the risk for this bias is reduced. This could be supported by having several researchers involved in the study, and comparing their conclusions.

Theory/Perspective Triangulation: This kind of triangulation is based on that validity can be obtained by interpreting data from different theoretical perspectives. It could for example be done by collecting data on a subject from a behavioral and a cognitive viewpoint.

Methods Triangulation: This technique involves comparing the data collected from several methodologies such as quantitative and qualitative methods. The difficulty is that two different methodologies are not always appropriate for the same research question. How-
However, the idea is that if several researchers come to the same conclusion using different methods, the results are more likely to be accurate.

1.3 Work Process and Outline

The work presented in this thesis has evolved by using empirical studies to gather results. These studies have been used both in industrial and academic environments. This section aims to present the different environments that have been used, and also serves as a description of how the research has evolved. First, the settings where studies have been performed are discussed in more detail in Section 1.3.1. Then, a description of how the work has evolved and a presentation of one unpublished study are provided in Section 1.3.2. Last, the contexts and the research methodologies used in the different studies are discussed and compared in Section 1.3.3.

1.3.1 Thesis Setting

In this thesis, two primary sources for conducting the studies have been used (i.e. industry and academia). These two kinds of environments have different characteristics and a more elaborated discussion of their characteristics is provided in Chapter 6. The environments are presented in the sections below.

1.3.1.1 Ericsson AB, Karlskrona (Ericsson)

The research in this thesis project have been conducted together with Ericsson AB in Karlskrona (further on denoted as Ericsson), an ISO 9001:2000 certified company. During the studies, the organization has had about 400 employees working in different software development projects. Such projects typically included 60-120 persons for 12-18 months. The employees of the organization were organized in a matrix organization [59].

The framework for software processes (e.g. guidelines for how to document processes) within the organization was developed by a functional area [59] similar to a Software Engineering Process Group (SEPG) [85]. However, the actual processes in the organization were mostly developed within each functional area or unit (e.g. a test unit had a test process of their own) and the organization also had some standard processes that all departments should follow (e.g. a process for inspecting artifacts). In addition, the organization followed a project management model called PROPS.
Ericsson develops real-time solutions for mobile telecommunication systems. This means that the products are real time products that are sold to different mobile telephone operators. The domain and the customer situation result in that requirements have to be gathered from operators, end users, standards and regulations, and so forth. This means that the requirements situation is rather complex and hence also the matter of deciding what to include and not include in a product or release.

1.3.1.2 Blekinge Institute of Technology (BTH)

When not being able to perform studies in industry for different reasons (e.g. cost, time, and schedule constraints), or when trying out new studies through pilot studies, it is possible to carry out such studies in an academic setting. At Blekinge Institute of Technology (further on denoted as BTH), there exist a number of different study programmes of which the software engineering programme is one. The software engineering program is a three years programme resulting in a bachelor’s degree, with the possibility to extend it with one and a half years of studies to get a degree of Master of Science with a major in software engineering. At the bachelor’s programme, the students perform three projects with different characteristics (more detailed information about the projects in Chapter 6). Because these projects are available, it is possible to perform research studies in both classroom environments and project environments. At the Masters level, both students from the bachelor’s programme at BTH and other students (both national and international) take courses in software engineering, without any project based courses.

In classroom environment, the students could be first year students as well as master students or Ph. D. students. In project environment, students could be first, second or third year students at the software engineering program. The difference between these two alternatives is that the students in a project environment are more close to reality (i.e. with customers, budgets, quality constraints etc.) and hence seem to be a better alternative when performing empirical studies as a substitute for an industrial study. A more detailed description of different alternatives can be found in Chapter 6, where the suitability of students as subjects when performing prioritizations are evaluated.
1.3.2 Work Process

This section aims to give the reader a view of how the work has progressed to what is present in this thesis. The work process is illustrated in Figure 1.1.

As can be seen in this figure, the work started with a Process Management (PM) study. This study was conducted at Ericsson within the scope of an improvement work of the requirements management process. This work resulted in the papers that form the basis for Chapters 3 and 4. The next step in the cooperation was done through launching a requirements engineering (RE) study to get inputs for further studies. This investigation was performed through distributing a questionnaire about the requirements process, with the aim to find areas to improve. This investigation did not result in any paper since no interesting research results were obtained. Nevertheless, it is presented in this thesis as a part of the background information (see Section 1.3.2.1).
The next step in this process was to take input from this investigation to address issues raised in the result. Because of different circumstances (discussed in Section 1.3.2.1), requirements prioritization has been further investigated in this thesis. This has been done through first provide a foundation for which techniques that exist (Chapter 2). After this presentation, an experiment evaluating different prioritization techniques is discussed (Chapter 5), followed by a discussion of how to test different techniques before introducing them in an industry setting (Chapter 6).

As can be seen in Figure 1.1, the chapters are not presented in a chronological order since Chapter 2 is presented before Chapters 3 and 4. Chapter 2 was written as a description of how requirements prioritization could be conducted when working with product requirements. However, as can be seen in Section 1.1, there are lots of similarities between products and processes and much of the information in this chapter is applicable when doing prioritizations in process development as well. This has led to that all presented papers involve prioritization in one way or another. Hence, it was most suitable to use Chapter 2 as a way of introducing the area for the reader on which the subsequent chapter could rely.

### 1.3.2.1 Requirements Engineering Study at Ericsson (RE study)

As can be seen in Figure 1.1, the RE study was launched after the PM study at Ericsson. Since the qualitative part of the PM study was focusing on the requirements process, the knowledge from that study was used as input for the RE study. The RE study aimed to find out which requirements areas that were most important to improve and how important it was to improve these areas. The intention was to include all different departments and roles at Ericsson in order to get the voices and needs of all different perspectives in the organization.

The RE study was conducted by creating a questionnaire with four main questions about different issues in the requirements process. The questionnaire was reviewed by personnel from both BTH (academic) and Ericsson (industrial) before distributing the questionnaire by e-mail to 190 randomly chosen participants. The sample was chosen by simply selecting every second employee at each department and unit. However, 16 of these 190 respondents were not able to answer the questionnaire due to different reasons (e.g. out of office, not suitable role). This left 174 persons able to answer the questionnaire and 66 of these answered. However, one person’s answer was not taken into account since it was not complete in any
of the questions. This person was eliminated from the study and the response-rate including acceptable answers reached slightly more than 37 percent of the sample (18.5 percent of the organization).

As stated previously, there were four main questions in the questionnaire. These four questions focused on the following:

1. Most important area; this question aimed to see which areas that were in most need for improvements.
2. Importance of improving; this question aimed to see how important it was to improve the current requirements management process.
3. The relation between functional and non-functional requirements; this question aimed to find out which of the functional and non-functional requirements the respondents perceived as the most problematic.
4. The relation between the Main Requirements Specification (MRS) and the Requirements Specification (RS)\(^1\); this question aimed to see which of the MRS and the RS the respondents perceived as the most problematic.

The questions raised and the alternatives given were based on the knowledge obtained from the work in the PM study (see Figure 1.1), both in terms of personal experience and documentation from the project. This knowledge, together with information elicited from literature (books and articles) in the requirements management field made up the basis for the questions in the questionnaire. Three of these questions (1, 3, and 4) were based on the 100-dollar test (consult Chapter 2 for more information) while one question (2) was based on a five level Likert scale (i.e. an ordinal scale with five levels with labeled alternatives \([70]\)).

In addition to these four questions, the respondents got three demographic questions (i.e. role, organizational location, projects involved in) and one open ended question (to get opinions about the questionnaire and get some further input they thought were important but not a part of the questionnaire). In Figure 1.2 and Figure 1.3 the results from questions one and two are presented. The results of the other questions are not discussed here since they do not provide any information vital to the discussions of this the-

\(^1\) The MRS is on a higher level than the RS (i.e. the requirements in the MRS are broken down to more specific requirements in the RS).
sis, and since the response rate on these were lower (and hence not as trustworthy).

**Figure 1.2:** Result of Question 1: How is the relative importance divided between the following 10 areas to improve in the management of requirements at Ericsson today?

The results of Question 1 (Figure 1.2) show that four areas seem to be more important than others. These four areas are elicitation (20%), validation (13.5%), changes (13%), and prioritization (11%). When analyzing these, it is possible to note that all these four areas relate to improvements of the early phases of a development initiative:

- **Elicitation** is about finding the right requirements.
- **Validation** ensures that the right requirements are realized.
- **Prioritization** finds the most important requirements to realize.
- **Changes** consider how and why changes occur and how well such changes are handled.

Elicitation, validation and prioritization are directly related to finding the “right” requirements. Changes, on the other hand, are rather a way to rectify wrong decisions. However, if the change management process works badly, and if many requirements are changed during a development initiative (for any reason), it indicates that the
other three areas have potential for improvement (why do we have to do changes?). Hence, the fact that changes are regarded as important supports the theory that there is room for improvement in the early phases (elicitation, validation, and prioritization). If these three areas are handled in a good manner, there would be less changes and thus less need for improving the change process.

Further, several of the respondents (independent of how they ranked different improvement areas) who answered with an open-ended response pushed on that the requirements must be handled correctly from the beginning. They stated that it is important to set the right scope of the project from the beginning and to deliver the “right” products. Several said that limiting the scope from the beginning through prioritization is the most urgent measure in the requirements process. They thought that it is better to start with a narrow scope and add things during the project instead of removing or changing requirements. Further, some respondents asked for better and clearer requirements from the beginning in order to be sure what should be developed. One respondent argued that the people in the projects must learn to know the customer needs, requirements, environment, and profitability.

With only the answers from the first question, it is possible to see if one area is more important than another, and how much more. However, due to that the areas are only weighted in relation to each other, no absolute measure is given of how important it actually is to improve. Hence, the second question aimed to get an impression of whether they are just important in relation to each other, or if they really were important. In Figure 1.3, it is possible to study the result of this question.

In Figure 1.3, it is possible to note that 96.5 percent of the respondents think that it is important (61%) or very important (35.5%) to improve the current way of dealing with requirements. Only 3.5 percent of the respondents were neutral to whether improvements were needed or not. The conclusion drawn from this result is that most personnel think that it is important to improve the areas mentioned in the questionnaire.

It should be noted that even though there seems to be a need for improving the current practices at Ericsson, the organization can be regarded as successful with profitable products on the market. This shows that even if an organization is successful, there is room for improvements to make the organization even more successful.
The importance of the four activities identified in the RE study have been emphasized in the research literature (e.g. [33][40][60][86]), which implies that these areas are not unique for Ericsson. The major question was of course in what area to start the further research. One natural way would be to start with elicitation since this was the highest prioritized area. Another would be to start by investigating change requests of historical projects to find out the origin of change requests historically (and thereby finding ways to prevent such changes). However, prioritization was chosen as the primary research area for several reasons:

- Interest for prioritization was raised when prioritizing improvement areas in this study and the PM study.
- Prioritization facilitates validation and elicitation by giving a better understanding of requirements [44].
- Several responses stressed the importance of prioritizing requirements better in the open ended question.
- Prioritization has been an area commonly brought up during informal discussions with personnel at Ericsson.
- Prioritization is rather stand alone and is possible to try out in different environments (e.g. with students), which is good when
time and resource limitations make it hard to conduct studies in
the industrial environment.

Even though requirements prioritization has been chosen as the
primary research area and the topic of this thesis, the other areas are
still included in the research. For example, in parallel to the studies
presented in this thesis, work with change requests is being per-
formed together with Ericsson. This work is ongoing and no final
results are possible to present in this thesis.

Except for the questions and results presented above, further ques-
tions existed that have not been brought up here. Questions three
and four are not brought up since there was a rather low response
rate on these questions. These two questions were aimed to get a
further understanding of factors influencing which areas were most
important. There were also three demographic questions in the
questionnaire. The information from these questions could have
been used to perform an analysis of how different roles regarded
the importance of different improvement areas (as in Chapter 4).
However, the response rate varied very much between different
departments and roles (0-60%), which led to the decision of not
performing such an analysis. However, if just performing a brief
analysis, different roles and departments generally had different
order of the ranks, even though the four areas mentioned above
mostly were seen as the four most important areas.

1.3.3 Studies Performed in this Thesis

As can be seen in Section 1.2.1, there exists three major ways of
conducting empirical research (experiments, case studies, and sur-
veys). Further, there are two ways to collect information when ask-
ing people about different things (qualitative and quantitative).
Finally, there exist at least three different arenas where research
studies could be performed. One is to ask or study people when
performing their daily work (field). Another way is to create a labo-
atory environment with people and thereby to create an environ-
ment where studies can be performed (as when using students in
experiments). The third, and last way, is to study literature to create
an understanding or draw conclusions on previously made research.
These different ways of how to conduct research is gathered in
Table 1.1 and each chapter is related to what ways of research that
are performed within the chapter.
In Chapter 2, a qualitative literature survey about requirements prioritization is presented. Chapter 3 presents an industrial (field) case study that involved both a qualitative and quantitative part while also involving a literature survey. Chapter 4 presents results from the same study but this chapter only focuses on the quantitative result and hence it is classified as case study done in an industrial setting (field) and based on quantitative analysis. Chapters 5 and 6 present two quantitative experiments about requirements prioritization that were made in laboratory environments. However, Chapter 6 also includes a survey where the result of the experiment is compared to other results in field settings. Finally, Section 1.3.2.1 presents an industrial (field) case study where a quantitative analysis was made. A further presentation of what each paper included and the overall contribution is presented in the next section.

It should be noted that this presentation of research approaches is not meant to be exhaustive. Other research approaches exist but the above presented are those that are relevant for the research presented in this thesis. Further, it is possible to note that some combinations are more used than other. For example, when literature is marked, survey is also marked. This is due to that a literature study could hardly be performed without doing a survey of existing literature (e.g. a literature case study would more or less be a summary of the literature source). Further discussions of the implications of this table could be interesting, but are not a part of the scope of this thesis. Instead, the contribution of this thesis is presented in the next section.

Table 1.1: Ways of Conducting Research. The different ways of performing studies (presented in 1.2) are presented in the first column while the chapter number (or section number) is presented in the first row. A mark in the corresponding cell indicates that it is a major component of the part of the study that is presented in the chapter.

<table>
<thead>
<tr>
<th></th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>1.3.2.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Experiment</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Case Study</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Survey</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Qualitative</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quantitative</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Field</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Laboratory</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Literature</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1.4 Contribution

The discussion in this section is based on the five papers that are included in the thesis and the aggregated results of these. The presentation of the contributions starts with a presentation of the different papers that constitute the chapters of the thesis (Section 1.4.1). After this presentation, a discussion about what the results of these chapters actually imply and what the overall contributions of the collection of chapters are (Section 1.4.2).

1.4.1 Chapters Included in the Thesis

The aim of this section is to give a brief presentation of the papers that are included in this thesis. The different papers are not discussed but the presentation consists of an abstract of each paper. It should be noted that the papers included in this thesis (Chapters 2-6) are presented in their original forms, except for minor changes to make the chapters fit better into the thesis.

1.4.1.1 Chapter 2: Requirements Prioritization [9]

This chapter provides an overview of techniques for prioritization of requirements for software products. Prioritization is a crucial step towards making good decisions regarding product planning for single and multiple releases. Various aspects of functionality are considered, such as importance, risk, cost, etc. Prioritization decisions are made by stakeholders, including users, managers, developers, or their representatives. Methods are given how to combine individuals’ prioritizations based on overall objectives and constraints. A range of different techniques and aspects are applied to an example to illustrate their use.

1.4.1.2 Chapter 3: Identification of Key Factors in Software Process Management – A Case Study [6]

When conducting process related work within an organization, it is important to be aware of which factors are most important to consider. This chapter presents an empirical case study performed to find key success factors in process management. One factor, namely synchronization of processes, was considered as much more important within the studied organization than within the studied literature. This shows that more research might be needed in this area. The study further shows that it is important to relate process improvement work to the properties of the affected organization and that the key factors identified are highly interrelated.
1.4.1.3 Chapter 4: Differences in Views between Development Roles in Software Process Improvement - A Quantitative Comparison [7]

This chapter presents a quantitative study that evaluates how different roles in a software development organization view different issues in software process improvement. The study was conducted at Ericsson, including the traditional roles of software development. The respondents of the study were asked five different questions related to process improvement. The result was that the different roles disagreed in three of the questions while they agreed in two of the questions. The disagreement was related to issues about importance of improvement, urgency of problems, and threat against successful process management, while the questions where the roles agreed focused on communication of the processes (documentation and teaching). It is concluded that it is important to be aware of and take into account the different needs of different roles and that looking into other areas (e.g. marketing and requirements engineering for products) could be beneficial when conducting process improvements.

1.4.1.4 Chapter 5: Requirements Prioritisation: An Experiment on Exhaustive Pair-Wise Comparisons versus Planning Game Partitioning [45]

The process of selecting the right set of requirements for a product release is highly dependent on how well we succeed in prioritizing the requirements candidates. There are different techniques available for requirements prioritization, some more sophisticated than others. In order to compare different techniques, a controlled experiment was conducted with the objective of understanding differences regarding time consumption, ease of use, and accuracy. The requirements prioritization techniques compared in the experiment are the Analytical Hierarchy Process (AHP) and a variation of the Planning Game (PG), isolated from Extreme Programming. The subjects were 15 Ph.D. students and one professor, who prioritized mobile phone features using both methods. It was found that the straightforward and intuitive PG was less time consuming, considered easier to use, and perceived as more accurate than AHP.

1.4.1.5 Chapter 6: Using Students as Subjects in Requirements Prioritization [8]

When conducting research in software engineering, the ultimate goal is usually to come up with results applicable in industry. However, it is not always possible to get industrial professionals to act as subjects in research studies. Instead, students are commonly used as
representatives for professionals since they are more convenient to use. This chapter presents an experiment on requirements prioritization that was performed with classroom students as subjects. The result of the experiment is compared to the results of similar prioritizations made in other classroom studies, student projects, literature and in an industrial case study. The objective of this comparison was to evaluate in which cases students successfully could be used as subjects in experimentation. The result indicates that students in a classroom environment are less suitable than students in projects as representatives for professionals in studies of this kind. Experience is often mentioned as a factor to determine whether students are suitable or not as subjects. However, commitment seems to be a more important factor in this study. It is concluded that it is important that further research is performed in order to evaluate under what circumstances students are suitable, and what factors influence suitability.

1.4.2 The Focus and Contribution of this Thesis

As already have been discussed, the focus of this thesis is in the area of requirements prioritization. Instead of the common focus of requirements prioritization, i.e. in product development, this thesis widens the subject to also include requirements prioritization in process development. The thesis consists of five different chapters; all with different aims and scopes (see chapters for details). The first chapter (Chapter 2) focuses on the topic of requirements prioritization and is primarily focused towards product development (since it was written for a book about requirements prioritization in product development). Nevertheless, many of the techniques, practices etc. that are presented in the chapter, could also be applied in requirements prioritization in process development.

In Chapter 3, an industrial case study is presented. The aim with this study was to find which software process areas that were most important to improve. After eliciting which areas that seemed most important to improve (by a qualitative study and a literature study), prioritization was used to find which areas were most important and most urgent to improve. This shows that requirements prioritization seems to be well suited when comparing the importance or urgency between different issues. Chapter 4 further looks on the result of this prioritization and notes that different roles have different agendas when it comes to process improvement. The chapter further concludes that it seems important to look into other areas to be able to cope with issues related to such differences (e.g. mar-
Introduction

Marketing, and requirements engineering for product development). For example, role differences are similar to how marketing research looks at segments of customers (discussed in Chapter 2).

In Chapter 5, an experiment is presented where two different prioritization techniques are compared and evaluated. This experiment was conducted in an academic setting (laboratory environment) and the subjects were supposed to prioritize features for mobile telephones. The results showed that one technique was superior with regard to time consumption, ease of use, and accuracy. However, it should be noted that the number of requirements (or features) in this study was limited to 8 and 16. This amount of requirements is not very realistic since companies have much more requirements when developing software (e.g. Ericsson has between 1500 and 2000 requirements in a release). Nevertheless, the results could be more valuable when looking into the process development area since a smaller amount of requirements are generally prioritized (e.g. between five and seven in the process management study, and 10 in the requirements management study). It is common that evaluations of prioritization techniques use too few requirements when assessing requirements prioritization for product development is common. This could imply that the results from these studies are in fact more applicable in process development than product development even though it is not exactly the same kind of requirements that are prioritized.

The last chapter in this thesis (Chapter 6) presents a study with students as subjects that had both educational and research objectives. The objective from a research perspective was to investigate how trade-offs are handled when planning releases of a software product. From this perspective, the study performed was a failure since the subjects did not prioritize as professionals normally do in industry, which meant that they had no problems when planning the releases. Instead, it was possible to get insight in what makes students suitable as subjects. In this case, commitment was regarded as the most influential factor since other studies, with students in projects, indicated that students could be suitable as research subjects when prioritizing requirements. The result of this, in comparison to the other results of this thesis, indicates that it might be better to let the students in such settings prioritize issues for which they care (e.g. educational improvement issues). By using an alternative requirements set (of which they care), the students would probably provide more reliable results and thereby help improving the knowledge about prioritization techniques.
This thesis shows that it is possible to use requirements prioritization in both process and product development (improvement issues are prioritized in Chapter 3 and 4 while product requirements are prioritized in Chapter 5 and 6). Further, when discussing prioritization techniques for product requirements in Chapter 2, experiences from prioritizations of improvement issues in Chapter 4 were used to show some drawbacks of one of the techniques (100-dollar test). This knowledge could be very valuable in the continuation of research about prioritization techniques in software engineering. It is common that studies of requirements prioritization involves toy systems with too few requirements to really reflect prioritization of product requirements in an industrial setting. Nevertheless, the results of such studies could probably be used when performing prioritizations of process requirements in an industrial setting. Further, to really get trustworthy results about prioritization techniques aimed at industrial application in product development, it is necessary that the number of requirements in such studies grow significantly.

The discussion above gives a view of the contributions of this thesis on a general level. In each of the subsequent chapters (Chapters 2 to 6), these issues are discussed in more detail without necessarily discussing the relation between prioritization of processes and products. This means that the current section serves as a compilation of the results, while Chapter 2 to 6 provide in-depth discussions about the specific issues investigated in each chapter. Before presenting these chapters, possible future work in the areas discussed in this thesis are presented and discussed in the next section.

1.5 Future Work

Within the areas brought up in this thesis, there exists a number of potential research questions that could be further investigated. In this section, four interesting research areas are presented and a number of research questions are outlined for each one.

1.5.1 Software Process Improvement

The results in Chapter 3 indicate that synchronization of processes might be an area where further research ought to be conducted. Due to the fact that synchronization was the least discussed factor within the literature studied and that it was the most important factor in the studied organization, it is an area that might have received
too little attention in the past. Since this was found in one organization and it is hard to know whether this is a common problem, further research could investigate if other organizations encounter the same problems. If some do, what are the characteristics of such organizations?

Further, the results in Chapter 3 show that there are strong dependencies between the different factors. This implies that it is not enough to just consider the factors that are most important at the moment, but also other factors that are regarded as important in literature and in other reliable sources. In Section 1.5.2, dependencies of product requirements are brought up as an area where future research is needed. The fact that dependencies exist between both product requirements and process requirements strengthens the argumentation for the similarities between software processes and products. The future research about dependencies in software products and processes might hence be conducted partially in conjunction.

In Chapter 4, it was concluded that different roles within the studied company had different opinions of what was important to improve. First of all, it would be interesting to see if different companies and settings have the same differences in views. This study and some previous studies have indicated this, and it is worth further studies to investigate how this affects the process improvement work and how this information could be used effectively. Further, studies that investigate if other demographic choices (e.g. age, gender, functional area) also find differences in the opinions about software process improvement would be interesting.

In general, it would be beneficial to increase the understanding regarding the importance of roles when it comes to views and opinions of software process improvement [41]. The agreements and disagreements are part of the environment in which any improvements should be made and a good understanding is most likely very important to have a firm foundation for improvements. As stated in Chapter 4, this issue has been known for long in other areas, especially in the marketing field (but has also been considered lately in relation to requirements of software products). Hence, other disciplines could probably provide important information that could be used in software process development (probably with some modification) and research might be necessary to investigate what could be used and what customizations should be made.
1.5.2 Prioritizing Product Requirements

Requirements engineering is a field with much research activity. One journal, several workshops, and one large annual international conference are devoted to requirements engineering. Nevertheless, the existing work in the area of requirements prioritization is limited even though the need for prioritizing software requirements is acknowledged in the research literature [43]. Especially, few empirical validations of different prioritization techniques and methods exist. Instead, it is common that new techniques and methods are introduced and they seem to work well, but the scalability of the approach has not been tested (e.g. [71]). However, there exist some studies that have evaluated different prioritization techniques (e.g. [44][45]). Unfortunately, such empirical evaluations most often focus on toy systems with a few requirements (seldom more than 20). This is not really providing any final evidence of whether one technique is better than another even though some preliminary evidence could be found. One of the few industry studies, for example, found that AHP was not usable with more than 20 requirements since the number of comparisons became too many for the practitioners [54]. Hence, more studies are needed where prioritization methods are used in industry.

A further question that seldom is addressed in requirements prioritization research is the question of how much sophistication is actually needed. Many techniques and methods are developed and they become more and more complex with the goal to provide more help for practitioners. However, the results are seldom used in industry (see also Section 1.2). Instead, professionals use simple methods such as numerical assignment. Practitioners live in a different environment than experimental subjects (often students) and are more limited by time and cost constraints (see Chapter 6). Hence, an important question to answer is how much sophistication (and thereby complexity) is actually necessary and desirable by practitioners?

The above issue leads to another open question about when a technique or method is suitable. Existing empirical studies seldom discuss factors such as company size, time-to-market limitations, number of stakeholders, domain, etc. Instead, focus is put on finding if a technique or method is better than another one (as for example in Chapter 5). A more sound approach would be to test different approaches in various environments to get some understanding when different prioritization techniques, aspects, etc. are
suitable. In [28], a framework for evaluating pair programming is suggested. In this framework; independent (e.g. technique), dependent (e.g. quality), and context variables (e.g. type of task) are proposed for evaluating different programming techniques. A similar framework for requirements prioritization would be beneficial.

Another question that is really important in the area of requirements prioritization is dependencies between requirements. The impact of dependencies is tremendous in prioritizations since choosing from the prioritized set is very much dependent on how the requirements relate to each other. Prioritization techniques (such as AHP) assume that requirements are independent even though it is known that they seldom are [69]. A question that is really important for requirements prioritization is to find ways to handle dependencies in an efficient way. Some work has been done in this area (e.g. [12]) but much more needs to be done in order to handle dependencies efficiently in the prioritization process [19].

As can be seen in Chapter 2, there are several different aspects that could be taken into account when performing prioritizations (e.g. importance, cost, and strategic benefit). Different authors present which aspects that are important and should be taken into account (e.g. [3][42][54]), but there does not seem to be studies that actually have investigated which aspects industrial professional use, or consider especially important. Studies investigating which aspects industrial professionals use and which they regard as important would provide valuable knowledge for the area of requirements prioritization.

The last area that is brought up here is the area of functional and non-functional requirements. As can be seen in Chapter 2, functional and non-functional requirements are very different even though they have a serious impact on each other. Prioritizing these two entirely together or separately might not be the best solution. Approaches where prioritizations of functional and non-functional requirements could be combined in an efficient way are necessary. Different methods that seem suitable for prioritizing non-functional requirements are available (e.g. Quality Grid [52], Conjoint Analysis [29]) and it would be interesting to evaluate these empirically in industrial settings. Furthermore, finding ways to combine such approaches with approaches more directed to functional requirements would be a challenge.
1.5.3 Using Change Data in Volatility Analysis

As can be seen in Section 1.3.2.1, the RE study at Ericsson resulted in that four areas were considered more important than the others. The conclusion was that three of these were related to ‘do the right thing’, namely elicitation, validation, and prioritization. One interesting result of this is that the fourth improvement issue, change management, could be investigated to get information on how to ‘do the right thing’. Material from the change management work (i.e. change requests) can be studied in order to find improvement opportunities in the other parts [84]. For example, if it is found that many change requests affect requirements originated from customers, requirements from customer might get special attention in the elicitation, validation and prioritization process. Or, for example, if it is possible to identify some type of requirements (e.g. requirements related to standards) that are very prone to changes, it might be possible to wait for implementation of these until they are regarded as more stable. As seen here, information from change requests could provide very much important knowledge that could be used to find patterns in the changes and from these take action for improvement. Most companies have some sort of a change management database, and this data could be used for these purposes in addition to only handle changes. Some initial work has been done in this direction (e.g. [60]) but the work is in its early stages and more work is needed to get a way of handling this information in an efficient way.

1.5.4 Using Students as Subjects in Software Research

Practice-based education through projects has been requested for educational purposes in order to provide industrial relevance in the education [14]. Based on the results of the study presented in Chapter 6, the project-based education could also be valuable in terms of research purposes. If having more education based on practical work, it is possible to get commitment and connection to reality that is hard to obtain in a classroom environment. This leads to the hypothesis that the students might be more suitable to use for research purposes as well. This combination would be very beneficial for students, researchers, and industry, since both good students and good research come out of the combination.

It is hard to generalize, but it seems like classroom students are not suitable for experiments that involves commitment and “project dependent decisions”. This also implies that exercises of the kind
presented in Chapter 6 are not very suitable for classroom students even though other studies might be suitable. However, as stated above, it is hard to draw any general conclusions when this commitment and connection to reality is required.

Based on the above discussion, it seems very important that studies are performed that investigates in which situations students are suitable as subjects and in which they are not. Some research has already been done in the area (see Chapter 6) but in order to really get knowledge of when students are suitable, more research is required. This research would ultimately lead to some sort of classification of when students are useful as research subjects. Such classifications should build on the nature of the problem to evaluate, the experience of the students, the needed commitment etc.

### 1.6 Summary

As discussed in Section 1.1, there exist several similarities between software processes and software products. One such similarity is that needs, or requirements, exists both when developing software processes and software products. In this thesis, this similarity is investigated further by discussing different prioritization techniques and by applying such techniques in different environments, for different purposes. As shown in the remainder of this thesis, it is possible to apply prioritizations both when prioritizing improvement issues for a software process and when prioritizing requirements for a software product. It is also shown that there exist further similarities between these two areas. For example, it is shown in Chapter 4 that different roles in a software organization have different needs when it comes to which processes to improve. This is similar to the different needs that users of a software product commonly have.

The studies presented in the following chapters use a wide range of different research methods to explore the area of requirements prioritization. For example, literature studies, industrial case studies, and experiments in a lab environment are applied. The results of these show similarities between software processes and software products even though the results are not final, and further research is necessary. A number of different findings, in different areas, are presented in this chapter and these are based on the result from the studies performed in Chapters 2 to 6. Each of these chapters has different focus and scope. The summary below gives a very con-
Introduction

densed description of each chapter, to give an overview of where to
find discussions about different issues:

- Chapter 2, discusses requirements prioritization of product
requirements and investigates which different techniques that
exist, stakeholders that should be a part of prioritizations, differ-
ent aspects to consider, and so forth.
- Chapter 3, presents a study about software process improve-
ment that identified and prioritized key factors in software proc-
ess improvement to get an understanding of which factors that
are most important to improve.
- Chapter 4, continues the analysis from the study presented in
Chapter 3 and investigate how the priorities differs between
roles and discusses how such information can be used.
- Chapter 5, presents an experimental evaluation of two prioriti-
ization techniques where the two are compared in order to pro-
vide knowledge of which one that is easiest to use, least time
consuming, and most accurate.
- Chapter 6, presents a study where different cases of prioritiza-
tions are compared in order to get an understanding of which
kind of subjects that are suitable for evaluation of prioritization
techniques.

1.7 References

Addison Wesley, Boston MA.

Requirements Engineering Activities as a Decision-Making Proc-

Upper Saddle River, NJ.

ing of Software Processes’, *International Software Engineering
Research Network (ISERN)*, Technical Report ISERN-97-10, Fraun-
hofer Institute for Experimental Software Engineering, Kaiserslau-
tern, Germany.

Elicitation for Descriptive Software Process Modeling’, *Procee-
ings of the Third International Conference on Product-Focused Software Processes Improvement (PROFES), Lecture Notes in Computer Science 2188, Springer-Verlag, pp. 312-325.


Complex decision-making situations are not unique to software engineering. Other disciplines, such as psychology, and organizational behavior have studied decision-making thoroughly [1]. Classical decision-making models have been mapped to various requirements engineering activities to show the similarities [1]. This chapter primarily focuses on requirements prioritization, an integral part of decision-making [45]. The intention is to describe the current body of knowledge in the requirements prioritization area.

In software engineering, the quality of a product is often determined by the ability to satisfy the needs of the customers and users [5][49]. Hence, getting the “right” requirements and planning suitable releases with the right functionality is a major step towards the success of a project or product. If the wrong collection of requirements is implemented and users resist using the product, it does not matter how solid the product is or how thoroughly it has been tested.

Most software projects have more candidate requirements than can be realized within the time and cost constraints. Prioritization helps to identify the most valuable requirements from this set by distinguishing the critical few from the trivial many.
The process of prioritizing requirements provides support for the following activities (e.g. [29][51][53][54]):

- for stakeholders to decide on the core requirements for the system.
- to plan and select an ordered, optimal set of software requirements for implementation in successive releases.
- to trade off desired project scope against sometimes conflicting constraints such as schedule, budget, resources, time to market, and quality.
- to balance the business benefit of each requirement against its cost.
- to balance implications of requirements on the software architecture and future evolution of the product and its associated cost.
- to select only a subset of the requirements and still produce a system that will satisfy the customer(s).
- to estimate expected customer satisfaction.
- to get a technical advantage and optimize market opportunity.
- to minimize rework and schedule slippage (plan stability).
- to handle contradictory requirements, focus the negotiation process, and resolve disagreements between stakeholders.
- to establish relative importance of each requirement to provide the greatest value at the lowest cost.

The list above clearly shows the importance of prioritizing and deciding what requirements to include in a product. This is a strategic process since these decisions drive the development expenses and product revenue as well as making the difference between market gain and market loss [1]. Further, the result of prioritization might form the basis of product and marketing plans, as well as being a driving force during project planning. Ruhe et al. summarize this as: "The challenge is to select the 'right' requirements out of a given superset of candidate requirements so that all the different key interests, technical constraints and preferences of the critical stakeholders are fulfilled and the overall business value of the product is maximized" [44].

Of course, it is possible to rectify incorrect decisions later on via change management, but this can be very costly since it is significantly more expensive to correct problems later in the development
process [3]. Frederick P. Brooks puts it in the following words: "The hardest single part of building a software system is deciding precisely what to build. [...] No other part of the work so cripples the resulting system if done wrong. No other part is more difficult to rectify later." [8]. Hence, the most cost effective way of developing software is to find the optimal set of requirements early, and then to develop the software according to this set. To accomplish this, it is crucial to prioritize the requirements to enable selection of the optimal set.

Besides the obvious benefits presented above, prioritizing requirements can have other benefits. For example, it is possible to find requirements defects (e.g. misjudged, incorrect, and ambiguous requirements) since requirements are analyzed from a perspective that is different from that taken during reviews of requirements [30].

Some authors consider requirements prioritization easy [51], some regard it of medium difficulty [53], and some regard prioritization as one of the most complex activities in the requirements process, claiming that few software companies have effective and systematic methods for prioritizing requirements [36]. However, all these sources consider requirements prioritization a fundamental activity for project success. At the same time, some text books about requirements engineering (e.g. [7][43]) do not discuss the matter of requirements prioritization to any real extent.

There is no "right" requirements process and the way of handling requirements differs greatly between different domains and companies [1]. Further, requirements are typically vaguer early on and become more explicit as the understanding of the product grows [46]. These circumstances imply that there is no specific phase where prioritization is made, rather, it is performed throughout the development process (more about this in Section 2.4.2) [11][34]. Hence, prioritization is an iterative process and might be performed at different abstraction levels and with different information in different phases during the software lifecycle.

Prioritization techniques can roughly be divided into two categories: methods and negotiation approaches. The methods are based on quantitatively assigning values to different aspects of requirements while negotiation approaches focus on giving priorities to requirements by reaching agreement between different stakeholders [35]. Further, negotiation approaches are based on subjective meas-
ues and are commonly used when analyses are contextual and when decision variables are strongly interrelated. Quantitative methods make it easier to aggregate different decision variables into an overall assessment and lead to faster decisions [13][46]. In addition, one must be mindful of the social nature of prioritization.

There is more to requirements prioritization than simply asking stakeholders about priorities. Stakeholders play roles and should act according to the goals of that role, but they are also individuals with personalities and personal agendas. Additionally, many organizational issues like power etc. need to be taken into account. Ignoring such issues can raise the risk level for a project. Even though negotiation is important to consider, this chapter focuses primarily on quantitative methods for prioritizing requirements.

This chapter is structured as follows: First a discussion of different aspects that could be used when prioritizing (Section 2.1) is presented. Next, some prioritization techniques and characteristics are discussed (Section 2.2), followed by a discussion of different stakeholders' situations that affect prioritization in Section 2.3. Section 2.4 discusses additional issues that arise when prioritizing software requirements and Section 2.5 provides an example of a prioritization. Finally, Section 2.6 summarizes the chapter.

### 2.1 Aspects of Prioritization

Requirements can be prioritized taking many different aspects into account. An aspect is a property or attribute of a project and its requirements that can be used to prioritize requirements. Common aspects are importance, penalty, cost, time, and risk. When prioritizing requirements based on a single aspect, it is easy to decide which one is most desirable (recall the example about the speed of a car). When involving other aspects, such as cost, customers can change their mind and high priority requirements may turn out to be less important if they are very expensive to satisfy [32]. Often, the aspects interact and changes in one aspect could result in an impact on another aspect [46]. Hence, it is essential to know what effects such conflicts may have, and it is vital to not only consider importance when prioritizing requirements but also other aspects affecting software development and satisfaction with the resulting product. Several aspects can be prioritized, and it may not be practical to consider them all. Which ones to consider depend on the specific situation, and a few examples of aspects suitable for software
projects are described below. Aspects are usually evaluated by stakeholders in a project (managers, users, developers, etc.)

2.1.1 Importance

When prioritizing importance, the stakeholders should prioritize which requirements are most important for the system. However, importance could be an extremely multifaceted concept since it depends very much on which perspective the stakeholder has. Importance could for example be urgency of implementation, importance of a requirement for the product architecture, strategic importance for the company, etc. [34]. Consequently, it is essential to specify which kind of importance the stakeholders should prioritize in each case.

2.1.2 Penalty

It is possible to evaluate the penalty that is introduced if a requirement is not fulfilled [53]. Penalty is not just the opposite of importance. For example, failing to conform to a standard could incur a high penalty even if it is of low importance for the customer (i.e. the customer does not get excited if the requirement is fulfilled). The same goes for implicit requirements that users take for granted, and whose absence could make the product unsuitable for the market.

2.1.3 Cost

The implementation cost is usually estimated by the developing organization. Measures that influence cost include: complexity of the requirement, the ability to reuse existing code, the amount of testing and documentation needed, etc. [53]. Cost is often expressed in terms of staff hours (effort) since the main cost in software development is often primarily related to the number of hours spent. Cost (as well as time, cf. Section 2.1.4.) could be prioritized by using any of the techniques presented in Section 2.2, but also by simply estimating the actual cost on an absolute or normalized scale.

2.1.4 Time

As can be seen in the section above, cost in software development is often related to number of staff hours. However, time (i.e. lead time) is influenced by many other factors such as degree of parallel-
ism in development, training needs, need to develop support infrastructure or complete industry standards, etc. [53].

2.1.5 Risk

Every project carries some amount of risk. In project management, risk management is used to cope with both internal (technical and market risks) and external risks (e.g. regulations, suppliers). Both likelihood and impact must be considered when determining the level of risk of an item or activity [40]. Risk management can also be used when planning requirements into products and releases by identifying risks that are likely to cause difficulties during development [37][53]. Such risks could for example include performance risks, process risks, schedule risks etc. [51]. Based on the estimated risk likelihood and risk impact for each requirement [1], it is possible to calculate the risk level of a project.

2.1.6 Volatility

Volatility of requirements is considered a risk factor and is sometimes handled as part of the risk aspect [37]. Others think that volatility should be analyzed separately and that volatility of requirements should be taken into account separately in the prioritization process [32]. The reasons for requirements volatility vary, for example: the market changes, business requirements change, legislative changes occur, users change, or requirements become more clear during the software life cycle [16][46]. Irrespective of the reason, volatile requirements affect the stability and planning of a project, and presumably increase the costs since changes during development increase the cost of a project. Further, the cost of a project might increase because developers have to select an architecture suited to change if volatility is known to be an issue [32].

2.1.7 Other Aspects

The above list of aspects has been considered important in the literature but it is by no means exhaustive. Examples of other aspects are: financial benefit, strategic benefit, competitors, competence/resources, release theme, ability to sell, etc. For a company, we suggest that stakeholders develop a list of important aspects to use in the decision-making. It is important that the stakeholders have the same interpretation of the aspects as well as of the requirements.
Studies have shown that it is hard to interpret the results if no guidelines about the true meaning of an aspect are present [33][34].

### 2.1.8 Combining Different Aspects

In practice, it is important to consider multiple aspects before deciding if a requirement should be implemented directly, later, or not at all. For example, in the Cost-Value approach, both value (importance) and cost are prioritized to implement those requirements that give most value for the money [27]. The Planning Game (from XP) uses a similar approach when importance, effort (cost), and risks are prioritized [2] (this method is discussed in more detail in Chapter 5). Further, importance and stability (volatility) are suggested as aspects that should be used when prioritizing while others suggest that dependencies also must be considered [10][32]. In Wiegers' approach, the relative value (importance) is divided by the relative cost and the relative risk in order to determine the requirements that have the most favorable balance of value, cost, and risk [53]. This approach further allows different weights for different aspects in order to favor the most important aspect (in the specific situation).

There are many alternatives of combining different aspects. Which aspects to consider depend very much on the specific situation and it is important to know about possible aspects and how to combine them efficiently to suit the case at hand.

### 2.2 Prioritization Techniques

The purpose of any prioritization is to assign values to distinct prioritization objects that allow establishment of a relative order between the objects in the set. In our case, the objects are the requirements to prioritize. The prioritization can be done with various measurement scales and types. The least powerful prioritization scale is the ordinal scale, where the requirements are ordered so that it is possible to see which requirements are more important than others, but not how much more important. The ratio scale is more powerful since it is possible to quantify how much more important one requirement is than another (the scale often ranges from 0 - 100 percent). An even more powerful scale is the absolute scale, which can be used in situations where an absolute number can be assigned (e.g. number of hours). With higher levels of measurement,
more sophisticated evaluations and calculations become possible [18].

Below, a number of different prioritization techniques are presented. Some techniques assume that each requirement is associated with a priority, and others group requirements by priority level. When examples are given, importance is used as the aspect to prioritize even though other aspects can be evaluated with each of the techniques. It should be noted that the presented techniques focus specifically on prioritization. Numerous methods exist that use these prioritization techniques within a larger trade-off and decision making framework (e.g. EVOLVE [21], Cost-Value [27] and Quantitative Win-Win [44]).

2.2.1 Analytical Hierarchy Process (AHP)

The Analytic Hierarchy Process (AHP) is a systematic decision-making method that has been adapted for prioritization of software requirements [41][47]. It is conducted by comparing all possible pairs of hierarchically classified requirements, in order to determine which has higher priority, and to what extent (usually on a scale from one to nine where one represents equal importance and nine represents absolutely more important). The total number of comparisons to perform with AHP are \( n \times (n-1)/2 \) (where \( n \) is the number of requirements) at each hierarchy level, which results in a dramatic increase in the number of comparisons as the number of requirements increases. Studies have shown that AHP is not suitable for large numbers of requirements [35][38]. Researchers have tried to find ways to decrease the number of comparisons (e.g. [23][50]) and variants of the technique have been found to reduce the number of comparisons by as much as 75 percent [28].

In its original form, the redundancy of the pair-wise comparisons allows a consistency check where judgment errors can be identified and a consistency ratio can be calculated. When reducing the number of comparisons, the number of redundant comparisons are also reduced, and consequently the ability to identify inconsistent judgments [30]. When using other techniques (explained below) a consistency ratio is not necessary since all requirements are directly compared to each other and consistency is always ensured. Some studies indicate that persons who prioritize with AHP tend to mistrust the results since control is lost when only comparing the requirements pair-wise (see Chapter 5 and [35]). The result from a prioritization with AHP is a weighted list on a ratio scale. AHP is
evaluated in comparison to Planning Game (see Section 2.2.7) in Chapter 5 and more detailed information about AHP can be found in [27], [47] and [48].

2.2.2 Cumulative Voting, the 100-dollar Test

The 100-dollar test is a very straightforward prioritization technique where the stakeholders are given 100 imaginary units (money, hours, etc.) to distribute between the requirements [33]. The result of the prioritization is presented on a ratio scale. A problem with this technique arises when there are too many requirements to prioritize. For example, if you have 25 requirements, there are on average four points to distribute for each requirement. Regnell et al. faced this problem when there were 17 groups of requirements to prioritize [41]. In the study, they used a fictitious amount of $100,000 to have more freedom in the prioritizations. The subjects in the study were positive about the technique, indicating the possibility to use amounts other than 100 units (e.g. 1,000 or 10,000). Another possible problem with the 100-dollar test (especially when there are many requirements) is that the person performing the prioritization miscalculates and the points do not add up to 100 (see Chapter 4). This can be prevented by using a tool that keeps count of how many points have been used.

One should only perform the prioritization once on the same set of requirements, since the stakeholders might bias their evaluation the second time around if they do not get one of their favorite requirements as a top priority. In such a situation, stakeholders could put all their money on one requirement, which might influence the result heavily. Similarly, some clever stakeholders might put all their money on a favorite requirement that others do not prioritize as highly (e.g. Mac compatibility) while not giving money to requirements that will get much money anyway (e.g. response time). In both these cases, the solution could be to limit the amount spent on individual requirements [33]. However, the risk with such an approach is that stakeholders may be forced to not prioritize according to their actual priorities.

2.2.3 Numerical Assignment (Grouping)

Numerical assignment is the most common prioritization technique and is suggested both in RFC 2119 [6] and IEEE Std. 830-1998 [26]. The approach is based on grouping requirements into differ-
ent priority groups. The number of groups can vary, but in practice, three groups are very common (e.g. [33][51]). When using numerical assignment, it is important that each group represents something that the stakeholders can relate to (e.g. critical, standard, optional), for a reliable classification. Using relative terms such as high, medium, and low will confuse the stakeholders [53]. This seems to be especially important when there are stakeholders with different views of what high, medium and low means. A clear definition of what a group really means minimizes such problems.

A further potential problem is that stakeholders tend to think that everything is critical [32][51]. If customers prioritize themselves, using three groups; critical, standard, and optional, they will most likely consider 85 percent of the requirements as critical, 10 percent as standard, and 5 percent as optional (see Chapter 6 for more on this issue). One idea is to put restrictions on the allowed number of requirements in each group (e.g. not less than 25 percent of the requirements in each group) as suggested in Chapter 5. However, one problem with this approach is that the usefulness of the priorities diminishes because the stakeholders are forced to divide requirements into certain groups [29]. However, no empirical evidence of good or bad results with such restrictions exists. The result of numerical assignment is requirements prioritized on an ordinal scale. However, the requirements in each group have the same priority, which means that each requirement does not get a unique priority.

### 2.2.4 Ranking

As in numerical assignment, ranking is based on an ordinal scale but the requirements are ranked without ties in rank. This means that the most important requirement is ranked 1 and the least important is ranked \( n \) (for \( n \) requirements). Each requirement has a unique rank (in comparison to numerical assignment) but it is not possible to see the relative difference between the ranked items (as in AHP or the 100-dollar test). The list of ranked requirements could be obtained in a variety of ways, as for example by using the bubble sort or binary search tree algorithms [30]. Independently of sorting algorithm, ranking seems to be more suitable for a single stakeholder because it might be difficult to align several different stakeholders' views. It is also possible to combine the different views by taking the mean priority of each requirement but this might result in ties for requirements which this method wants to avoid.
2.2.5 Top-Ten Requirements

In the top-ten requirements approach, the stakeholders pick their top-ten requirements (from a larger set) without assigning an internal order between the requirements. This makes the approach especially suitable for multiple stakeholders of equal importance [32]. The reason to not prioritize further is that it might create unnecessary conflict when some stakeholders get support for their top priority and others only for their third priority. One could assume that conflicts might arise anyway if, for example, one customer gets three top-ten requirements into the product while another gets six top-ten requirements into the product. However, it is important to not just take an average across all stakeholders since it might lead to some stakeholders not getting any of their top requirements [32]. Instead, it is crucial that some essential requirements are satisfied for each stakeholder. This could obviously result in a situation that dissatisfies all customers instead of satisfying a few customers completely. The main challenge in this technique is to balance these issues.

2.2.6 Which Prioritization Technique to Choose

Table 2.1 summarizes the presented prioritization techniques based on measurement scale, granularity of analysis, and level of sophistication of the technique.

<table>
<thead>
<tr>
<th>Technique</th>
<th>Scale</th>
<th>Granularity</th>
<th>Sophistication</th>
</tr>
</thead>
<tbody>
<tr>
<td>AHP</td>
<td>Ratio</td>
<td>Fine</td>
<td>Very Complex</td>
</tr>
<tr>
<td>100-dollar test</td>
<td>Ratio</td>
<td>Fine</td>
<td>Complex</td>
</tr>
<tr>
<td>Ranking</td>
<td>Ordinal</td>
<td>Medium</td>
<td>Easy</td>
</tr>
<tr>
<td>Numerical Assignment</td>
<td>Ordinal</td>
<td>Coarse</td>
<td>Very Easy</td>
</tr>
<tr>
<td>Top-ten</td>
<td>-</td>
<td>Extremely Coarse</td>
<td>Extremely Easy</td>
</tr>
</tbody>
</table>

A general advice is to use the simplest appropriate prioritization technique and use more sophisticated ones when a more sensitive analysis is needed for resolving disagreements or to support the most critical decisions [38]. As more sophisticated techniques generally are more time consuming, the simplest possible technique ensures cost effective decisions. The trade-off is to decide exactly how "quick and dirty" the approach can be without letting the quality of the decisions suffer. It should also be noted that there exist
2.2.7 Combining Different Techniques

The techniques in Table 2.1 represent the most commonly referenced quantitative prioritization techniques. It is possible to combine some of them to make prioritization easier or more efficient. Some combinations of the above techniques exist and probably the best known example is Planning Game (PG) in eXtreme Programming (XP) [2]. In PG, numerical assignment and ranking are combined by first dividing the different requirements into priority groups and then ranking requirements within each group (more about this method in Chapter 5). Requirements triage is an approach where parallels are drawn to medical treatment at hospitals [15]. Medical personnel divide victims into three categories: those that will die whether treated or not, those who will resume normal lives whether treated or not, and those for whom medical treatment may make a significant difference. In requirements prioritization, there are requirements that must be in the product (e.g. platform requirements), requirements that the product clearly need not satisfy (e.g. very optional requirements), and requirements that need more attention. This means that the requirements are assigned to one of three groups (numerical assignment) and requirements that need more attention are prioritized by any of the other techniques (AHP, ranking, 100 points etc.). In this approach, not all requirements must be prioritized by a more sophisticated technique, which decreases the effort.

The two examples above show that it is possible to combine different techniques for higher efficiency or to make the process easier. Which method or combination of methods is suitable often depends on the individual project.

2.3 Stakeholders in the Prioritization Process

As mentioned in Chapter 1, market-driven and bespoke software development has some different characteristics. The similarities and differences between market-driven and bespoke development also apply when prioritizing software requirements. In a bespoke project, only one or a few stakeholders must be taken into consider-
ation while everyone in the whole world might serve as potential customers in market-driven development. Table 2.2 outlines some of the differences between market-driven and bespoke development that affects requirements prioritization.

As can be seen in Table 2.2, there are large differences between these two extremes and different projects have to consider different ways to handle, and hence prioritize, requirements. Table 2.2 shows the two extremes in software development; a real case probably falls somewhere in between. For example, it is possible that a company delivers for a market, but the market is limited to a small number of customers (e.g. telecommunication systems are only bought by telephone operators). The discussion here focuses on three different "general" scenarios: one customer, a number of "known" customers, and a mass-market.

Table 2.2: Differences Between Market-driven and Bespoke Development [9]

<table>
<thead>
<tr>
<th>Facet</th>
<th>Bespoke Development</th>
<th>Market-driven Development</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main Stakeholder</td>
<td>Customer organization</td>
<td>Developing organization</td>
</tr>
<tr>
<td>Users</td>
<td>Known or identifiable</td>
<td>Unknown, may not exist until product is on market</td>
</tr>
<tr>
<td>Distance to users</td>
<td>Usually small</td>
<td>Usually large</td>
</tr>
<tr>
<td>Requirement Conception</td>
<td>Elicited, analyzed, validated</td>
<td>Invented (by market pull or technology push)</td>
</tr>
<tr>
<td>Lifecycle</td>
<td>One release, then maintenance</td>
<td>Several releases as long as there is a market demand</td>
</tr>
<tr>
<td>Specific RE issues</td>
<td>Elicitation, modeling, validation, conflict resolution</td>
<td>Steady stream of requirements, prioritization, cost estimating, release planning</td>
</tr>
<tr>
<td>Primary goal</td>
<td>Compliance to specification</td>
<td>Time-to-market</td>
</tr>
<tr>
<td>Measure of success</td>
<td>Satisfaction, acceptance</td>
<td>Sales, market share</td>
</tr>
</tbody>
</table>

2.3.1 One Customer

In a one customer situation, there is only one customer's priorities that needs to be considered (from the customer/user perspective). Many of the present software development processes are based on one customer and assume that this customer is available throughout the project [9]. For example, eXtreme Programming has an "on-site customer" as one of the core practices (the focus is on having one customer even though this customer could represent a market) [2]. One important issue to consider when having a one-customer situation is that the customer and the end-user(s) are not always the
same. In this case, the person who prioritizes and the persons who will use the system may not have the same priorities [21]. Such situations are of course undesirable since it may result in reduced use of the product. In this case, it would be better to involve the end-users in prioritizing the requirements since they are the ones who know what they need. For example, if the customer is an employer, and the user is an employee of the company buying the product, this may result in conflicts. It is possible to imagine features that are desirable to an employer, but not an employee.

2.3.2 Several Known Customers

When having several customers, the issue of prioritization becomes more difficult since the customers may have conflicting viewpoints and preferences [1]. This introduces the challenge of drawing these different customer views together [34]. The ultimate goal in these situations is to create win-win conditions and make every stakeholder a "winner" [4]. If one perspective is neglected the system might be seen as a failure by one or several of the stakeholders [1]. Hence, it is of tremendous importance that all stakeholders are involved in this process since the success of the product ultimately is decided in this step. A discussion on how to make trade-offs between different stakeholders is provided in Section 2.3.5.

2.3.3 Mass-Market

When developing for a mass-market, it is not possible to get all customers to prioritize. When eliciting information for prioritization in a mass-market situation, different sources exist [31]: internal records (e.g. shipments, sales records), marketing intelligence (e.g. information from sales force, scientists), competitor intelligence (e.g. information about competitors' strategies, benchmarking competitors' products) and marketing research (e.g. surveys, focus groups). When conducting marketing research, the sample must be representative for the intended market segment (group of consumers with similar needs) [31]. For example, if developing products for large companies, it is meaningless to involve small companies in the focus groups or the surveys. Hence, it is very important to decide which market segments should be the focus of the product before performing the prioritization.

The result from a prioritization for a mass-market product could provide a good base for analyzing which requirements are high pri-
Priorities for all different market segments. By using this information, it is possible to identify which parts of a system should be common for all market segments and which parts should be specifically developed for specific market segments. This way of dealing with requirements is valuable when developing software product lines [12].

One way of dealing with the problem that all possible users are not known or accessible is to use the concept of 'personas' that originated in marketing and has been used in system design [22]. These 'personas' are fictional persons, representing market segments. They have names, occupations, possessions, age, gender, socioeconomic status, etc. Further, they are based on and inspired by real people that are supposed to use the developed product. This information is gathered from ethnographies, market research, usability studies, interviews, observations, and so forth. The intention is to help the developing organization focus the attention on 'personas' that the system is and is not designed for, and to give an understanding of these target 'personas'. Further, 'personas' enhance engagement and reality by providing fictional users of the system. The developing organization can use the 'personas' in decision-making (and prioritization) through asking questions like:

- Why are we building this feature (requirement)?
- Why are we building it like this?

When having such explicit but fictive users of the system, the organization can get an understanding of which choices the 'personas' would make in different situations.

### 2.3.4 Stakeholders Represented in the Prioritization

Since requirements can be prioritized from several different aspects, different roles must also be involved in the prioritization process to get the correct views (e.g. product managers prioritize strategic importance and project managers prioritize risks). At least three perspectives should always be represented: customers, developers, and financial representatives [15]. Each of these stakeholders provides vital information that the other two may neglect or are unable to produce since customers care about the user/customer value, developers know about the technical difficulties, and financial representatives know and care for budgetary constraints and risks [15]. Nevertheless, it is of course suitable to involve all perspectives (beside these three) that have a stake in the project or product.
2.3.5 Trade-Off between Different Stakeholders

In both market-driven and bespoke projects, there can be several different stakeholders with different priorities and expectations of the system. How to make trade-offs between several stakeholders with different priorities is an issue that is commonly mentioned as a problem by product managers in software organizations. First, this could be a problem when having one or a few very strong stakeholders since their wishes are often hard to neglect (i.e. when the big customer says jump, the company jumps). Second, "squeaky wheel" customers often get what they want [34][54].

In such situations, it is important to have a structured way of handling different stakeholders. Regnell et al. adjusts the influence of each stakeholder by prioritize them for different aspects [41]. This can be considered by weighting market segments based on for example: revenue last year, profit last release, size of total market segment, number of potential customers, etc. The weighting aspects depend on the strategy most suitable in the current market phase ([39]). Priorities are then used to weigh each stakeholder in the prioritization process. This approach is also possible when dealing with specific stakeholders even though the aspects on which the priorities are based might be different.

The weighting of the stakeholders could be performed in the same way as ordinary prioritization, and the techniques described in Section 2.2 could be used to provide the weights (preferably the techniques based on a ratio scale since these will provide distances of importance between the stakeholders).

2.4 Using Requirements Prioritization

Requirements prioritization needs to consider several different aspects, techniques, and stakeholder situations. This section presents additional issues to consider and ways of dealing with such issues.

2.4.1 Abstraction Level

Requirements are commonly represented at different levels of abstraction [20], which causes problems when prioritizing requirements. One reason is that requirements on higher abstraction levels tend to get higher priority in pair-wise comparisons [35]. For exam-
Requirements Prioritization

2

Deciding on the level of abstraction can be difficult and depend very much on the number of requirements and their complexity. With a small number of requirements, it might be possible to prioritize the requirements at a low level of abstraction while it might be a good idea to start with requirements at a high level and prioritize lower levels within the higher levels later when having many requirements to prioritize [53]. AHP supports this approach of decomposing requirements into different hierarchical levels in order to decrease the number of comparisons. In other cases, it might even be a good idea to just prioritize the high level requirements, and then letting the subordinate requirements inherit the priorities. If choosing this approach, it is important that all stakeholders are aware of this inheritance [53].

Regnell et al. discuss the problem of having a lot of requirements to prioritize [41]. They grouped the requirements to make the prioritization easier. The requirements were divided into a low level (original requirements) and a higher level (requirements were grouped based on relationships). This approach not only reduces the number of requirements to prioritize but also deals with dependencies of requirements [46]. Grouping requirements based on requirements dependencies (e.g. which requirements must be implemented together) would make further analysis of the requirements easier since requirements that are grouped together would not compete for priorities. According to the result of the study, forming coherent groups was easy and the stakeholders successfully prioritized at both levels.

2.4.2 Reprioritization

When developing software products, it is likely that new requirements will arrive, requirements are deleted, priorities of existing requirements change, or that the requirements themselves change [21][35]. Hence, it is of tremendous importance that the prioritization process is able to deal with changing requirements and priorities of already prioritized requirements. When prioritizations are on
an ordinal (e.g. ranking and numerical assignment) or absolute scale (estimating cost) this does not introduce any major problems since the new or changed requirement just need to be assigned a value, or a correct priority. Such iterations of the numerical assignment technique have been used successfully [15].

When using prioritization on a ratio scale (such as AHP), the situation becomes more complex since all requirements should be compared to all others to establish the correct relative priorities. However, it is possible to tailor this process by comparing new or modified requirements with certain reference requirements and thereby estimating the relative value. For example, when using the 100-dollar test it is possible to identify the two requirements with higher and lower ranking, and then establish the relative value in comparison to these and normalize the weights (of the complete requirements set). However, this means that the original process is not followed and the result might differ from a complete reprioritization even though the cost versus benefit of such a solution might be good enough. Cost and benefit must be taken into consideration when choosing a prioritization technique.

Further, it is important to not forget that priorities of already implemented requirements can change; especially non-functional requirements. Techniques such as gap-analysis (see Section 2.4.5) could be successfully used to prioritize already implemented requirements in order to take these into account in a reprioritization.

### 2.4.3 Non-Functional Requirements

Previously in this chapter, no differences in analyzing functional and non-functional (quality attributes) requirements have been discussed. The previously presented methods can be used with both kinds of requirements and sometimes it is preferable to prioritize them together. Nevertheless, it is not always advisable to prioritize functional and non-functional requirements together, for the same reasons that requirements at different abstraction levels should not be prioritized together. Differences between functional and non-functional requirements include, but are not limited to [32][43][52]:

- Functional requirements usually relate to specific functions while non-functional requirements usually affect several functions (from a collection of functions to the whole system).
• Non-functional requirements are properties that the functions or system must have, implying that non-functional requirements are useless without functional requirements.

• When implemented, functional requirements either work or not while non-functional requirements often have a "sliding value scale" of good and bad.

• Non-functional requirements are often in conflict with each other, implying that trade-offs between these requirements must be made.

Thus, it is not always possible, or advisable, to prioritize both types of requirements together. For example, if there is one functional requirement about a specific function and one non-functional requirement regarding performance, it could be hard to prioritize between them. In such cases, it is possible to prioritize them separately with the same or even with different techniques. Some techniques exist that are especially suitable for prioritizing non-functional requirements. One such approach (originating from marketing) is conjoint analysis where different product alternatives are prioritized based on the definition of different attribute levels [19]. It should be noted that there does not seem to be a need to include all levels of all attributes (e.g. faster response time is always preferable). However, since trade-offs often are present when it comes to such attributes (e.g. maintainability vs. performance), one idea is to include comparisons where trade-offs are taken into consideration.

2.4.4 Introducing Prioritization into an Organization

As with other technology transfer situations, it is recommended to start small with one or a few of the practices (e.g. using numerical assignment to prioritize importance and cost) and then add more sophistication (and thereby complexity) as need and knowledge increase. Since introducing and improving prioritization is a form of process improvement, rules and guidelines for software process improvement should be applied (e.g. changes should be done in small steps and should be tested and adjusted accordingly [25]). A good idea could be to monitor future extensions by measuring process adherence and satisfaction of the involved stakeholders (both internally and externally). This way, it is possible to continuously measure the process and thereby determine when the process gets too heavy by calculating the cost versus benefit of each extension.
2.4.5 Evaluating Prioritization

Both for the reasons of improving and adjusting the prioritization process, and for improving and adjusting a product, it is necessary to evaluate the result of prioritizations in retrospect. For both purposes, it is important that information about the priorities is kept since these provide the best information for analyzing both the product and the process [34]. This includes information about both selected and discarded requirements from a release [42]. When having access to this information, it is possible to do post mortem analysis to evaluate if the correct requirements were selected and if they fulfilled the stakeholders' expectations. If they did not, it is possible to change the process and the product for subsequent products/releases to get better prioritizations and more satisfied stakeholders.

One way of evaluating if the correct priorities were assigned is through gap-analysis where the 'gap' between perceived level of fulfillment of a requirement and the importance of the requirement is calculated [24]. The result shows how well each requirement, or type of requirement, is fulfilled according to how important the stakeholders think the requirements are. In this case, the requirements with the largest gaps get the highest priorities for improvement (PFI) [24]. This makes it possible to improve parts of the product with a low level of fulfillment, but it could also be used to tune the process to avoid such situations again.

2.4.6 Using the Results of Requirements Prioritization

The results of a prioritization exercise must be used judiciously [35]. Dependencies between requirements should be taken into consideration when choosing which requirements to include. Dependencies could be related to cost, value, changes, people, competence, technical precedence, etc. [14][45]. Such dependencies might force one requirement to be implemented before another, implying that it is not possible to just follow the prioritization list. Another reason for not being able to solely base the selected requirements on the priority list is that when the priority list is presented to the stakeholders, their initial priority might have emerged incorrectly [35]. This means that when the stakeholders are confronted with the priority list, they want to change priorities. This is a larger problem in techniques where the result is not visible throughout the process (e.g. AHP).

The product may have some naturally built-in constraints. For example, projects have constraints when it comes to effort, quality,
duration, etc. [46]. Such constraints makes the selection of which requirements to include in a product more complex than if the choice were solely based on the importance of each requirement. A common approach to make this selection is to propose a number of alternative solutions from which the stakeholders can choose the one that is most suitable based on all implicit context factors (e.g. [21][34][44][46][53]). By computerizing the process of selecting nominated solutions, it is possible to focus the stakeholders' attention on a relatively small number of candidate solutions instead of wasting their time by discussing all possible alternatives [17]. In order to automate and to provide a small set of candidate solutions to choose from, it is necessary to put some constraints on the final product. For example, there could be constraints that the product is not allowed to cost more than a specific amount, the time for development is not allowed to exceed a limit, or the risk level is not allowed to be over a specific threshold.

2.5 An Example of a Requirements Prioritization

To illustrate the different aspects, prioritization techniques, trade-offs between stakeholders, and combinations of prioritization techniques and aspects, an example of a prioritization situation is given. The method used in this example is influenced by a model proposed by Wiegers but is tailored to fit this example [53]. The example analyses 15 requirements (R1-R15) in a situation with three known customers (see 4.5.2). The analysis is rather sophisticated to show different issues in prioritization but still simple with a small amount of requirements. While many more requirements are common in industry, it is easier to illustrate how the techniques work on a smaller example. Each of the 15 requirements is prioritized according to the different aspects presented in Section 2.1. Table 2.3 presents the aspects that are used in the example together with the method that is used to prioritize the aspect and from which perspective it is prioritized.

As can be seen in Table 2.3, all prioritization techniques presented in Section 2.2 are used. However, two clarifications are in order. First, numerical assignment for time (7) and risk (3) uses a different number of groups to show varying levels of granularity. The customer importance is prioritized both by the top-ten technique and the 100-dollar technique depending how much time and cost the different customers consider reasonable.
As can be seen in Table 2.3, all prioritization techniques presented in Section 2.2 are used. However, two clarifications are in order. First, numerical assignment for time (7) and risk (3) uses a different number of groups to show varying levels of granularity. The customer importance is prioritized both by the top-ten technique and the 100-dollar technique depending how much time and cost the different customers consider reasonable.

To make the prioritizations more effective, requirements are further refined. First, requirements R1 and R2 are requirements that are absolutely necessary to get the system to work at all. Hence, they are not prioritized by the customers but they are estimated when it comes to cost, risk, etc. since R1 and R2 influence these variables no matter what. This is a way of using the requirements triage approach presented in Section 2.2.7. Further, two groups of requirements have been identified as having high dependencies (must be implemented together) and should hence be prioritized together. Requirements R3, R4, and R5 are grouped together as R345, and requirements R6 and R7 are grouped into R67.

The next step is to prioritize the importance of the requirements. In the case at hand, the three known customers and the product manager prioritize the requirements. Furthermore, these four stakeholders are assigned different weights depending on how important they are deemed by the company. This is done by using the 100-dollar test (see Section 2.3.5) to get the relative weights between the stakeholders (the weights are recalculated from 0-100 to 0-1 in this example). Table 2.4 presents the result of the prioritization. In the

---

**Table 2.3: Aspects to Prioritize**

<table>
<thead>
<tr>
<th>Aspect</th>
<th>Prioritization Technique</th>
<th>Perspective</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strategic Importance</td>
<td>AHP</td>
<td>Product Manager</td>
</tr>
<tr>
<td>Customer Importance</td>
<td>100-dollar / Top-ten(^a) Requirements</td>
<td>Customers</td>
</tr>
<tr>
<td>Penalty</td>
<td>AHP</td>
<td>Product Manager</td>
</tr>
<tr>
<td>Cost</td>
<td>100-dollar</td>
<td>Developers</td>
</tr>
<tr>
<td>Time</td>
<td>Numerical Assignment (7)</td>
<td>Project Manager</td>
</tr>
<tr>
<td>Risk</td>
<td>Numerical Assignment (3)</td>
<td>Requirements Specialist</td>
</tr>
<tr>
<td>Volatility</td>
<td>Ranking</td>
<td>Requirements Specialist</td>
</tr>
</tbody>
</table>

\(^a\) The top-ten technique is modified to a top-four technique in this example due to the limited number of requirements

---

As can be seen in Table 2.3, all prioritization techniques presented in Section 2.2 are used. However, two clarifications are in order. First, numerical assignment for time (7) and risk (3) uses a different number of groups to show varying levels of granularity. The customer importance is prioritized both by the top-ten technique and the 100-dollar technique depending how much time and cost the different customers consider reasonable.

To make the prioritizations more effective, requirements are further refined. First, requirements R1 and R2 are requirements that are absolutely necessary to get the system to work at all. Hence, they are not prioritized by the customers but they are estimated when it comes to cost, risk, etc. since R1 and R2 influence these variables no matter what. This is a way of using the requirements triage approach presented in Section 2.2.7. Further, two groups of requirements have been identified as having high dependencies (must be implemented together) and should hence be prioritized together. Requirements R3, R4, and R5 are grouped together as R345, and requirements R6 and R7 are grouped into R67.

The next step is to prioritize the importance of the requirements. In the case at hand, the three known customers and the product manager prioritize the requirements. Furthermore, these four stakeholders are assigned different weights depending on how important they are deemed by the company. This is done by using the 100-dollar test (see Section 2.3.5) to get the relative weights between the stakeholders (the weights are recalculated from 0-100 to 0-1 in this example). Table 2.4 presents the result of the prioritization. In the
As can be seen in this Table 2.4, the different stakeholders have different priorities, and it is possible to combine their different views to an overall priority. The weights (within parenthesis after each stakeholder) represent the importance of each customer and in this case, the product manager is assigned the highest weight (0.35). This is very project dependent. In this case, the mission of this product release is to invest in long-term requirements and attract new customers at the same time as keeping existing ones. As also can be seen, C1 used the top-ten technique and hence the priorities were evenly divided between the requirements that this customer regarded as most important. The list to the far right presents the final priority of the requirements with the different stakeholders and their weights taken into consideration. This calculation is possible since a ratio scale has been used instead of an ordinal scale.

The next step is to prioritize based on the other aspects. In this case, the Priority from Table 2.4 is used to express Importance in Table 2.5. It should also be noted that requirements R1 and R2 (absolutely necessary) have been added in Table 2.5. Table 2.5 shows a prioritized list of our requirements (based on IP). With this information there are two options: 1) pick prioritized items from the top of the list until the cost constraints are reached, 2) analyze table, the three customers are denoted C1-C3 and the product manager is denoted PM.
further based on other prioritized aspects, if prioritizations of additional aspects are available. The example has two major constraints: 1) the project is not allowed to cost more than 65% of the total cost of the elicited requirements, and 2) the median risk level of the requirements included is not allowed to be higher than 2.5.

Based on the constraints in the example, we first try to include the requirements with the highest IP. The result of this is presented in Table 2.6 where the list was cut when the sum of costs reached 65% of the total cost of elicited requirements.

### Table 2.5: Descending Priority List Based on Importance and Penalty (IP).

\[
IP(R_X) = RP_I \times W_I + RP_P \times W_P
\]

where \( RP \) is the Requirement Priority, and \( W \) is the Weight of Importance (I) and Penalty (P).

<table>
<thead>
<tr>
<th>Requirement</th>
<th>I (0.7)</th>
<th>P (0.3)</th>
<th>IP</th>
<th>Cost</th>
<th>Time</th>
<th>Risk</th>
<th>Volatility</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.11</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>R2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.13</td>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>R8</td>
<td>0.19</td>
<td>0.2</td>
<td>0.2</td>
<td>0.07</td>
<td>1</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>R67</td>
<td>0.19</td>
<td>0.09</td>
<td>0.16</td>
<td>0.10</td>
<td>6</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>R10</td>
<td>0.18</td>
<td>0.01</td>
<td>0.13</td>
<td>0.24</td>
<td>2</td>
<td>3</td>
<td>11</td>
</tr>
<tr>
<td>R14</td>
<td>0.1</td>
<td>0.16</td>
<td>0.12</td>
<td>0.01</td>
<td>1</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>R345</td>
<td>0.11</td>
<td>0.02</td>
<td>0.08</td>
<td>0.03</td>
<td>3</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>R9</td>
<td>0.06</td>
<td>0.12</td>
<td>0.08</td>
<td>0.05</td>
<td>5</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>R15</td>
<td>0.03</td>
<td>0.17</td>
<td>0.08</td>
<td>0.05</td>
<td>5</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>R12</td>
<td>0.06</td>
<td>0.06</td>
<td>0.06</td>
<td>0.11</td>
<td>4</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>R11</td>
<td>0.02</td>
<td>0.14</td>
<td>0.06</td>
<td>0.02</td>
<td>3</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>R13</td>
<td>0.05</td>
<td>0.03</td>
<td>0.05</td>
<td>0.04</td>
<td>7</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>Total/Median:</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

### Table 2.6: Selected Requirements Based on IP and Cost.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>IP</th>
<th>Cost</th>
<th>IP/Cost</th>
<th>Time</th>
<th>Risk</th>
<th>Volatility</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>1</td>
<td>0.11</td>
<td>9.09</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>R2</td>
<td>1</td>
<td>0.13</td>
<td>7.69</td>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>R8</td>
<td>0.2</td>
<td>0.07</td>
<td>2.8</td>
<td>1</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>R67</td>
<td>0.16</td>
<td>0.1</td>
<td>1.59</td>
<td>6</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>R10</td>
<td>0.13</td>
<td>0.24</td>
<td>0.54</td>
<td>2</td>
<td>3</td>
<td>11</td>
</tr>
<tr>
<td>Total/Median:</td>
<td>2.48</td>
<td>0.65</td>
<td>21.71</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.6 shows that we managed to fit within the cost constraints but could not satisfy the risk constraint. As a result, the project
becomes too risky. Instead, another approach is taken to find a suitable collection of requirements. In this approach, we take the IP/Cost ratio into consideration. This shows which requirements provide most IP at the least cost. In this case, we try to set up a limit of only selecting requirements that have an IP/Cost-ratio higher than 1.0. The result is presented in Table 2.7

Table 2.7: Selected Requirements Based on Cost and IP/Cost Ratio.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>IP</th>
<th>Cost</th>
<th>IP/Cost</th>
<th>Time</th>
<th>Risk</th>
<th>Volatility</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>1</td>
<td>0.11</td>
<td>9.09</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>R2</td>
<td>1</td>
<td>0.13</td>
<td>7.69</td>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>R8</td>
<td>0.2</td>
<td>0.07</td>
<td>2.8</td>
<td>1</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>R67</td>
<td>0.16</td>
<td>0.1</td>
<td>1.59</td>
<td>6</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>R14</td>
<td>0.12</td>
<td>0.01</td>
<td>11.7</td>
<td>1</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>R345</td>
<td>0.08</td>
<td>0.03</td>
<td>2.71</td>
<td>3</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>R15</td>
<td>0.08</td>
<td>0.05</td>
<td>1.5</td>
<td>5</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>R11</td>
<td>0.06</td>
<td>0.02</td>
<td>2.94</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>R13</td>
<td>0.05</td>
<td>0.04</td>
<td>1.17</td>
<td>7</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td><strong>Total/Median:</strong></td>
<td><strong>2.73</strong></td>
<td><strong>0.56</strong></td>
<td><strong>41.19</strong></td>
<td><strong>3</strong></td>
<td><strong>2</strong></td>
<td></td>
</tr>
</tbody>
</table>

Table 2.7 shows the cost constraints are still met (even nine percent less cost) while also satisfying the risk constraint. Comparing Tables 2.6 and 2.7 shows that the IP-value of the second candidate solution is higher which indicates that the customers are more satisfied with the product and the IP/Cost ratio is almost doubled. The second candidate solution satisfies 91 percent (2.73/3) of the IP aspect, compared to 83 percent in the first candidate solution. The fact that the second alternative costs less and is less risky also favors this choice. Nevertheless, the above example is not optimal since cost was constrained at 0.65 and other combinations of requirements may be more optimal for the selection.

This type of release planning is known in operational research as the binary knapsack problem [11]: maximize value when the selection is bounded by an upper limit. However, the difference between a classical knapsack problem and the problem faced above is that release planning is a "wicked problem" [11]. This means that an optimal solution may not exist, that every release planning is unique, that no objective measure of success exists, etc. [11]. In addition, the values of the aspects in the above example are estimates and subjective measures in comparison to objective measures such a length, weight, and volume. Instead of finding the optimal
set, different alternative solutions should be discovered and the alternative that seems most suitable should be chosen [11]. This implies that the purpose with prioritization is not to come up with a list of final requirements, but rather to provide support for good decisions. In comparison to the above example, real projects generally have more requirements, and more complex dependencies [11]. However, this example was meant to show how different aspects can be used to handle trade-offs between different (sometimes conflicting) aspects. It is also possible, as illustrated, to fine-tune an existing technique or method to suit a company specific situation.

2.6 Summary

This presents a number of techniques, aspects, and other issues that should be thought of when performing prioritizations. These different parts together form a basis for systematically prioritizing requirements during software development. The result of prioritizations suggests which requirements should be implemented, and in which release. Hence, the techniques could be a valuable help for companies to get an understanding of what is important and what is not for a project or a product. As with all evaluation methods, the results should be interpreted and possibly adjusted by knowledgeable decision-makers rather than simply accepted as a final decision.

This chapter primarily focuses on requirements prioritization as an activity in product development. However, as discussed in Chapter 1, several similarities between product and process development exist. This indicates that it is possible to use several of the above discussed techniques and recommendations when prioritizing requirements in process development and improvement. The next two chapters present a study where the 100-dollar test is used as prioritization technique for prioritizing software process improvement issues. In these chapters, it is shown that it is suitable to use prioritization also in process development, and different stakeholders have different priorities as well as personal agendas.
2.7 References


The most serious problems in software organizations typically concern organizational procedures and cultural behavior [12]. These problems are not something individuals within the organization generally can fix themselves. Therefore, a comprehensive and long-term focus on the software process is required to solve them [12]. Further, to compete well in today’s marketplace, it is a pre-condition to have best-practice engineering standards in place, measuring the conformance, and continually trying to improve [6].

Despite the need of focusing on the processes, users of processes often have an inner resistance against defined processes and the change of processes [5]. It is also common that the defined processes are disregarded or deviated from (e.g. [3], [4]). To solve these problems, knowledge about important factors to consider is required. One of the reasons for having defined processes is to be able to share knowledge in an organization by communicating this knowledge through documented processes [17]. Therefore, people working with processes should reuse the knowledge others have gained in process related work.

This chapter presents a case study where the key factors for successful management and evolution of the software process were studied. The study was launched as it was viewed as essential to identify key success factors for managing the software process
before introducing an improvement program. The key factors are found through an empirical case study involving three parts: a qualitative part (interviews), literature part (survey), and a quantitative part (questionnaire).

The remainder of this chapter is structured as follows. Section 3.1 outlines how the empirical study was performed. Section 3.2 presents the results that were obtained within each part of the study. In Section 3.3, a combined analysis of the three parts is presented and the factors found are ranked in relation to each other. Section 3.4 discusses the implications of the findings and what conclusions that could be drawn. Finally, Section 3.5 summarizes the results and discusses further work within the area.

3.1 Method

The objective of this chapter is to identify key factors for managing the software process. The factors are identified through analyst and method triangulation as described in Chapter 1, i.e. the use of several different approaches to identify the key factors. Moreover, the intention is to study whether the factors identified in a specific software developing organization are also the factors that are found in literature. The triangulation is done by combining the three parts of the empirical study to retrieve one overall result.

The three different parts constituting the overall study were done and combined as follows. First, a number of interviews were conducted and the key factors for software process management, according to these interviews, were identified. Second, a literature survey was done and key factors for successful software process management and improvement according to literature were identified. Third, a questionnaire was used to capture the viewpoint of the whole organization. The questionnaire was based on the identified factors from the interviews and the literature survey. Finally, the results were combined by identifying the most important factors from the questionnaire and relating that to the findings with respect to these factors in the interviews and the literature survey respectively. The three parts were regarded as equally important during this process and no weights were assigned to make certain parts more important than others. The result of this process was the identification of a set of key factors for software process management.
In this section, the goal of the study is firstly introduced in Section 3.1.1. The different parts of the study are then presented in the subsequent three sections with the interviews in Section 3.1.2, the literature survey in Section 3.1.3 and the questionnaire in Section 3.1.4. Finally, the validation using method triangulation is further discussed in Section 3.1.5.

3.1.1 Goal of Empirical Study

The initial goal of the empirical study was to get an understanding of the processes in the studied organization. The intention was to investigate how people perceived the defined processes and to do a mapping of which defined processes that are useful and what problems that are encountered in others. It was also the aim to assess how mature the processes were in the organization. A side effect of this work was that several factors were identified as key factors in the organization. Due to that the design was not primarily intended to find such factors, the six identified key factors were never compared with each other in the questionnaire.

3.1.2 Qualitative Part (Interviews)

The interviews focused on one part of the software process, namely the requirements engineering process. The interviews were part of a project aiming at evaluating the current requirements engineering process and identifying potential improvements. In the interviews, the objective was to capture the tacit knowledge about the processes that resides with the co-workers.

The interview questions were fairly open and there were much room for discussions about the factors that were mentioned. 25 persons from all operative departments within the organization were interviewed. These persons were selected by asking the functional manager of each department to assign personnel for the interviews. The result was that all departments were not represented by the same amount of personnel. Three persons were part of the team that conducted the interviews: one employee within the organization, one external consultant, and one researcher.

3.1.3 Literature Part (Survey)

A detailed literature survey was conducted after the interviews. The main reason for this was that both researchers (i.e. the authors of
this chapter) have general knowledge in the area of process management. Performing the literature survey prior to the interviews could lead to that we were “fishing” for the key factors identified in literature. The general (but not in-depth) knowledge about processes implied that no preconceived thoughts about exactly which factors that were key factors were present. The intention was that the interviews should capture the viewpoints of the personnel and hence the discussions should not be tainted by the factors that can be found in the literature.

In addition, the literature survey was performed after the interviews for two reasons. First, it was possible to see whether the problems mentioned in the interviews also were mentioned in the literature about processes. Second, it was possible to find out if there were additional concerns mentioned in the literature that were not mentioned in the interviews. The search criteria that were used in the literature survey were based on the interviews as well as the general knowledge in processes by the researchers. This pre-knowledge resulted in that additional factors were found in the literature.

### 3.1.4 Quantitative Part (Questionnaire)

The objective of the questionnaire was to obtain a larger sample and also a more representative sample for the whole organization than in the interviews. The questionnaire was formulated based on the findings in the interviews and the additional information obtained from the literature survey.

The questionnaire consisted of 23 questions, plus demographics and one open-ended question. The first 18 questions were multiple-choice questions. Questions 19-23 were weighted questions where the respondents had the possibility to rank their answers. In addition to ordinary ranking, they had the possibility to put weights on their answers by apportioning 100 points between the answering-alternatives (see Chapter 2 for details about the 100-dollar test). This made it possible to see how much more important some factors were than others (in contrast to ordinary ranking where you can only see that one alternative is more important than another).

The questionnaire was sent out to 84 persons in the organization (not the same persons as in the interviews). 65 persons answered the questionnaire, which is equivalent to 17 percent of the organization (software development part), and 77 percent of the sample.
This sample was symmetrically distributed both vertically (i.e. management levels) and horizontally (e.g. departments, units).

### 3.1.5 Validation of Results

To avoid misinterpretations in the questionnaire, three persons (both from academia and the organization) reviewed the questions. Further, a pilot test was performed with three respondents from the organization to test the questionnaire. This test resulted in some minor adjustments of some questions and the order of the questions.

As mentioned, the objective was to ensure internal validity of the obtained results through triangulation. In Chapter 1, four different types of triangulation were presented. Two of these have been used to ensure validity of this study. First, analyst triangulation was used to verify that the results in the interviews were interpreted correctly. In this triangulation, the three interviewers discussed their individual results and reached consensus about the findings. Further, methods triangulation was used between the three conducted parts of the study, as presented in Figure 3.1 (the bidirectional arrows represent validation). This triangulation compared the results in order to validate that the result from each study corresponds to the other studies.

#### Figure 3.1: Methods Triangulation of Results.
3.2 Result

In this section, the results of the three parts of the study are presented. However, not all factors that were mentioned in the literature and the two empirical parts are presented here. Many more factors were discussed (e.g. training, goal setting, measurements, process staffing) but in order to keep the focus on the vital few, i.e. the key factors, only the factors being viewed as most important in the study are presented.

As can be seen in Figure 3.1, the interviews and the literature survey served as input to the questionnaire. The interviews explicitly found nine important factors. From the literature survey, it is possible to generate a long list of important factors (depending how they are counted) and an approximate number is around 14 factors. However, only factors that were found in several books or articles were included. When designing the questionnaire, the union of these two sources was used as a foundation. This resulted in approximately 14 factors (depending how they are counted) that were considered in the questionnaire. Hence any factor included in the questionnaire was either viewed as important in the interviews or in literature, or in both.

Based on the result from the three parts, four factors identified in the interviews and two factors identified in the literature were viewed as being key factors. The criteria for considering a factor as a key factor lies in the frequency and to what extent the factor was discussed in each part of the study, and that it was considered important in at least two parts. In the interviews and the literature part, the identified key factors were those that were mentioned the most, and mostly discussed. In the questionnaire it was those that were ranked highest.

In the listing below, it is in parenthesis indicated whether the factor initially were identified in the interviews or the literature survey:

- Change management (interviews), this factor is related to the necessity of keeping the defined process under configuration control and ensure that it is updated and is in accordance with the actual or proposed way of working.
- Synchronization (interviews), the process steps must be synchronized so that exit criteria, including documents, are consistent to entry criteria in a following step.
• Baselining the current way of working (interviews), process descriptions should be derived bottom-up, i.e. they should be based on the current way of working.

• Documentation (interviews), the documentation (e.g. granularity level, location) of the processes should be suited to the need of the users.

• User involvement (literature), the actual users of the process must be involved in the definition of a process. This is partially related to baselining the current way of working. However, it also includes that the users should be involved in the changes of a defined process.

• Management commitment (literature), management must provide money and resources to show their commitment and that process management is important.

Next, the support for these six factors in the three parts of the study is discussed.

### 3.2.1 Qualitative Part (Interviews)

Almost all of the respondents in the interviews expressed that they are in need of usable process descriptions. Several respondents felt that they did not see the whole picture of the development process, and hence they could not see what their contributions were. The impact was that motivation could suffer and they did not have a real commitment towards their work. Several factors were mentioned as areas where today’s process handling could be improved. In particular, the findings with respect to the six key factors are as follows.

#### 3.2.1.1 Change Management

The respondents argued that process descriptions often are made and then forgotten and not updated. As a side effect, the descriptions are not always used since they often do not reflect the current way of working but rather as the organization has worked in the past.

#### 3.2.1.2 Synchronization

The respondents argued that synchronization and integration between different processes is a must (e.g. synchronization between design/implementation and test processes). The respondents seemed to find it hard to see a continuum in the overall process because defined processes were not synchronized and integrated. Unsynchronized processes could for example mean that a first
phase has X as exit criteria while a following phase has X and Y as entry criteria, which results in the second phase having problems starting its work (due to the lack of Y).

3.2.1.3 Baselining the Current Way of Working
Baselining the current way of working was one of the factors most mentioned. Respondents argued that defined processes were “forced” on them and they had no active part in defining the process. This often meant that the process descriptions were not suited for their real working environment but rather were considered as an ideal picture. Hence, they were often neglected. Further, several respondents argued that the most urgent measure was to describe and baseline current work procedures; no improvement initiatives would have any effect until this was done. One respondent expressed himself rather clear: “The processes MUST be anchored in the real way of working”.

3.2.1.4 Documentation
Some respondents argued that it is important that process descriptions have the right level of detail. Often, the process descriptions were either too detailed (people stop thinking) and too specific to reuse, or too fuzzy, so that they had to be tailored so much for specific needs that they were not usable without much further work. The respondents argued that processes must be easy to adapt and be dynamic to suit different circumstances. Further, the respondents argued that the defined processes were too extensive for people to take their time to read them and there are too many descriptions of each process.

The main reasons for why the process descriptions were not followed were that they were hard to find, hard to understand and they did not fit the purpose, they argued. The essence of this part is what one of the respondents said: “The process must be an aid in the work instead of a load”.

3.2.1.5 User Involvement
The respondents of the interviews did not mention the involvement of users explicitly. However, this factor was hidden in other factors and several times it was implicitly expressed that they did not think that the future users of processes were involved enough in the development of defined processes (e.g. in baselining the current way of working).
3.2.1.6 Management Commitment

Management commitment was not mentioned explicitly in the study. Nevertheless, this factor was hidden in other factors (e.g. baselining, change management) and several respondents argued implicitly that they did not think that management gave them enough time and resources to improve their way of working.

3.2.2 Literature Part (Survey)

Based on the interviews, a literature survey was conducted to find the most important factors in process management, according to the literature in the field. Many factors were found in this survey. However, the discussion below presents only those six factors that were concerned as key factors in the overall study.

3.2.2.1 Change Management

The problem with change management occurs when the way of working changes but not the written process description. Conradi and Dybå [3] report that none of the twelve companies they investigated had any routines for updating defined processes. Curtis et al. [4] further argue that deviations from defined processes often originate from the lack of updating the defined processes. Gilb [6] also aligns to these statements and argues that defined processes must not be static when there is better know-how in the organization.

3.2.2.2 Synchronization

Synchronization was not a frequently mentioned factor in the literature. However, Humphrey [8] argues that the focus should be put on defining what input and output that are expected from a process/phase. Pfleeger [17] also aligns to this and states that a defined process could require design before coding but allow many different design techniques to be used. This means that the input and output of design and coding are defined, but coding does not care how the design was developed. Further, some work has been done about synchronization of more formal processes. However, the company investigated does not have highly formalized process descriptions and they do not intend to introduce more formal processes than the current.

3.2.2.3 Baselining the Current Way of Working

The majority of models that handles process improvement discuss the importance of baselining the current way of working (e.g. [9][13][15]). This means that the process description describes the current way of working (descriptively) rather than how the employ-
ees are supposed to work (prescriptively). This implies that the risk of describing an ideal process (which is a common reason for deviations) is reduced.

### 3.2.2.4 Documentation

Researchers within the field seem to agree that the documentation should be easy to read, and not too formalistic but rather useful and necessary (e.g. [1][3][6][18]). The main reason is that no matter how good the defined process is, people do not consider using it if they cannot understand the documentation or if it is too extensive.

The level of detail of a defined process should much depend on the people who will use the documents and how complex the task is [10]. Less experienced people need a lower abstraction level than more experienced people (e.g. [3][4]). Further, the level of detail also affects the level of formalism introduced in the defined process; a higher abstraction level provides more freedom to own decisions and creativity. Therefore, different levels of detail are desirable to satisfy the needs of all the process users and it is important that the process descriptions facilitate the process to be tailored to suit such different needs [4].

One important thing to remember is that defined processes should impose consistency and structure. Defined processes could be very flexible when the level of detail is right. Steltzer and Mellis [19] ranked tailoring processes as the fourth most important out of ten success factors, based on a study performed at 56 software companies.

### 3.2.2.5 User Involvement

Steltzer and Mellis [19] ranked user involvement factor as number two out of the ten most important success factors (for process improvement). Besides this study, several authors discuss the importance of user involvement (e.g. [1][8]). User involvement is primarily needed for two reasons. First, their first hand experience of the processes is valuable in the development (i.e. they know which part that is in most need of development/improvement). Second, participation reduces the resistance to the developed processes [19].

### 3.2.2.6 Management Commitment

Management Commitment was ranked as number one of the ten factors Steltzer and Mellis ranked [19]. Further, several authors discuss management commitment as very important (e.g. [7][14]). The
essence of management commitment is that management must provide time and resources for the staff if process initiatives should be successful. Management should also show that they believe in the initiative.

As described above, the material from the interviews and the literature survey was used as input to the design of the questionnaire. The results are presented below.

### 3.2.3 Quantitative Part (Questionnaire)

23 questions were asked in the questionnaire. Due to the large amount of data, the result of each individual question is not provided here. However, a general discussion that summarizes the answers relating to each key factor is provided. As indicated in Section 3.1.1, the initial intention of the study was not to identify these key factors, and hence the factors were not compared directly to each other. This means that it is hard to perform relevant statistical tests and care must be taken when running such tests.

#### 3.2.3.1 Change Management

A question about change management showed that 66 percent felt that process descriptions were updated too seldom or never. At the same time 25 percent felt that they were updated as they should while 9 percent thought that the descriptions were updated too often. Change management was also ranked higher than inconsistent process descriptions but was ranked lower than synchronization and baselines in one of the weighted questions.

#### 3.2.3.2 Synchronization

When the respondents got the question if different departments/units had an understanding for each other's work, 63 percent answered that they had not. Further, when they answered how well they thought that processes were synchronized, 41 percent answered bad (33%) or very bad (8%). 50 percent did answer that the processes “could be better” synchronized while only 9 percent answered that they were “good”. None of the respondents answered “very good” to this question.

This result is also shown in the answer to which factors that the respondents regarded as most important in a work description/process, where “workflow” (with entry and exit criteria for documents, artifacts etc.) was regarded as the single most important fac-
tor. This indicates that it was in the interfaces between processes the problems were located.

When the respondents got the question what they saw as the largest threats against successful process improvement they clearly indicated that “Processes are not synchronized between departments” was the most threatening factor. “Synchronization between processes” was also ranked as the most urgent problem to resolve.

Nevertheless, when the respondents were given the question if they discuss process-related faults with the people that are affected, 21 percent answered “always”, 34 percent answered “often” while 29 percent answered “sometimes”.

3.2.3.3 Baselining the Current Way of Working
When the respondents were asked to rank the urgency between seven problems they had encountered, they ranked “understand and document our current way of working” as the second most important factor, slightly after “synchronization between processes”. When the respondents were asked who they believed were the most active users of process descriptions, 17 percent answered “staff” while the rest answered “quality persons” (27%) or “management” (56%). This indicates that the defined processes are not a baseline to the work environment of the staff, but rather to management or quality persons’ work.

3.2.3.4 Documentation
To the question “how detailed are process descriptions today?” 37 percent answered that the descriptions had an adequate level, 42 percent answered that it depended on which description it was, while “too general” (16%) was a more common answer than “too detailed” (5%). This result shows that 63 percent of the respondents thinks that the documentation of processes could be improved, at least some of the descriptions. “Inconsistent process descriptions” was ranked after synchronization, change management, and user involvement to the question of how the relative threats against successful process improvement were divided. Further, tailoring of the processes and their documentation was ranked as the third most important factor of which problem that was the most urgent to resolve.
3.2.3.5 User Involvement
To the question “Who develops processes and process descriptions today?”, 50 percent answered that the staff were not involved, 8 percent did not know and 42 percent stated that the users were involved. To the follow-up question: “According to your opinion, who should develop processes and process descriptions?” 92 percent answered that the users should be involved. Further, when the respondents were asked about the most important factors in process improvement, “user involvement” was ranked as the most important.

3.2.3.6 Management Commitment
The respondents to the questionnaire generally felt (66%) that they did not have enough time devoted to develop structured solutions to problems. Further, 52 percent stated that they wanted to devote more or much more time to improvement activities while 37 percent wanted to devote the same amount of time and 11 percent wanted to devote less (8%), much less (0%), or no time (3%). When the respondents were asked about the most important factors in process improvement, they ranked “management commitment” as the second most important.

3.3 Combined Analysis
In the discussion below, the results that were obtained when triangulating the three studies are presented. Hence, the discussion focuses on relating the findings from each of the studies to each other. In Section 3.3.8, an analysis of how the results from the different studies relate to each other is presented to summarize the findings.

3.3.1 Change Management
The literature states that change management of defined processes is a key issue in process management. In both the interviews and the questionnaire, the respondents argued that process descriptions were updated too seldom. Further, the descriptions were not updated when the actual work methods were improved. Hence, they did not reflect the actual way of working. A good change management process should also facilitate the possibility to manage a process baseline [16] as discussed in Section 3.3.3.
3.3.2 Synchronization

The studied literature did not discuss synchronization very much even if some authors discuss the area to a certain extent. However, the interviews indicated that synchronization of processes was a problem within the organization. This was validated through the questionnaire where 91 percent answered that they were not satisfied with the synchronization of processes. This indicates that it is one of the largest problems in the studied organization. Further, one respondent in the open-ended question in the questionnaire argued that changes were made without consideration of the stakeholders in the process and in connected processes.

Respondents in the questionnaire answered that they discussed process related problems with other people. It seems like these discussions only relate to issues that are specific for the process/department. This response indicates that people generally are good at communicating within the departments, but they are not very good at discussing with people from other departments or processes. Kock [11] states that many problems are divided between several departments/processes and hence the interaction between departments/processes must be improved in order to gain the knowledge that are possessed by all involved parties.

3.3.3 Baselining the Current Way of Working

In the literature, baselining the process (descriptive modeling) is discussed as a key activity in process management [2]. The people in the investigated organization seem to support this statement, in both the interviews and the questionnaire. Further, people in the interviews and the open-ended question in the questionnaire, argued that defined processes were developed without involvement from the staff and without having the process well anchored in reality. The respondents of the questionnaire support this when they answered that the staff should be more involved in the development of processes. This issue is therefore tightly connected to the issue of user involvement in Section 3.3.5.

3.3.4 Documentation

Documentation was often mentioned as a problem in the interviews. The questionnaire showed that the “inconsistent process descriptions” was ranked after change management, synchronization, and user involvement as threat to successful process manage-
ment. However, a rather large part of the respondents answered that the quality of the documents varied between different descriptions. The literature also discusses documentation as a problem; most often, the documentation is too extensive without a reason (e.g. [6][18]). Adaptation (tailoring) of processes was discussed and considered as important during the interviews and in the literature survey. It was also rather highly ranked in the questionnaire after the already proven important factors: synchronization/interfaces between processes/departments, and baselines.

### 3.3.5 User Involvement

According to the questionnaire, the wrong persons develop the process descriptions in the investigated organization. The answers indicate that most people do not consider the staff as being a part of the development, but the majority of the respondents think the staff should be a part of this activity. These figures clearly indicate that the staff wants to be a more active part in the development of defined processes. One respondent that answered the open-ended question, further strengthens this argumentation when talking about the people developing processes at present: “these developers do not have an understanding of how people work in reality”, which means that if usable process descriptions should be developed, the affected staff should be a part of the development. The issue of having affected users involved is also discussed much in the literature, and was also mentioned implicitly in the interviews at a few occasions.

### 3.3.6 Management Commitment

Management commitment was a factor that was highly ranked in the questionnaire as well as in the literature studied. Further, one respondent argued in the open-ended question of the questionnaire that management tried to shortcut the process whenever a project became critical. This respondent further argued that it would be nice to hear that when a project becomes critical it is the most important time to follow processes; if management does not believe in the processes, their staff will not.

People in the interviews did not mention management commitment explicitly. However, time was mentioned several times in relation to other tasks. They argued that they did not get the time they wanted.
to improve their work but all their time was assigned to projects that produced products.

The questionnaire clearly indicated that not enough time is devoted to structured solutions to problems. This shows that the staff really wants to devote time to process activities, but management does not provide the right amount of time and resources.

### 3.3.7 Validity

The internal validity of the presented study is addressed by triangulating (see Section 3.1.5) three different sources of information: interviews, a literature survey and a questionnaire. Moreover, the rather large sample size and an even spread across the organization help ensuring the internal validity. The results from the study has also been presented and discussed in-depth in the organization to validate that the results are interpreted correctly.

The external validity (i.e. the degree of generalization) from a case study is much more difficult to determine. However, it should be noted that most of the success factors identified in the organization were supported by the literature, which indicates that the results are not specific for the organization. Moreover, the size of the organization and that it delivers products on a world market (in contrast to a small domestic consultant company) at least indicate that it is not likely that the findings are valid for this organization only. However, it is hard to determine to what extent the findings are possible to generalize.

### 3.3.8 Factor Importance

The next step is to see how important each factor was in each of the three parts of the study. Here, an attempt has been made to quantify the importance of the factors by ranking (more about ranking in Chapter 2). The importance of the factors in the questionnaire was relatively easy to rank because weights were used in the questionnaire. The factors in the literature survey and the interviews were harder to rank. Therefore, pair-wise comparisons between all factors in each part of the study were performed, which made it easier to compare the factors than through a direct ranking.

When doing the pair-wise comparisons, different aspects were considered as determination of which factor that was most important.
In the interviews, the number of times the factor was mentioned and how thoroughly it was discussed determined the rank. In the literature study, the number of articles found that deal with the factor, how they perceived the factor, and to what extent it was examined were considered. In the questionnaire, the weights and answers provided in the questionnaire were used as a basis for the rank.

The fact that researchers performed the comparisons between the different parts of the study might look subjective at a first glance. However, the quantitative results were rather clear and unambiguous due to that the respondents set the weights themselves. The qualitative results are also seen as rather reliable due to the analysts triangulation (see section 3.1.5) and to the half-year experience at the organization investigated. The factors in the literature survey were the hardest to compare. However, with the study of Steltzer and Mellis as a starting point and with an extensive study according to the aspects mentioned above, the result of this ranking could also be seen as reasonably reliable. Further, both authors have some experience in the area of process management, which made the comparisons of “literature factors” easier.

The result is shown in Table 3.1, where the lowest numbers are considered as most important. The discussion about the implications of this result is presented in Section 3.4.

<table>
<thead>
<tr>
<th>Factor</th>
<th>Qualitative</th>
<th>Quantitative</th>
<th>Literature</th>
<th>Total:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Baselining</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>1. Synchronization</td>
<td>1</td>
<td>1</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>3. User Involvement</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>4. Management Commitment</td>
<td>6</td>
<td>4</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>5. Change Management</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>13</td>
</tr>
<tr>
<td>6. Documentation</td>
<td>3</td>
<td>6</td>
<td>5</td>
<td>14</td>
</tr>
</tbody>
</table>

### 3.4 Discussion

As seen in Table 3.1, baseline and synchronize the software process is considered as the two most important factors when adding the three parts of the study together. To provide a baseline is considered rather important in all three studies and therefore becomes important overall. Synchronization, on the other hand, is ranked as
the most important factor in both organization-specific parts of the study but is ranked lowest in the literature part.

User involvement and management commitment are ranked as third and fourth important respectively. These two were ranked highest in the literature survey but ranked lowest in the interviews. However, user involvement was ranked second in the questionnaire and management commitment fourth in the questionnaire, which must be seen as they still are very important within the organization. Both these might have been higher ranked if they had been addressed more directly in the interviews. This was a negative thing when not “fishing” (see Section 3.1.3) for results. On the other hand, factors like synchronization might not have been mentioned in the interviews if “fishing”.

Regarding change management, the results of each part of the study corresponded very well with each other. All three parts of the study ranked the change management issue as the fourth or fifth highest ranked. Still, 75 percent of the respondents argued that the defined processes were not updated as they should. Documentation was ranked third in the interviews but was ranked as 5 and 6 in the literature survey and the questionnaire respectively. This shows that even if the organization have some problems with documenting the processes, it is not as important as other issues. Still, the majority is not satisfied with the current documentation.

Overall, the result of the three studies aligns rather well with each other. The rankings between literature and this particular organization are different but that is not very surprising. The big difference lies in the issue of synchronization. Synchronization is ranked highest in the organization-specific part while lowest in the literature part. This is, according to us, a very interesting result. The major question is if this is something that is organization specific or this is a common issue in other organizations as well. Organizations with the same properties (e.g. size, organizational structure) might be affected by the same problems while it might not be as evident in smaller organizations.

The difference between the organization and the literature might be an indication that further studies ought to be conducted and that the subject of synchronization is not discussed well enough in industry and the research community. The difference between the literature and the conducted study could originate in that the litera-
The results further indicate that it is very important that the people that perform process improvements are well aware of which problems an organization has. In some organizations, documenting the processes might be the largest problem. However, in this organization it was considered as the least important problem (of the key factors). Remember that management commitment was considered as the most important in the study performed by Steltzer and Mellis [19] but was passed by user involvement in this study. This means that we cannot draw any general conclusions about which areas to improve in all companies. Nevertheless, it is important to be aware of which areas that could be an issue in process improvement. Historic empirical studies might be a very good indicator on which areas to start with when launching an improvement initiative in a company. It should however be noted that the focus of this chapter has been on the key factors identified. Nevertheless, other factors (e.g. communication and training) also influence a company, and should not be neglected.

A concrete example that supports the discussion above is the one about synchronization of processes. In the literature, synchronization was not a commonly discussed issue while the studied organization prioritized this area as the most important. Therefore, in this organization, it seems to be the area where it is most promising to reach the largest benefits.

The studied organization has a history of developing processes within each department without a common instance for coordinating the improvement work, which has resulted in unsynchronized processes. We do not argue about if this is a good or bad strategy for improving processes, it has both strengths and weaknesses. However, as the results indicate, it seems to be important that even if an organization focuses on developing processes within each department, it is important to coordinate the interfaces of the processes between the departments.

The above discussions indicate that it is important to be aware of prioritization techniques as the one presented in this chapter. With the use of such techniques, it is possible to get the users to prioritize which issues are most important in their organization. To prioritize the improvement issues seems to be a good way of finding the most suitable improvements for the specific company. Without the priori-
itization in the questionnaire, it would be hard to know which issues that were the most important.

It should however be noted that the key factors are not separate, without interactions between them. Baselining the current way of working is tightly connected with user participation. If no users are involved in when providing a baseline, the baseline will probably not reflect the current way of working. Further, different processes and different persons need baselines of their own processes. If these are not synchronized, they will not work due to unsynchronized interfaces.

In order to get the users of processes to be involved (to baseline and synchronize), management must commit to the process work and provide time and resources for the users. Management must also provide some kind of instance that is responsible for synchronizing the processes. When the processes are documented, the users must also be a part of the development in order to provide an adequate level of the documentation. When all these parts are in place, a good working change management of defined processes must be in place in order to have process descriptions that are not out of date. In this work, management must provide time and resources so that the users of defined processes could be a part of the updating procedure so that the processes are documented and synchronized in a correct way.

The discussion above could probably go on forever. The important thing to remember when conducting process related work is to identify the factors that are in most need for improvement, focus on these, without neglecting to consider what implications this have on the other factors. The fact that dependencies exist between different improvement issues show the similarities to release planning of software products, where dependencies have to be considered (see Chapter 2). In software process improvements, these dependencies must also be taken into account when planning the “releases”.

3.5 Summary

This chapter presents a study conducted at Ericsson. The study is divided into three different parts: a qualitative part (interviews), a literature part (survey), and a quantitative part (questionnaire). In
this chapter, six different areas have been identified as key areas for successful process management:

1. Baselining the current way of working
2. Synchronization between processes
3. User involvement
4. Management commitment
5. Change management
6. Documentation

In order to determine which factors are most important and find the key success factors, the three parts of the study have been triangulated. Further, a ranking of the different factors has been made from the result of each part of the study.

The result of the ranking shows that baselining the current way of working and synchronization between processes are considered as the most important areas. In the literature study, synchronization of processes was considered as the least important part of process management. This indicates that it is hard to generalize results in this area. Historical studies and results are a very good point to start from but it is important to consider the individual differences in specific organizations.

The results from this study further indicate that synchronization of processes might be an area where further research ought to be conducted. Due to the fact that this was the least discussed factor within the literature studied and that it was the most important factor in the studied organization, it is an area that might have received too little attention in the past. However, no general conclusions could be drawn from this study. The study just considers one specific case and it is important that further research is conducted in order to find out if other organizations encounter the same problems. If some do, it would be interesting to investigate what characteristics organizations with synchronization problems have.

Further, the results show that the different factors are very tightly connected. If a successful process improvement program is to be started, it is not enough to just consider the factors that are most important at the moment, but also other factors that are regarded as important in literature and in other reliable sources.
As mentioned earlier in this chapter, prioritization could be a good way to find out which factors are most important within an organization. When performing prioritizations, it is also possible that different persons have different views of what is important. In the next chapter, the quantitative part of this study is further examined and an analysis of how different roles prioritized the improvement issues is presented.

3.6 References


People are in their nature different. They think differently, behave differently, and require different things. This has been realized for example in the marketing field, where companies and organizations divide customers into market segments, based on their inherent characteristics. Such market segments are often based on geographic (e.g. country), demographic (e.g. age), psychographic (e.g. lifestyle), and/or behavioral (e.g. brand loyalty) data [10]. Psychographic segmentation could include for example which profession a customer has because it is likely that people with similar professions have similar characteristics.

Within software development, people within an organization have different responsibilities and work tasks [6]. This means that they have different professions or roles and it is likely that people with similar roles have the same opinion about things. Further, within an organization, different roles are often divided into different departments, units, divisions, and so forth. These different organizational homes contain the same kind of people and they run into the same kind of troubles in their work. Hence, it is probable that different roles have different priorities when it comes to what should be improved in a company.

Some work around this has been done in the area of software engineering where different perceptions of different roles have been
noticed. For example, Conradi and Dybå [2] showed that managers and developers had different opinions regarding if formal routines were efficient as a communication medium or not. Generally, it showed that managers were more positive than the developers to written routines. However, no distinction of what kind of developer or manager was made. In Svahnberg [9], a study was performed to see how different roles formed groups in an architecture assessment. The conclusion was that people formed groups based on their roles when prioritizing quality attributes and based on their experience when assessing a software architecture. This result shows that different persons with similar characteristics seem to have the same opinion about different issues (as in the idea of market segmentation).

In software process improvement, it is important to know what to improve before improving. People that are going to use an improved process know the current problems very well and should be asked in order to find the right areas to improve [3]. It is obviously convenient to just take an average of responses in order to get the view of what is most important within an organization and then implement the most important areas throughout the whole organization. However, as is noticed in the marketing field and the two papers mentioned above; different roles may have different priorities. The main question is if this is true in software process improvement as well. Do programmers, testers, managers and so on have different issues on their personal agendas?

It is important to be aware of differences in opinion for two reasons. First, it is important to know which roles that are interested in which improvements. Second, it has been reported that people have an inner resistance to change [3]. If knowing the priorities of different roles, it is easier to provide these with the “correct” improvements at the same time as it is easier to sell-in the concept. This is especially true when people refuse changes due to the “not invented here syndrome” [3]. Knowing their personal agendas, it should be easier to convince them about the fitness of the proposed improvement.

Based on the above discussion, the hypothesis of this work is that different roles have different personal agendas, and hence priorities, when it comes to software process improvements. This chapter aims to find out if the opinions about issues in software process improvement differ between roles and if such differences could be used to provide more tailor-made process improvements within an
organization. This is done through further investigating the results that were presented in Chapter 3. This chapter focus on is the quantitative part of the study presented in Chapter 3, performed at Ericsson with several different kinds of roles involved.

In Section 4.1, the performed study is presented. In Section 4.2, the result of the different questions in the study are presented. Further, Section 4.3 presents an analysis of the results. Section 4.4 provides a discussion of how the result affects the process improvement work and Section 4.5 summarizes the chapter.

4.1 Method

The performed study was conducted as a part of the case study presented in Chapter 3 and the result is based on the questionnaire that was a part of the study. In this section, the foundation for the results obtained in the study is described. First, the design of the study is presented (Section 4.1.1). After this, the different roles are identified and presented (Section 4.1.2). Finally, possible threats to this study are discussed (Section 4.1.3).

4.1.1 Design of Study

As can be seen in Chapter 3, this study was done in three steps; a qualitative part, a literature survey, and a quantitative part. The results of this overall study is not presented in this chapter but is discussed in Chapter 3. Instead, this chapter focuses on differences between how different roles perceive the software processes in the organization. To be able to focus on different opinions of different roles, only the quantitative part is included in this chapter. The reason for this is that the roles were not defined until the quantitative study.

As can be seen in Chapter 3, the questionnaire included eighteen multiple-choice questions, five weighted questions, four demographic questions (i.e. age, gender, department and role), and one open-ended question. In the weighted questions, the 100-dollar test (consult Chapter 2 for more information about this technique) was used for prioritizing improvement issues. This means that not only the order of the alternatives were outlined but also to which extent one alternative was more important than another.
In order to get all views from within the organization, all horizontal and vertical levels were represented (i.e. all management levels and functional areas) in the quantitative study. Further, this sample was evenly distributed between the horizontal and vertical levels (i.e. the number of respondents correlated to the number of persons at the functional area/management level).

As can be seen in Chapter 3, the questionnaire sum up to 28 different questions. In order to do a comprehensive analysis and discussion, only the weighted questions are included in this chapter. The main reason for selecting the weighted questions is that it is possible to see not only the ranking, but also the weight the respondents have assigned to the alternatives. Further, the weighted questions were used as validation of the multiple-choice questions, which implies that the multiple-choice questions are implicitly included in the weighted questions.

### 4.1.2 Roles

In this chapter, the focus is on different roles within the studied organization and the study aims to find how they perceive different issues. A number of roles were identified in the demographic questions of the questionnaire. The internal names of the roles have been mapped to common roles within software engineering literature in order to make them more general. These mappings of internal roles to general roles have been validated through a review with people from the organization. However, not all roles could be mapped to general roles and in those cases, no references to literature are provided. Further, not all roles presented in this chapter could be elicited from the demographic questions. Hence, a mapping of roles to functional homes is performed on those (see Others below) that did not have an explicit role. Below follows an explanation of each role:

- **Functional Managers** are responsible for maintaining technical competency, staffing, organizing, and executing project tasks within their functional areas [5].
- **Project Managers** plan, direct, and integrate the work efforts in order to achieve the project's goals [5].
- **Quality Assurance Supervisors** establish and administers inspections and controls the quality related work [5].
Sub-Project Managers inherit the characteristics from the project manager but manage different sub-parts of the main project (e.g. a design project).

Team Leaders manage small teams within the sub-projects.

Others (Staff), this category involves the personnel that have none of the above-mentioned explicit roles and can be seen as the staff of the organization. A division of this role into explicit roles such as programmer and tester is performed below.

This division of roles means that all roles except for the Functional Managers are a part of the development projects. The Functional Managers provide resources to the projects, as explained in Nicholas [5]. This division also means that each project consists of one Project Manager. Each Project Manager has a number of Sub-Project Managers that run the sub-projects within the main project. Further, each Sub-Project Manager has a number of Team Leaders that administer and lead the teams in the sub-project. Staff (or Others) can have several functions and organizational homes, even though they do not have explicit roles. They may, for example, be testers, designers, programmers, requirements specialists, and so on.

When running the study in the industrial environment, the answers of different functional areas were most interesting for the company. However, from a research point of view, it is more interesting to see the views of different roles. Hence, the Staff has been mapped to specific roles, based on their respective functional area.

The functional areas that the Staff are distributed on are the following (the internal names of the functional areas have been mapped to names of functional areas in literature, and have been validated with the organization):

- Design and Programming is responsible for program design and implementation of the system [6].
- Software Engineering Process Group facilitates the definition, maintenance and improvement of the software processes within the organization [11].
- Maintenance and Supply is responsible for maintenance, training, installation/delivery, and support of the system on customer sites [6] [7].
- Strategic Product Management is responsible for planning activities related to the product and the product line [4] and writes a high level requirements specification for each project.
• System Architecture/Requirements Analysts breaks down the high level requirements specification to low level requirements specification and generates a system-level description [6].

• Test verifies that the product works properly and according to its specification [6].

Note that the roles (e.g. Functional and Sub-Project Managers) could also be included in these functional homes. However, the aim of dividing the staff to the functional areas is that it is possible to categorize the staff into roles, based on their organizational homes. This means that we have:

• Designers and Programmers
• Software Engineering Process Group members
• Maintenance and Supply staff
• Strategic Product Managers
• System Architects/Requirements Analysts
• Testers

All identified roles are presented in Table 4.1 together with the abbreviation used and the number of respondents for the role. In this table, it is possible to see that there were 63 respondents to the questionnaire of which 30 had filled in explicit roles and 33 were assigned roles according to their functional homes.

### Table 4.1: Role Distribution.

<table>
<thead>
<tr>
<th>Role</th>
<th>Abbreviation</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functional Managers</td>
<td>FM</td>
<td>6</td>
</tr>
<tr>
<td>Project Managers</td>
<td>PM</td>
<td>7</td>
</tr>
<tr>
<td>Quality Assurance Supervisors</td>
<td>QAS</td>
<td>2</td>
</tr>
<tr>
<td>Sub-Project Managers</td>
<td>S-PM</td>
<td>5</td>
</tr>
<tr>
<td>Team Leaders</td>
<td>TL</td>
<td>10</td>
</tr>
<tr>
<td>Designers and Programmers</td>
<td>D&amp;P</td>
<td>10</td>
</tr>
<tr>
<td>Software Engineering Process Group</td>
<td>SEPG</td>
<td>1</td>
</tr>
<tr>
<td>Maintenance and Supply</td>
<td>M&amp;S</td>
<td>8</td>
</tr>
<tr>
<td>Strategic Product Managers</td>
<td>SPM</td>
<td>2</td>
</tr>
<tr>
<td>System Architects</td>
<td>SA</td>
<td>9</td>
</tr>
<tr>
<td>Testers</td>
<td>T</td>
<td>3</td>
</tr>
<tr>
<td><strong>Total:</strong></td>
<td></td>
<td><strong>63</strong></td>
</tr>
</tbody>
</table>
4.1.3 Threats to Validity

One possible threat to this study is that the roles of D&P, SEPG, M&S, SPM, SA, and T was not elicited from the persons themselves but rather from their functional homes. This could mean that the role descriptions might not fully correspond to their actual work responsibilities. However, during the design of the questionnaire, people from within the organization participated when specifying the explicit roles. Hence, those who have not stated their roles should be considered as staff in their functional home implying that they perform tasks that are typical for their functional home. Hence, they should perform tasks that are in line with the responsibility of the department. A person located in for example the Design and Programming functional area, most probably performs the specific activities of that functional area.

Another threat in relation to the roles is that some roles were only represented by a few respondents (i.e. QAS, SEPG, and SPM). The threat is that their answers could reflect their personal opinion rather than the role's opinion. However, if more people had represented these roles, the sample would not have been evenly distributed (as described in Section 4.1.1). Therefore, in this sample, these small groups must be seen as representative for their role. However, no discussions will be based on these roles. Further, this issue is only considered as a threat in the descriptive part (Section 4.2) since a threshold value is introduced in the analysis section (Section 4.3) in order to reduce the threat.

The normalization made (see Section 4.2) could also be considered as a threat since it might not correspond to what the person's real answers would be. Eight respondents gave answers that did not add up to 100. Three of these had filled in all alternatives of the question, and one had distributed the points equally between the different alternatives (i.e. 33, 33, 33). Out of the remaining four that actually missed/neglected to fill in an alternative, one summed up to more than 100, which probably was just a miscalculation and the ratio should be the same between the alternatives. These five persons are hence not regarded as threats to the validity.

The three remaining persons missed one alternative or more, and had a sum less than 100. One of these missed/neglected one alternative and had a sum of 80. Another missed/neglected two alternatives and had a sum of 90. The third missed/neglected three alternatives and had a sum of 95. The one that had 95 could proba-
bly be classified into miscalculation because it was just 5 points and it was a question with seven alternatives (see Table 4.4). The two remaining are harder to determine, it could have been a miscalculation and it could have been that one alternative that should have been marked was missed. This means that these two persons could be a threat to validity.

A final threat that often is a problem in questionnaires is that it is not evident that the respondents interpreted the questions and answers similarly. However, this should have been prevented since discussions about the language used within the organization were held and the questionnaire was tailored in order to use the terminology that is used within the organization (e.g. internal names of roles instead of general names). Further, a discussion about the interpretation of the questions and answers were held when the pilot study was performed.

When having the method described, the roles outlined and the threats discussed, it is possible to discuss what results that were obtained in the study. This is done in the next section.

4.2 Result

The questionnaire was sent out to 84 persons in the organization. 65 persons answered the questionnaire, which is equivalent to a response rate of 77%. The questionnaire was sent out to 22 percent of the organization (software development part), which resulted in that the respondents represent 17 percent of the organization. This sample was symmetrically distributed both vertically (i.e. management levels) and horizontally (e.g. functional areas). However, as can be seen in Table 4.1, only 63 persons are included in the discussions in this chapter. This is because two persons did not answer the weighted questions, and are hence removed.

In this section, the result of the five weighted questions is presented. The section is divided into five parts, of which each discusses one question. The heading for each section is named according to the original number of the question in the questionnaire.

The number in each cell in the tables represents the mean relative weight of the alternative in comparison with the other alternatives for that role. This means that each row in the tables sums up to 100.
In each table, the answering alternative that has the highest value of each role is marked white while the alternative that has the lowest value is marked black. In some cases, several alternatives have been marked with black or white because some alternatives resulted in equal values. In the tables below, it could also be noted in some cases that values that seem to have equal values are not marked in the same colors. This is because when rounding the numbers, they may get the same value (e.g. 23.5 and 24.3 are both rounded to 24).

The N-value in the upper left corner of each table represents the number of respondents on that particular question. In this study, two persons have been removed from one question each. The reason for this was that they did not answer that particular question. Further, eight persons had answers that did not add up to 100. In order to not lose these persons’ answers, their answers were normalized so that their answers added up to 100 (i.e. \(\frac{\text{Oldvalue}}{\text{Oldsum}} \times 100\)). For example, if a person had a sum of 95 and answered 35 on one of the alternatives, the normalized value became \(\frac{35}{95} \times 100\), which is \(~37\) on that alternative.

In the following sections, each question and its answers are presented and discussed. A more formal analysis is presented in Section 4.3.

### 4.2.1 Question 19

The question and the answering alternatives given in Question 19 were:

When conducting process improvement, how is the relative importance divided between the following 5 factors?

1. Close cooperation of departments/units
2. Shared vision between management and staff
3. Management commitment (providing resources, time, etc.)
4. Affected users of the process involved
5. Setting realistic objectives and goals

In Table 4.2, it is evident that the different roles within the organization have different opinions of what is important and what is not. A brief look in the table reveals that all alternatives but one (alt. 4) are represented as the least important factor. Further, all alterna-
tives have been represented as the most important by at least one role.

4.2.2 Question 20

The question and the answering alternatives posed in Question 20 were:

How is the relative threat against successful process management divided between the following 5 factors?
1. Changes in work procedures are not reflected in process descriptions
2. Changes in process descriptions are not reflected in work procedures
3. Wrong persons develop the descriptions
4. Processes are not synchronized between departments
5. Inconsistent process descriptions (different notations, levels, locations, etc.)

When looking at the result of Table 4.3, a similar result as in the previous question could be identified. In this table, all but two alternatives have been chosen as largest threat to successful process improvement. Further, all but one alternative have been chosen as the smallest threat. If looking at the total, “processes are not synchronized” was regarded as the largest threat. However, the Designers and Programmers regarded this as the least threat of the alternatives. The Designers and Programmers (together with Maintenance and Supply) instead regarded the persons who develop the process descriptions as clearly the largest threat (8 points more that the second largest) while four other roles considered this as the smallest threat to successful process management. This shows that there are rather large differences on how the different roles regard the threats for successful process management.
4.2.3 Question 21

The question and the answering alternatives given in Question 21 were:

How is the relative urgency divided between the following 7 problems that must be resolved?

1. Interfaces between departments/units
2. Synchronization between processes
3. Creating a good change management of processes
4. Make it possible to tailor processes to certain needs
5. Understand/document our current way of working
6. Create a holistic view over the development
7. Enforce tools to support our work

In Table 4.4 it can be seen that the views of what are the most urgent problems to resolve are also rather scattered within the organization. Here five out of the seven alternatives have been chosen as the most urgent while four have been chosen as the least urgent. In this case, it seems like creating a holistic view, enforce tools, and having a good change management process are those areas that are least urgent to resolve. However, when studying the Designers and Programmers, it is possible to see that these thought that enforcing tools were the most urgent. The Designers and Programmers was the only role that thought this was urgent while oth-
ers thought it was not urgent at all (e.g. Testers and Project Managers). It could also be seen that some roles (e.g. Project Managers and System Architects) considered documenting the current way of working as a very urgent problem to resolve (between 6 and 9 points before the second most urgent) while Functional Managers considered this as a not very urgent area (only 7 points). Based on the above discussion, it seems like the roles have rather scattered views of what is urgent and what is not.

4.2.4 Question 22

The question and the answering alternatives posed in Question 22 were:

How is the relative importance divided between the following 6 factors regarding content of a work description/process?

1. Communication scheme between people in the process
2. Workflow (with entry and exit criteria’s for documents, artifacts, etc.)
3. Role definitions
4. Templates
5. Work instructions/guidelines/checklists
6. Training Material

| Table 4.5: Answers to Question 22. |
|-----------------|---|---|---|---|---|---|
| N = 62          | 1 | 2 | 3 | 4 | 5 | 6 |
| FM (6)          | 13| 38| 10| 11| 18| 10|
| PM (7)          | 23| 31| 19|  7| 14|  5|
| QAS (2)         | 15| 25| 15| 20| 23|  3|
| S-PM (5)        |  6| 33| 21| 11| 17| 12|
| TL (10)         | 12| 25| 21| 17| 20|  6|
| D&P (9)         | 15| 21| 15| 16| 26|  8|
| SEPG (1)        |  7| 25| 30| 15| 20|  3|
| M&S(8)          | 13| 27| 24| 10| 23|  4|
| SPM (2)         | 15| 25| 40| 10|  8|  3|
| SA (9)          | 15| 24| 14| 15| 23|  8|
| T (3)           |  2| 50| 28| 10|  7|  3|
| Total:          | 13| 28| 19|13| 20|  7|
In Table 4.5, there are clearly two alternatives that were considered as the most (workflow) and least (training material) important contents of a process. Those three that did not consider workflows as the most important way of documenting the processes, considered this as the second most important. When it comes to the least important, training material was regarded as least important by all but two roles. However, the Testers thought this as the second least important issue while Sub-Project Managers considered it as the third least important issue. Both Testers and Sub-Project Managers considered communication scheme as the least important. The result of this is that even though there are slight differences between the roles, the overall opinion is rather aligned between the roles. This is not at least evident when looking at the totals of the answers (workflow is 8 points more important than the second most important and training material is 6 points less important than the second least important).

4.2.5 Question 23

The question and the answering alternatives given in Question 23 were:

How is the relative effectiveness divided between the following 5 ways to communicate/teach processes?

1. Games
2. Role-plays
3. Workshops
4. Seminars
5. Presentations

Table 4.6 reveals that the views within the organization are rather similar for this question (as in the previous one). Workshops are regarded as the most effective way to communicate processes while games are regarded to be the least effective. All but three roles considered workshops to be most effective, two of these considered workshops as the second most effective while the third role (SEPG) considered it to be the second least effective way. However, SEPG is only represented by one respondent, which might mean that this answer reflects this person’s individual view. When it comes to the least effective, all but two roles considered games as the least effective way to communicate processes. One of the two that did not consider games as least effective was the Testers, which considered
it second least important. The other, SEPG, considered it as the third least important. To conclude, the different roles are aligned in their view of what is most and least important. This is also shown in the large differences of weight in the totals (workshops are 7 points more effective than the second most important, and the games are 7 points less important than the second least important.

### Analysis

The above results indicate that there are both differences and similarities between the roles. A correlation analysis is conducted to investigate this further and to study whether there is an agreement or disagreement between different roles. The correlations are based on the ranks and not on the actual weights even though the benefit with the 100-dollar test is lost by this approach. This was done since it was considered more important to identify agreements in terms of ranking than actual values. Moreover, it would have been hard to interpret the correlation for the actual values. This correlation study was performed by using the Pearson product-moment correlation coefficient since the data involve a number of ties.

To minimize the risk for correlations based on individual viewpoints and randomness, some measures have been taken. First, the roles with less than three respondents are removed from the analysis. It was considered whether a higher threshold should have been set. However, this would have resulted in that the testers would
have been left out from the analysis, which was judged as being unfortunate since tester is one of the most common roles. Thus, the following roles are removed from the analysis: QAS, SEPG and SPM. Secondly, it was decided to be rather conservative when claiming that the different roles were of the same opinion. It was decided that the actual correlations were of less interest than judging whether correlations were positive or negative. In addition, it was decided to look at the number of high correlations as a measure of agreement between different roles.

The results from the correlation analysis are presented in Tables 4.7 to 4.11. The number of observations (O) presented in the tables is equal to the number of answering alternatives on the different questions.

In Table 4.7, it can be seen that the Functional Managers mostly have a different opinion than the other roles. The only really high (> 0.8) positive correlations are between Sub-Project Managers, and Designers and Programmers, as well as between Designers and Programmers, and Team Leaders. However, the table also includes a number of negative correlations, which indicate disagreement between roles. This analysis supports that the different roles have rather different views regarding question 19.

<table>
<thead>
<tr>
<th></th>
<th>FM</th>
<th>PM</th>
<th>S-PM</th>
<th>TL</th>
<th>D&amp;P</th>
<th>M&amp;S</th>
<th>SA</th>
</tr>
</thead>
<tbody>
<tr>
<td>FM</td>
<td>0.12</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S-PM</td>
<td>-0.12</td>
<td>0.2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TL</td>
<td>-0.24</td>
<td>0.8</td>
<td>0.7</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D&amp;P</td>
<td>-0.12</td>
<td>0.6</td>
<td>0.9</td>
<td>0.9</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M&amp;S</td>
<td>-0.24</td>
<td>0.6</td>
<td>-0.1</td>
<td>-0.5</td>
<td>-0.3</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>SA</td>
<td>-0.12</td>
<td>0.6</td>
<td>0.6</td>
<td>0.0</td>
<td>0.2</td>
<td>0.2</td>
<td>1</td>
</tr>
<tr>
<td>T</td>
<td>0.36</td>
<td>-0.6</td>
<td>-0.6</td>
<td>0.0</td>
<td>-0.2</td>
<td>0.8</td>
<td>0.2</td>
</tr>
</tbody>
</table>

In Table 4.8, it can be seen that there is a larger number of high positive correlations. However, it is also notable that there is a number of very high negative correlations. The results for this question show that there is some agreement between certain roles, but on the other hand there are also large disagreements. Overall, the results show that there are agreements and disagreements, as well as neutral relations between different roles.
In Table 4.9, it is visible that the correlations are in general low and there are a number of negative correlations, although mostly rather low. Once again the analysis indicates that there is a disagreement between people having different roles.

Table 4.8: Correlations, Question 20.

<table>
<thead>
<tr>
<th></th>
<th>FM</th>
<th>PM</th>
<th>S-PM</th>
<th>TL</th>
<th>D&amp;P</th>
<th>M&amp;S</th>
<th>SA</th>
</tr>
</thead>
<tbody>
<tr>
<td>FM</td>
<td>0.9</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S-PM</td>
<td>0.97</td>
<td>0.85</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TL</td>
<td>-0.24</td>
<td>0.8</td>
<td>0.7</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D&amp;P</td>
<td>-0.9</td>
<td>0.8</td>
<td>-0.97</td>
<td>-0.3</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M&amp;S</td>
<td>0.29</td>
<td>0.48</td>
<td>0.09</td>
<td>0.87</td>
<td>0.1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>SA</td>
<td>0.97</td>
<td>0.85</td>
<td>1</td>
<td>0.49</td>
<td>-0.97</td>
<td>0.09</td>
<td>1</td>
</tr>
<tr>
<td>T</td>
<td>0.5</td>
<td>0.6</td>
<td>0.49</td>
<td>-0.1</td>
<td>-0.6</td>
<td>-0.1</td>
<td>0.49</td>
</tr>
</tbody>
</table>

Table 4.9: Correlations, Question 21.

<table>
<thead>
<tr>
<th></th>
<th>FM</th>
<th>PM</th>
<th>S-PM</th>
<th>TL</th>
<th>D&amp;P</th>
<th>M&amp;S</th>
<th>SA</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM</td>
<td>-0.04</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S-PM</td>
<td>0.12</td>
<td>0.35</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TL</td>
<td>0.37</td>
<td>0.7</td>
<td>0.65</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D&amp;P</td>
<td>-0.34</td>
<td>-0.63</td>
<td>0.12</td>
<td>-0.2</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M&amp;S</td>
<td>-0.3</td>
<td>0.56</td>
<td>0.24</td>
<td>0.47</td>
<td>-0.04</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>SA</td>
<td>0.06</td>
<td>0.54</td>
<td>0.18</td>
<td>0.76</td>
<td>0.07</td>
<td>0.52</td>
<td>1</td>
</tr>
<tr>
<td>T</td>
<td>0.75</td>
<td>0.42</td>
<td>0.61</td>
<td>0.87</td>
<td>-0.24</td>
<td>0.12</td>
<td>0.53</td>
</tr>
</tbody>
</table>

Table 4.10: Correlations, Question 22.

<table>
<thead>
<tr>
<th></th>
<th>FM</th>
<th>PM</th>
<th>S-PM</th>
<th>TL</th>
<th>D&amp;P</th>
<th>M&amp;S</th>
<th>SA</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM</td>
<td>0.66</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S-PM</td>
<td>0.33</td>
<td>0.37</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TL</td>
<td>0.52</td>
<td>0.66</td>
<td>0.83</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D&amp;P</td>
<td>0.85</td>
<td>0.37</td>
<td>0.26</td>
<td>0.54</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M&amp;S</td>
<td>0.59</td>
<td>0.83</td>
<td>0.77</td>
<td>0.94</td>
<td>0.49</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>SA</td>
<td>0.94</td>
<td>0.63</td>
<td>0.23</td>
<td>0.57</td>
<td>0.92</td>
<td>0.57</td>
<td>1</td>
</tr>
<tr>
<td>T</td>
<td>0.26</td>
<td>0.37</td>
<td>0.83</td>
<td>0.89</td>
<td>0.31</td>
<td>0.71</td>
<td>0.35</td>
</tr>
</tbody>
</table>
In Table 4.10, it is notable that all correlations are positive and there are also more high correlations than for the other questions. The correlation analysis supports the interpretation in the previous section, i.e. the different roles are mostly in agreement with respect to question 22.

Table 4.11: Correlations, Question 23.

<table>
<thead>
<tr>
<th>O = 5</th>
<th>FM</th>
<th>PM</th>
<th>S-PM</th>
<th>TL</th>
<th>D&amp;P</th>
<th>M&amp;S</th>
<th>SA</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM</td>
<td>0.9</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S-PM</td>
<td>1.0</td>
<td>0.9</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TL</td>
<td>0.3</td>
<td>0.4</td>
<td>0.3</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D&amp;P</td>
<td>0.85</td>
<td>0.61</td>
<td>0.85</td>
<td>0.49</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M&amp;S</td>
<td>0.9</td>
<td>0.7</td>
<td>0.9</td>
<td>0.5</td>
<td>0.97</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>SA</td>
<td>0.9</td>
<td>0.7</td>
<td>0.9</td>
<td>0.5</td>
<td>0.97</td>
<td>1.0</td>
<td>1</td>
</tr>
<tr>
<td>T</td>
<td>0.2</td>
<td>0.0</td>
<td>0.2</td>
<td>0.7</td>
<td>0.61</td>
<td>0.6</td>
<td>0.6</td>
</tr>
</tbody>
</table>

The result in Table 4.11 shows an even clearer agreement between the different roles in comparison to Table 4.10. Further, the result of this analysis also supports the findings in Section 4.2.

In summary, there is a disagreement between the roles regarding questions 19, 20 and 21 and an agreement between the different roles for questions 22 and 23. There is a disagreement regarding important issues and threats for process improvement as well as issues that need to be addressed. However, there is an agreement when it comes to process communication issues both with respect to the written material and ways of communicating the process. These disagreements and agreements are important input to any process improvement activity at the company. Knowledge of the disagreements may be used when discussing improvements and the knowledge of the agreements help when it comes to communicating any changes made.

4.4 Discussion

The presentation of the results in Sections 4.2 and 4.3 shows a variety of results. In questions 19, 20 and 21, the different roles have a rather large difference in terms of what is highest prioritized. In questions 22 and 23, the different roles agree rather much on that specific issues are higher prioritized than others. The first question is: Why is there a difference with regard to agreement between the
first three and the last two questions? To this question, there is probably not a straightforward answer. However, when looking at the characteristics of the questions, it is possible to see a difference between the two (questions 19-21 and questions 22-23). The first three questions are concerned with issues about the importance of improvements, the urgency of problems, and threats against successful process management. The last two questions are focused on the content of the process definitions and how to teach the processes.

The difference lies in the nature of the topic. The first three questions are about what to improve and the last two are about how to communicate the processes. As described in Chapter 1, the organization has a SEPG that administrates processes and gives instructions about processes. In these guidelines, there are rules of what to include in a process description and how to document it. The guidelines are based on several years of experience within the company and should provide a good solution. Further, processes have been communicated in the organization for several years and a good practice for communicating processes has been established. This might be one of the reasons to why there are no major differences in how the different roles answered these questions.

As previously stated, it is evident that the different roles have different priorities in the first three questions. This verifies the suspicion that opinions differ between different roles, as was stated in the introduction. However, this discovery also reveals some questions. The main questions in relation to this are:

- Why do different roles have different priorities?
- How could this information be used in a good way?

The first question is relatively easy to answer. As stated in the introduction, but also in Chapter 2, it is a well-known fact that different kinds of persons (or roles) have different priorities, and it would be strange if this would not apply within a software development organization.

The second question is trickier to answer. A first thought would be to personalize every effort in the process improvement work. However, this is not very realistic since it is not possible to solve all kinds of problems with this approach (e.g., if improving the synchronization, several or all roles must be involved). Hence, some kinds of problems must be solved at a general level. These would generally
be the problems that are ranked the highest in total (with all roles involved). However, when it comes to specific issues where one or a few roles think that one issue is important but the others think it is unimportant (e.g. Designers and Programmers want to enforce tools in question 21), it should be possible to personalize these efforts to specific roles. Without an analysis on the role level (as presented here), this would not be possible.

Just as in marketing and product development, it is important to satisfy the different needs of the different users of the processes. Therefore, it seems natural to investigate what requirements the different roles have on the processes. When having both the information of the importance in total and with regards to different roles, it is possible to decide what should be done at a general and a role-specific level. This is similar to what is done when developing products for different markets. When developing products in a product line setting, common functionality (general improvement issues) are put into the core assets while product specific functionality (role-specific improvement issues) are just put into the specific product (or process) [1]. This parallel has also been discussed by Sutton and Osterweil [8].

This means that it seems rational to first address the issues that are common for the whole organization and then trying to solve the issues that are regarded as most important by different roles. The discussion in this chapter has shown that there are differences between roles and that it might be beneficial to try to tailor different improvement issues on a role level. Based on this result, it seems important to investigate and understand the interests of different roles in addition to an understanding about the interest of the organization as a whole.

4.5 Summary

This chapter presents a study that aims at finding out whether there are disagreements between different roles when it comes to process improvement issues in a software environment. The study is based on a quantitative questionnaire and contains five weighted questions. The result of the study is that the respondents disagree in three out of the five questions while they agree in two of the questions. When analyzing the questions with this knowledge, it is possible to see that the three questions where the different roles disagree focus on issues about importance of improvements, urgency of
problems, and threats against successful process management. The two questions where they agree, on the other hand, focus on communication of the processes (documentation and teaching).

The result of this chapter should not be interpreted as that there is always disagreement in some kinds of questions and always agreement in others. Instead, the results indicate that it is important to have different roles in mind when prioritizations of process issues are conducted and not only focusing on the sum of all roles. With an understanding of the different needs of different roles, it is possible to provide improvements tailored for specific roles. This helps satisfying needs of different stakeholders and probably helps to overcome resistance of process improvements. Finally, it seems like it is important to look at other areas that are not directly related to software process improvement. In this chapter, parallels have been drawn to other areas where different types of artifacts/persons get different treatment based on their needs (e.g. marketing, and product line development).

Marketing, product management, and product line development are closely related to requirements engineering (each of these areas is mentioned in relation to prioritization of product requirements in Chapter 2) and the areas overlap to a large extent. When discussing prioritization, the result of this chapter indicates that large similarities exist between priorities of process requirements and priorities of product requirements (as suggested in Chapter 1). This also means that it is possible to use common prioritization techniques for product requirements when prioritizing process improvement issues. In the next chapter, an evaluation of two different prioritization techniques for product requirements is presented.

### 4.6 References


Chapter 5

Requirements Prioritisation: An Experiment on Exhaustive Pair-Wise Comparisons versus Planning Game Partitioning

Empirical Assessment in Software Engineering (EASE 2004), Edinburgh, Scotland

Lena Karlsson, Patrik Berander, Björn Regnell, and Claes Wohlin

Software requirements need to be prioritized when the elicitation process has yielded more requirements than can be implemented at once. In Chapter 2, a number of different prioritization techniques that could be used for this purpose were presented. These different techniques have different level of sophistication, and highly sophisticated techniques may not be practical without tool support. There exist a number of different tools to use for requirements prioritization but not all software organizations have enough resources to buy or develop a tool. This makes it interesting to investigate techniques that do not need computer support.

This chapter describes an experiment aimed at comparing two requirements prioritization techniques. The intention with the experiment is to compare a rudimentary prioritization technique (Planning Game) with a more sophisticated one (Analytical Hierarchy Process). The main variables that are investigated are the difference in time-consumption, accuracy, and ease of use. The experiment was performed during a one-day session with 15 Ph.D. students and one professor as subjects. Instead of real requirements, the subjects prioritized features of mobile phones, which is a well-known product with a range of features to choose from.

In order to investigate the trade-off between different aspects in the prioritization, the prioritization was performed with respect to both

Introduction
Price and Value. The experiment also aimed at investigating if the preferred choice of prioritization technique depended on the number of features involved.

As expected, the results indicate that the more rudimentary technique is less time-consuming and a majority of the subjects found it easier to use. Most subjects also found the results from the rudimentary technique more accurate, which is a bit surprising.

The chapter is structured as follows. Section 5.1 explains and discusses the matter of requirements prioritization in general and the two compared techniques in particular. Section 5.2 describes the design of the experiment and brings up some validity issues. Further, Section 5.3 presents the results discovered in the experiment while Section 5.4 discusses what the results may imply. Finally, the chapter is summarized in Section 5.5.

5.1 Requirements Prioritization

As can be seen in the discussion put forward in Chapter 2, requirements prioritization can be used for several purposes even though the main application is to distinguish the critical few from the trivial many. In order to select the correct set of requirements, the decision makers must understand the relative priorities of the requested requirements [15]. By selecting a subset of the requirements that are valuable for the customers and can be implemented within budget, organizations can become more successful on the market. As shown in Chapter 2, there are several different techniques to choose from when prioritizing requirements. Some techniques are based on more or less structured sorting algorithms, while others use pair-wise comparisons or numerical assignment [5].

The two techniques compared in this chapter are (1) the Analytical Hierarchy Process (AHP) that is based on exhaustive pair-wise comparisons [12], and (2) the Planning Game (PG) [1] that uses a sorting algorithm to partition the requirements. The two techniques are further described below.

5.1.1 Analytical Hierarchy Process (AHP)

As can be seen in the description of AHP in Chapter 2, AHP is a decision-making technique based on pair-wise comparisons that can be been used to prioritize requirements in software development
Karlsson et al. [8] performed an evaluation of six different prioritization techniques based on pair-wise comparisons, including AHP. The authors concluded that AHP was the most promising approach because it is based on a ratio scale, is fault tolerant, and includes a consistency check. AHP was the only technique in the evaluation that satisfied all these criteria. Furthermore, it includes a priority distance, i.e. a ratio scale, while the other approaches only provided the preferred order. However, because of the sophistication of the technique, it was also the most time-consuming technique in the investigation.

5.1.2 Planning Game (PG)

In the last years, there has been an increased use and interest in agile methodologies, such as Extreme Programming (XP). Agile methodologies are based on streamlined processes, attempting to reduce overhead such as unnecessary documentation. The interest and use of agile methodologies have been both from industry and academia. Tom De Marco has aligned to this interest and have expressed that “XP is the most important movement in our field today” [2].

XP is composed of 12 fundamental practices, of which Planning Game (PG) is one. For the purpose of this experiment we have isolated PG despite that the practices likely affect each other [1]. PG is used in planning and deciding what to develop in a XP project. In PG, requirements (written on so called Story Cards) are elicited from the customer. When the requirements have been elicited, they are prioritized by the customer into three different piles: (1) those without which the system will not function, (2) those that are less essential but provide significant business value, and (3) those that would be nice to have [1].

At the same time, the developers estimate the time required to implement each requirement and, furthermore, sort the requirements by risk into three piles: (1) those that they can estimate precisely, (2) those that they can estimate reasonably well, and (3) those that they cannot estimate at all [2]. Based on the time-estimates, or by choosing the cards and then calculating the release date, the customers prioritize the requirements within the piles and then decide which requirements that should be planned for the next release [10].

The result of this easy and straightforward technique is a sorted vector of requirements. The requirements are represented as a rank-
ing on an ordinal scale without the possibility to see how much more important one requirement is than another.

### 5.1.3 Cost-Value Trade-off

As discussed in Chapter 2, it is often not enough to just prioritize how much value the requirement has to the customers. Often other aspects such as risk, time, cost, and requirements dependencies should be considered before deciding if a requirement should be implemented directly, later, or not at all.

Karlsson and Ryan [6] use AHP as an approach for prioritizing both Value and Cost in order to implement those requirements that give most value for the money. The data can be further used to provide graphs to visualize the Value to Cost ratio between the requirements.

In PG, a similar approach is taken when requirements are prioritized based on both customer value and implementation effort. The information that can be extracted from PG should hence be possible to use in the same way as it is used by Karlsson and Ryan [6] with the difference that the result from PG is based on an ordinal scale instead of a ratio scale.

### 5.2 Method

This section describes the experiment approach and execution as well as the analysis performed by the researchers¹. Finally, it is concluded with a number of validity issues.

#### 5.2.1 Experiment Approach

The experiment was carried out with a repeated measures design, using counter-balancing [11][16]. The 16 subjects in the convenient sample included 15 Ph.D. Students in their first or second year, and one professor. The experiment was carried out during a one-day session, which included an introduction to the task, the experiment itself, a post-test, and finally a concluding discussion of the experiment implementation. In addition, before the experiment a pre-test

---

¹ For more information, see http://serg.telecom.lth.se/research/packages/ReqPrio/
was performed, and a few weeks after the experiment a second post-test was conducted.

The two requirements prioritization techniques described above (Section 5.1) were used as input to the experiment, but were modified in order to be further comparable. The system aspect of AHP was not considered, and thus there is only one level of the hierarchy in this investigation [12]. Neither were any of the possible ways of reducing the number of comparisons used, thus the pair-wise comparisons were exhaustive.

PG was modified so that the piles were labeled according to Value and Price: (1) Necessary, (2) Adds to the value, and (3) Unnecessary, and (1) Very high price, (2) Reasonable price, and (3) Low price, respectively. In practice, PG is performed by a customer representative and a developer, but in this experiment each subject had to play both roles.

5.2.1.1 Research Hypotheses
The goal of the experiment is to compare two prioritization techniques and to investigate the following hypotheses:

1. The average time to conclude the prioritizations is larger when using AHP.
2. The ease of use is considered higher for PG.
3. AHP reflects the subjects’ views more accurately.

The objective dependent variable average time to conclude the prioritizations was captured by measuring each subject’s time to conclude the tasks. The subjective dependent variables ease of use and reflecting the subjects’ views were captured by questionnaires after the experiment.

5.2.1.2 Pilot Experiment
A pilot experiment was performed before the main study to evaluate the design. Six colleagues participated and they prioritized 10 features each, with both techniques. After this pilot experiment, it was concluded that the experiment should be extended to 8 and 16 features in order to capture the difference depending on the number of factors to prioritize. Another change was to let the subjects use the techniques and aspects in different order to eliminate order (learning) effects. Further, changes to the AHP sheets included to remove the scale and instead use the “more than” and
“less than” signs so that the participants would not focus on the numbers, and to arrange the pairs randomly on each sheet.

5.2.1.3 Pre-Test
Before the session, the subjects were exposed to a pre-test in order to get a foundation for sampling. A questionnaire was sent out by e-mail in order to capture the knowledge about mobile phones and the subjects’ knowledge and opinion of the two prioritization techniques. The pre-test was used to divide the subjects into groups with as similar characteristics as possible.

Another objective with the pre-test was to investigate how well the subjects could estimate the price of mobile phone features. Nine of the 16 subjects stated that they consider buying a new mobile phone at least every second year, and therefore we believe that their knowledge of mobile phone prices is fairly good.

5.2.1.4 Experiment Execution
The domain in this experiment is mobile phones and according to the pre-test, all subjects were familiar with this context. The factors to prioritize were mobile phone features, for example SMS, Games, WAP, Calendar, etc. In this experiment, the prioritization aspects were “Value for me”, which corresponds to how important and interesting the subject finds the feature, and “Added price on the phone”, which is an estimation of how much the feature might add to the actual mobile phone price. Note that this is not the same as development cost, which would be difficult for laymen to estimate.

The Value aspect has probably been regarded by most of the subjects when buying or considering buying a mobile phone. The Price aspect may also be accounted for since considering buying and comparing mobile phones gives a clue of how much the features add to the price. Thus, there is a trade-off between Value and Price when buying a mobile phone.

One intention of the experiment was to investigate if a different number of requirements would affect the choice of preferred technique. Therefore, half of the subjects were asked to prioritize 8 features, while the other half prioritized 16 features. Another intention was to investigate if the order in which the techniques were used would affect the choice of preferred technique. Therefore, half of the subjects started with AHP and half started with PG. The order of the Value and Price was also distributed within the groups in
order to eliminate order effects. Thus, the experiment was performed using a counter-balancing design, as shown in Table 5.1.

<table>
<thead>
<tr>
<th>Subject</th>
<th># of features</th>
<th>Tech 1</th>
<th>Tech 2</th>
<th>Aspect 1</th>
<th>Aspect 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>8</td>
<td>AHP</td>
<td>PG</td>
<td>Price</td>
<td>Value</td>
</tr>
<tr>
<td>B</td>
<td>8</td>
<td>AHP</td>
<td>PG</td>
<td>Price</td>
<td>Value</td>
</tr>
<tr>
<td>C</td>
<td>16</td>
<td>AHP</td>
<td>PG</td>
<td>Price</td>
<td>Value</td>
</tr>
<tr>
<td>D</td>
<td>16</td>
<td>AHP</td>
<td>PG</td>
<td>Price</td>
<td>Value</td>
</tr>
<tr>
<td>E</td>
<td>8</td>
<td>AHP</td>
<td>PG</td>
<td>Value</td>
<td>Price</td>
</tr>
<tr>
<td>F</td>
<td>8</td>
<td>AHP</td>
<td>PG</td>
<td>Value</td>
<td>Price</td>
</tr>
<tr>
<td>G</td>
<td>16</td>
<td>AHP</td>
<td>PG</td>
<td>Value</td>
<td>Price</td>
</tr>
<tr>
<td>H</td>
<td>16</td>
<td>AHP</td>
<td>PG</td>
<td>Value</td>
<td>Price</td>
</tr>
<tr>
<td>I</td>
<td>8</td>
<td>PG</td>
<td>AHP</td>
<td>Price</td>
<td>Value</td>
</tr>
<tr>
<td>J</td>
<td>8</td>
<td>PG</td>
<td>AHP</td>
<td>Price</td>
<td>Value</td>
</tr>
<tr>
<td>K</td>
<td>16</td>
<td>PG</td>
<td>AHP</td>
<td>Price</td>
<td>Value</td>
</tr>
<tr>
<td>L</td>
<td>16</td>
<td>PG</td>
<td>AHP</td>
<td>Price</td>
<td>Value</td>
</tr>
<tr>
<td>M</td>
<td>8</td>
<td>PG</td>
<td>AHP</td>
<td>Value</td>
<td>Price</td>
</tr>
<tr>
<td>N</td>
<td>8</td>
<td>PG</td>
<td>AHP</td>
<td>Value</td>
<td>Price</td>
</tr>
<tr>
<td>O</td>
<td>16</td>
<td>PG</td>
<td>AHP</td>
<td>Value</td>
<td>Price</td>
</tr>
<tr>
<td>P</td>
<td>16</td>
<td>PG</td>
<td>AHP</td>
<td>Value</td>
<td>Price</td>
</tr>
</tbody>
</table>

The experiment was conducted in a classroom with the subjects distributed in the room. Each subject was given an experiment kit consisting of the AHP sheets and the PG cards.

For AHP, one sheet per aspect and person had been prepared with all possible pair-wise combinations of the features to compare. For the purpose of eliminating order effects, the order of the pairs was randomly distributed so every subject got a different order of the comparisons. With 16 features to compare, there was $16(16-1)/2 = 120$ pair-wise comparisons for Value and Price, respectively. With 8 features, there was $8(8-1)/2 = 28$ pair-wise comparisons for both Value and Price. In between each pair in the sheets there was a scale where the difference of the requirements’ Value or Price was circled, see Figure 5.1.

In order to be able to try different scales, no scale numbers were written on the sheets. Instead, a scale with 9 different “more than”, “equal” and “less than” symbols was used. The further to the left a symbol was circled, the more valuable (or expensive) was the left
feature than the right one. If the features were equally valuable (or expensive) the “equal” symbol was circled.

For PG, the subjects were given two sets of cards (one for Value and one for Price) with one mobile phone feature written on each. The cards were partitioned into three piles, separately for the Value aspect and the Price aspect, see Figure 5.2. The piles represent (1) Necessary, (2) Adds to the value, and (3) Unnecessary, for the Value aspect, and (1) Very high price, (2) Reasonable price, and (3) Low price, for the Price aspect.

Within the piles, the cards were then arranged so that the most valuable (or expensive) one is at the top of the pile and the less valuable (or expensive) are put underneath. Then the three piles were put together and numbered from 1 to 8 and 1 to 16 so that a single list of prioritized features was constructed for each aspect.

The subjects were given approximately 2 hours to conclude the tasks, which was enough time to avoid time-pressure. During the experiment, the subjects were instructed to note the time-consumption for each prioritization. Further, the subjects had the possibility to ask questions for clarification.
5.2.1.5 Post-Test 1

The subjects handed in their experiment kit after finishing the tasks and were then asked to fill out a post-test. This was made in order to capture the subjects’ opinions right after the experiment. The test included the questions below, as well as some optional questions capturing the opinions about the techniques and the experiment as a whole. The questions were answered by circling one of the symbols “more than”, “equal” or “less than”.

1. Which technique did you find easiest to use?
2. Which technique do you think gives the most accurate result?
3. Which technique do you think is most sensitive to judgmental errors?

5.2.1.6 Post-Test 2

After completing the analysis, the subjects were, in a second post-test, asked to state which technique they thought gave the most accurate result. They were sent two sheets (one for Value and one for Price) with two different lists of features, corresponding to the results from the PG and AHP prioritizations. The post-test was designed as a blind-test, thus the subjects did not know which list corresponded to which technique, but were asked to select the list they felt agreed the best with their views. The ratio scale from AHP was not taken into consideration, and neither was the pile distribution from PG. This was necessary in order to get comparable lists.

5.2.2 Analysis

The analysis of the experiment was divided between two independent researchers, in order to save time and to perform spot checks so that the validity could be further improved. The analysis was performed with Microsoft Excel™ and the computing tool MATLAB™.

Two different scales were tried for the AHP analysis: 1~ 5 and 1~ 9. According to Zhang [17] the scale 1~ 5 is better than 1~ 9 at expressing human views and therefore the scale 1~ 5 was used when compiling the prioritization ranking lists. Furthermore, Saaty [12] has calculated random indices (RI) that are used in the calculation of the consistency ratios. Unfortunately, this calculation only includes 15 factors while this experiment included as many as 16 factors. However, the RI scale was extrapolated and the RI for 16 factors was set to 1.61.
5.2.3 Validity

The experimental design involves some threats to validity, which we have tried to prevent. Using the counter-balancing design, the order effects have been balanced out since the subjects were randomly given different orders to perform the techniques and using the aspects. Therefore, we believe that the order of the techniques and aspects does not affect the results.

It is also possible that the subjects could become fatigued during the experiment. Especially the subjects who perform the tasks with 16 features may get tired or bored, which in turn may affect the concentration. This has been tested during the analysis, by calculating the consistency for AHP and the results indicate that there is no significant difference in consistency depending on the number of features (see Table 5.8).

Group pressure and the measure of each subject's time to complete the task might impose time-pressure, which can affect the results. However, it may not be a large problem since there is no major correlation between the time and the consistency in the results (see Table 5.9). Therefore we can argue that time-pressure does not affect the performance of the prioritization.

Another possibility is that the subjects get practice during the experiment and unconsciously get an opinion on the context using the first technique, which will affect the result for the second technique. Especially when using PG first, it may affect the AHP performance. This is not the case. Although the mean values in Table 5.10 indicate a difference in the consistency, the hypothesis tests show that the difference is not significant.

The number of subjects was only 16, which reduces the generalizability, i.e. there is a threat that the findings are specific to this particular group or context. On the other hand, Ph.D. students may have similar views as the requirements engineers and customers who are intended to use the techniques in practice [4]. It is also likely that the subjects are not taking the prioritization as seriously as a requirements engineer or customers would in a real project (see Section 5.3.7 and Chapter 6).

Unfortunately, the scales with “more than” and “less than” in the AHP sheets were accidentally switched so that it could be interpreted in the opposite way than was intended (see Figure 5.1). This
caused some confusion during the experiment. However, the interpretation was explained and clarified and therefore this should not be considered a threat to validity.

It would have been valuable to start the session with an introduction explaining each feature in the prioritization to clarify their meaning. However, the subjects had their own interpretation of the features, which was the same throughout the experiment and therefore this should not affect the result.

5.3 Results

This section presents some of the results found during analysis. First, the three hypotheses are discussed, and then some other interesting findings are described.

5.3.1 Hypothesis 1: The average time to conclude the prioritizations is larger when using AHP

As expected, the time to conclude the prioritization is larger with AHP than with PG, for both aspects. As Table 5.2 shows, the difference in time between the two techniques is 6.1 minutes for 8 features and 14.7 minutes for 16 features. The time increase in percent from 8 to 16 features for AHP is 88 percent, while the same for PG is only 48 percent. Thus, a larger number of objects to prioritize affect the time-consumption for AHP more than for PG, at least when using 8 and 16 features.

<table>
<thead>
<tr>
<th>Number of features</th>
<th>Aspect</th>
<th>AHP</th>
<th>PG</th>
<th>Diff.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Value</td>
<td>7.8 min</td>
<td>3.6 min</td>
<td>4.2 min</td>
</tr>
<tr>
<td></td>
<td>Price</td>
<td>6.4 min</td>
<td>4.5 min</td>
<td>1.9 min</td>
</tr>
<tr>
<td>Total:</td>
<td></td>
<td>14.2 min</td>
<td>8.1 min</td>
<td>6.1 min</td>
</tr>
<tr>
<td>16</td>
<td>Value</td>
<td>12.6 min</td>
<td>6.5 min</td>
<td>6.1 min</td>
</tr>
<tr>
<td></td>
<td>Price</td>
<td>14.1 min</td>
<td>5.5 min</td>
<td>8.6 min</td>
</tr>
<tr>
<td>Total:</td>
<td></td>
<td>26.7 min</td>
<td>12.0 min</td>
<td>14.7 min</td>
</tr>
<tr>
<td>% increase</td>
<td></td>
<td>88 %</td>
<td>48 %</td>
<td></td>
</tr>
</tbody>
</table>

This can also be seen in Figure 5.3 where the median values are higher for AHP than for PG, and the difference between 8 and 16 features is larger for AHP than for PG. Additionally, the box plot
indicates that the subjects’ time to conclude the prioritization with AHP are more dispersed.

In Table 5.3, it is possible to see that the subjects have in average used less time per feature when they had more features to prioritize. It is particularly interesting to see that it takes less time per feature to perform PG partitioning with 16 features than with 8.

One could expect that it should be more complex to perform PG with more features but this result show that it is even faster with more features. However, there might be a breakpoint when the number of features is too great and it becomes hard to obtain an overview.

Four hypothesis tests (see Table 5.4) were performed, for 8 and 16 features respectively, and one for each aspect. The frequency distribution was plotted in histograms to check the distribution. Due to the not normally distributed sample, we chose a non-parametric
test, the Wilcoxon test. The hypothesis tests show that on the 5 percent level there is a significant time difference for three of the four cases. In the fourth case, the Price aspect on 8 features, the test shows that the difference is only significant on a higher level. This is illustrated in Table 5.4, where the p-value is lower than 5 percent in three of the four cases.

### Table 5.4: Wilcoxon Tests for the Time Difference.

<table>
<thead>
<tr>
<th>Number of features</th>
<th>Aspect</th>
<th>Wilcoxon p-values</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>Value</td>
<td>0.0117</td>
</tr>
<tr>
<td></td>
<td>Price</td>
<td>0.0781</td>
</tr>
<tr>
<td>16</td>
<td>Value</td>
<td>0.0098</td>
</tr>
<tr>
<td></td>
<td>Price</td>
<td>0.0039</td>
</tr>
</tbody>
</table>

### 5.3.2 Hypothesis 2: The ease of use is considered higher for PG

Immediately after the experiment the subjects filled out the first post-test that, among other things, captured the opinions of the techniques’ ease of use. Among the 16 subjects, 12 found PG more or much more easy to use than AHP. Only 3 found them equally easy and 1 stated that AHP was more easy to use, see Table 5.5. As can be seen in this table, 12 of the 16 subjects favored PG over AHP, which is 75 percent of the subjects. A qualitative discussion about what the subjects thought about the techniques, and possible reasons for the result, are presented in Section 5.3.9.

### Table 5.5: Results from the First Post-Test: Ease of Use.

<table>
<thead>
<tr>
<th>Number of Features</th>
<th>AHP</th>
<th>Equal</th>
<th>PG</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Much More</td>
<td>More</td>
<td></td>
<td>More</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>16</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Total</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

It seems as if the subjects prioritizing 16 features are a bit more skeptical to PG than those prioritizing 8 features. This could indicate that the more features makes it more difficult to keep them all in mind.
5.3.3 Hypothesis 3: AHP reflects the subjects’ views more accurately

Right after the experiment, the subjects performed the first post-test that captured which technique the subjects expected to be the most accurate. As Table 5.6 illustrates, a majority of the subjects expected PG to be better, while less than a fifth expected AHP to be better. A qualitative discussion about what the subjects thought about the techniques, and possible reasons for the result, are presented in Section 5.3.9.

<table>
<thead>
<tr>
<th>Number of Features</th>
<th>Favour</th>
<th>Equal</th>
<th>Favour PG</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>1</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>16</td>
<td>2</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>Total:</td>
<td>3</td>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td>Total %</td>
<td>19%</td>
<td>25%</td>
<td>56%</td>
</tr>
</tbody>
</table>

For most subjects, the actual ranking that was captured in the analysis differed somewhat between the two prioritization techniques. In order to evaluate which technique that gave the most accurate results, a second post-test was sent out to the subjects. This was done a few weeks after the experiment was performed, when the analysis was finished. The result of this test is presented in Table 5.7.

<table>
<thead>
<tr>
<th>Number of features</th>
<th>Aspect</th>
<th>Favour AHP</th>
<th>Equal</th>
<th>Favour PG</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>Value</td>
<td>0</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>Price</td>
<td>4</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>Value</td>
<td>3</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Price</td>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Total</td>
<td>9</td>
<td>8</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>Total %</td>
<td>28 %</td>
<td>25 %</td>
<td>47 %</td>
<td></td>
</tr>
</tbody>
</table>

As Table 5.7 shows, the most common opinion was that PG reflects the subjects’ views more accurately. Half of the ones that have stated that both techniques are equally accurate actually had the same order in the lists. An interesting observation is that PG
was actually not as good as the subjects expected even if it was clearly better than AHP (compare Tables 5.6 and 5.7).

5.3.4 Judgement Errors

Another question at the first post-test was which technique the subjects expected to be most sensitive to judgmental errors. The objective was to find out the subjects’ views, although it has been shown that AHP is insensitive to judgmental errors due to the redundancy in the pair-wise comparisons [8][12]. However, among the subjects 75 percent expected AHP to be most sensitive. Perhaps this is because the AHP-technique “feels like pouring requirements into a black-box” as one of the subjects stated. It may be difficult to trust something that you are not in control of.

5.3.5 Consistency Ratio

The consistency ratio (CR) describes the amount of judgment errors that is imposed during the pair-wise comparisons. The CR is described with a positive value and the lower CR value, the higher consistency. Saaty [12] has recommended that CR should be lower than 0.10 for the prioritization to be considered trustworthy. However, CR exceeding the limit 0.10 is frequently used in practice [6].

The CR limit above is only valid for the scale 1~9, and in this experiment the scale 1~5 was used instead. Therefore, the limit for acceptable CR will be lower. The average consistency ratios for scale 1~5 are presented in Table 5.8.

<table>
<thead>
<tr>
<th>Aspect</th>
<th>Number of features</th>
<th>Scale 1-5</th>
<th>Wilcoxon p-values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>8</td>
<td>0.11</td>
<td>0.3270</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>0.08</td>
<td></td>
</tr>
<tr>
<td>Price</td>
<td>8</td>
<td>0.10</td>
<td>0.6744</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>0.12</td>
<td></td>
</tr>
</tbody>
</table>

The frequency distribution for the consistency was plotted in histograms to check the distribution. The data was not normally distributed and therefore we chose a non-parametric test. The Wilcoxon test shows that on the 5 percent level, there is no significant difference in consistency depending on the number of features priori-
tized. It was decided not to exclude any of the prioritzations, even though CR was high, in order to keep all available data.

In order to investigate if the time spent on each comparison affects the consistency, the correlation between the parameters was calculated. The Pearson correlation coefficients indicate an insignificant correlation between the time and the consistency, positive for the Value aspect and negative for the Price aspect, see Table 5.9.

<table>
<thead>
<tr>
<th>Pearson Correlation Coefficient</th>
<th>Value</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 features</td>
<td>0.06</td>
<td>-0.25</td>
</tr>
<tr>
<td>16 features</td>
<td>0.26</td>
<td>-0.21</td>
</tr>
</tbody>
</table>

Generally, the absolute value of the correlation coefficient should be greater than 0.5 in order for the values to be considered correlated [13]. Thus, the conclusion is drawn that the consistency is not particularly influenced by the time consumption.

### 5.3.6 Order Effects

There is a chance that the order in which the two techniques are used can influence the result. Table 5.10 shows that the mean consistency ratio is a bit lower for the subjects who used PG before AHP. This may indicate that using PG can provide an image of one’s preferences that is not possible to get from using AHP. Therefore it may be easier to be consistent when PG precedes AHP.

<table>
<thead>
<tr>
<th>Mean Consistency</th>
<th>AHP-PG</th>
<th>PG-AHP</th>
<th>Mann-Whitney p-values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>0.11</td>
<td>0.08</td>
<td>0.6773</td>
</tr>
<tr>
<td>Price</td>
<td>0.12</td>
<td>0.10</td>
<td>0.6773</td>
</tr>
</tbody>
</table>

However, the hypothesis tests show that the difference is not significant on the 5 percent level. Due to the not normally distributed sample, we chose a non-parametric test, the Mann-Whitney test. The p-values are all larger than 5 percent, and we can therefore draw the conclusion that there is no significant difference depending on the order. This finding validates that the experiment analysis
has not suffered from any order effects since there is no significant difference between the two groups.

5.3.7 Distribution in Piles

For PG the subjects were asked to distribute the features in three different piles, dependent on Value and Price. In average, the respondents distributed 41 percent of the features in the middle pile (independent of aspect). This is a result that might not correspond well to how the features would have been distributed in a real case. One could assume that customers would put most of the features in the highest priority pile, which is often the case when customers need to prioritize between their wishes [9][14][15]. Therefore, this result might be somewhat misleading and further studies should clarify this condition. The next chapter presents a study where it is investigated how the choice of subjects affects the distribution of the requirements in different piles.

5.3.8 Prioritizing the Price Aspect

One problem that was identified before the experiment was that the respondents may find it difficult to prioritize the Price aspect, since it is hard to know the price of different features. However, the results show that the mean standard deviation in PG was lower when prioritizing the Price aspect than the Value aspect, see Table 5.11. This result shows that the respondents have been more united when prioritizing Price than Value, which is a rather expected result since the Price is a more objective aspect. Therefore, it is concluded that the Price aspect is not considered a threat to validity.

<table>
<thead>
<tr>
<th>Table 5.11: Mean Standard Deviation.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>8 features</td>
</tr>
<tr>
<td>16 features</td>
</tr>
<tr>
<td>Value</td>
</tr>
<tr>
<td>1.73</td>
</tr>
<tr>
<td>3.02</td>
</tr>
<tr>
<td>Price</td>
</tr>
<tr>
<td>1.25</td>
</tr>
<tr>
<td>2.79</td>
</tr>
</tbody>
</table>

5.3.9 Qualitative Answers

In the post-test performed right after the experiment, the subjects had the opportunity to answer some optional questions about their general opinion. Opinions about AHP include “effort demanding but nice”, “it feels like a black-box wherein you pour requirements”, “good but boring”, “it feels like you loose control over the priorit-
zation process”, and “straightforward”. Opinions about PG are for example “fast and easy”, “lets the respondent be creative”, “intuitive”, “prone to errors”, “good overview”, and “logical and simple”. These opinions correspond well with the results of the captured subjective dependent variables: ease of use and expected accuracy, discussed in prior sections.

5.3.10 Price-Value Graphs

In order to illustrate the possibility of using the Cost-Value approach for requirements selection, two examples of Cost-Value graphs are available in Figures 5.4 and 5.5 (PG and AHP with 8 features). However, in this experiment, we use the term Price instead of Cost. The graphs are made in order to visualize the results from the experiment and to see how much the two techniques differ regarding Price-Value graphs.

The three areas in the graphs represent different grades of contribution [7] and the lines visualize which Value to Price ratio each requirement has, as explained in [6]. The upper line in each graph divides those features that had more than 2 in Value to Price ratio from those that had between 2 and 0.5. The lower line in each graph divides those features that had between 2 and 0.5 from those with a ratio below 0.5 [6]. The Price and Value markings for AHP are based on the mean of the subjects’ relative weight of the features. In PG, the markings are based on the median of the subjects’ ranking number (since it is based on an ordinal scale).

In the case with 8 features, the two methods provide the same result when it comes to which features that are located in which area of the graph. The features Alarm and Vibrating call alert have in average a high Value to Price ratio (above 2) and therefore they would give high contribution to the fictive product. The features Color screen and WAP have a low Value to Price ratio (below 0.5), and would bring low contribution to the product. Finally, Calendar, Games, Notebook and Timer bring medium contribution (between 0.5 and 2 in Value to Price ratio).

The results indicate that it is possible to provide Price-Value (or Cost-Value) graphs with both PG and AHP. However, further studies are needed in order to validate if this result applies to other prioritizations.
Figure 5.4: Price-Value Graph for PG with 8 Features.

Figure 5.5: Price-Value Graph for AHP with 8 Features.
In practice, the Cost-Value diagram would be used to guide the decision-maker in the difficult requirements selection. Other factors such as market segmentation, product focus and time constraints, will also influence the requirements selection.

### 5.4 Discussion

Prioritization is a very important activity in requirements engineering because it lays the foundation for release planning. However, it is also a difficult task since it requires domain knowledge and estimation skills in order to be successful. The inability to estimate implementation effort and predict customer value may be one of the reasons why organizations use ad hoc methods when prioritizing requirements. For a prioritization technique to be used it has to be fast and easy to manage since projects often have strict time and budget pressure. Therefore, a strong argument for PG is that the time consumption is reasonable and the usage easy and intuitive.

In this experiment two groups prioritized 8 and 16 features, respectively, in order to investigate if there is a breakpoint between 8 and 16 where one of the methods is more efficient than the other. It was suspected that a greater number of requirements would eliminate the valuable overview in PG, since it would be difficult to keep all features in mind. However, this experiment only shows a slight tendency of less overview when prioritizing 16 features (see Table 5.5). Therefore, it is suspected that the breakpoint is at an even higher number of features, if a breakpoint exists at all.

Another interesting observation in this experiment was that the time-consumption did not affect the consistency in AHP. One could assume that if someone stresses through the comparisons, the consistency would be worse, but this did not happen. However, this is only initial results and with more difficult features to prioritize, the results might be different.

In practice, it is common that a larger number of requirements need to be prioritized. When the number of requirements grow, it is hard to get an overview. Therefore, visualization is very important in order to share information. This experiment showed that it should be possible to visualize the result of both AHP and PG. However, it should be further evaluated how the ordinal scale in PG affects the visualization.
In a real project, it may also be more valuable to use the ratio scale in order to differentiate requirements from each other in more detail. Thus, it may not be sufficient to determine which requirement that is of higher priority, without knowing to what extent. However, without tool support, AHP will be very time-consuming with a greater number of requirements, both to perform and to analyze.

Due to the small sample and the specific domain it is questionable if the results can be generalized to an industrial situation. Although the subjects may have opinions similar to decision-makers in industry, the context of mobile phone features is a bit too simplistic. The main weakness is that mobile phone features are on a high level and rather independent, while requirements in a real case often have dependencies. It is also possible that industrial experience would affect the results, although we believe that in a relative comparison between these two techniques, it is likely that the rudimentary PG technique would still be preferred.

In the experiment performed by Karlsson et al. [8], AHP was ranked as the superior technique in relation to the others. The main reasons were that AHP had reliable results, was easy to use, was fault tolerant and was based on a ratio scale. This experiment shows that PG is superior to AHP on all of these criteria except for that it is not based on a ratio scale. Therefore, it is interesting to imagine a combination of the two techniques similarly as proposed by Davis [3] (see Chapter 2 for more details). In this approach, only the requirements that need special attention are prioritized with a sophisticated technique (e.g. AHP). The approach assumes that most requirements are not put into the same pile when grouping the requirements, which might be common in an industrial situation. Therefore, some constraints might be needed in order to force the piles to be rather evenly distributed. With three piles, this could for example mean that no pile is allowed to have less than 25 percent of the requirements.

Based on the results from this experiment, it could not be concluded if a combination of the two techniques is efficient or not, or how such a combination should look like. However, we strongly believe that such a combination could be valuable and that it is worth evaluating. Therefore, it is recommended that a combination is tried in a separate experiment or case study, with more data points.
5.5 Summary

This chapter describes an experiment aimed at comparing two requirements prioritization techniques regarding time consumption, ease of use and accuracy in the result. The investigated techniques are the sophisticated Analytical Hierarchy Process (AHP), which is based on pair-wise comparisons and has a ratio scale, and the rudimentary Planning Game (PG), which is based on numerical assignment and ranking, and has an ordinal scale.

The results reveal that the intuitive and quick PG technique is superior with regard to time consumption, ease of use, and accuracy. The mean time consumption was higher when using AHP and the result is statistically significant in three of four cases. PG was considered easier to use by 75 percent of the subjects, although it is more preferred by those who prioritized 8 features. A blind-test performed after the experiment shows that 47 percent found the priority order from PG more accurate, while 28 percent favored the order from AHP and 25 percent found both priority orders equally accurate. However, it is concluded that a combination of the two techniques would further improve prioritization. By first using PG to get an overall picture of the problem and then use AHP for the most difficult decisions, you would, with reasonable effort, get an accurate priority list.

The generalizability of the study is limited due to the small sample and the specific context. A real project has requirements dependencies, and time and budget pressure to consider, which cause the decision-making to be far more difficult. However, we believe that PG is valid as prioritization technique, although it does not have the same elaborate and valuable attributes as AHP.

The main disadvantage of the experiment being the difficulty to generalize to industrial projects, it would be valuable to try the experiment out in a case study. The participating organization would then get knowledge about prioritization and perhaps find a technique that suits their needs.

The presented experiment design could also be used on more subjects to get a larger data set and thereby a stronger basis for conclusions. There are, as discussed, several other prioritization techniques presented in Chapter 2 that would be interesting to look into and compare to the presented techniques as well.
One interesting issue that was noticed in this experiment was that the subjects did not distribute the requirements into the piles of PG as would be expected in an industrial situation. In this experiment, the subjects distributed the requirements rather evenly between the piles while prioritizations in industry commonly results in a significant amount of requirements in the high priority pile. In the next chapter, this finding is further investigated and a number of different studies are observed in order to see what is influencing the suitability of subjects for prioritization studies.

5.6 References


Chapter 6

Using Students as Subjects in Requirements Prioritization

International Symposium of Empirical Software Engineering (ISESE '04)

Patrik Berander

When conducting research in software engineering, the goal is usually to come up with results applicable in industry. In order to obtain such results, industrial professionals are most suitable as subjects in research studies. Nevertheless, students are often used instead of professionals as subjects in empirical research [6]. The reason for this is that professionals are often impossible to use for reasons of cost and access while students are easy to access, cheap to use, and are willing to participate as a part of the courses they attend [6][16].

When conducting empirical research with students as subjects, it is not only the suitability for research that must be accounted for. An ethical dimension must also be taken into account, and that is the study’s suitability in the education. A study that should be performed with students must be in line with their education and preferably give them practical experience, state-of-the-art knowledge, industry relevance, and so forth [4]. This means that learning objectives of courses and research objectives of studies should be combined [6]. In fact, both the researchers and the students should receive value from the study [4].

The approach of using students as subjects could most often be seen as convenience sampling. Robson [13] states that: “Convenience sampling is sometimes used as a cheap and dirty way of doing
a sample survey. You do not know whether or not the findings are representative". This statement implies that the largest problem with convenience sampling is that it is hard to know if the sample is representative. The issue with students is often that it is hard to know if they are representative as professionals in experiments. Nevertheless, studies with students as subjects have made important contributions to empirical software engineering [5].

This chapter presents an experiment on requirements prioritization (i.e. separate the vital few from the trivial many), with students as subjects. The main objective of the experiment was to give the students an exercise in release planning. The prioritization technique used in this experiment was the Planning Game (PG), which is one of the practices in eXtreme Programming. The study presented in Chapter 5 shows that the students in a classroom environment do not prioritize similarly as in industry. Hence, some measures have been taken in this experiment to get the students to prioritize as similarly as possible to prioritizations made in industry.

Beside the objectives from an educational viewpoint, the research objective of the study was to evaluate in which cases students might be used successfully as subjects in research studies. The evaluation aimed at investigating what environmental factors that might affect the students’ capabilities as research subjects in software engineering in general, and in requirements engineering and prioritization in particular. This evaluation was done by studying additional cases (from industry, literature, students in projects, and students in a classroom environment) that have made similar prioritizations.

As can be seen above, the study contains two parts, one experiment performed in a classroom environment with students, and one part where the result from this experiment is compared with results from studies in other environments. In this chapter, the former part is referred to as the experiment while the latter part is referred to as the study. Based on the result of this comparison, it is discussed under which circumstances students are useful as subjects and how different types of education may influence the suitability of students as subjects.

This chapter is structured as follows. In Section 6.1, requirements prioritization is briefly described. In Section 6.2, the method used in the experiment with the classroom students is presented. Section 6.3 presents the results from the experiment while Section 6.4 analyses the result in comparison to the other cases in the same area but
in other environments. Section 6.5 gives a discussion of the implications of the findings in this study and discusses this in relation to related work. Finally, Section 6.6 presents a summary of the chapter.

6.1 Requirements Prioritization

As discussed in Chapter 2, prioritization aims at selecting and implementing a subset of the elicited requirements and still produce a system that meets the most essential needs of the stakeholders, and by this provide quality for the customers [2][7][15][19]. The activity of requirements prioritization could be performed using different techniques. Examples of techniques presented in Chapter 2 are the Analytic Hierarchy Process (AHP) [14], the 100-dollar test [11], and numerical assignment [17][19]. There are also methods that combine different techniques. For example, Planning Game (PG) combines numerical assignment and ranking and is one of the 12 fundamental practices of eXtreme Programming (XP) [1].

In Chapter 5, a more detailed description of PG was given and this is not repeated here. However, as indicated in Chapters 2 and 5, one commonly mentioned problem with numerical assignment is that stakeholders tend to think that everything is critical [10][17][19], which results in that most of the requirements are put into the pile with highest priority. Thus, with this problem, the underlying meaning with prioritization is lost (i.e. if all requirements are put into the high priority pile, they are not really prioritized). However, this problem is reduced in PG since PG also introduces a ranking of the requirements.

In Chapter 5, it was also discussed how to reduce the impact of the above issue by putting restrictions on how many requirements that are allowed in each pile (e.g. not less than 25 percent of the requirements in each pile). With this approach, it is possible to get requirements divided between the piles more evenly. However, one problem with this approach is that the usefulness of the priorities diminishes due to that the stakeholders are forced to divide requirements into certain piles [8]. Another problem could be that requirements are made up just to put them in the low priority pile. This means that the number of requirements increase even though the number of important requirements is not increased. However, no empirical evidence of either good or bad results of setting restrictions has been found.
In the previous chapter, PG was compared to AHP. In that experiment, students were used as subjects, and the phenomenon with most requirements in the high priority pile (as suggested in literature) did not arise. Instead, the requirements were fairly evenly divided between the different piles. The experiment presented in this chapter was launched in order to see if another set of students prioritize the requirements less evenly between the piles. If not, why do they not prioritize similarly as described in the literature?

### 6.2 Method

The experiment presented in this chapter was performed in a classroom environment. The students that were used as subjects were fourth year Software Engineering Master’s students. The experiment was performed within an optional course in Project and Quality Management at Blekinge Institute of Technology in Sweden. The students in the course were both national and international students and all course participants took part in the experiment since it was a mandatory part of the course. The main objective of the experiment was not to use it for research purposes but rather as a planning exercise for the students. The purpose of the exercise was to make the students aware of the difficult trade-offs that must be made when developing software products. As a side effect, it was possible to observe the result and collect empirical research data.

The purpose of the experiment from a research point of view was to observe how the subjects planned releases. In order to really show the students the difficult trade-offs in release planning, and to get research results on how such trade-offs are handled, some attempts were made to get them to prioritize in a similar way as in industry (i.e. put most of the requirements in the high priority pile).

In order to get the students to act as real customers, a couple of pilot studies were conducted with Ph.D. students to see how they could be stimulated to give most of the requirements highest priority. After two pilot studies and a couple of hours of discussion, some conclusions were drawn. First of all, in the study presented in Chapter 5, and in the pilot studies, predetermined mobile phone features were used as requirements. In order to get the students to really being able to act as customers, it was hypothesized that they should come up with their own features (further on called requirements). The intention was that they should care more about the requirements they should prioritize and negotiate when they had
created them themselves. However, the question was if they would care enough to distribute the requirements similar to industry.

Second, in order to really get the students to prioritize as customers, it was seen as important to motivate them to act as customers and convince them that they were allowed to put many of the requirements in the same pile. If the students would not respond to these two stimulations, a second research objective was formulated: What are the reasons for why students do not prioritize as in industry?

6.2.1 Experiment

When performing the experiment, the students were randomly divided into groups of four or five persons and the experiment was performed in a two-hour session. In order to give the participants a view of the task, an introduction to the domain and the exercise was given. The domain chosen in this experiment was the mobile telephone domain since the students ought to have rather equal knowledge in this domain. In the introduction, the students were also given a list of the functionality in the mobile phone/network they should develop for (e.g. camera, MMS, Bluetooth, positioning, etc.). The experiment was conducted as shown in Figure 6.1.

Figure 6.1: Experiment Process
In the first step of the experiment (1), the students got the task to come up with 10-20 requirements, give each requirement a unique number, and write it on post-it notes. An important issue at this stage was that everyone in the group should develop the requirements together and everyone in the group should agree about the meaning of each requirement in order to avoid different interpretations. After the requirements were written down and all group members agreed on the meaning of each requirement, each requirement was duplicated for use in the next step.

In the second step (2a and 2b), each group was divided into two sub-groups by random. Half of the persons in each group were assigned the role of customers and the other half was assigned to be developers (the developing company). At this stage, the different roles were separated and the developers were relocated to another room together with one set of the requirements. The relocation was made in order to not let the different roles know what the task of the other role was.

First (2a), the developers were given the task to put relative cost of implementing each requirement. This was made through giving them 1000 points (representing cost to implement) to distribute between the requirements at hand (consult Chapter 2 for more information about the 100-dollar test). Second (2b), the customers were given instructions to put the requirements in three piles (Less important, Important, Very important). Further, the customers were also given instructions to rank the requirements within the piles in order to get a prioritized list. Before giving the customers these instructions, they were motivated to act as real customers. This was done by repeating that they should act as demanding customers (you of course want everything), and they were instructed that they were allowed to put as much as 85 percent of the requirements in one pile. This number was set in order to get them to understand that they were allowed to distribute the requirements unequally between the piles.

In the next step (3), the groups were reunited by relocating the customers to the developers’ room. The task was to plan three releases of the product together, based on the priorities of the customers and the cost estimates of the developers. The restriction was that 300-350 points should be put in each release. When the negotiation was finished, the groups documented and handed in the results.
The last step (4) in the exercise/experiment was to do another negotiation. This time, the conditions were changed. Instead of having three releases with 300-350 points in each, the students should plan three releases with a different amount in each release. In the first release, they were allowed to put 150-200 points. In the second they were allowed to put 300-350 points, and in the third release, they were allowed to put 450-550 points. The intention with this change was to show that other factors (e.g. time-to-market, learning-effects, work with platforms) could have effect on the release planning.

6.3 Result

When performing the experiment, the 20 students from the course that were present were divided into 5 randomly composed groups. The random composition of groups was made through giving each student a number from one to five and resulted in that students with different backgrounds were mixed. Each of the initial groups consisted of four students but some students arrived late. These late students were put into existing groups, which resulted in that some groups contained five persons. In the section below, the result for each step in the exercise/experiment is presented, and the numbers within parenthesis represent the step illustrated in Figure 6.1.

6.3.1 Elicitation of Requirements (1)

The groups started by eliciting requirements for the mobile phones. Some groups had some problems to come up with requirements in the beginning but after a while, all groups were very eager to find out new requirements. Each group came up with between 10 and 18 requirements. The requirements were very varying and examples of requirements are (shortened): “record videos”, “remote control of other devices”, “find location of friend”, and “digital TV”.

6.3.2 Cost Estimations of Requirements (2a)

When performing the cost estimations, the developers had problems to do the estimations due to their limited domain knowledge. However, after some discussions within the groups, they agreed on estimations. The assigned points on each requirement varied rather much. The span was from 5 to 615 points. However, 615 was an extreme outlier with the second most costly requirement at 250 points.
6.3.3 Prioritization of Requirements (2b)

The students that acted as customers directly started to put the requirements into different piles. As stated earlier, the students were motivated to act as much as customers as possible. The result was that in average, 31 percent of the requirements were put into the Less important pile, 35 percent in the Important pile, while 34 percent were put in the Very important pile. The result of each group's distribution can be viewed in Figure 6.2. The numbers in the figure represent the number of requirements put into each pile.

As can be seen in this figure, Groups 4 and 5 put most of their requirements in the Important pile (I). Group 1 put most of their requirements in the Very important pile (VI), Group 2 most of the requirements in the Less important pile (LI), while Group 3 put equally many in Very important and Less important.

6.3.4 Negotiation 1 (3)

In the first negotiation, the students were allowed to put 300-350 points in each release. The result was that most developers succeeded in giving the customers the Very important requirements in the first release, the Important requirements in the second and the Less important in the third. Only two of the groups needed to put Very important requirements in the last release. One of these was Group 1, with a requirement (which was Very important) estimated
to 615 points. However, this group is not taken into account in the analysis because their dominant requirement could not be included in one single release. Therefore, this group will not be further discussed in relation to the negotiations.

### 6.3.5 Negotiation 2 (4)

In the second negotiation, the conditions were changed and the students were told to put different amounts of points in each release. The result of this change was not as revolutionary as might be expected since none of the four groups needed to put Very important requirements in the last release. Hence, the result was that only minor adjustments were needed in the releases and in most cases, the releases were even more coupled to their priority (i.e. Less important requirements in the third release) than in the first negotiation.

### 6.4 Analysis

In this section, the results presented in Section 6.3 are analyzed in comparison to other studies performed in the same area. This analysis is divided into five different sections. First, a short résumé of the study presented in Chapter 5 is presented (Section 6.4.1). Second, two student projects that have distributed requirements into piles are presented (Section 6.4.2). Third, literature about numerical assignment is presented (Section 6.4.3). Then a presentation of how an industrial company has divided their requirements in a requirements specification is given (Section 6.4.4). Last, a combined analysis of the four preceding sections and the experiment is presented (Section 6.4.5).

#### 6.4.1 Students in Classrooms

In Chapter 5, a similar experiment as in this chapter is presented. In that study, 15 Ph.D. students and one professor were included to compare AHP against a variant of Planning Game. Half of the subjects prioritized 8 predetermined features while the other half prioritized 16. In the experiment, three different piles were used:

- Unnecessary
- Adds to the value
- Necessary
The result when prioritizing 8 features was that (on average) 31 percent of the requirements were put into the Unnecessary pile, 47 percent of the requirements were put into the Adds to the value pile, while 22 percent of the requirements were put into the Necessary pile. When prioritizing 16 features, 23 percent of the requirements were put into the Unnecessary pile, 43 percent of the requirements were put into the Adds to the value pile, while 34 percent of the requirements were put into the Necessary pile. These results show that, on average, the requirements were rather distributed between the three piles, which correspond well to the experiment presented in this chapter.

6.4.2 Students in Projects

At the software engineering programme, at Blekinge Institute of Technology, the students have several project-based courses during their education. At the first year, the students conduct an individual project with real customers. This project is conducted by one student and lasts for approximately 200 hours. At the second year, the students conduct a small team software engineering project. In this project, the students are divided into groups of approximately five persons. Each person should work 280 hours and the customers are most often industrial companies. In the large team software engineering project at the third year, approximately 10-20 software engineering students, one business administration student, and two human-computer interaction students participate and each student work full time for 20 weeks (800 hours) with industrial companies as customers. This approach results in that the projects conducted are very realistic and similar to projects in industry that develops bespoke software.

In order to see how the students and their customers prioritize the requirements in these projects, one small and one large team software engineering project have been studied. Unfortunately, none of these two projects used a numerical assignment prioritization with three piles. Instead, both groups used a numerical assignment prioritization with two piles.

In the small team software engineering project, 5 persons worked with the development of a distributed system to present 3D presentations on several screens and/or projectors. The requirements specification comprised 18 functional and 18 non-functional requirements. 30 (83%) of the requirements were prioritized as Primary, while 6 (17%) were prioritized as Secondary.
In the large team software engineering project, 11 software engineering students developed a system to visualize the benefits of autonomous entities interacting via distributed networks in a marine environment. The requirements specification comprised 69 functional and 49 non-functional requirements. Out of these 107 (91%) were Mandatory and 11 (9%) Optional.

As can be seen in these figures, both projects had a rather large portion of the requirements in the high priority pile. Even though it cannot be directly compared with the result from prioritizations with three piles, it looks like these students put more requirements in the high priority pile and less in the low priority pile than the classroom students.

6.4.3 Reference Literature

In literature it is commonly stated that customers tend to think that everything is critical and that they prefer putting most of the requirements in the pile with highest priority [10][17]. Wiegers [19] has put some numbers on how customers prioritize and states that if they prioritize themselves, customers probably will establish 5 percent of the requirements as May, 10 percent as Should, and 85 percent as Must (if having three piles; May, Should, and Must). It is not clear where these numbers originates from, but it could be suspected that they are estimations based on experiences from industrial companies. However, the numbers do not correspond very well to how students in a classroom environment distributed the requirements, as can be seen in Sections 6.3 and 6.4.1.

6.4.4 Industry

Within the research collaboration between Blekinge Institute of Technology and industry, companies often indicate that customers put most requirements into the high priority pile when using numerical assignment. A common statement by employees in these organizations is that “nearly all requirements are Must requirements”. One of the companies that are using numerical assignment as prioritization technique use the following piles:

- May, an optional requirement of the system
- Should, recommended to include in the system
- Must, an absolute requirement of the system
- Must not, absolute prohibition of the system
In order to get some numbers to verify the suspicions that most requirements are Must requirements, the number of requirements in each pile has been counted. This was done by choosing the main requirements specification for a typical project within the organization. This resulted in that out of 69 requirements, 5 were Should, 1 was Should not, 59 were Must, and 4 were Must not. As can be seen, Should not was not prescribed in the list presented earlier. This means that this requirement does not follow the internal prioritization standard. However, if aligning the piles to the aforementioned piles of three, it is possible to merge the Should not and Must not piles with the Should and Must piles respectively. The reason for this is that it is possible turn around Must not requirements (e.g. ‘performance must not be decreased’ could be ‘performance must be equal or increase’). This means that the Should not requirement that is not following the internal standard do not affect the results of the study. Hence, the following piles are used in this study for comparison:

- May, 0 requirements (0%)
- Should, 6 requirements (8.7%)
- Must, 63 requirements (91.3%)

These numbers correspond well with the numbers presented by Wiegers (Section 6.4.3) while they are very different from the numbers presented in Sections 6.3 and 6.4.1.

### 6.4.5 Comparison

As can be seen in the discussions in the previous sections, there are rather large differences between classroom students, students in projects, reference literature, and the industry case. When dividing requirements into piles, it is possible to use several different names and explanations of the piles. In this chapter, several sources have been studied and the names and explanations of the piles differ. In order to compare the different studies with each other, parts of RFC 2119 [3] are used. RFC 2119 specifies five piles of requirements: May, Should, Should not, Must, and Must not. However, the piles Must not and Should not have been excluded in order to make the different sources comparable according to three piles.

The translation between the individual names and the names prescribed in RFC 2119 means that requirements that were in the low priority pile (e.g. Unnecessary, Less important) have been translated
to May requirements while requirements in the high priority pile (e.g. Necessary, Very important) have been translated to Must requirements. In Figure 6.3 the different cases are illustrated in order to visualize the differences.

In this figure, PG8 and PG16 refer to the experiment presented in Chapter 5 (résumé in Section 6.4.1), and PQM refer to the experiment/exercise presented in Section 6.3. STP and LTP refer to small and large team software engineering projects respectively (Section 6.4.2). Wiegers refers to the numbers from literature presented in Section 6.4.3, while Industry refers to the industrial case, presented in Section 6.4.4.

As can be seen in the figure, there are three groups of similar results, considering distribution. PG8, PG16 and PQM provide rather similar results. Wiegers and the industry case provide similar results, and these have much more requirements in the Must pile than the classroom students had. The two project studies also form a group. This is mostly dependent on that these only used two piles (converted to Must and May in the figure).

As can also be seen in the figure, there is a very large difference between the classroom studies and the studies that are performed in
industry (Wiegers and Industry). The studies based in industry have much more requirements in the Must pile and much less in the Should and May pile (the industrial case did not even have any requirements in the May pile). This shows that students in classroom studies do not prioritize in the same way as it is done in industry. However, if thinking strictly about prioritization, the classrooms students have done better prioritizations (theoretically) because they have distributed the requirements more.

It is harder to draw general conclusions about the results of the prioritizations made in the student projects since they have used fewer piles. However, it is possible to do some reasoning about the result. First of all, it is evident that the project students have put much more requirements in the Must pile than the classroom students. Further, the project students have put much less requirements in the May pile than the classroom students.

If these projects had extended their prioritizations with a Should pile, it would probably mean that there would be less requirements both in the Must and the May pile. This means that, without knowing the exact distribution, the students in projects put much less requirements in the May pile than the classroom students. This result indicates that the students in projects are more similar to industry than the classroom students are.

It could be questioned if the conversion from Optional and Secondary to May is correct in the student projects. The alternative would be to convert them to Should requirements. However, by converting them to Should instead of May, the similarities with literature and industry would have been even larger. In order to not draw too extensive conclusions of this uncertain data, it was decided that they should be converted to May instead of Should.

To conclude this section, it is possible to see that classroom students do not seem to be good subjects when doing research on distribution of requirements with numerical assignment. In order to being able to use classroom students for this purpose, they have to be better aligned to industry. Students participating in projects seem to be more similar to industry cases and hence more suitable for experimental usage in cases of requirements prioritization.
6.5 Discussion

In the past, several studies have been conducted that try to evaluate if students are suitable or not as research subjects within software engineering (e.g. [6][9][12]). The studies have come up with both cases where students are suitable and where they are not. As can be seen in the previous sections of this chapter, students were both seen as suitable and not suitable, depending on the environment in which they were used.

This section discusses the appropriateness of using students as subjects. In Section 6.5.1, literature about students as subjects and the results of this chapter is combined in order to investigate in which areas students could be suitable as subjects. In Section 6.5.2, experience and commitment are discussed as factors that influence the suitability of students as subjects.

6.5.1 Suitability of Students as Subjects

Different studies have been conducted that discuss the appropriateness of students as subjects. Some of these studies have found that there are no significant differences compared to professionals (e.g. [6]) while others have found that there are significant differences (e.g. [12]).

The fact that different studies come up with different results is not very surprising. In some areas it is suitable to use students and in others it is not. However, it is very important to clarify under which circumstances students are useful and not. In the experiment presented in this chapter, for example, it was concluded that classroom students were not suitable to use when evaluating how professionals or customers prioritize their requirements.

On the other hand, in an experiment by Höst et al. [6], it was concluded that students were representative when it comes to problems focusing on project impact assessment. However, when looking in different literature sources that have cited that paper, it is evident that the paper sometimes is used as a motivation for using students, even though the studies have nothing to do with project impact assessment. This combination could be very dangerous since situations in which students are not representative could be considered as representative because of a misinterpretation of the cited reference. For example, if Höst et al. [6] was referred in relation to the
experiment presented in this chapter, it could be concluded that developers and customers in an industrial setting distribute the requirements well between the different piles of importance.

Another article that sometimes is used as a motivation for using students in research is Tichy [18], who gives eight hints for reviewing empirical work. One of these hints is named “Don’t dismiss a paper merely for using students as subjects” where he outlines four different situations where it is acceptable to use students as subjects. The essence of these situations is that students could preferably be used when doing initial research in a subject. This could for example mean that the researchers could study behaviors, trends, and performing pilot studies with students before using empirical studies in industrial situations.

The results of the above studies and discussions lead to some conclusions where students generally could be used as subjects. The different settings that are seen as appropriate are presented and discussed below.

6.5.1.1 Pilot Studies
The most obvious one, and the one most commonly mentioned and accepted in literature, is to use students as subjects in pilot studies (e.g. [4][18]). By doing this, it is possible to try a study, obtain preliminary evidence, control factors, show the relevance, develop experimental kits, and so forth [4]. This means that students could successfully be used to identify trends and to try studies before running them in a more costly and complex environment, i.e. in industry practice.

6.5.1.2 Validating Education
Another area where students could successfully be used is when validating education. As can be seen in Höst et al. [6], students could be used to validate that the education given is appropriate for giving the students a good foundation for working in industry. These kinds of experiments are very valuable for verifying that the appropriate skills are taught at the university. Such an experiment is of course only possible with students. However, if knowing in what areas students at a specific university are comparable with professionals, it is also possible to use the students as replacement for professionals in research, and draw more valid conclusions. For example, the students in the study by Höst et al. [6], could probably be representative as professionals in areas of project impact assessment. However, it is not possible to draw the conclusions that all
students at all universities always are representative in this area. Different universities have different curricula and teaching methods, which of course affects the suitability of the students for use in empirical studies in the area.

6.5.1.3 Worst Case Scenarios
A further area where students could be used as representatives for professionals is when doing studies with “worst case” scenarios, as presented by Kuzniarz et al. [9]. In this paper, the authors argue that if having two methods A and B, and the effectiveness of B is to be studied, the worst case is if the subjects have more knowledge in A than in B. If the subjects find B better under these circumstances, it is most probable that others (including professionals) find B better as well. However, when performing studies under such conditions, it is important to carefully document the knowledge and education of the students used as subjects. In the experiment conducted in [9], the students in the “worst case” group had education and practical experience with the Unified Modeling Language (UML) while less experience with UML extended with graphical stereotypes. In such situations, it is highly probable that the result will be in the same direction when using professionals that have experience with UML but not with UML extended with graphical stereotypes.

6.5.1.4 Trends and Behaviors
When doing studies with two independent methods that the students have no prior knowledge of, it is more uncertain what conclusions that can be drawn. In such a case, it could depend very much on factors that could not be controlled in the experiment. For example, if comparing two programming techniques, A and B, and no prior knowledge could be documented, experience with other things such as design, other programming languages, code inspections and so forth could affect the results. This kind of information could be very hard to document and control but this kind of experiments should not be neglected just because of this.

Even though the above information could be hard to control, students could show a clear advantage for one method over another. With such results, it could be suspected that the professionals will have results in the same direction, although maybe not as much or maybe more than the students [18]. Further, the clearer the advantage is, the more probable it is that the results are in the same direction in other environments as well.
Remus presents a study where students were worse than professionals on making decisions through a decision support system [12]. However, as Remus states, it is possible that even if the students are worse on making decisions, it is possible that they respond similarly to a variety in environment factors and/or experimental manipulations. This means that even if students produce fewer lines of codes per hour, they could increase or decrease their productivity as much as professionals, if using a new development methodology (like pair programming [1]). Hence, it is not possible to just look at the difference and similarities in performance between students and professionals but also on the relative difference in response to treatment (e.g. improvement in percent). Further, the relative order between two methods could be the same but not necessarily the relative difference.

6.5.2 Experience and Commitment

As can be seen in the results (Section 6.3) and analysis (Section 6.4) presented in this chapter, the distributions of requirements in piles differ rather much in the different situations (i.e. classroom, project, industry, literature). If looking at these results, it seems like students in a classroom environment are not representative as customers/professionals when doing prioritizations of requirements. However, students performing projects seem to be more similar to an industrial setting, even though the projects in this study divided the requirements into two piles instead of three. One question that arises is about which factors that influence the suitability of students in different studies.

Prior experience is sometimes mentioned as a factor influencing if students can be used or not in an empirical study [5]. However, in the experiment performed in this chapter, experience does not seem to be an issue (since the classroom students have made “better” prioritizations). This means that there might be other factors that influence the suitability of students as subjects in this kind of experiments.

If excluding the literature case in this chapter, three cases were based on running projects (the industry case and the student projects) while three cases were based on classroom studies. The cases that involved a project had much more Must and much less May requirements than the classroom students. This seems to indicate that the customers in the projects have more at stake and therefore puts more requirements into the Must pile. In the classroom
studies, the customers will get nothing even if they prioritize every requirement as a Must requirement. Hence, it is easier to spread them between the piles.

This leads the discussion into a matter of commitment. In the classroom studies, the students have no commitment to the experiments they are involved in. In a real project, much more is at stake and the students must take responsibility for everything in the project. This means that they have to make the customer happy at the same time as they want to stay within budget and time schedules. The customers put high demands on the project while the students must think about what they could do within their budget.

The argumentation about commitment is further strengthened by the fact that some of the students in the experiment presented in this chapter have been involved in the project courses described in Section 6.4.2. In these projects, they (together with the customer) prioritized more similar to industry practice than in the classroom study. This means that they know the difficulties of prioritization but they are not affected when not having to commit themselves to a real project.

6.6 Summary

This chapter presents an experiment on requirements prioritization with students in a classroom environment as subjects. The result of the experiment is that the prioritizations do not align well with the result of prioritizations in industry practice, even though the students were stimulated to act as in an industrial setting. Further, a comparison is made between the result from the conducted experiment and other cases where similar prioritizations have been made in different environments. This shows that requirements prioritizations made by students and customers in projects seem to be more similar to industry. However, the projects presented in this chapter prioritized on a different scale than the other cases. Hence, no definitive conclusions can be drawn about the suitability of students in projects. However, the results indicate that students in projects are at least more suitable than students in a classroom environment for studies when commitment is an issue. In studies with other objectives, classroom students might be more suitable than students in projects.
In literature, a commonly mentioned factor of the suitability of students is the experience of the students. This means that fourth year students should be more suitable than second year students (because of their increased knowledge in developing software). However, the students that were represented in the classroom studies in this study were fourth year students and Ph.D. students. The students in the projects, on the other hand, were second and third year students. This indicates that the most experienced students (based on their level of education) were the least suitable subjects. Hence, it seems that commitment is a more important factor than experience in this study. This is of course not the factor that is most important in all studies, but knowledge of which factors that are important in what kind of studies is very important to have when judging if students are suitable or not as subjects.

This means that students should be used in empirical studies, as long as they are suitable and the study fits into the students’ education. However, some general contexts where students most often seem to be suitable are:

- Students could be used as subjects in pilot studies, before running the studies in industrial environments.
- Students are the only ones that can be used when evaluating if the education is giving them the right skills.
- Students could be used when evaluating if a new technique is better than a known technique.
- Students could be used for identifying trends and behaviors.

By using students when evaluating different prioritization techniques, it is possible to perform more studies to a lower cost. For example, in Chapter 5, students were successfully used to evaluate two different prioritization techniques in comparison to each other. In such studies, it might be possible to evaluate prioritization techniques for prioritization in both product and process development. As can be seen in Chapter 1, prioritization of needs, and other related areas needs to be further researched. By using students as subjects, this can be achieved to retrieve results in a faster pace. However, more research about the suitability of students as subjects must be performed in order to identify under which circumstances students could be used as subjects. Until such results are present: use students as subjects with care!
6.7 References


