HELSINKI UNIVERSITY OF TECHNOLOGY
Faculty of Electronics, Communications and Automation

Lauri Halkola

R&D Process Optimization for a Customer and Order Management System

Master's thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Technology

Espoo, March 26, 2008

Supervisor Professor Raimo Kantola
Instructor MSc Miko-Janne Uskali

Author:	Lauri Halkola
Name of thesis:	R&D Process Optimization for a Customer and Order Management System
Date:	March 26, 2008 **Number of pages:** 10+66
Faculty:	Faculty of Electronics, Communications and Automation
Professorship:	S-38 Networking Technology
Supervisor:	Professor Raimo Kantola
Instructor:	MSc Miko-Janne Uskali

Due to the challenges posed by rapidly changing ICT markets, the evolution of agile software process models has been fast during the recent years. Experiences of them have been very positive. Adapting a new way of developing software is, however, often challenging and not all models are suitable for all types of projects.

This master's thesis aims to find a solution for shortening the time-to-market of company A's A1 system development process from 12 months to 4 months. The process encompasses all the activities from an initial business idea to the actual deployment on the market. The study focuses especially on improving the quality of requirements engineering, which is seen as the biggest bottleneck.

The study was carried out in three phases. In the first phase, new ideas were gathered from companies A, B, and C using different methods. The literature review was conducted during the second phase. In the third phase of the work, a new process model was developed based on the theory and findings in the first phase.

The recommendation for the new A1 process model follows an agile methodology called Scrum. The process consists of pre-game, development, and post-game phases. Requirements engineering activities take place during the continuous pre-game phase. The requirements are maintained in a product backlog list. The development phase consists of several 30-day development sprints. In the suggested model, new critical and well-prepared business requests can be processed further very quickly but poorly defined requests must be improved before getting further. Tight control throughout the process ensures that the quality of the requirements will improve from their current state. In the optimal case, the time-to-market can be only two months.

Keywords: software process, Scrum, requirements engineering, agility, time-to-market, quality

TEKNILLINEN KORKEAKOULU **DIPLOMITYÖN TIIVISTELMÄ**

Tekijä:	Lauri Halkola
Työn nimi:	Asiakas- ja tilaushallintajärjestelmän tuotekehitysprosessin optimointi
Päivämäärä:	26.3.2008 **Sivumäärä:** 10+66
Osasto:	Elektroniikan, tietoliikenteen ja automaation tiedekunta
Professuuri:	S-38 Tietoverkkotekniikka
Työn valvoja:	Prof. Raimo Kantola
Työn ohjaaja:	DI Miko-Janne Uskali

Nopeasti kehittyvien ICT-markkinoiden aiheuttamista haasteista johtuen ketterien ohjelmistokehitysmallien evoluutio on ollut nopeaa viime vuosien aikana. Myös niiden käyttökokemukset ovat olleet erittäin lupaavia. Organisaatioilla on kuitenkin usein vaikeuksia sopeutua työskentelemään uuden ohjelmistokehitysmenetelmän mukaisesti, eivätkä kaikki mallit sovellu kaikentyyppisille projekteille.

Tämä diplomityö pyrkii löytämään ratkaisun lyhentämään yritys A:n A1-järjestelmäkehitysprosessia 12 kuukaudesta 4 kuukauteen. Prosessi käsittää kaikki aktiviteetit alustavasta liiketoimintaideasta varsinaiseen tuotantoon vientiin. Tutkimuksessa keskitytään erityisesti kehittämään vaatimusmäärittelyn laatua, jota pidetään suurimpana pullonkaulatekijänä nykyisessä prosessissa.

Tutkimus suoritettiin kolmessa vaiheessa. Ensimmäisessä vaiheessa uusia ideoita kerättiin yrityksistä A, B ja C eri metodeita käyttäen. Aiheeseen liittyvään kirjallisuuteen paneuduttiin vaiheessa kaksi. Työn kolmannessa vaiheessa kehitettiin uusi A1 prosessimalli perustuen teoriaan ja ensimmäisen vaiheen havaintoihin.

Suositus uudeksi A1-prosessimalliksi noudattelee ketterää ohjelmistokehitys-menetelmää nimeltään Scrum (~ aloitusryhmitys rugbyssa). Prosessi koostuu lämmittely-, kehitys- ja jälkipelivaiheista. Vaatimusmäärittelyaktiviteetit suoritetaan jatkuvassa lämmittelyvaiheessa. Vaatimuksia ylläpidetään tuotelistassa. Kehitysvaihe koostuu useista 30-päiväisistä kehityspyrähdyksistä. Ehdotetussa mallissa kriittiset ja hyvin valmistellut uudet liiketoimintavaatimukset voidaan prosessoida eteenpäin hyvinkin nopeasti, mutta huonosti määriteltyjä pyyntöjä täytyy parantaa ennen kuin ne pääsevät eteenpäin. Tiukka kontrolli läpi koko prosessin varmistaa, että vaatimusten laatu paranee nykytilanteeseen verrattuna. Optimaalisessa tapauksessa koko tuotekehitysprosessi voi kestää ainoastaan kaksi kuukautta.

Avainsanat: ohjelmistoprosessi, Scrum, vaatimusten määrittely, ketteryys, nopeus, laatu

Preface

This master's thesis was carried out for company A. The work was a part of the company's internal IT process development activities that were aiming to improve the company's competitiveness in the long run.

I would like to thank my supervisor, Professor Raimo Kantola, for his valuable guidance and ideas. I would also like to thank my instructor Miko-Janne Uskali for offering me an interesting topic and guiding me throughout the process. I would also thank all my Finnish, Swedish, Norwegian, and Indian colleagues who contributed to this research in one way or another. It was great to notice that so many of you were truly enthusiastic about the topic and you really wanted to give your opinion on it.

Finally, I would like to express my gratitude to my family for all the support I have got during my studies.

Helsinki, March 26, 2008

Lauri Halkola

Table of contents

Abbreviations

AM	Agile Modeling
ASD	Adaptive Software Development
BIT	Business IT Management
BR	Business Request
B2B	Business-to-Business
CEO	Chief Executive Officer
CMG	Change Management Group
CR	Change Request
CRM	Customer Relationship Management
DP	Decision Point
DSDM	Dynamic Systems Development Method
EP	Evaluation Point
ERP	Enterprise Resource Planning
FAA	Functional and Architecture Analysis
FDD	Feature-Driven Development
FST	Feasibility Study Template
ICT	Information and Communications Technology
ITCM	IT Continuity Management
IEEE	Institute of Electrical and Electronics Engineers
IT	Information Technology
MD	Man Day
MDE	Master Development and Test Environment
OOPSLA	Object Oriented Programming Systems Languages and Applications
PROMO	Process Model
QURE	Quality through Requirements
RFP	Request for Proposal
RAD	Rapid Application Development
RUP	Rational Unified Process
SD	Small Development
SRS	Software Requirements Specification
R&D	Research and Development
XP	eXtreme Programming

Index of Figures

Index of Tables

1 Introduction

Note: Because of some delicate information in this thesis, code names are used instead of the real names of the examined companies, projects, and systems. Organization ABC refers to the whole name of the organization. Companies A, B, and C refer to the country organizations of organization ABC. Projects A1, B1, and C1 are responsible for developing the corresponding systems A1, B1, and C1 for companies A, B, and C. For the same reason, the names of the interviewees are not mentioned in the list of references.

Organization ABC is one of the leading companies within the telecommunications market in the Nordic and Baltic region. However, the boundaries between different branches of the ICT industry are becoming fuzzier, which makes it possible for a larger number of companies to compete in the same market. Also the emerging new technologies and services can be seen as a major threat to traditional companies. To maintain or even strengthen its position in the very competitive market, organization ABC needs to continuously improve its internal processes.

Nowadays, there is a huge variety of different software process models. Despite of the many options, only a few models are and have been popular on the larger scale. Most of the companies used to adapt the waterfall model or one of its modifications for decades. The waterfall model is a very straightforward method, which progresses step by step from requirements engineering to maintenance. It is extremely suitable for large and predictable software projects.

As the ICT market evolves with an accelerating pace, the projects have become less predictable. The business requirements may change from day to day and the product lifecycles are shorter than ever. In order to meet these challenges, more agile software process models have been developed during the recent years. The results have been very promising and, today, some of these models are used worldwide. However, adapting a totally new way of developing software is always difficult and not all models are suitable for every project.

The purpose of this master's thesis is to find a solution for shortening the time-to-market of company A's A1 system development project from 12 months to 4 months. The process encompasses all the activities from an initial business idea to the actual deployment on the market. A second goal is to improve the quality of requirements engineering to ensure that, in the beginning of the project execution phase, all the business requirements are at the required level, so that the IT project team will be able to complete the project without unexpected surprises in the later phases of the project.

The ultimate focus will be on optimizing and finding new ways for the project planning phase, which is seen as the biggest bottleneck, but also the entire process will be considered. The study is limited to cover the requirements engineering process for consumer products within company A.

The A1 system is considered to be the most important business system within company A. Customer service representatives use the system whenever they are interacting with the customers to handle transactions such as creating orders and changing customer information. By optimizing the A1 R&D process, company A will be able to respond faster to competitors' actions; achieve competitive advantage by getting new innovations and services faster to the market; improve the quality of the requirements engineering process so that the final products will better meet the customer needs; and reduce costs by avoiding unnecessary work during the R&D process.

The research process was divided into three phases. In the first phase, the aim was to get familiar with the current A1 process, identify the possible problems, and find suggestions for improving the process. The phase also included a field study at companies B and C trying to find new practices for developing software. The literature review was conducted during the second phase. It involved searching for information regarding new theories and solutions to meet the set goals. In the third phase of the work, the theory and the findings from the first phase were adapted to the current practices of the A1 project. Please note that the structure of this thesis does not follow the phases of the study.

The software process models are introduced in Chapter 2. The chapter aims to give an overview of the conventional waterfall and incremental process models that are the basis for many newer models. After a comparison of the ten most interesting agile and conventional process models, the winner of the comparison is presented at length. Chapter 3 has been dedicated to requirements engineering. The chapter describes the importance of requirements engineering, the characteristics of a good requirement, traditional requirements engineering process, and some useful requirements engineering practices.

Chapter 4 describes and analyzes the research methods used. The current A1 system development process is described in Chapter 5. The description gives an overview of the process from both an IT and a business perspective. Identified problems and bottlenecks are introduced at the end of the chapter. Chapter 6 discusses the findings relating to companies B and C. The chapter mostly concentrates on describing the interesting aspects that are conducted differently to the A1 project. Because company C uses a totally different, more agile approach in their software projects than companies A and B, that process is described in detail.

The new A1 software process model is presented in Chapter 7. Although the main emphasis is on describing the enhanced requirements engineering process, also the entire A1 process is discussed. The chapter also introduces some of the useful practices that should be used in the development of A1. Detailed step-by-step guidelines for implementing the recommendation in practice are presented at the end of the chapter. The thesis is concluded in Chapter 8.

2 Software process models

Nowadays, there is a huge variety of different software process models. The models have evolved over time and new ones are constantly being introduced. Despite of many alternatives, only a few of the models have established a firm foot fold in the global software industry. This chapter shortly introduces two of the major prescriptive process models and gives a quick overview of agile process models. The winner of a comparison with ten of the most interesting process models is introduced in more detail.

2.1 Definitions

It is obvious that due to the large range of publications concerning software engineering, there are numerous definitions for software terminology. However, the basic content is usually the same. A software process is "a set of activities whose goal is the development or evolution of software". A software process model can be described as "a simplified representation of a software process, presented from a specific perspective". [Som06]

Conventional (or sometimes called "prescriptive") process models [Pre05] "define a distinct set of activities, actions, tasks, milestones, and work products that are required to engineer high-quality software." In other words, conventional models bring order to the chaos of R&D by providing a stepwise roadmap for a project.

Agile process models [Pre05] combine a philosophy and a set of development guidelines that were stated in the "Manifesto for Agile Software Development" in 2001. "The philosophy encourages customer satisfaction and early incremental delivery of software; small, highly motivated project teams; informal methods; minimal software engineering work products; and overall development simplicity." This way, the roadmap is usually much more flexible than in projects following some of the conventional process models.

2.2 Comparison of software process models

Practically all software process models are based on two of the most traditional conventional models: the waterfall model and the incremental model. Despite of this, significant differences can be found between the various models.

2.2.1 Waterfall model

The waterfall model is the oldest framework for software engineering [Pre05] [Sch02] [Som06] [Vli04]. It was originally introduced by W. W. Royce in 1970 although a similar approach can already be found in publications from the early 1960's [Vli04]. The waterfall model suggests a systematic, sequential approach to developing software and it consists of five phases [Vli04] [Som06]:

1. Requirements engineering. Goals and constraints for the project are set and they are documented in the requirements specification.
2. Design. System requirements are derived from the user requirements. The purpose is to provide details about software architecture, interfaces, and components that are required in implementing the changes.

3. Implementation. The changes are implemented according to the design documentation. Unit testing is performed separately for each unit.
4. Testing. The individual units are put together and tested. Integration testing is a systematic technique for testing interfaces while constructing the software architecture [Pre05]. The goal of regression testing is to re-execute a subset of tests that have already been conducted to ensure that the new functionality does not break the existing system. Finally, the system is validated by a customer through appropriate acceptance testing.
5. Maintenance. The changes are deployed and taken into use. Maintenance team is responsible for correcting errors.

The model is based on careful planning, tight control, and extensive documentation. Before moving to the next phase, all the essential deliverables must at least be verified or tested to make sure that the system has been built correctly and meets the set user requirements. This is due to the fact that if the project team needs to return to one of the previous phases during the R&D process, this might cause a significant increase in budget and delivery time. Figure 1 illustrates the waterfall model as it is described in [Vli04].

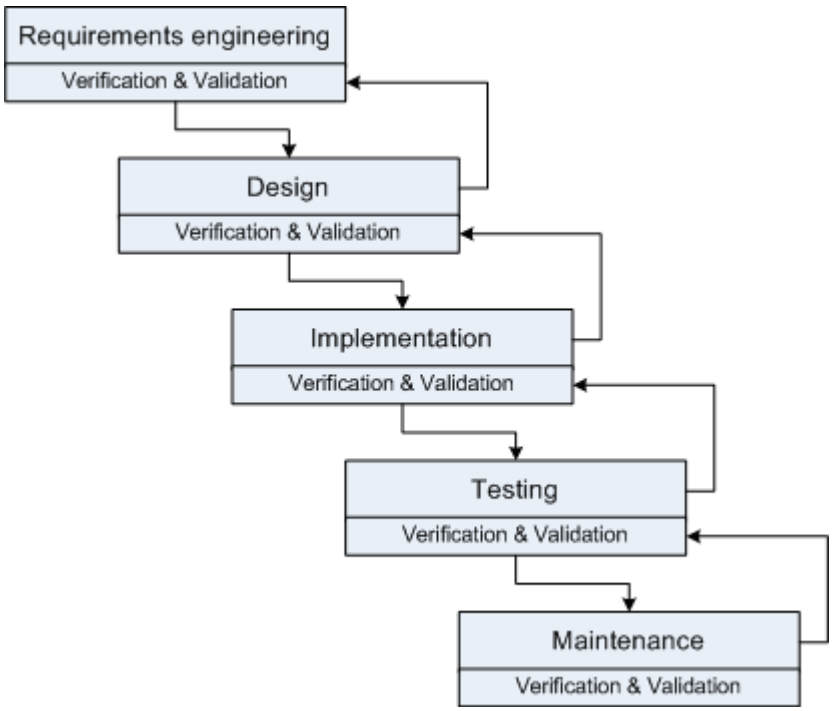

Figure 1. Waterfall model.

2.2.2 Incremental model

The incremental model utilizes the same practices as the waterfall model. However, the software is deployed in much shorter increments. Each linear sequence produces new features and creates value for the company. Normally, the first increment focuses on the essential key features and the later increments provide additional functionality. The main advantage of the incremental model is its flexibility.

Figure 2 shows how a large-scale project can be conducted in an iterative manner using the incremental model [Vli04]:

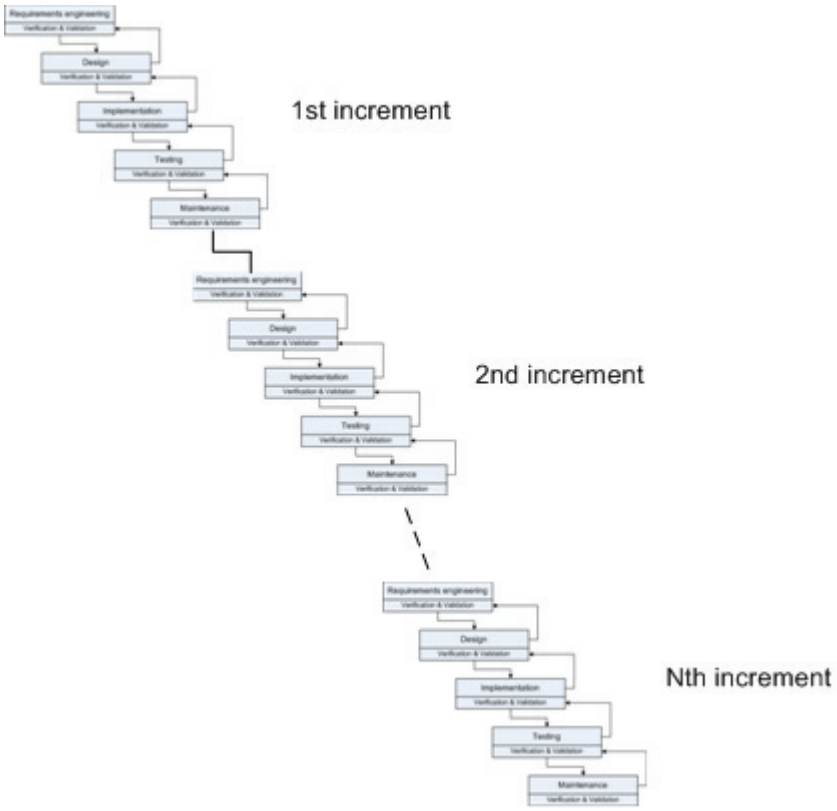

Figure 2. Incremental model.

2.2.3 Agile process models

Because of the challenges posed by the rapidly changing ICT markets, the evolution of the agile software process models has been fast during the recent years. Experiences of them have been very positive. Some of the well-known agile process models include eXtreme Programming (XP), Scrum, Crystal, Feature-Driven Development (FDD), Dynamic Systems Development Method (DSDM), Adaptive Software Development (ASD), Rapid Application Development (RAD), Agile Modeling (AM), and the Rational Unified Process (RUP). [Abr02] [Agi08] [Pre05]

Practically all of agile process models share a few common characteristics that are stated in the "Manifesto for Agile Software Development" (see "Appendix B: Agile Manifesto"). The main advantages of agile process models compared to conventional models are introduced in Figure 3 [Ver06]. The progress of the work is always visible through frequent deliveries. For the same reason, agile process models create value for the business from the very beginning of the process. When using agile process methodology, IT is more adaptable to changing business needs, because plans may be modified more often. As a result of these three benefits, agile software development involves less risk than conventional process models.

However, agile process models do not suit all kinds of projects and adapting a new way of developing software is often challenging. A potential disadvantage of agile software development is the fact that the development costs of smaller deliveries may be higher than in conventional projects, which have larger economies of scale. But quite often, agile process models become less expensive since they are more efficient, focus on the right things, and fulfill the business needs better.

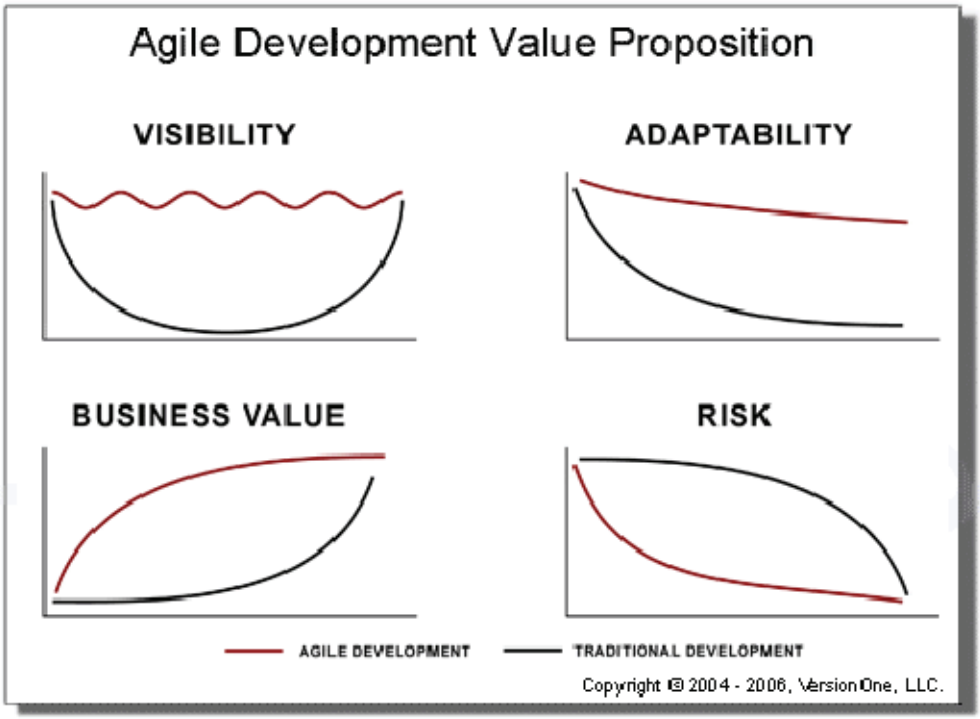

Figure 3. Agile development value proposition.

2.2.4 Comparison

Due to the fast and continuous evolution of the new and existing software process models, there are practically no comprehensive comparisons made of them. In order to evaluate the suitability of different software process models for project A1's purposes, I conducted a study regarding ten of the most interesting models. These models include Waterfall, Incremental, Spiral, Crystal, Scrum, Feature-Driven Development (FDD), Dynamic Systems Development Method (DSDM), Adaptive Software Development (ASD), Rapid Application Development (RAD), and eXtreme Programming (XP). The models were chosen after a preliminary evaluation during a literary review.

The actual study is based on twelve variables that have been weighted according to their importance to the A1 project. Each variable is actually a statement, which is evaluated on a four-step scale from 1 (false) to 4 (true). The statements are allocated under five categories: agility (weight of 35%); general limitations (weight of 28%); low risk and fast adaptation (weight of 24%); quality (weight of 10%); and costs (weight of 3%).

Agility is measured with three variables: requirements can be changed at a late stage of the project; short time-to-market; and suitability for continuous incremental development. Also general limitations are covered by using three statements: suitability for developing packet software; scalability for large projects (more than 40 persons involved); and suitability for distributed development and outsourcing. Risk and adaptation time are evaluated using four variables: organization ABC has experience of the model; the model has proved to be successful (positive experiences from other projects); lots of literature available; the model adopts suitable methods. Quality simply tells us how good software can usually be produced using a specific model and costs reflect the expected development expenditure.

The results of the comparison are presented in "Appendix A: Comparison of software process models". The winner of the comparison, Scrum, is introduced in the next section.

2.3 Scrum

The roots of Scrum date back to the 80's when Takeuchi and Nonaka [Tak86] introduced a collection of thoughts regarding an adaptive, quick, and self-organizing product development process, which originated from Japan [Abr02]. However, the first formal description of Scrum was presented at OOPSLA'96 by Ken Schwaber and Jeff Sutherland [Sch02b]. The name, Scrum, is derived from an activity that occurs during a rugby match [Pre05]. The Scrum methodology has been developed for managing the agile systems development process [Sch02b]. It is based on empiricism, self-organization, and action. It does not tell precisely how the software should be developed, but Scrum rather explains the roles of the team members and how they should interact. Scrum also introduces certain practices that should be followed throughout the process in order to succeed in producing a high-quality system in a constantly changing environment. Efficient management activities help in improving the existing engineering practices, such as testing, in an organization [Abr02]. Scrum can be used in both creating new and maintaining old software [Sch02b]. All Scrum principles are consistent with the Agile Manifesto, which was created and signed in 2001 by 17 of the world's leading software engineers who had been developing new agile software development methods including eXtreme Programming, Scrum, Dynamic Systems Development, Adaptive Software Development, Crystal, Feature Driven Development, and Pragmatic Programming [Agi08] [Bec01]. The full Agile Manifesto can be found in "Appendix B: Agile Manifesto". Scrum has been used successfully in thousands of companies since its introduction.

2.3.1 Process

Quite often, the Scrum process is described as consisting of three phases: pre-game, development, and post-game phase [Abr02] [Sch96]. The process is described in Figure 4.

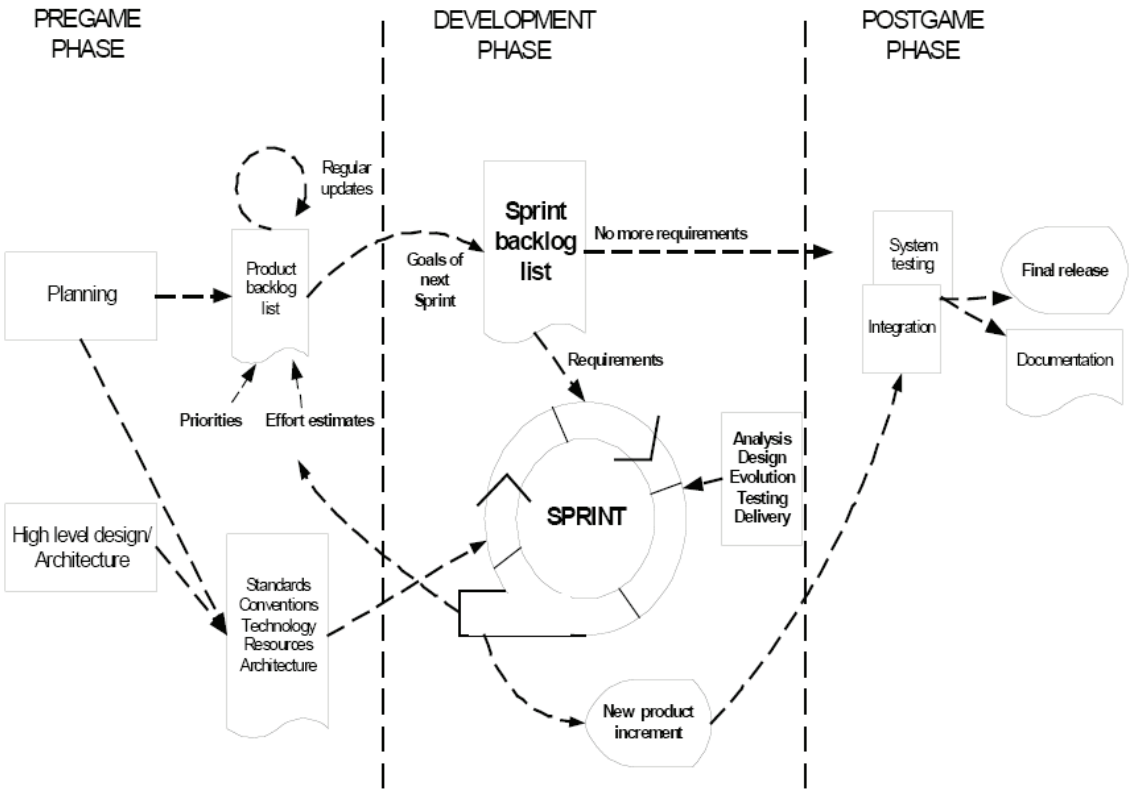

Figure 4. The Scrum Process [Abr02].

The pre-game phase is a continuous process that takes place ahead of the actual development phase. In case of totally new software, the pre-game phase plays an essential role in the beginning of the process and requires plenty of time before the first sprint can be started. In case of later sprints or maintenance upgrades, the pre-game phase is performed in the background and does not interrupt the development work. [Abr02]

Pre-game begins with planning, which focuses on defining the requirements and the project framework. The requirements are actively gathered from all over the organization and they are added in a product backlog list. They can originate from, e.g., product managers, customer services, sales and marketing division, or software developers. Project framework encompasses the definition of the project team; tools and other resources; risk assessment and controlling issues; training needs; and verification management approval. This framework is usually defined before the first iteration. High-level design and architecture planning are conducted for those new items in the product backlog list that may cause certain problems. In the beginning of the new software project, this phase is done thoroughly for all the items in the product backlog. [Abr02]

The development phase is an incremental process which is divided into several iterations. Each iteration consists of a one-day sprint planning meeting, a 30-day sprint, and a one-day sprint review meeting. Each day is further divided into daily Scrums that take 24 hours. New functionality is demonstrated at the end of each sprint. [Abr02]

Post-game is performed to close the release. A release may contain one or more sprints. At this point, no new items are allowed to be added in the sprint backlog. The post-game phase involves system & integration testing; other needed testing; documentation; and all the deployment activities. [Abr02]

2.3.2 Roles

The Scrum Master is a management role in a Scrum team. He or she works in the interface between management and the Scrum Team. The person is in charge of setting up the Scrum practices and ensuring that rules are followed. One of his or her main tasks is to organize a Daily Scrum meeting. In the meeting, the Scrum Master evaluates the project's status according to the team's comments. Based on the findings, the Scrum Master takes necessary actions immediately, e.g., to help a team member who has been struggling with a trivial task for several days. It is essential to remove impediments and make fast decisions in order to achieve the highest possible productivity. In general, it is better to proceed with some decision than no decision at all. The Scrum Master also participates in nominating a Product Owner and creating the Sprint Backlog. In a conventional project, the Scrum Master would be a team leader although there are significant differences between these two roles. [Abr02] [Sch02b]

The Product Owner (sometimes called a product manager or a project manager) is solely in charge of maintaining the Product Backlog via continuous interaction with all the stakeholders. The Product Backlog should be visible for everyone. It is essential that there is only one person maintaining the content and priority of a single Product Backlog in order to avoid any confusion that could otherwise arise. Everyone in the organization must respect the Product Owner's decisions and no one is allowed to tell the Scrum Teams to bypass the set priorities. [Abr02] [Sch02b]

In the beginning of each sprint, a cross-functional Scrum Team commits to implementing certain functionality from the Sprint Backlog. The team should consist of analysts, designers, testers, and coding engineers. It is preferable to have at least one very experienced engineer as part of the team to mentor junior engineers. A Scrum Team has the authority to decide on the needed actions and to organize itself in order to achieve the goals of each Sprint. Nobody is allowed to affect their work during the Sprint, but the team is allowed to ask others for advice and counseling. This freedom does not mean that there are no rules – in fact, the team must follow the existing charters, standards, conventions, architectures, and technology strictly. Experience shows that an open working environment suits Scrum Teams absolutely best because an open office allows people to communicate more easily, makes it easier to get together, and facilitates self-organization. [Abr02] [Sch02b]

Although most team members are assigned full time to the team, certain engineers having specific expertise may be available only part time. The team size should be from four to eight persons – according to Schwaber and Beedle, a smaller size would reduce productivity gains, while a larger size would generate too much complexity. At the end of a Sprint, the Scrum Master or a Project Manager may change the composition of the team to improve performance. In the shorter term, however, this harms the productivity gained from self-organization. [Abr02] [Sch02b]

If there is more than one Scrum Team working on a project, the interaction and dependencies between the teams should be minimized and the cohesion of the work within each team maximized. Successful projects have been run with up to ten Scrum Teams. On these occasions, it is preferable to coordinate the work with a daily Scrum of Scrums, which means that the Scrum Masters of every team gather after the Daily Scrum in their own Daily Scrum. In other words, there should be a "leading" Scrum Master who takes responsibility for the higher level Scrum meeting. In a conventional project, this role would be quite close to that of a project manager (since project manager leads team leaders). [Abr02] [Sch02b]

2.3.3 Practices

Product Backlog list

Schwaber and Beedle state that "Product Backlog is an evolving, prioritized queue of business and technical functionality that needs to be developed into a system". In other words, Product Backlog includes all the requirements that any of the stakeholders think are necessary for the product. These requirements can be features, functions, technical solutions, enhancements, and bug fixes that should be included in future releases. The requirements engineering process behind the Product Backlog depends on the organization in question, which means that the process can be either formal or informal. All the Product Backlog items are prioritized according to their importance to the organization. Items with high priority should be specified in more detail than items with lower priority. Items having the highest priority are seen as crucial to implement already in the next Sprint or release. Initially, the Product Backlog can be only a collection of thoughts, but at the later stages it should be developed into something more specific. [Sch02b] Often, product backlog items are described as user stories (see Section 3.5.3) [Agi08]. A story point, which describes a development effort estimate, should be defined for each user story. There is no a single definition for a story point because the formula depends on the project in question. Story points are defined by Scrum teams in an activity called Planning Poker. [Coh08]

The Product Owner is responsible for maintaining the Product Backlog. Sometimes the arising ideas are not well specified and, in this case, the Product Owner turns these ideas into features or enhancements to be developed. He or she works with different stakeholders to estimate how long it will take to develop a particular item. The time estimation is an iterative process and it is revised when the stakeholders get more information regarding the backlog item (what is the product like, what kind of architectural changes are needed, etc.). [Sch02b]

Sprint Planning Meeting

A two-phase Sprint Planning Meeting is held in the first day of each Sprint. At the first meeting, the most important stakeholders including the Product Owner, management, users, and the team figure out what functionality to build during the next Sprint. The Product Owner leads the discussion based on the Product Backlog and the results achieved in the previous Sprint (see Sprint Review). The goal is to choose the backlog items that the team or teams are able to implement in the next Sprint. The time estimates in the Product Backlog play a crucial role in the decision-making process. In the

absence of detailed specifications, the team is free to choose how it implements the functionality. [Sch02b]

At the second meeting, the team works by itself to determine a list of tasks relating to how it will reach the Sprint goals. The tasks should describe in detail the work needed to convert the chosen Product Backlog items into working software, each task taking about fourteen to sixteen hours to finish. The team can ask help from, e.g., system architect or some other special expert, if needed. The task list is called the Sprint Backlog. [Sch02b]

Sprint
The Sprint lasts exactly thirty calendar days. The first day is spent for the Sprint planning meeting and the last day for the Sprint review. The rest of the time, each Scrum team is free to strive towards accomplishing the Sprint goals as it wants. Turning complexity into a predictable product is a challenge that requires the best effort from everyone in the team. Although close team work is essential, the success of the Scrum team often depends on a couple of brilliant individuals. [Sch02b]

During the Sprint, the team has two mandatory working tools: 1) Daily Scrum meetings and 2) the Sprint Backlog. Every team member must attend the Daily Scrum meetings in person or via conference call. The Sprint Backlog must be kept up-to-date. The team is expected to modify the Sprint Backlog during the Sprint if it finds that more or fewer tasks are needed – no one else is allowed to touch this list. In case the team discovers that it is not able to implement everything that was promised, the Scrum master immediately identifies a solution together with the Product Owner. The problem can usually be overcome by removing a task that does not affect meeting the Sprint goals. [Sch02b]

No one has the authority to tell the team how to proceed with its tasks or add more functionality to the Sprint. The Sprint always fixes the development time, costs, and delivered quality. The only dynamic variable is the delivered functionality because the functionality can be changed as long as the team meets the Sprint goals. If the customer is not satisfied with the demonstrated functionality in the Sprint Review, the unimplemented parts are re-entered onto the Product Backlog and reprioritized. [Sch02b]

There is an option to cancel the Sprint, if the Sprint Backlog items become obsolete for some reason during the Sprint or if the team meets an obstacle that cannot be overcome. Despite of the fact that the team has the power to cancel its own Sprint, this should be done very rarely since terminations consume resources and cause extra costs. [Sch02b]

Daily Scrum (Meeting)
Each Scrum team holds its own Daily Scrum meeting every working day at the same place and at the same time – preferably the team should start the day with the Daily Scrum. It is a 15-minute status meeting where each team member answers three fundamental questions [Pre05]: 1) What did you do since the last Scrum meeting?; 2) Do you have any obstacles?; and 3) What will you do before the next meeting? The

meeting, e.g., improves communication and efficiency; helps finding and solving potential problems; speeds up decision-making; and gives everyone a good progress status update. The Daily Scrum provides management with useful information regarding psychological factors, such as a person's motivation or problems at home, which could be almost impossible to detect by reading a written status report.

The Scrum Master is in charge of conducting the Daily Scrum and keeping it on schedule [Coc06]. The atmosphere should be supportive [Yip07]. Earlier it was suggested that the Daily Scrum should be held in a separate room, having at least a whiteboard, a conference phone, a table, and chairs for the team [Sch02b]. Nowadays, the trend is to organize the Daily Scrum as a stand-up meeting without chairs [Yip07]. This makes the meeting more productive because people tend to be more focused on the core content of the meeting and expressing themselves in fewer words than in a traditional meeting context. The team should stand in a circle around a whiteboard (or table). In addition to the Scrum team, everyone who is interested to know about the day-to-day operation of the project can attend an individual Daily Scrum meeting. These guests must stand outside the team circle and are strictly forbidden to interject comments or to have side conversations.

The Daily Scrum is not a design session, which means that the team members should concentrate on producing very compact answers that provide only the necessary facts needed by the Scrum Master to identify potential impediments. The Scrum Master is responsible for finding ways to remove impediments and making the decisions necessary to achieve the Sprint goals. If the Scrum Master is not able to make the decision in the Daily Scrum, he or she has to provide it to the whole team within an hour of the end of the meeting. Sometimes the Scrum Master can organize a follow-up meeting for a bigger issue which came up in the Daily Scrum. [Sch02b]

Sprint Review (Meeting)
A Sprint Review meeting takes four hours on the last day of the Sprint. The team presents the built product increment to management, customers, users, and the Product Owner who assesses the introduced functionality. A good practice is to compare the Sprint goal and the Product Backlog with the actual results of the Sprint and introduce the reasons for any variances. The presentation should include also a high-level description of the applied system and technical architecture; which functionality was built; and what are its strengths and weaknesses. [Sch02b]

2.4 Summary

Most software process models are based on the waterfall model and the incremental model. Agile process models have been developed to meet the challenges posed by the fast-changing ICT markets. Ten of the most interesting agile and conventional process models were evaluated according to their suitability for the purposes of system A1. The winner of the comparison, Scrum, is an agile model that has proved to be successful in thousands of software projects worldwide.

3 Requirements engineering

This chapter presents the importance of requirements engineering in every software development process. The chapter also introduces the characteristics of a good requirement and the activities a successful requirements engineering process should include.

3.1 Definitions

A requirement is "anything that drives design choices" [Law97]. In other words, it is "something that the product must do or a quality it must have" [Rob06]. Requirements arise either because the type of product demands certain functions or qualities or because a stakeholder wants that requirement to be part of the delivered product [Rob06] [Wie03]. The IEEE Standard Glossary of Software Engineering Terminology [IEE90] defines a requirement as

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

As we can see, there is no single unambiguous definition for a requirement – practically everyone has a slightly different view on it. However, it is important to notice that requirements do not only come up based on users' or developers' wishes but there may be a huge variety of different types of sources that should be taken into account.

Requirements engineering covers all of the activities involved in discovering, documenting, managing, and testing a set of requirements systematically for a system. The term engineering implies that systematic and repeatable techniques should be used to ensure that system requirements are complete, consistent, relevant, etc. [Som04].

Quite often in software literature, software requirements are classified according to the type of the requirement. A functional requirement specifies an action that a system must be capable of performing. It can arise from the work that the stakeholders need to do [Rob06]. A non-functional requirement describes the property or quality of the system, such as performance, reliability, usability, and legal restrictions [Rob06]. A constraint is a global issue that somehow limits development. Constraints can arise from standards or they can emerge from software, hardware, project, or business constraints [Rob06].

Another way to classify requirements is to sort them into three levels. Business requirements represent high-level objectives of the organization or customer of the system – these people include decision-makers in a leading position. Their requirements often reflect the current and future business practices they want to adopt. Sometimes business requirements tend to overlap with user requirements although business requirements are usually of a higher level. [Cou05] [Wie03] User requirements describe user goals or tasks that the users must be able to perform with the system. [Wie03] These should include the features or attributes that the system must have [Cou05].

Technical details should be avoided – the best way is to look at the system as a black box. Technical requirements describe how the system should be implemented from the technical point of view [Kau07].

The relation between the three types and three levels of requirements is shown in Figure 5. According to Marjo Kauppinen [Kau07], the main focus in too many projects is on technical and functional requirements.

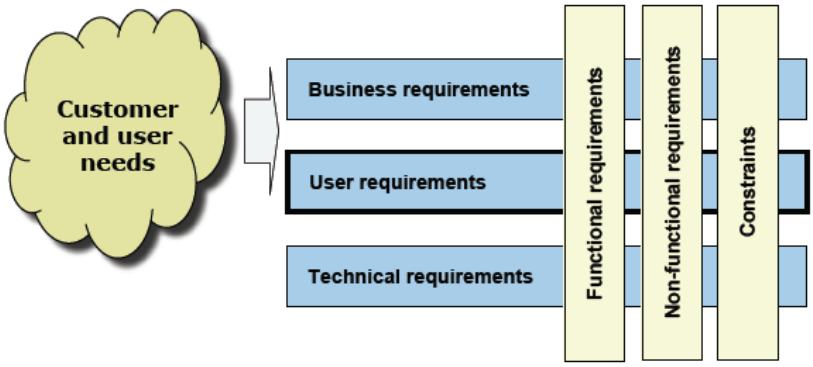

Figure 5. The relation between the three types and three levels of requirements [Kau07].

A project stakeholder can be anyone who has any real interest in the project. Stakeholders participate actively particularly in the requirements gathering phase and determine the product. Potential stakeholders can be, e.g., customers, users, developers, testers, technical experts, project managers, marketing department, and product managers. All the stakeholders must be individually named. [Rob06] [Pre05]

3.2 Importance of requirements engineering

Requirements engineering is often underrated by organizations although it actually plays the most significant role in the whole R&D process [Rob06]. A correct piece of software cannot be developed without knowing the correct requirements. Correct requirements come from understanding the work that the product is intended to support. According to the statistics provided by both Steve McConnel [McC04] and Jerry Weinberg [Wei97], over 60 percent of errors during the R&D process originate in the requirements activity. In addition to this, a study conducted in IBM in the middle of the 90's [Kan94] shows that the relative cost of finding an error at the later stages of an R&D process increases almost exponentially over time (see Figure 6). This means that if an error is found during the requirements engineering phase, the cost to fix the bug would be 1 unit of money. But if the error is found in the maintenance phase, the cost would be 368-fold.

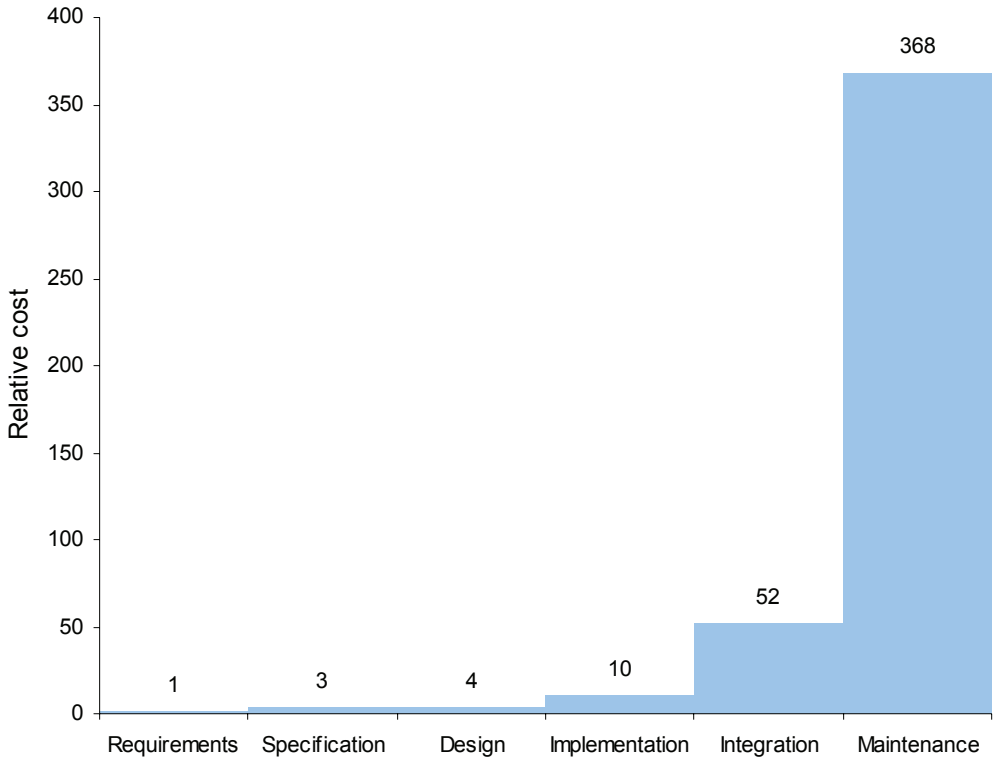

Figure 6. Cost to detect and correct a fault.

The final results of the QURE research project [QUR01] describe several benefits of a properly conducted requirements engineering process. Requirements describe the collectively accepted objectives and are an agreement between the customers and suppliers on what is to be developed. Requirements improve the commitment and motivation of project members and communication between different stakeholders. They are also the basis for systematic project planning, specification, and design. Well-documented requirements support testing and reduce rework. Requirements engineering is considered to be equally important for agile software projects and for conventional projects – there is just a small difference in the way the requirements are discovered and managed.

3.3 Characteristics of a good requirement

There is no single definition for a good requirement, but probably the most complete list can be found in the IEEE Recommended Practice for Software Requirements Specifications [IEE98]. A good requirement should be correct; unambiguous; complete; consistent; ranked for importance and/or stability; verifiable; modifiable; and traceable. Adjectives, such as understandable, necessary, concise, and reachable can be also found from the literature [Kau07] [Les07]. All these characteristics are explained below based on the slightly modified IEEE specifications.

A requirement is correct if, and only if, each requirement stated therein is one that the software should meet to respond to a real need. The requirement must be compared to any related specifications, documents, and standards to ensure that it agrees with them. Sometimes the customer or user can determine if the requirement correctly reflects the actual needs. [IEE98]

A requirement is unambiguous if, and only if, every requirement stated therein has only one interpretation. Each characteristic of the final product should be described using a single unique term. The term should be explained in a glossary if the term used in a particular context could have multiple meanings. The requirement should be unambiguous to all stakeholders despite of their background. However, there is a risk that a representation which improves the requirements specification for the developer may be counterproductive in that it diminishes understanding by the user and vice versa. [IEE98]

A requirement is complete if, and only if, it includes the following elements [IEE98]:

a) All significant requirements, whether relating to functionality, performance, design constraints, attributes, or external interfaces. In particular any external requirements imposed by a system specification should be acknowledged and treated.
b) Definition of the responses of the software to all realizable classes of input data in all realizable classes of situations. Note that it is important to specify the responses to both valid and invalid input values.
c) Full labels and references to all figures, tables, and diagrams in the requirement and definition of all terms and units of measure.

Consistency refers to internal consistency. If a requirement does not agree with some higher-level document, such as a system requirements specification, then it is not correct. [IEE98]

A requirement is ranked for importance and/or stability if each requirement in it has an identifier to indicate either the importance or stability of that particular requirement. Typically, all of the requirements are not equally important: some requirements may be essential, especially for life-critical applications, while others may be desirable. Each requirement in the requirement should be identified to make these differences clear and explicit. Identifying the requirements in the following manner helps [IEE98]:

a) Have customers to give more careful consideration to each requirement, which often clarifies any hidden assumptions they may have.
b) Have developers to make correct design decisions and devote appropriate levels of effort to the different parts of the software product.

A requirement is verifiable if, and only if, every requirement stated therein is verifiable. A requirement is verifiable if, and only if, there exists some finite cost-effective process with which a person or machine can check that the software product meets the requirement. In general, any ambiguous requirement is not verifiable. Non-verifiable requirements include general statements, such as “works well,” “good human interface,” and “should usually happen.” These requirements cannot be verified because it is

impossible to define the terms “good,” “well,” or “usually.” The statement that “the program should never enter an infinite loop” is non-verifiable because the testing of this quality is theoretically impossible. [IEE98]

A requirement is modifiable if, and only if, its structure and style are such that any changes to the requirements can be made easily, completely, and consistently while retaining the structure and style. [IEE98]

A requirement is traceable if the origin of each of its requirements is clear and if it facilitates the referencing of each requirement in future development or enhancement documentation. [IEE98]

A requirement is understandable if, and only if, the reader can easily understand the meaning of the requirement [Kau07]. It is important to note that although readers may have very different technical and business backgrounds, they should nevertheless get the same picture of the requirement.

A requirement is necessary if, and only if, it brings real business value to the company. There should always be a clear business case behind the requirement. However, the requirement does not necessarily increase revenues, but it may reduce costs in many ways or bring in benefits in the longer term. [Les07]

A requirement is concise if, and only if, it is stated in a declarative language that is brief and easy to read. Yet, the requirement should convey the essence of what is required. [Les07]

A requirement is reachable if, and only if, there is a great chance that it can be implemented for the available money; with the available resources; and in the available time. [Les07]

However, reaching all these goals is almost impossible and if a company tries to reach certain specific goals (e.g., complete, unambiguous, and understandable) at the same time, the costs may skyrocket. [Dav93]

3.4 Requirements engineering process

Every organization differs in terms of organizational culture; the level of experience; the ability of the people; the nature of the business; the size and resources; and the type of the systems it develops. There can even be significant differences within a single company. Due to these and many other differences, there is no single requirements engineering process which is ideal for all organizations. In fact, the process should be defined separately for each organization’s purposes. In this section, I will introduce the most traditional requirements engineering process model, which can be customized according to the organization’s needs. [Kot98]

Depending on the reference, the traditional requirements engineering process includes three to seven phases [Wie03] [Som04] [Vli04] [Pre05] [Tha05] [Som06]. The most common activities are elicitation, analysis, specification, and validation. In addition to these, good requirements management practices are needed. Ultimate customer focus and involvement of all the stakeholders throughout the process is essential. General business goals should also be taken into account in performing each phase. [Pre05]

3.4.1 Elicitation

During requirements elicitation the requirements analysts or a requirements engineering team gathers needs of the project stakeholders by helping them to understand and articulate their explicit and hidden problems. It is worth breaking down the problem into smaller pieces, such as use cases. There are various techniques that can be used in the elicitation process – the most appropriate method should be chosen according to the context. These techniques include collaborative sessions (e.g., focus group, Joint Application Design, brainstorming), interviews, questionnaires, ethnography, prototyping, documentation, modeling, role-playing, checklists of non-functional requirements, Delphi technique, and domain analysis. The requirements analysts should not rely too heavily on the "free" customer input, because an individual opinion does not necessarily represent the real target user's viewpoint. A vision and scope document containing the product's business requirements must be written (as the final output). [Vli04] [Tha05] [Wie03] [Pre05] Experience has shown that communication between stakeholders can be problematic and communication gaps arise easily [Les07]. Therefore it is essential to get all the important stakeholders involved from the inception of the requirements engineering activity and continue their involvement throughout the process [Cou05]. This way, the stakeholders should not be surprised by the results of the process and they feel they made the requirements discoveries with the requirements engineering team.

3.4.2 Analysis

The gathered needs should be analyzed carefully. The main emphasis in the requirements analysis phase is on getting a deeper understanding of the software to be developed. The goal is to present the requirements to such a quality that the management can create project estimates and the developers can proceed with actual development activities [Wie03]. The needs are analyzed using some conceptual model [Tha05]. Often it is helpful to present the requirements in multiple ways [Wie03], e.g., using user scenarios, context diagrams (including identification of external parties), and prototypes [Wie03].

It is challenging to present requirements at an ideal level. Therefore, many companies employ expert users to provide a link between users and developers. Expert users have an ability to express the functional requirements in a way that can be easily translated into a design feature but is still understandable by end users. [Les07]

Requirements are divided into three categories (see Section 3.1): business requirements, user requirements, and technical requirements. Each of these main categories should be categorized further into three subcategories (see Section 3.1): functional requirements

(what functions/services a system must be capable of performing), non-functional requirements (the properties of the system, including performance, reliability, and usability), and constraints (standards, software/hardware constraints) [Kau07].

The requirements analysts should conduct a feasibility study (see Section 3.5.2) to see whether a single requirement could be implemented at acceptable cost. Identification of risks, potential conflicts with other requirements, and technical obstacles are also parts of the feasibility study. In the case when the requirement affects multiple subsystems, it must be apportioned among all the related systems. All the requirements should be carefully prioritized from the customer's or the target user's perspective. Various analytical approaches can be used for this purpose. [Wie03] [Pre05] [Som06]

3.4.3 Specification

All the requirements should be documented in a consistent, accessible, and reviewable way [Wie03]. A requirements specification can be a written document, a collection of graphical models, a set of usage scenarios, a prototype, or any combination of these [Pre05]. Large conventional projects (elephant projects) should build and use a complete template for documenting requirements [Rob06]. Each requirement should be uniquely labeled and the quality attributes must be specified. Many organizations adapt the SRS template described in IEEE Standard 830-1998 (IEEE 1998b), but there are various competing templates, too (e.g., the Volere requirements specification [Rob06, pp. 451-]). Agile projects, on the other hand, should not write a complete specification, but rather concentrate on enhancing the preliminary requirements description (e.g., in the Wiki) [Wie03]. However, it is recommended that also some of the formal topics be covered in the agile documentation.

3.4.4 Validation

The validation step includes reviewing requirements and organizing informal inspections. The goal is to verify that the requirements are understood correctly and conform to specified standards. As described in Section 3.2, it would be extremely expensive to correct a wrongly-defined requirement later in the development process. Some kind of a validation step should be performed at every stage of the requirements process in order to ensure a high-quality product. The process should include reviewing documents, conducting traceability analysis, and creating test cases and acceptance criteria for requirements. The characteristics of a good requirement are described in Section 3.3. When no further improvements are needed, the approved set of requirements is created. [Pre05] [Tha05] [Wie03]

3.4.5 Management

The requirements for large software systems tend to change over time because the stakeholders' understanding of the problem improves during the process [Som06]. Requirements management is a set of activities that help in identifying, controlling, tracking, and changing requirements at any time as the project proceeds [Pre05]. The project needs to have a process for proposing changes and evaluating their potential costs and impact on the project [Wie03]. A good practice is to establish a change control board composed of key stakeholders, for handling these issues. The board is

responsible for evaluating the proposed Change Requests (CRs) and deciding which ones to accept and which ones to reject. It also sets priorities for the changes and may allocate them to the development roadmap.

Other continuous activities that belong to requirements management are conducting requirements-change impact analysis for each proposed requirement, establishing a baseline, controlling versions of requirements documents, maintaining a history of requirements changes, tracking the status of each requirement, measuring requirements volatility, and using a requirements management tool. There are many ways for conducting these activities, such as maintaining traceability matrixes or single documents. All of the chosen practices should be mentioned in the process description. [Wie03]

3.5 Requirements engineering practices

There are various techniques that could be used during the requirements engineering process. Due to the nature of this thesis, this section introduces four practices that are extremely suitable for agile process models. Other practices were discussed in Section 2.3.

3.5.1 Focus group

A focus group is probably the best way for gathering data from several participants in a short period; covering only a few questions without a lot of depth from each participant; and discussing topics that are not sensitive or likely to be influenced by the opinions of others. However, if you intend to get a lot of details from end users and a large number of participants is not of primary importance, interviews are the best choice. Surveys, on the other hand, are best suitable for collecting a large number of responses. [Cou05]

The focus group is one way to improve your understanding of a target audience that you have very limited information about. It is extremely suitable in the beginning of the requirements engineering process and when starting to create fundamentally new products. In the focus group session, six to ten people are brought together to discuss their opinions on topics introduced by a moderator. A separate note-taker is needed to help the moderator. The session typically lasts from one to two hours. One of the major advantages of a focus group is the fact that group dynamics bring up topics that the moderator would otherwise not think of asking. [Cou05]

3.5.2 Feasibility studies

A feasibility study should be conducted in the beginning of the requirements engineering process [Som06]. It is a short study that aims to answer the following questions:

1) Does the system contribute to the overall objectives of the organization?
2) Can the system be implemented using current technology and within given cost and schedule constraints?
3) Can the system be integrated with other systems which are already in place?

The feasibility study is based on a set of preliminary business requirements, an overall description of the system, and a description of how the system is intended to support business processes. The study encompasses information assessment, information collection, and report writing. Information can be gathered from different experts and managers who know the system well. The feasibility study should be completed within two or three weeks. The resulting report should recommend whether or not it is worth carrying on with the requirements engineering and system development process. [Som06]

3.5.3 User story

User stories are primarily used by eXtreme Programming and Scrum to express requirements [Les07]. A single story is only a couple of sentences long and should fit on a small index card. Therefore, more formal acceptance tests must be written for each user story by the customer to determine later whether the validation conditions are met satisfactorily. The work is performed in varying levels of detail, and, at each level, the work is broken further down into smaller tasks [Agi08].

According to the Ron Jeffries [Agi08], user stories have three critical components: cards (physical medium), conversation (discussion surrounding the stories), and confirmation (tests and validation that verify story completion). More specifically, a good user story should be: independent, negotiable (not an explicit contract concerning features – details are co-created by the customer and programmer during development), valuable to the business, estimatable, small (should not be more than eight to sixteen hours of effort), and testable (should include validation criteria).

User stories should not be mixed with use cases or user scenarios. As a rule of thumb, a user story is less formal than a use case and shorter than a user scenario. [Pre05]

3.5.4 Product backlog

The product backlog is a Scrum practice for listing requirements in a prioritized order. It is explained in more detail in Section 2.3.3.

3.6 Summary

The majority of errors during the R&D process originate from requirements engineering. Therefore, requirements engineering should be considered as the most essential part of the whole R&D process. The requirements engineering process includes four phases: elicitation, analysis, specification, and validation. Good management practices are needed throughout the process. Useful requirements engineering practices for agile process models encompass, e.g., eliciting requirements in focus groups, conducting feasibility studies, describing requirements in the form of user stories, and maintaining requirements in a product backlog.

4 Evaluation of the research methods

4.1 Research methods

The scope of this master's thesis was very large and, therefore, I needed to use numerous sources for collecting reliable data. Chapters 2 and 3 are based on nearly 30 references regarding the software process models and requirements engineering. Many agile models have evolved significantly during the past few years and I have tried to use the most recent findings in my thesis. Because there is a huge variety of references available relating to my topic, it also means that many alternative views have been presented. Quite often, I have referred to the original founder of the specific methodology in question and his or her later improvements.

The findings in chapters 5 and 6 are based on the internal documents and my personal findings of over 30 individual interviews within companies A, B, and C. The documents are highly confidential and I do not have the permission to publish the names. They encompass detailed descriptions of the ABC IT process model and the related sub-processes. I have also had access to internal documentation of projects A1, B1, and C1 including all the project plans and the running project documents.

I met people from different levels and positions of the ABC organization (IT project managers, software developers, product managers, steering group leaders, etc.) during the research phase. The interviews followed a basic structure that is described in "Appendix D: Templates for the interviews". Despite of this, they were quite informal and I paid emphasis on the most interesting findings that came up during the discussion. Also the interviewees' areas of expertise affected my questions. I aimed to get the person's true personal opinion regarding the topic by asking neutral questions, in order to form an objective view later on.

Due to the fact that the software and requirements engineering processes differed somewhat in all of the three companies, the answers were comparable only within each country. However, projects A1 and B1 shared many common problems. As a big surprise, business and IT representatives within each of the companies had noticed many similar problems in the process although they naturally talked from a little bit different viewpoints.

4.2 Analysis

The references encompass many new, important books and web articles relating to the topic. Which are the most essential is a matter of opinion, but I have tried to use all the relevant references when collecting my data. A large variety of modern references ensures that my findings are not based only on a single opinion, and the theory has been proved to work successfully also in practice (e.g., case examples) over time.

I made a clear choice to conduct a small number of qualitative interviews instead of collecting quantitative data. Through qualitative interviews I had the opportunity to gain a deeper knowledge of the existing processes and identify the problems and bottlenecks.

By choosing a heterogeneous group of interviewees, I made sure to get as broad a perspective to the topic as possible. The cohesion of the answers proves that the findings are relevant and I had enough interviewees.

However, there is always room for improvement. Especially going through the suggested methods with all of the key stakeholders during and at the end of the process it would have been fruitful to ensure that they agree with the suggestion and to get their input on how the proposal should be modified. Of course, I have continuously kept both my instructor and IT manager up-to-date of the progress of the study and asked for comments on my findings. However, an opportunity to do this kind of iterative work on a larger scale would have produced additional reliability.

5 Current A1 Process Model

The A1 process model enforces the organization's general ABC IT process model. This chapter describes first the ABC IT process model and continues with introducing the current A1 process model. The main emphasis will be on describing the requirements engineering phases of these processes.

5.1 ABC IT process model

5.1.1 Overview

The ABC IT process model (ABC IT PROMO) defines an overall work model for IT development projects and is an application of the ABC process model. The ABC IT model is a waterfall model (see Section 2.2.1) consisting of several sub-processes and deliverables as shown in Figure 7. However, project management practices are described in the ABC process model (ABC PROMO). Because the ABC IT process model does not suggest detailed development methodologies, every project can choose a suitable methodology for its purposes. [ABC06]

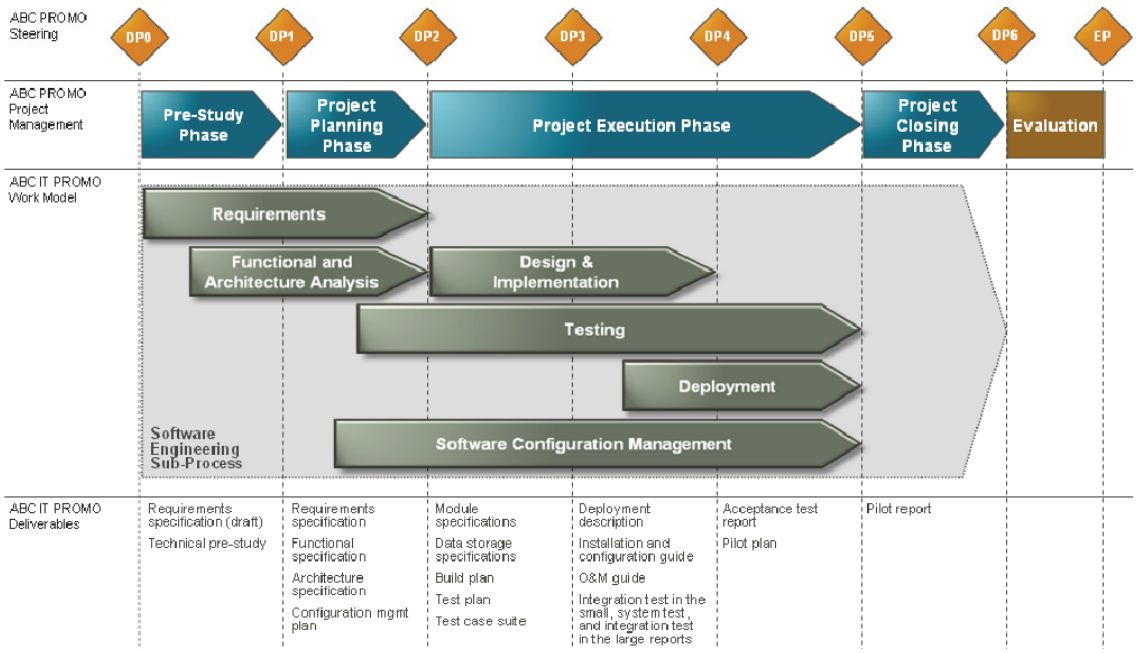

Figure 7. ABC IT process model.

Decision points (DPs) give a rhythm to the development process. They are external control points for verifying that the project meets the required standards. Basically, the project is not allowed to proceed to the next phase before the project steering group gives a formal approval. At each decision point, the steering group must decide whether to accept, reject, or require some clarifications before proceeding to the next phase. Many of the ABC IT model's sub-processes take place between more than two decision points, so, in practice, most of the activities can continue even if the project needs to wait for a decision. The time between DP0 and DP2 is allocated to different kinds of

project preparation activities, the actual project is executed between DP2 and DP5, and the closing activities take place between DP5 and the evaluation point (EP). [ABC06]

There are eight decisions points altogether. These are thoroughly explained in "Appendix C: Decision Points". Their main objectives are described briefly here [ABC06]:

- DP0 – ideas are approved & permission is given to start pre-studies
- DP1 – pre-studies are approved & permission is given to start project planning
- DP2 – the project plan is approved & permission is given to begin the design
- DP3 – the design is approved & permission is given to start implementation based on the design
- DP4 – deliverables are approved & permission is given to start hand-over preparations
- DP5 – final results are approved, hand-over completed & permission is given to initiate closing
- DP6 – the end of project is approved
- DPE – the business case and benefits are evaluated in an evaluation report

5.1.2 ABC requirements process

"The purpose of the ABC requirements process is to document the agreed requirements of the solution to be built within the project. The requirements must be presented in a form that is understandable to both the customer as well as the development team. Requirements define what the system should do. The requirements provide a goal for the project, as well as the criteria for the acceptance and validation of the solution upon the delivery." [ABC06] The ABC IT process model [ABC06] specifies the essential activities in the requirements phase in Table 1.

Table 1. Activities during the ABC requirements process.

Role(s)	• Analyst • Stakeholders
Tasks	• Collect and analyze requirements. • Finalize requirements (define and specify the set of requirements to be implemented). • Inspect with the customer and stakeholders that the requirements are correct (before DP1 and before DP2).
Input	• The scope of the solution is decided upon (which features, when to deliver, who will use it and for which purpose?) and it is concrete enough for the customer and development team. • All relevant stakeholders are available and they are committed to contributing to the requirements. • The DP1 and DP2 deliverables are defined with the customer.
Output	• Vision of the solution (i.e. problem statement) • Stakeholders and actors • Functional requirements of the solution (i.e. use cases) • Non-functional (i.e. supplementary) requirements • Prioritization of the requirements • Open issues (if any) and assumptions
Related Templates	• Requirements specification

5.1.3 ABC functional and architecture analysis

"The ABC Functional and Architecture Analysis (FAA) process prepares a solution for the customer needs (how the solution meets the requirements). This solution is a product of activities that carry the customer's requirements into the foundation of the design and implementation work, i.e. functional specification and architectural specification. The FAA process provides both the technical and methodological guidelines, as well as the scheduling basis for the design and implementation work. Implementing prototypes to ensure the feasibility of the solution can support the FAA process work." [ABC06] ABC IT process model [ABC06] specifies the essential activities in the functional and architecture analysis phase in Table 2.

Table 2. Activities in the ABC functional and architecture analysis.

Role(s)	• Analyst • Architect • Designer • (Tester)

Tasks	• Functional analysis
	  ○ Analyze functionality (complexity, feasibility, sanity, stakeholder effects, missing functionality)
	  ○ Describe the system's functionality (use cases)
	• Architecture analysis
	  ○ Define the system's architecture
	  ○ Analyze opportunities for reusing existing solutions
	  ○ Check IPRs

Input	• Available information from the requirements process and Project Management • Enterprise-wide architectural strategies and policies • Reusable assets
Output	• Solution's functionality in terms of structured and categorized use cases and sequence diagrams • Solution's architecture • The interconnection between solution's functionality, architecture and the design and implementation • Ability to refine the solution based on analyzed changes in the functionality, architecture, quality attributes, or scheduling. • UI visualizations, UI navigation charts • Possible prototype(s) • Requirement feasibility • Requirement traceability • Solution references • Technology references • Technical roadmaps, future plans
Related templates	• Technical pre-study (from ABC process model) • Functional specification • Architecture specification

5.2 A1 process model

5.2.1 A1 system

A1 and its dependable systems provide a holistic solution for order entry, order management, billing, and customer care. The products and processes have been implemented using the ABC target architecture. The development of the A1 system was started in 2003 and there have been several deployments. The goal was to reduce IT

costs dramatically by replacing all the existing legacy order, delivery, and billing systems with the target architecture. Due to various reasons, the migration path has been delayed and the architecture became more complex.

Nowadays, A1 is used to serve millions of company A's customers and it is the master for managing customer and order data. However, there are still several dozens of interfaces and dependable systems. New orders and changes in existing orders are received from the channel applications. A1 also manages the overall order management process. There is a close coupling between systems – channel applications are totally dependent on the data and logic of the A1 system. A1 is developed using one of the leading packaged software although a lot of customization is needed, too.

5.2.2 Overview of the A1 process

The A1 process is based on the ABC IT model, but it is customized to some extent for the purposes of the A1 project. The development is divided into releases and small development packages. Releases are conducted two to four times a year and the average time to get an initial idea to the market is about a year. Changes in the releases are usually rather big in terms of required man days and they encompass new products for consumers and B2B customers, process changes, and usability changes. Small development packages, on the other hand, are much smaller and they are delivered approximately six times a year. Small development changes are often more critical than release changes, but they should not cause a major regression testing impact on the existing system. The time-to-market is about 8 months. Operational maintenance is handled by a different unit in the A1 organization.

Company A has outsourced the development activities to two vendors. However, common project management is still handled by A1, but both vendors have their own sub-managers (sub-project manager, team leaders, test manager, etc.). The A1 management team consists of 9 members and they have the following roles: project manager, test manager, external systems manager, environment manager, deployment manager, end-to-end manager, and four business request (BR) managers. Part of the development takes place on-site in Helsinki although most of the routine tasks are performed offshore in a few locations. Depending on the project phase, there are dozens of consultants working in the project.

5.2.3 Requirements engineering process

In the A1 process, most of the business requirements are defined for an initial business idea before DP1. The exact requirements engineering process varies somewhat depending on the nature of the idea and the person the request is coming from. A general process flow is described in Figure 8 from the perspective of company A's product management.

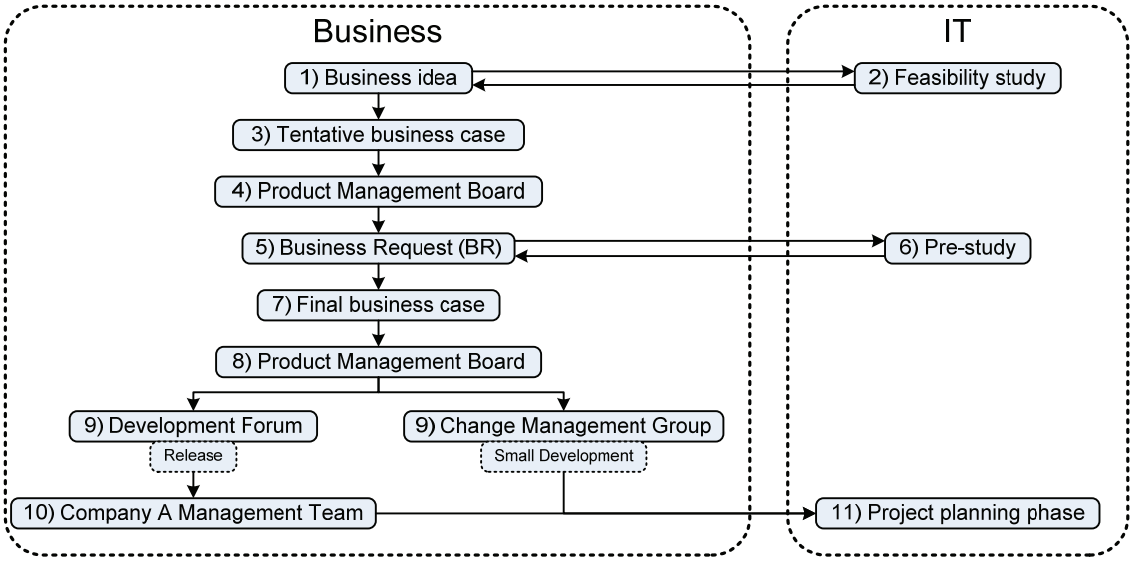

Figure 8. Current A1 requirements engineering process flow.

1) Business idea

New business ideas can come from almost anybody, but quite often they are proposed by one of the product managers. Ideas arise all the time and the business needs to continuously manage emerging ideas. Each business unit filters and refines established ideas before they are further processed in the requirements engineering pipeline. If the idea is very simple and it only affects the A1 system, a business representative may create a business request (BR) straight away (see step 5).

2) Feasibility study

In a case when a business finds out, possibly after the tentative discussions with IT, that the idea may cause more complex changes in the A1 system and it may also affect many external systems, a feasibility study request must be submitted. In practice, about a fifth of new business ideas require a feasibility study. The feasibility study process is described in more detail in Section 3.5.1.

3) Tentative business case

When a business has enough knowledge about the business requirements regarding the change, they write a tentative business case. a business case includes an overall description of the business idea, all the defined business requirements, the associated costs, and the estimated revenues or other benefits over time.

4) Product Management Board

Based on the business case, the business decides whether to go further with the idea or disregard it.

5) Business Request (BR)

If the idea seems to have potential, the business creates a BR and sends it to IT department. BR should include as detailed business requirements as possible, so that IT will be able to conduct a technical pre-study later on.

6) Pre-study
The purpose of the pre-study is to figure out a high-level technical and functional solution of the problem. Pre-studies are only conducted when the requirement is complex and it may affect many related systems. Even if IT does not necessarily conduct a complete pre-study for all BRs, they prepare answers to key technical questions. Pre-studies are normally conducted between DP0 and DP1. The pre-study process is described in more detail in Section 5.2.4.

7) Final business case
After IT has provided an answer to a BR including R&D cost and time estimates, the business can finalize the business case. If the business case still looks positive, the BR is raised on a release or a small development candidate list.

8) Product Management Board
Product Management Board prioritizes BRs and decides which changes are brought to releases and small development packages. Each department has a specific slot in releases and small development packages for their requests. For example, product management might be allowed to include 750 man days of work in the next release. There is a person allocated to maintaining a prioritized list in Excel format that includes the priorities and workload estimates. However, the actual prioritization takes place in the meeting and it is based on a voice vote. The group meets once a week and it includes only participants from product management.

9) Development Forum or Change Management Group
The Development Forum prepares the final proposal to suggest which BRs will be included in the next release. The forum is too big for detailed decision-making, but some business ideas can be discussed there if they impact many departments. The suggestion encompasses all requests from the different departments. The Change Management Group, on the other hand, proposes the final list of BRs that should be included in the next Small Development Package.

10) Company A Management Team
The Company A Management Team gives the final approval for the proposals which were agreed in the Development Forum and the Change Management Group. The team consists of the CEO of company A and all the department managers.

11) Project planning phase
Preparations for a new project are started based on the approved list, which includes the BRs that were qualified to a particular release or a small development package. The activities are described in more detail in Section 5.2.5 and most of them take place between DP1 and DP2.

5.2.4 A1 pre-study phase

The requirements engineering process that was described in Section 5.2.3 begins already before DP0 and continues until DP1. Someone could argue that the requirements engineering process actually continues until the end of the whole R&D

process since many requirements change over time. This is absolutely true, but the later activities are not explained in this chapter.

The A1 pre-study phase does not strictly follow the ABC IT process model. Pre-studies are conducted once in a while and the main purpose is to determine workload estimates, so that the business is able to decide before DP1 which BRs it wants to be included in the release. The stakeholders that are involved in the pre-study phase include BR managers, a few senior consultants, external system experts, and business representatives. Depending on the situation either a BR manager or a senior consultant prepares a high-level solution for the problem and the stakeholders meet once or twice to discuss the open issues. Other activities in the A1 pre-study phase include defining the release or small development project's scope and schedule.

The official pre-study phase between DP0 and DP1 takes about 4 to 5 weeks. The exact time depends mainly on the schedules of decision forums. When the pre-studies and the release or the small development scope are approved, the A1 development manager is given the permission to start the project planning.

5.2.5 A1 project planning phase

A1 project planning encompasses vendor selection using the Request-For-Proposal (RFP) process, preparing project plans, and allocating people to the required roles. The planning phase between DP1 and DP2 lasts about 5-6 weeks.

A1 RFP process

1) The A1 development manager sends RFP material to potential vendors straight after DP1. The material is based on the conducted pre-studies, but it cannot include any confidential material. RFP materials are often prepared already at the end of pre-study phase.
2) The A1 development manager organizes a Question & Answer session together with the potential vendors in order to ensure that both parties have a common understanding regarding the material.
3) Potential vendors are expected to return their first bids 2-2.5 weeks from receiving the RFP material.
4) The A1 development manager conducts a quick analysis and cuts down the number of potential vendors for the second round and sends the second set of RFP material. In the second set of RFP material the work is usually divided into functional parts for each of the subcontractor in order to have the possibility to choose more than one vendor.
5) Short-listed vendors are expected to return their second bids within 4-5 working days from receiving the RFP material.
6) The A1 development manager conducts a quick analysis, modifies the combination of the functional parts, and sends the third set of RFP material.
7) Short-listed vendors have 4-5 working days to submit their final bids.
8) After receiving the final bids, company A selects the vendors for the project.

Project plans are written using the organization's ABC standard templates and the people are allocated to suitable roles according to their competence. These two activities are not in the scope of this thesis.

5.2.6 A1 project methodology

Feasibility study process

1) A business representative fills in a Feasibility Study Template (FST)
2) The FST is sent to the Pre-Study IT Council
3) The persons in charge are appointed
   - Main responsible person and the sub-responsible persons (usually from IT side)
   - Usually one person from each of the related systems
   - Bottlenecks
     - No dedicated persons or even time reserved for these tasks, but the people participate on top of their full normal duties.
     - Many consultants are used in this activity. More internal resources should be employed (efficiency).
4) The group meets a couple of times to discuss the case and provides a high-level functional and technical solution, which equals the result of a normal pre-study
5A) If the business representative who submitted the FST feels that there still is a business case, a new Business Request (BR) is created (however, a BR is nowadays created first)
5B) If the business representative who submitted the FST feels that the suggested solution does not meet the goals or is too expensive, the planned change is left on the table

Business Request Process

1) A business representative fills in a Business Request (BR) template
2) The template is sent to the A1 Small Development Team using a distribution list
3) One of the four BR managers feeds the required details of the BR to the Quality Center
4) The Small Development Team prepares an estimate of the A1 work regarding the BR
   - If the workload is more than 50 MDs, the BR is tentatively set to one of the upcoming major releases
   - If the workload is less than 30 MDs, the BR is tentatively set to one of the upcoming small development packages
   - If the workload is between 30 and 50 MDs, the BR is a borderline case and, e.g., also the estimated work needed in the related systems is taken into account
5) Before DP0, A1 Change Management Group (CMG) sends a request to key business representatives to prioritize their BRs
   - In major releases, each of the four business sectors has a slot of approximately 750 MDs to be used
   - In SD packages, each of the four business sectors has a slot of approximately 40 MDs to be used
   - Representatives of Business IT Management (BIT), B2B Sales, and Customer Service are in charge of prioritization
6) After the prioritization, the A1 project team asks the representatives of all the related systems whether they are able to implement the change according to the project schedule. If this is not possible, the BR will be replaced with another BR.

5.3 Findings from the interviews

The findings in this chapter are based on several interviews that were conducted within company A in October 2007. A total of more than 20 people from IT and business units gave their opinion on the A1 R&D process. For more details on the methodology, see Chapter 4.

5.3.1 Software process

The A1 process is too rigid due to the fact that the releases are big – even the size of small development packages has grown too much. Business requirements usually change while the BR is waiting to get into a release. Nowadays, the rigid process almost prevents changing the requirements during the R&D process and, therefore, business sometimes tries to estimate future needs by asking more than is needed. Reducing the size of the release would shorten the time-to-market and make the process more flexible.

Another way to significantly shorten the time-to-market would be to first build a simplified solution for pilot selling. A fully automated version could be implemented on top of the simplified solution after the success of the product has been proved on the market. Thereby, IT does not need to make extra effort to develop an automated version if the product fails on the market. The risk in implementing the solution in two phases is that IT may actually need to do almost double the work, if they cannot utilize the simplified solution when enhancing the functionality.

IT solutions should be as generic as possible. It does not make sense to implement many similar product specific solutions individually again and again if the same output could be achieved by creating a holistic solution and then reusing it. The main problem with starting this practice is the fact that no one wants to spend money in a general solution that does not bring any direct business benefits.

The experience shows that consultants in the small development team are two times more effective than in big releases. At least five reasons were found to explain the effectiveness: the small development team consists of top employees; only system & integration testing is needed (no user acceptance testing or regression testing), wages are paid applying the “time-material” principle, the schedules are extremely tight, and the requirements are strictly defined at the time the implementation starts.

If company A wants to shorten the time-to-market and improve its requirement engineering process, an agile process methodology called Scrum could be an interesting alternative. Company C, which is an affiliated company to company A, has achieved great results using Scrum.

5.3.2 Requirements engineering

The fundamental problem in the current A1 process is the varying quality of the requirements engineering process. Business and IT have somewhat different needs what comes to freezing the requirements. Also the level of required details in each phase divided opinions.

The quality of the requirements engineering process varies a lot. For example, sometimes functional requirements are defined precisely enough already before BR creation, but quite often this is not the case. There are even examples where a business has recognized at the last minute before going into production that an essential requirement is missing from the implementation because no one remembered to request it. In order to avoid these kinds of disasters, the product managers or other persons who submit the business requests should know the technical systems much better than they do now. However, if something is unclear, system experts should be available, on short notice, for a 30-minute initial discussion regarding the problem, in order for business to submit the right kind of request. If the design is made by an inexperienced vendor, business should specify the functional requirements in more detail than normally to avoid misunderstandings. Frequent communication between all the stakeholders throughout the process is essential. These issues are discussed in Section 5.3.4.

The requirements should be always unambiguous. Today, most last-minute changes occur due to the fact that business requirements are not fully documented and many things are agreed on only through e-mails. This is particularly the case with the external systems. A good rule of thumb is that the implementation should be always comparable to the requirements document.

Using different types of check lists might improve the quality of requirements engineering and, at least, prevent forgetting important things when defining the requirements. Check lists could include yes/no type of questions to define all the relevant information, such as “do we need to pay compensation”. The BR will not be forwarded until all the details have been filled in.

Business and IT have somewhat different needs when it comes to freezing the requirements. IT hopes to get the final approval for the detailed functional requirements before DP1 at the latest, whereas business wants to have the option to change requirements until DP4. As one of the business representatives stated: “It is impossible to freeze all the business requirements before DP2 for example because the final name of the product can be decided by DP3 and the final price, which depends on the market situation, can be decided by DP4.” Currently, IT requires too detailed requirements when creating a BR that business sometimes asks the wrong things by accident.

According to IT representatives, the current R&D process works only if the detailed functional requirements can be specified and frozen before DP1. Technical requirements must be frozen after the design phase at DP3. Business and IT agree that business should be in charge of specifying the business needs and functional requirements. It is IT’s task to discover how the business need can be fulfilled and find the optimal solution for the current systems by conducting the design. However, close cooperation is needed all the time to verify that both parties have understood everything the same way.

Often, business does not know the exact business requirements early enough because the business project is running behind the IT project although the projects should progress the other way around. Such BRs should not be included in a release or an SD package.

5.3.3 Needed roles

The A1 project should establish a specialized requirements engineering team which would conduct all the feasibility studies and pre-studies. Each requirement manager would be in charge of end-to-end requirements engineering, writing the required descriptions, and preparing the RFP material. The team should consult any relevant stakeholder whose expertise is needed in refining the requirements. The team would also suggest whether the BR should be implemented in small development or release. Neither business nor IT representatives were sure whether the requirements engineering team should consist purely of company A employees or should there be also consultants participating in the process. If the consultants were involved in the requirements engineering process, they would be able to create better design solutions later on when they truly understand the business case. However, the biggest disadvantage would be that company A would need to provide confidential information to vendors that they can utilize when negotiating the new contract.

Another important new role would be the A1 system architect. The architect would be responsible for commenting on A1 system development related matters from a technical point of view, helping business in creating BRs, evaluating pre-studies from the A1 perspective, and estimating the approximate amount of man days required for implementing a specific BR.

The key stakeholders who should be involved in the requirements engineering process include the A1 requirements engineering team, business representatives, and self-service system representatives.

5.3.4 Cooperation between stakeholders

The backbone of succeeding in the IT projects is close cooperation between the various stakeholders. Practically all the failures or negative last-minute surprises in the previous A1 projects have taken place due to misunderstandings or a lack of communication between business and IT. Business complains that, e.g., they do not necessarily get answers to feasibility study requests or they are not informed whether the BR has been accepted to the next A1 release or not. Disappointments in the current process have lead to a situation where the official process is often bypassed to get things done faster. For example, the BR is sometimes created before submitting the feasibility study request. Obviously, this reduces the quality and slows down the process as a whole.

In order to improve the quality of the BRs and make them more realistic, an IT representative should be present in the meetings of the business project's first phase. This would also help IT to understand the usage scenarios. If IT representatives cannot participate in business meetings, alternatively business should have, on short notice, an opportunity to have an initial 30-minute discussion together with an IT expert regarding

the topic. During the design phase, IT should ask guiding questions in common meetings with business to verify that business has taken everything into account in their solution. As proposed earlier, check lists might help in this problem, too.

The schedules of the related systems should be synchronized better, so that the new service would be available in every system concurrently. A general problem with achieving this goal is that the self-service channels are not able to start defining their functional requirements until they get A1 requirement specifications.

As a general guideline, everybody should anticipate as much as possible since the actual pre-study and design phases progress so fast.

5.3.5 Decision-making

Some people claim that the irregular meeting schedules of the multiple decision boards are the biggest bottlenecks in the decision-making process, but many people do not agree with that. They say that employees at company A prepare badly for the decision boards. For example, the related material may be sent as late as on the previous evening prior to meeting and it is often poorly prepared. In theory, decisions concerning the well-prepared proposals could be done very quickly, even by using e-mails. If thinking about getting an approval for DP2, one way for ensuring the quality of the material would be to send preliminary material 1 to 2 weeks before the meeting and ask whether the decision-makers need further information. The final material could then be sent a few days before the actual meeting with the appropriate corrections.

At the moment, most of the new business ideas get through to IT processes without filtering. This overloads the IT department and, thus, business should cut down the amount of initial ideas before creating feasibility study requests. It is crucial to prioritize emerging ideas from the very beginning, so that the most important matters get immediate attention. In fact, prioritization should be a continuous process underlying the IT and business projects. The fundamental problem in the prioritization is how to compare ideas that cannot be measured in the same way. For example, certain ideas have a clear price tag, whereas certain changes only improve the process itself or provide a general solution that can be used in the future. The line-up of the forums should be more carefully planned than nowadays in order to get the best out of the meetings. The top executives do not necessarily have as good input to give compared to the employees who are truly interested in the decisions.

5.3.6 Schedule

A1 release schedules should be planned to support business needs. Three large marketing campaigns set the boundaries for the deployment times over the year. Business requires that the maximum time to get new products to the market would be 2 to 7 months from the time when there are clear business requirements. The variation in the R&D time depends on the complexity of the solution. IT should be able to implement minor changes in existing products or any kind of package solutions within 2 to 3 months due to the strict competition. Implementation of the simplest changes, such as changing the name and price of the product, should take only a few days. Both IT and

business agreed that company A should perhaps allocate more time to the requirements specification phase, because this would reduce the time needed for actual implementation activities after DP2. Another way to shorten the time-to-market would be to sign longer-term contracts with the subcontractors instead of spending a lot of time in RFP rounds before each release. This way, there would be almost no work between DP1 and DP2.

5.3.7 Other

The problem with vendors preparing the pre-studies and giving the same pre-studies to them as RFP material is that the vendors can ask as much money as they want, because no one else precisely evaluates their suggestions.

Often, stakeholders waste their time on sitting in design meetings only for the reason that there may be some crucial thing to be discussed at some point of the meeting. It would be important to distinguish between the different phases of the R&D process clearer and stay focused on the real topic of the meeting, so that only the key persons relating to the topic would need to participate. This would lead to significant savings in R&D costs and boost development.

5.4 Summary

The ABC IT process model is a waterfall model, but it does not specify how the software should be developed. Instead, every project can choose the methodologies that best suit their purposes. However, the project must complete certain activities before getting approval to proceed to the next phase from the project steering group. These milestones are called decision points.

The A1 R&D process is divided into releases and small development packages. The time-to-market of a release is about 8 months. According to the interviews, the A1 process is too rigid and there are many things that could be done better. These included the software process in general, requirements engineering, new allocation of roles, cooperation between stakeholders, decision-making, and schedules.

6 Requirements Engineering in the B1 project and an overall company C model

This chapter discusses the core findings regarding the interviews at companies B and C. The first part of the chapter describes some of the differences and similarities between the A1 and B1 R&D processes. The second part of the chapter introduces an overall company C process model concentrating on the requirements engineering practices.

6.1 B1 requirements engineering process

6.1.1 Background

The B1 system at company B corresponds to the A1 system in terms of functionality, architecture, and complexity. The system is based on the same package software than A1 and it is developed using the ABC IT process model. However, the roles and certain detailed practices differ somewhat from A1 and the core findings are represented in this chapter.

6.1.2 Software process

In theory, about 50 percent of the B1 work could be done outside the releases in individual product projects. Thus, new products that do not have an impact on integrations outside the B1 system will be delivered as individual deployments.

6.1.3 Requirements engineering

The B1 project has implemented a very effective process for evaluating emerging business requirements (BR). Process leaders receive BRs from business forum every Monday morning. They are expected to give short answers to each BR preferably by the end of the same day.

The answers must be provided to the following questions

1) Does the BR require changes into any other systems than B1? Which systems will be affected?
2) When could the change be implemented and in which release? There is a dedicated person in charge of planning the release schedules and scopes.
3) What are the estimated workload and costs?

Experience shows that the business requirements should be more precise than currently before they are given to vendors. The requirements gathering should be lead by B1 process managers and conducted together with business. Their brand new internal investigation suggests that company B employees should be in charge of requirements engineering, including high-level design. This way, company B ensures that the requirements really meet the business needs and they maintain their touch with software development, so that the consultants would not be able to mislead them later on. Consultants, on the other hand, should be responsible for detailed design, build, and test phases. During these phases, the B1 management team only follows up the development.

6.1.4 Needed roles

Most of the roles needed in B1 development are practically the same as in A1 development – only the titles differ to some extent. The only major difference is the fact that the B1 project has three system architects whereas A1 does not have anybody with this role. B1 system architects are primarily used in the design phase and they participate in the design meetings whenever needed. Each system architect is responsible for specific BRs. They look at the problem from the B1 point of view and evaluate how well the suggested solution meets the technical guidelines and suit current integrations. Their main task is to come up with new kinds of solutions if they find that the original suggestion would not work well.

6.1.5 Decision-making

Company B's decision-making forums should look more at the future when deciding what is to be implemented in the B1 releases. New platforms or reusable functions might help in creating new things faster. For example, a general bundling solution would be needed to make it easier to bundle products in the future, but it is very challenging to convince business to support this kind of BRs. Despite of this, B1 is clearly ahead of A1 in anticipating the future business benefits of a solution that does not bring any direct revenues.

6.1.6 Schedule

During the year 2007 four B1 releases were deployed. Internal investigation argues that this is too many from the sourcing point of view and, therefore, the number of releases will be reduced to three on year 2008. Compared to the A1 project, the RFP round in B1 takes a couple of weeks less although the B1 project changes their vendor combination very often.

6.2 Company C requirements engineering process

6.2.1 Background

Company C is a relatively new player on the market and it is much smaller than companies A and B. Company C has a simple IT architecture and CRM/ERP systems have been coded using Java. The ABC IT process model is not followed closely in the development, but the DP points are the same. All the ongoing software projects use Scrum as the development methodology. It is applied as literally as possible. Also the company-wide decision-making processes support Scrum. Most of the findings have been gathered during a visit to company C through observing actual work in their biggest CRM/ERP project and interviewing several business and IT representatives. The current time-to-market of the most critical business requirements can be as short as 6 to 8 weeks from the time when the functional requirements are defined in detail.

6.2.2 Software process

The software process tries to follow Scrum as closely as possible. Due to the lack of resources, company C is only able to deploy two or three sprints in the quarter. The deployment always takes half a day on Sunday. The scope is never changed during the

sprint, but the modifications can be done in later sprints. A new product can be brought to the market in six weeks, if all the resources in all of the systems focus on it.

6.2.3 Requirements engineering

The Company C requirements engineering process is described in Figure 9. The process consists of nine phases until the initial business idea can enter the development process.

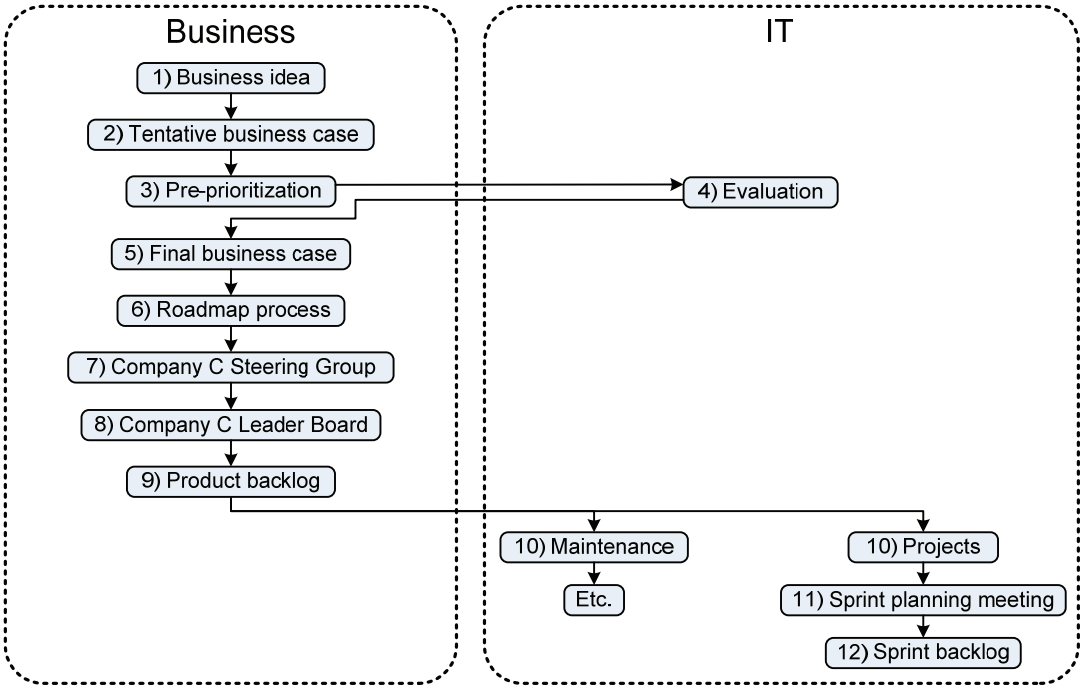

Figure 9. Company C requirements engineering process.

1. New business idea

Anybody, quite often the product manager, from any of the business departments (Marketing & Products, Sales, Customer Care, Finance, etc.) can come up with a new business idea.

2. Tentative business case

The person writes an overall description of what he or she wants. The business case should include all the information that is needed for processing the idea further. Basically, the person specifies the idea in as much detail as possible from a functional point of view. The level 3 department manager (there are 5 levels at company C) either confirms that the idea should be processed further or disregards it – this is an ongoing activity.

3. Pre-prioritization

Because there are more business requests than company C has the capacity to carry out, a quick pre-prioritization activity is needed. A team of three persons has the authority to pre-prioritize emerging BRs and filter out the ones that do not seem to be potential at all. The team consists of the head of the company C steering group, the head of the IT department, and the head of the technical service department. These persons meet every

two weeks. The decisions are based on experience and knowledge, because the persons have combined a technical and a business perspective. For each BR, they give a priority from 1 to 5.

4. Evaluation
Each BR that is qualified in pre-prioritization is evaluated by the IT department. If many BRs arrive at the same time, they begin from the BRs with the highest priority. The system experts define the technical requirements and estimate how much work and which resources are needed. The process takes a couple of weeks.

5. Final business case
Based on the IT evaluation, the BR owner finalizes the business case, providing answers to all the necessary questions. If the case still looks positive, the BR is added on the roadmap.

6. Roadmap process
The head of the company C steering group is in charge of the general roadmap process. This person represents the marketing department, but he or she should also have some technical background. The roadmap-related decisions are made in quarter meetings and monthly meetings.

As the name may suggest, quarter meetings are held once in a quarter. A large group of people from different business and IT departments participates in the meeting. Most of the new items on the roadmap are introduced and prioritized in this meeting. New ideas need to be put on the roadmap before the meeting with a preliminary priority. The final priority order is usually set by the head of the steering group based on the input he or she has got earlier. The person always considers the related issues from an objective point of view. Sometimes the head of the steering group consults department leaders and they can choose one of their requests to be included in a certain position on the roadmap. The roadmap is always kept up-to-date and the roadmap items are reprioritized if necessary.

The deadline for adding new items on the roadmap before the next quarter is set by the head of the steering group. It is usually a few days before the quarter meeting. If the request arrives late, it is handled as a “quarter after the next quarter” issue.

Monthly meetings take place between the quarter meetings. The participating group of people is rather small and it includes the head of the steering group, two system analysts, a system architect, and three persons from IT development. The group has the authority to change the priorities of the roadmap items, but only some critical new business ideas may be introduced in these meetings.

7. Company C Steering Group
The final roadmap needs to be approved by the Company C Steering Group and Company C Leader Board. The steering group meets every two weeks and it includes representatives from the major business and IT departments.

8. Company C Leader Board
The Company C leader board meets every week and the members include the CEO of company C and his or her directors. The roadmap is usually on the agenda in the last meeting before the next quarter. The head of the steering group introduces the following to the directors: the previous quarter (what was promised, what was achieved, what are the reasons behind it) and a roadmap for the next quarter. If the leader board feels positive about the roadmap, they will give their final approval for it.

9. Product backlog
Three company C's system analysts (a synonym for a product owner) create and maintain a product backlog. The backlog items are mostly derived from the roadmap, but it also includes some small changes and production defects. It is important to note that small changes are not included in the roadmap but can be taken up by the system analysts and added directly to the backlog. Small changes encompass minor requirements and the bugs that fit in the scope of sprints.

The product backlog should always be up-to-date two sprints ahead, but there may be issues even for the next quarter and the quarter after that. At the moment, the backlog items are documented in Excel, because it can be kept up-to-date easier than Wiki, which was used earlier. Every requirement is processed by system analysts who normally refine them significantly. System analysts should identify the whole value chain and also evaluate the impacts on the related systems and interfaces. They use a pre-defined template for writing the detailed functional requirements. However, they do not tell how the BR should actually be implemented.

10. Maintenance/Projects
Depending on the nature of the BR, the backlog items with the highest priority are dealt out to maintenance and project teams.

11. Sprint planning meeting
As described in Section 2.3.3, the first day of the development sprint is used for sprint planning activities. Product owners, system analysts, and Scrum teams look at the product backlog and pick up as many BRs as they will be able to implement during the sprint. Detailed technical requirements are defined during the sprint.

12. Sprint backlog
The chosen product backlog items are written to the sprint backlog.

6.2.4 Needed roles

Surprisingly, company C has outsourced many important expert roles, such as system architects and system analysts, to subcontractors. In contrary to many other organization ABC's software projects, the system architect is seen as a very essential role at company C and 100% of his or her time is often allocated to a single IT project. Usually, the vendors ask clarifying questions first from the system architect since he or she also has good contacts and knowledge of the related systems . Some of company C's employees feel that there is a huge risk involved in using consultants in key roles.

6.2.5 Cooperation between stakeholders

Company C aims to cooperate closely with their vendors throughout the R&D process. This brings in many benefits involving, e.g., less documentation. If something is unclear in the functional requirements during the sprint, the Scrum team can ask for clarification from the appropriate people. If there would be a long delay in the answer, the team may go ahead and choose a solution themselves.

6.2.6 Other

Company C has a totally different view on subcontracting compared to company A. Subcontractors are usually hired using long, 1 to 3-year contracts. They work most of the time as they were normal employees. Many consultants work for the same vendor, but also consultants from different vendors work well together.

6.3 Summary

The B1 project uses many of the same practices as the A1 project, because they are both based on the ABC IT process model. On the other hand, company C uses a totally different process model called Scrum in all its software projects. Also their requirements engineering process supports the agile way of working.

7 Recommendation

This chapter suggests a new software process model for the A1 project. The recommendation focuses on describing the enhanced requirements engineering process, but it also suggests a general process model and a few useful practices to be used in the future. The recommendation ends with a detailed step-by-step suggestion how the A1 development manager should in practice change the current process.

7.1 New A1 software process model

In order to significantly shorten the time-to-market, reduce the development costs, and make the software process more flexible, the A1 system should be developed using an agile software process model. This requires major operational and cultural changes within the whole organization.

The new A1 software process model is based on Scrum methodology. The best results will be achieved by applying the Scrum practices in combination with the best current practices.

7.1.1 Roadmap

The new yearly A1 roadmap consists of four bigger releases and eight small development sprints. Four of the small development sprints are deployed at the same time with the releases, so there are actually only eight deployments during the year. The development schedules and deployment times have been planned in a way that supports company A's business needs, but also technical limitations have been taken into account. Requirements engineering (~ pre-game phase) is a continuous process behind the actual development activities. An example of the potential roadmap is introduced in Figure 10 (the deployment times are marked with stars). Please note that the schedule is schematic, it is not synchronized with the real suggestion because the deployment times are secret.

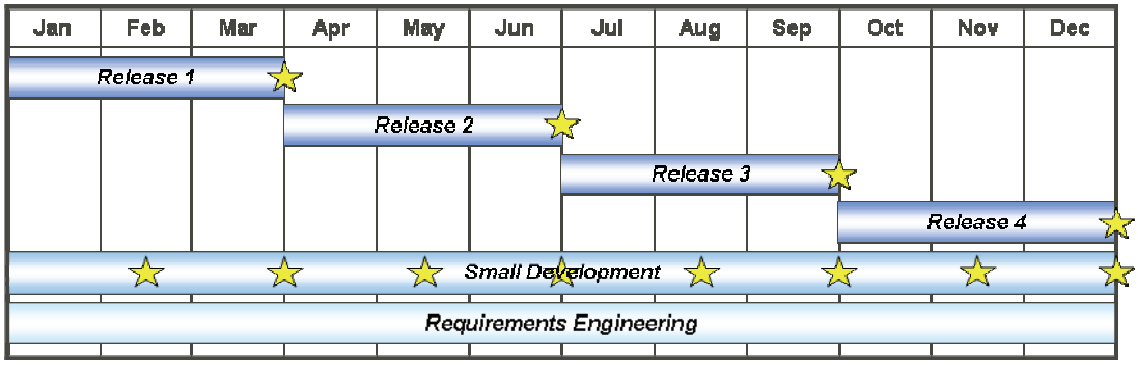

Figure 10. New yearly A1 roadmap.

7.1.2 A1 process

The A1 process consists of pre-game, development, and post-game phases. The pre-game phase actually runs continuously under the release process, so only the development and post-game phases belong to the actual release or small development

process. The main activities within these three phases are described in Figure 11. The figure also shows the relation to the ABC IT process model's decision points (DPs).

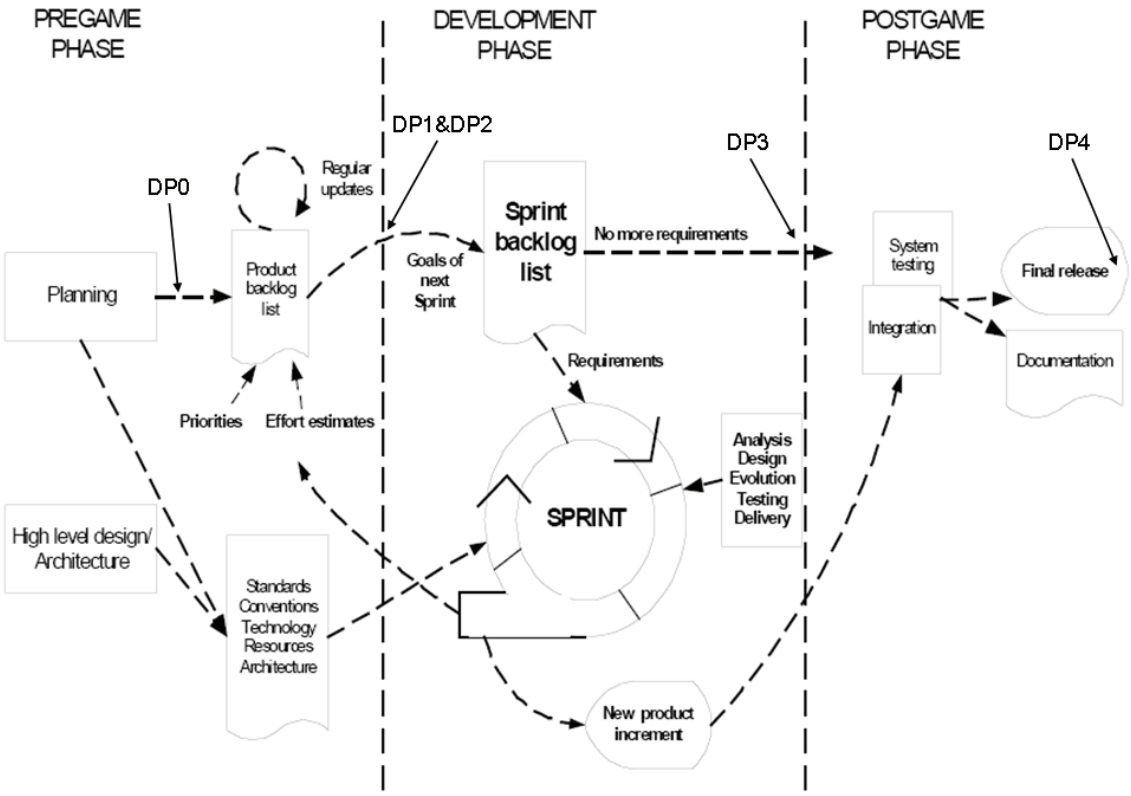

Figure 11. New A1 process model.

A1 release process

The A1 release process includes two development sprints and a test sprint. Each sprint takes exactly 30 days, so the total length of the A1 release process is 90 days. During a single development sprint, the Scrum team conducts the design, build, system testing, and rough integration testing. The Scrum team performs the complete integration testing, regression testing, performance testing, user acceptance testing, and all the deployment activities during the test sprint. The A1 release process flow is described in Figure 12.

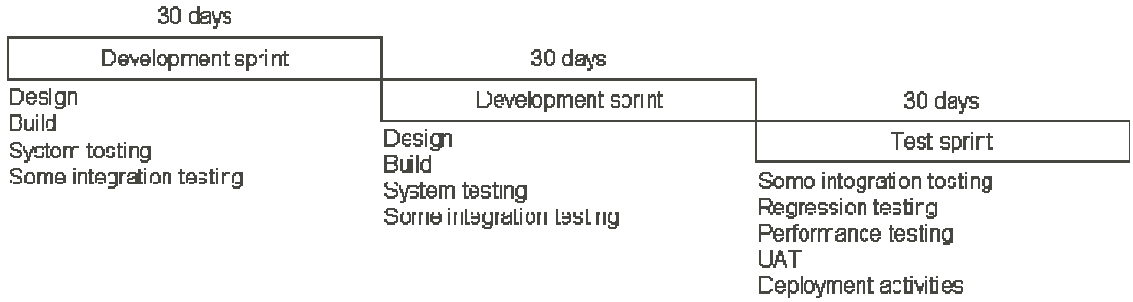

Figure 12. New A1 release process.

A1 small development process

The A1 small development process consists of a 30-day development sprint and a 14-day test sprint. In other words, the whole process takes only 44 days. The activities during the development sprint encompass design, build, and system testing. Integration testing, performance testing, user acceptance testing, and deployment activities are performed during the test phase. It is important to note that the small development package can include only BRs that do not cause a need for regression testing. The A1 small development process is introduced in Figure 13.

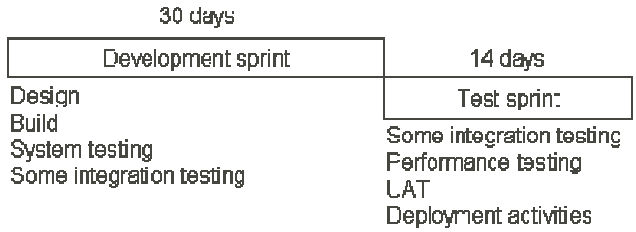

Figure 13. New A1 small development process.

7.2 A1 pre-game phase

The pre-game phase follows the Scrum methodology (see Section 2.3), but it has been customized to some extent for company A's purposes. It is essential to pay attention to high-quality requirements engineering from the very beginning of the process. The requirements must always meet the criteria for a good requirement (see Section 3.3) and there are a few great techniques that can be used throughout the process (see Section 3.5). Some of the other useful practices are discussed in Section 2.3.3. The A1 requirements engineering process during the pre-game phase is introduced in Figure 14.

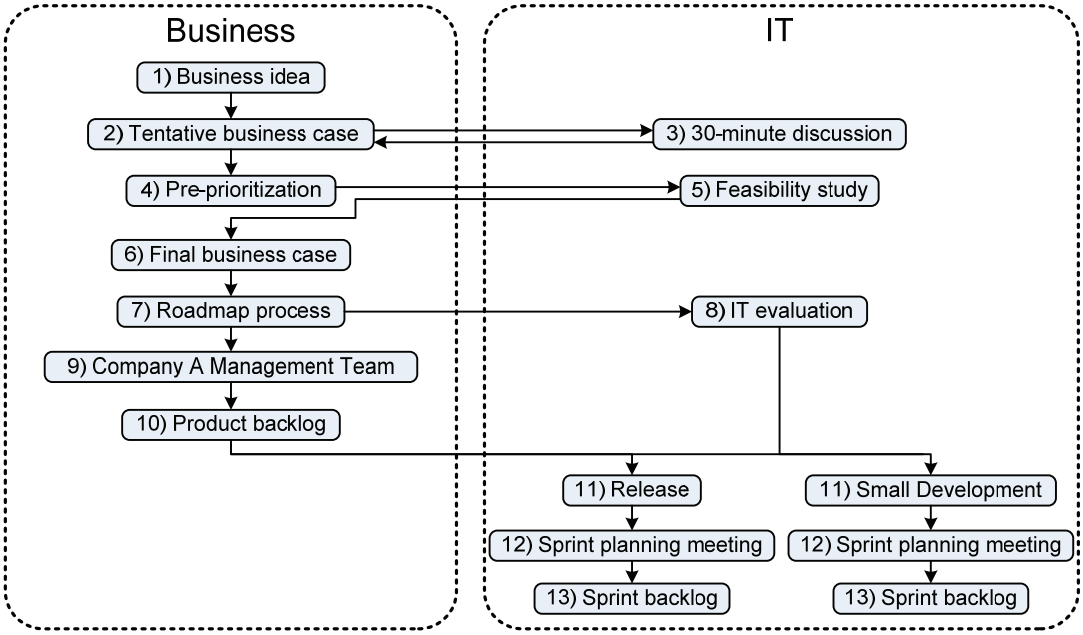

Figure 14. New A1 requirements engineering process.

1. New business idea
Anybody, quite often a product manager or a customer service representative, from any of the business departments (products, B2B sales, customer service, business IT management, etc.) can come up with a new business idea. New ideas can be also explored using several techniques (see Section 3.4.1).

2. Tentative business case
The person writes an overall description of what he or she wants. The business case should include all the information that is needed for processing the idea further. Basically, the person specifies the idea in as much detail as possible from a functional point of view. Each department manager or an appointed person from the department confirms that the idea should be further processed or disregards it – this is an ongoing activity.

3. 30-minute discussion
If the business representative is not able to define all the required details alone, he or she has an opportunity to discuss the idea with a technical system analyst. Usually this helps business in creating the right kind of a BR. However, the result of the discussion may be also that the business representative must come up with a totally different approach to the idea, because there are technical limitations that cannot be overcome. A good way to ensure that system analysts have time for these 30-minute discussions would be to allocate for example the whole Monday afternoon to this activity.

4. Pre-prioritization
Because there are more business requests than company A has the capacity to carry out, a quick pre-prioritization activity is needed. A team of five persons has the authority to pre-prioritize emerging BRs and filter out the ones that do not seem to have any potential. The team consists of managers from Business IT Management, Product Management, Production Line Management, IT, and Self Service Systems. These persons meet every two weeks. The decisions are based on experience and knowledge, because the persons have combined a technical and a business perspective. For each BR, the pre-prioritization group gives a priority from 0 to 5. If the BR gets the priority 0, it will be automatically filtered out from the process.

5. Feasibility study
Each BR that is qualified in pre-prioritization is evaluated by the IT department. If many BRs arrive at the same time, they begin from the BRs with the highest priority. The system experts define the high-level technical requirements and estimate how much work and which resources (maintenance, small development, or release) are needed. The process takes from a few days to a couple of weeks depending on the priority and the complexity of the BR. The feasibility study is conducted also for all the A1 related systems that the BR affects.

6. Final business case
Based on the feasibility study, the BR owner finalizes the business case, providing answers to all the necessary questions. If the case still looks positive, the business department adds the BR on the general roadmap.

7. Roadmap process
An appointed person from Business IT Management (BIT) is in charge of the general roadmap process. This person works in the interface between business and IT, so he or she has the most objective view on the whole requirements engineering process. The roadmap-related decisions are made in quarter meetings and monthly meetings.

The quarter meetings are held once in a quarter. A large group of people from different business and IT departments participates in the meeting. New ideas need to be put in the end of the roadmap before the meeting with a preliminary priority. The preliminary priorities are set by each of the business departments in cooperation with the BIT representative. In the beginning of the meeting, the new items are introduced and their importance is evaluated. All the BRs are set in the final priority order during the quarter meeting. The priorities of the old roadmap items are re-evaluated and new BRs are added where they suit best. No BR can have exactly the same priority as another BR. Obviously, different department managers argue about the priorities and, therefore, the BIT representative always has the final word in the matter. At the end of the meeting, this person introduces an updated roadmap for the next quarter and the meeting needs to give its approval for the roadmap. Sometimes the BIT representative consults department managers and they can choose one of their requests to be included in a certain position on the roadmap. The roadmap is always kept up-to-date and the roadmap items are reprioritized if necessary.

The deadline for adding new items on the roadmap before the next quarter is set by the representative of BIT. It is usually a few days before the quarter meeting. If the request arrives late, it is handled as a "quarter after the next quarter" issue.

Monthly meetings take place between the quarter meetings. The participating group of people is rather small and it includes the same persons as the pre-prioritization group. The group has the authority to change the priorities of the roadmap items, but only some critical new business ideas can be introduced in these meetings.

8. IT evaluation
System analysts in the requirements engineering team start refining the BRs as soon as they are put on the roadmap. This is an ongoing activity until the BR gets into the development pipeline (step 11). System analysts should identify the whole value chain and also evaluate the impacts on the related systems and interfaces. They use a standard template for writing the detailed functional and high-level technical requirements. Due to the dynamic market situation, the functional requirements may change slightly in the course of time. Therefore, also technical requirements must be improved all the time. However, in a case when the functional requirements change significantly, a new BR must be created.

9. Company A Management Team
The final roadmap needs to be approved by the Company A Management Team. The management team meets every week and the members include the CEO of company A and all the department managers. The roadmap is on the agenda in the last meeting

before the next quarter. The head of BIT introduces the following to the directors: the previous quarter (what was promised, what was achieved, what are the reasons behind it) and a roadmap for the next quarter. If the management team feels positive about the roadmap, they will give their final approval for it. This approval is, at the same time, a DP0 for the release and small development packages that are planned to be started within this particular quarter.

10. Product backlog
System analysts (a synonym for a product owner) create and maintain a product backlog. The backlog items are derived straight from the roadmap. It is important to note that system analysts have the authority to take up minor changes and add them directly to the backlog. These changes encompass, e.g., simple price or name changes of the current products.

The product backlog should always be up-to-date two sprints ahead, but there may be issues even for the next quarter and the quarter after that. The backlog should be maintained in a format that is easy to keep up-to-date – Excel is a good enough option, Wiki does not meet these criteria.

The IT steering group must approve a DP1 for the top 20 backlog items on the product backlog list before they can be taken to the sprint backlog. In the new process model, a DP2 is achieved in tandem with the DP1 since no RFP round is needed.

11. Release / Small development
Depending on the nature of the BR, the backlog items on top of the list are divided into A1 releases and small development sprints.

12. Sprint planning meeting
As described in Section 2.3.3, the first day of the development sprint is spent in sprint planning activities. Product owners, system analysts, and Scrum teams look at the product backlog and pick up as many BRs as they will be able to implement during the sprint. At this point, the functional requirements are finally frozen. Detailed technical requirements are defined during the sprint.

13. Sprint backlog
The chosen product backlog items are written in the sprint backlog.

7.3 A1 development and post-game phases

During the development and post-game phases, company A should use the common Scrum practices as such. They are described in Section 2.3. A DP3 must be approved after each development sprint. A DP4 can be achieved after the test sprint. DP5, DP6, and DPE are given following the old ABC IT process model.

7.4 Needed roles

In order to be able to implement the new A1 software process model, old roles must be replaced with some new roles.

A requirements engineering team is needed for conducting the activities during the pre-game phase. The team should include four system analysts (~ product owners) and a system architect. Two of the system analysts are consultants, so that each subcontractor has their own representative in the team. The system analysts are in charge of end-to-end requirements engineering. The most essential activity is to refine the functional requirements together with business in the form of user stories and translate the user stories into technical requirements. When the subcontractors get involved with the BRs already at the very beginning of the process, they are able to create better holistic solutions during the development phase. This also reduces the risk of misunderstanding the business needs. However, having also company A representatives involved in the requirements engineering process, it is ensured that the subcontractors are not able to mislead when, e.g., the parties derive the story points. A system architect is responsible for commenting on A1 system development related matters from a technical point of view, evaluating user stories from an A1 perspective, and estimating the approximate number of story points required for implementing a specific user story. The story point estimation is always done in cooperation with the Scrum team.

A1 release team consists of a project manager and three to four Scrum teams depending on the situation and how much can be implemented simultaneously in parallel. Each Scrum team has a Scrum master and five developers. The A1 small development team is a single Scrum team consisting of a Scrum master and five developers. The A1 maintenance team is similar to the small development team, but the maintenance organization is not in the scope of this thesis. All the members of the Scrum team are consultants, but A1 product owners (~ system analysts) and the system architect observe their work, so that they know what is going on. A project manager is responsible for normal day-to-day project management activities as in a conventional software projects.

In addition to the requirements engineering and Scrum teams, A1 needs four general roles for managing the whole process. The A1 development manager (~ IT manager) participates in the prioritization boards and takes care of resourcing. The A1 test manager is in charge of organizing the testing preparations and verifying that the tests are conducted properly. The A1 environment manager takes care of the various system environments, making sure that they support the development activities in the best possible way. The A1 deployment manager is responsible for planning the deployment activities from company A's perspective to ensure that they do not disturb the running business in any way. It may be possible to combine some of these roles, such as the environment manager and the deployment manager, if the workload seems to be reasonable.

7.5 Useful practices

As stated earlier, the new A1 software process model is mainly based on Scrum methodology and the related practices. In this chapter, however, I will point out certain other useful practices that would be worth implementing in the whole ABC organization. They are not out of tune with the Scrum methodology, but they rather support it.

7.5.1 Longer-term contracts

Despite of having certain advantages to them, frequent RFP rounds bring much less benefit than they cause harm for the whole development process. Hiring subcontractors using long, 1 to 3-year contracts would be a much better solution. This way, the consultants can work most of the time as they were normal employees – this improves the cooperation between different stakeholders. Only by eliminating the work between DP1 and DP2, the time-to-market would be 5 to 6 weeks shorter than earlier. The agreement should support high quality and an efficient way of working. Extra bonuses will be paid if the consultant surpasses the set goals. Having two or more subcontractors working simultaneously in A1 system development increases competition between the vendors and reduces the price of their offers.

7.5.2 Close cooperation

Currently the company A employees and subcontractors in the A1 project do not work as closely together as they should in order for company A to succeed in implementing the new A1 software process model. All the stakeholders must trust each other deeply in order to achieve the best results. What is beneficial to the company also should be beneficial to the vendors (see Section 6.2.6). This shortens the time-to-market, improves the quality of the process, reduces the need for extensive documentation, and reduces the costs.

7.5.3 Pilot selling

The fastest way to get a new product to the market is to first build a simplified solution of it for pilot selling. A fully automated version can be implemented on top of the simplified solution when the success of the product has been proved on the market. Therefore, IT does not need to make extra effort relating to automation if the product fails on the market. The risk in implementing the solution in two phases is that IT may actually need to do almost double the work if they cannot use the simplified solution when enhancing the functionality.

7.5.4 Generic solutions

At the moment, decision boards consider too often only the short-term benefits of the BRs. Many times, however, it would make sense to create a holistic long-term solution instead of implementing the same kind of functionality with slight variations again and again. For example, the current trend of bundling products in different packages takes a lot of effort from IT. Building and reusing a more generic bundling solution would lead to stunning future savings of money and time. The main challenge with starting this practice is the fact that no one wants to spend money in a general solution that does not bring any direct business benefits.

7.5.5 Daily Scrum of Scrums

Weekly project management meetings do not belong to Scrum methodology even in the bigger releases. Instead, the project can have a Daily Scrum of Scrums, which means that Scrum Masters from each team meet after the Daily Scrum to their own Daily Scrum. In other words, there should be a "leading" Scrum Master who takes responsibility for the higher level Scrum meeting. Most naturally, this person would be

the A1 project manager since he or she leads the team leaders. If daily meetings are not considered to be appropriate for the purposes of the Scrum masters, the Scrum of Scrums can be held also less frequently.

7.5.6 Check lists

A solution to the poor and varying quality of requirements engineering is to use different kinds of templates and check lists while defining the functional requirements. Check lists could include various yes/no type of questions to verify that the parties have not forgotten anything important. It would not be possible to submit the BR before all the details have been filled in.

7.6 Why this would work

Many findings in the literature and experiences in various companies worldwide support the hypothesis that Scrum is a very suitable model for company A's purposes. Also organization ABC's own experiences indicate that the suggested model and practices will improve the current ABC IT process model significantly.

The existing small development concept has brought approximately three times better results than bigger releases. This process is not yet agile, but it has many agile elements, such as tight time constraints (a short development cycle) and the way the small team works.

The suggested model suits the distributed software development process well since each Scrum team can work individually even in an offshore location if needed.

Because the functional requirements must be frozen before the development sprint can begin, no business involvement is needed or even allowed during the development phase. And, actually, the process is very flexible, because business is allowed to make minor changes in requirements for as long as they want before the freezing. This means that the last changes may be submitted as late as one and a half months before the deployment. However, business should try to do as few changes as possible after sending the BR for further processing.

In addition to the previous facts, the recommendation involves many other advantages that are discussed earlier in this thesis.

7.7 The next steps

Obviously, company A cannot change its whole process over a night since the adaptation takes a lot of time. To help all the stakeholders in successfully adapting the new practices, the process must be implemented very systematically step by step. It is important to note that quite often immediately after big changes in the software process, productivity may temporarily decrease. Therefore, a lot of patience is required from everybody.

Here is an overall 10-month action plan for the A1 development manager for implementing the suggested model:

Month 1

1. Familiarize yourself with the theory of Scrum and pay special attention to the requirements engineering practices that are represented in this master's thesis.
2. Discuss the suggested model with the key stakeholders and convince them of its excellence.
3. Get an approval from the higher managers to change the A1 software development process according to this model.
4. Create an exact action plan for the next months.
5. Organize an internal meeting to introduce the new model and explain what needs to be done next.
6. Organize an external meeting with the potential subcontractors to introduce the new model and explain what needs to be done next.
7. Start a project for solving the existing technical constraints that prevent taking a tighter deployment schedule into use (automation of regression testing, deployment activities, etc.).

Month 2

1. Organize more advanced training sessions for key stakeholders.
2. Form a requirements engineering team of five persons (two internal system analysts, two external system analysts, an internal system architect). If needed, recruit new people to this team. Set a clear goal and areas of responsibility for the requirements engineering team. Make sure that the requirements engineering team works according to the guidelines.
3. Set up a new pre-prioritization group together with a BIT representative.
4. Start pushing business to change their decision-making process according to the plan.
5. Be regularly in touch with all the key stakeholders: inform them of the progress; tell them what they should do; and follow up that they are in fact doing what they are supposed to.

Month 3

1. Focus on improving the quality of requirements engineering (user stories, how the requirements process works, etc.).
2. Make sure that everybody is still following the plan.
3. Prepare a new subcontracting model, which supports high performance. Long-term contracts are the future.
4. Prepare a new roadmap for releases and small development packages. Pay special attention to transition from the old A1 software process model to the new model. The new A1 process model can be launched in full scale as soon as the technical limitations have been overcome (the improvement project is ongoing). Get an approval for the new roadmap.
5. Continue interacting with all the key stakeholders.

Month 4

1. Make sure that the pre-game phase finally works as it has been described in this master's thesis and all the necessary supporting tools, such as the Master Development and Test Environment (MDE) tool, are in use. If not, then you must take the actions needed to improve the way of working.
2. Continue planning the launch of development and post-game phases. Keep tracking the ongoing improvement projects.
3. Get the representatives of all the related systems closer involved in the planning activities, so that a holistic company A wide process could be introduced in the future.
4. Continue interacting with all the key stakeholders.

Month 5

1. Continue the previous activities.

Month 6

1. Continue the previous activities.
2. Start using automated regression testing and test the enhanced deployment model in practice. (If they are not ready yet or they do not work as planned, the following activities will be delayed.)
3. Start practical preparations
    a. Conclude new types of contracts with the vendors
    b. Adapt the office facilities to support the Scrum methodology (a place to organize daily Scrums, whiteboards, etc.).
    c. Setup all the required environments.
    d. Appoint employees to the Small Development's Scrum team
    e. Etc.
4. Organize advanced training sessions.

Month 7

1. Start the first small development sprint using Scrum (pilot project). Make sure that all the suggested practices are followed literally. It is essential to learn the right way of working immediately.

Month 8

1. Evaluate the achieved results and improve the process together with the other stakeholders. Do not turn back to the old A1 process model even if the results are not promising.
2. Start the second small development sprint using Scrum.

Month 9

1. Evaluate the achieved results and improve the process together with the other stakeholders.

Month 10

1. Evaluate the achieved results and improve the process together with the other stakeholders.

2a. If the results are promising, start using the new A1 process model also in the release development.

   a. Allocate employees to the new release organization.
   b. Start the first full-scale Scrum process (pre-game, development, and post-game phases in place) with the first release and the third small development sprint.

2b. If the results of the third small development sprint are still much worse than when using the old A1 process model, consider returning back to the old process.

Please note that the suggested action plan is described in a very general level and it only encompasses some of the most essential A1 development manager's tasks during the ten months' timeframe. The A1 development manager is not able to change the whole A1 development process alone in such a dramatic way, because the process affects so many stakeholders and involves certain risks. Therefore, he or she must work in close cooperation with company A's other IT and business functions in order to win their confidence in the new process model and get them to participate in the numerous activities that are described in this document.

7.8 Summary

The new A1 process model is based on Scrum methodology. The process consists of pre-game, development, and post-game phases. Requirements engineering activities take place during the continuous pre-game phase. Along with a new process model, also certain new practices and roles must be adopted.

8 Conclusions

The goal of this master's thesis was to find a solution for shortening the time-to-market of company A's A1 system development process from 12 months to 4 months. The purpose was to focus on optimizing the requirements engineering process although the whole process needed to be re-evaluated. Improving the quality of the requirements definition was considered to be essential since the requirements affect the entire R&D process.

The A1 system is used for serving millions of company A's customers and it is a master for managing customer and order data. New orders and changes to existing orders are received from the channel applications.

Traditionally, large software projects have been implemented using a waterfall model or one of its subtle modifications. The problem with these conventional models is that they are very rigid and the time-to-market is several months. In other words, the waterfall model does not suit the current, fast changing markets. Agile software process models were developed in the 90's for overcoming the shortcomings of the conventional models. During the 21st century, many new agile models have been developed and some of them have become extremely popular. The main advantages of agile process models are the visible progress from the very beginning of the process through frequent deliveries, the shorter time-to-market of individual requests, lower risks, less obligatory documentation, and adaptability to changing business needs. However, agile processes suit relatively small software projects the best and changing the organizational culture from conventional to an agile way of working may be very challenging.

After an objective evaluation of different conventional and agile software process models compared to twelve important variables for A1 project, Scrum methodology ultimately achieved the highest marks in the comparison. The evaluation was conducted by me and it was based on the available literature.

The Scrum methodology is an agile process model that is divided into pre-game, development, and post-game phases. All the requirements engineering activities take place during the pre-game phase. The requirements can be gathered using different techniques and they are maintained in a product backlog list. The development phase consists of several 30-day development sprints. The first day of each sprint is spent in a sprint planning meeting where the top-prioritized requirements are fed from the product backlog list into a sprint backlog list. Although Scrum is an agile process model, it is scalable for big software projects since even up to ten Scrum teams may work simultaneously during a sprint. Scrum suggests also some new roles and methods for software development.

Requirements engineering plays an essential part in every software project. By creating high-quality requirements, companies can avoid unnecessary costs during the later phases of the process. Regardless of the type of the project, following the common rules in defining requirements and adapting the suggested practices is the key to success.

Company A's employees had some clear thoughts on the problems regarding the current A1 process model. As a conclusion, the existing process was considered to be too rigid and lacking sufficient support for business needs. In addition, they suggested certain new practices for the future. Also visits and interviews at companies B and C produced many great ideas for improving the process. The B1 software process model is very similar to the current A1 process model and they had noticed mostly the same kind of bottlenecks as company A's employees. The C1 process, on the other hand, is based on Scrum and the company's business processes support an agile way of working. They implement the Scrum methodology almost literally. However, the C1 process model cannot be copied directly to the A1 process because of certain fundamental differences between project types and business organization. Furthermore, the comprehensive existing documentation that was available from companies A, B, and C made it easier to form a picture of the current processes and find new ideas.

In order to meet the set goals, company A should adapt an agile working methodology. Implementing a slightly modified Scrum model with the best current practices would be the most functional solution for performing the requirements engineering and the A1 system development activities. This way, the future time-to-market from an initial idea to production could be as short as two months, supposing that the decision-making process flows very smoothly. However, this can be the case only with some relatively small, very critical business requests. Otherwise the time-to-market depends on the priority and complexity of the business request.

The basis for the new process is that a single development sprint takes only 30 days. In small development, this is followed by a 14-day test sprint including all deployment activities. A release consists of two 30-day development sprints and a 30-day test sprint. Therefore, the time-to-market of releases is a little bit longer than using small development sprints, but still much more flexible than using the current model. The scope of the release is never fixed because new items can be added in the backlog just before the next sprint begins.

The new requirements engineering process supports the Scrum methodology. New critical and well-prepared business requests can be processed further very quickly but poorly defined requests without a good business case can be dropped out of the process at the very beginning. Strict control throughout the process ensures that the quality of the requirements will improve. Despite of the strict control, there are only three decision-making boards to ensure high flexibility – less than three boards would make the decision-making less effective.

It is impossible to evaluate how the suggested solution will achieve the set goals, because the implementation of the new process model is only about to start. The process model seems to have a lot of potential and it may be possible that in the future, the shortest time-to-market will be much shorter than four months. Applying the suggested requirements engineering practices and controlling that they are truly followed guarantees that the quality of the requirements will improve significantly. This would have a huge impact on reducing the development costs of the A1 system.

Finally, it is important to notice that the represented solution is still far from complete. It gives only some general guidelines on how the A1 process should be modified. Topics for further studies include specifying a more detailed plan for implementing the recommendation, figuring out how the current technical limitations could be overcome (e.g., improving the configuration management and automating the regression testing), planning and setting up the metrics for measuring the new A1 process, customizing some of the requirements engineering tools that company A supports for system A1's purposes, and creating a holistic decision-making and prioritization solution for all the IT projects within company A. An ideal situation would be a case where all company A's IT projects used Scrum methodology throughout the whole process.

References

[ABC06] Organization ABC. 2006. ABC IT Process Model 2.0. Stockholm, Sweden.

[Abr02] Abrahamsson P., Salo O., Ronkainen J., and Warsta J. 2002. Agile Software Development Methods: Review and Analysis. Espoo, Finland: Otamedia. http://www.vtt.fi/inf/pdf/publications/2002/P478.pdf (December 27, 2007)

[Agi08] Agile Alliance. 2008. [Online]. Available: http://www.agilealliance.com/ (December 31, 2007)

[Bec01] Beck K, et. al. 2001. The Agile Manifesto. [Online]. Available: http://agilemanifesto.org/ (December 31, 2007)

[Coc06] Cochango. 2006. Scrum for Team System. [Online]. http://scrumforteamsystem.com/ProcessGuidance/Scrum/Scrum.html (December 28, 2007)

[Coh06] Cohn M. 2006. Planning Poker. [Online]. http://www.planningpoker.com/detail.html (March 3, 2008)

[Cou05] Courage C. and Baxter K. 2005. Understanding Your Users: A Practical Guide to User Requirements Methods, Tools, and Techniques. San Francisco: Morgan Kaufmann.

[Dav93] Davis A. 1993. Software Requirements: Objects, Functions, and States. Upper Saddle River, USA: Prentice Hall.

[IEE90] The Institute of Electrical and Electronics Engineers (IEEE). 1990. The IEEE Standard Glossary of Software Engineering Terminology. New York, USA.

[Kan94] Kan S., Dull S., Amundson D., Lindner R., and Hedger R. 1994. AS/400 software quality management. IBM Systems Journal, Vol. 33, No. 1, pp. 62–88.

[Kau07] Kauppinen M. 2007. Introduction to Requirements Engineering. Espoo, Finland. Helsinki University of Technology: Lecture and Lecture Notes.

[Kot98] Kotonya G. and Sommerville I. 1998. Requirements Engineering: Processes and Techniques. New York, USA: John Wiley & Sons.

[Law97] Lawrence B. 1997. Requirements happen. American Programmer, Vol. 10, No. 4, pp. 3-9.

[Les07] Leskelä T. 2007. Requirements at R&D. Espoo, Finland. Helsinki University of Technology: Lecture and Lecture Notes.

[McC04] McConnell S. 2004. Code Complete. 2nd Edition. Redmond, USA: Microsoft Press.

[Pre05] Pressman R. 2005. Software Engineering: A Practitioner's Approach. 6th Edition. New York, USA: McGraw-Hill.

[QUR01] QURE Research Project. 2001. Quality through Requirements (QURE). Helsinki, Finland.

[Rob06] Robertson J. and Robertson S. 2006. Mastering the Requirements Process. 2nd Edition. London, UK: Addison-Wesley.

[Sch96] Schwaber K. 1996. Scrum Development Process. Burlington, USA. [Online]. http://jeffsutherland.com/oopsla/schwapub.pdf (December 27, 2007)

[Sch02] Schach S. 2002. Classical and Object-Oriented Software Engineering. 5th Edition. Singapore: McGraw-Hill.

[Sch02b] Schwaber K. and Beedle M. 2002. Agile Software Development with Scrum. Upper Saddle River, USA: Prentice-Hall.

[Som04] Sommerville I. and Sawyer P. 2004. Requirements Engineering: A Good Practice Guide. New York, USA: John Wiley & Sons.

[Som06] Sommerville I. 2006. Software Engineering. 8th Edition. New York, USA: Addison-Wesley.

[Tak86] Takeuchi H. and Nonaka I. 1986. The New Product Development Game. Harvard Business Review, Vol. 64, No.1, pp. 137-146.

[Tha05] Thayer R. and Christensen M. 2005. Software Engineering. Volume 1: The Development Process. 3rd Edition. New Jersey, USA: John Wiley & Sons.

[Ver06] VersionOne. 2006. Benefits of Agile Development. [Online]. http://www.versionone.com/Resources/AgileBenefits.asp (March 18, 2008)

[Vli04] Van Vliet H. 2004. Software Engineering: Principles and Practice. 2nd Edition. New York, USA: John Wiley & Sons.

[Wei97] Weinberg J. 1997. Quality Software Management, Vol. 4: Anticipating Change. New York, USA: Dorset House.

[Wie03] Wiegers K. 2003. Software Requirements. 2nd Edition. Redmond, USA: Microsoft Press.

[Yip07] Yip J. 2007. It's Not Just Standing Up: Patterns of Daily Stand-up Meetings. [Online]. http://martinfowler.com/articles/itsNotJustStandingUp.html (December 28, 2007)

Appendix A: Comparison of software process models

		Suitable for continuous incremental development	Requirements can be frozen late	Short time-to-market	Suitable for developing packet software	Scalable for large projects (>50 developers)	Suitable for distributed development & outsourcing	Organization ABC has experience of the model	Utilizes suitable practices	Lots of literature available	Has been proved to be successfull	Good quality	Low costs	
		Agility			Limitations			Low risk and fast adaptation				Quality	Costs	Total
	Weight (0-100 %)	*5*	*15*	*15*	*15*	*8*	*5*	*7*	*5*	*5*	*7*	*10*	*3*	*100*
Evaluation (1 false - 4 true)	Waterfall	1	1	1	4	4	4	4	3	4	2	3	4	2,66
	Incremental	4	3	3	4	4	4	4	3	3	4	3	3	3,47
	Spiral	3	3	2	4	4	4	2	2	3	4	4	3	3,18
	Crystal	3	4	4	3	3	4	1	3	2	3	3	3	3,16
	Scrum	4	4	4	3	3	4	4	4	4	4	3	3	3,64
	Feature-Driven Development (FDD)	4	4	4	3	3	4	1	3	1	1	4	3	3,12
	Dynamic Systems Development Method (DSDM)	4	4	4	2	4	4	1	4	3	4	3	3	3,31
	Adaptive Software Development (ASD)	4	3	4	3	4	4	1	1	1	1	3	3	2,85
	Rapid Application Development (RAD)	4	3	4	3	4	2	1	1	2	2	3	3	2,87
	eXtreme Programming (XP)	4	4	4	3	2	4	1	4	4	4	4	3	3,45

Figure 15. Comparison of software process models. [Abr02] [Agi08] [Pre05] [Sch02] [Sch02b] [Som06] [Tha05] [Vli04]

Appendix B: Agile Manifesto

Manifesto for Agile Software Development [Bec01]

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on
the right, we value the items on the left more.

Signed and created by:
Kent Beck, Mike Beedle, Arie van Bennekum,
Alistair Cockburn, Ward Cunningham, Martin Fowler,
James Grenning, Jim Highsmith, Andrew Hunt,
Ron Jeffries, Jon Kern, Brian Marick,
Robert C. Martin, Steve Mellor, Ken Schwaber,
Jeff Sutherland, Dave Thomas

Appendix C: Decision Points

The following criteria describe the minimum set of issues which should be clear and done before each DP. If a criterion in a DP is not fulfilled and the decision is still GO, the exception and explanation should be written down in the DP meeting minutes. [ABC06]

DP0 Criteria
1. Inspections of the business needs 2. Project proposal accepted 3. Project responsibilities appointed for DP0-DP1 phase 4. Solution scope agreed 5. All relevant stakeholders identified 6. All critical viewpoints identified 7. Deliverables, budget and schedule determined for the DP0-DP1 and DP1-DP2 phases

DP1 Criteria
Requirements 1. Requirements can be approved by each and every stakeholder 2. Requirements within DP0 scope 3. Critical viewpoints considered 4. All major questions identified and listed *Functional & Architecture Analysis* 1. Basic architecture framework inspected 2. Non-functional requirements inspected 3. Technical pre-study completed

DP2 Criteria
Requirements 1. All requirements “complete” 2. All viewpoints considered 3. Requirements are of a good quality (explicit, understandable, feasible, verifiable, within the agreed scope) 4. The complete version of the requirements specifications can be approved by the customer and all stakeholders 5. There are no critical issues open, minor issues listed

Functional & Architecture Analysis

1. Functionality analyzed and determined, functional specification inspected
2. Architecture analyzed and determined, architecture specification inspected
3. (Optional) UI navigation charts and UI prototype inspected
4. The development life cycle for the system to be implemented is analyzed.

Testing

1. Functional specification document commented based on testing viewpoint
2. Requirements specification commented based on testing viewpoint
3. Has a testing strategy been developed for all significant changes in applications and infrastructure technology? Does the testing strategy address all phases of testing?

Configuration management

1. Configuration management plan can be approved
2. Environment for configuration management available

DP3 Criteria

Design and Implementation

1. Technical implementation is planned
    a. Module specifications inspected
    b. Build plan inspected
    c. Final UI specification and layouts inspected
2. Preliminary UI texts, terms, error messages and helps available
3. Development environment is ready

Testing

1. Testing is planned
    a. Test plan inspected
    b. Test cases are “complete” and inspected
2. Testing environment is ready for integration testing in the small

DP4 Criteria

Design and Implementation

1. Implementation is inspected
2. User interface is inspected

	Testing *1.* Integration testing in the small is passed, and the test report for integration testing in the small is inspected 2. System testing is passed, and the test report system testing is inspected *3.* Integration testing in the large is passed and test report for integration testing in the large is inspected *Deployment* 1. Deployment description is inspected 2. Operation and maintenance guide is inspected 3. Installation and configuration guide is inspected

DP5 Criteria	
	Testing 1. Project deliverables are validated 2. Acceptance testing is passed, test report for acceptance testing is inspected *Deployment* 1. System is deployed

DP6 Criteria	
	Security 1. TS requirements on security, availability and processing integrity are considered in the system design *IT Continuity Management* 1. Testing strategy for the continuous maintenance of the systems exists and covers unit, system, integration and user acceptance testing 2. IT continuity plan and procedures are established *Architecture Implementation* 1. The needed exemptions are handled according to the process *Operation and Maintenance* 1. Maintenance operation for the project deliverables is established according to the maintenance plan 2. Technical operation is established

DPE Criteria	
	Security 1. TS requirements on security, availability and processing integrity are verified *IT Continuity Management* 1. Deployment of the testing strategy for the continuous maintenance of the systems is verified 2. IT continuity management results of the ITCM plan are reviewed and tested *Architecture Implementation* 1. Evaluation report is reviewed by BCC Architecture *Operation and Maintenance* 1. Maintenance operation is secured according to the maintenance plans

Appendix D: Templates for the interviews

The following templates were used in the interviews at companies A, B, and C. The guiding questions and key areas of focus were not very binding and the exact topic of a single interview depended on the role of the interviewee in the organization.

A1-related interviews

- How does the A1 requirements engineering process currently work from your perspective? Describe in detail.
- What are the pros and cons? What are the biggest bottlenecks?
- How should things work in your opinion? Consider the timeframe and what is realistic.
- What kinds of results have been gained in the previous enhancement projects?

Key areas of focus

- Keep the main emphasis on project planning phases before DP2
- Ask more specific questions regarding certain methods or details of the process after the first interviews
- Try to find differences between the opinions of IT and business representatives

B1-related interviews

Key areas of focus

- Identify differences between the A1 and B1 process models
- Keep the main emphasis on requirements engineering
- Ask more specific questions regarding the most interesting methods or details of the process after the first interviews

Interviews at company C

Key areas of focus

- Try to form a big picture relating to the organization of company C and its core IT systems
- Find out how company C applies Scrum practices
- Observe how the sprint planning meeting and sprint review meeting are held in practice
- How does the requirements engineering process work?
- What kinds of roles are needed outside the Scrum project teams?
- Try to find differences between the opinions of IT and business representatives