Requirements Managements from a Life Cycle Perspective – Overview and Research Areas

HS-IDA-MD-01-013

Åsa G. Dahlstedt (asa@ida.his.se)

Department of Computer Science
Högskolan i Skövde
S-54128 Skövde, SWEDEN

Supervisor: Benkt Wangler

Submitted by Åsa G. Dahlstedt to University of Skövde as a dissertation towards the degree of M.Sc. by examination and dissertation in the Department of Computer Science.

[2001-12-06]

I hereby certify that all material in this dissertation, which is not my own work has been identified and that no material is included for which a degree has previously been conferred on me.

Signed: _____

Abstract

Requirements Engineering (RE) is nowadays considered to be an activity, that aims at supporting the whole lifecycle of an information system by: eliciting, documenting, validating, and managing the requirements of the system. This thesis aims at providing an overview of the area of Requirements Management (RM) and to identify important and interesting issues or areas where further research is needed.

RM includes two major areas; organising requirements and requirements change management. Organising requirements is concerned with structuring the requirements and storing additional relevant information about them e.g. attributes and traceability information. Requirements change management is concerned with dealing with changing requirement in a systematic way i.e. making informed decisions whether to implement a certain change or not, and support the implementation of approved changes.

In order to provided a broader view of RM, the literature study were complemented by an interview study of how RM is conducted in practice. This interview study shows that the effort resources spent on RM differs substantially between different organisations. Various reasons for these discrepancies are elaborated in the work, but one of the main reasons are the type of software development that is conducted in the organisation. There is a tendency that organisations that develop software products and continuously releases new versions of there products are more likely to spend resources on RM, compared with organisations that develop customers specific solutions in one shoot projects. The need to reuse requirements and knowledge, as well as the maturity of the RE/RM process, are other factors that affects the resources spent on RM. The RM activities performed in practice are concordant with the activities found in literature.

A number of areas where further research is needed were identified: requirements change management, dependencies between requirements, RM tools, and information management

Key Words: Requirements, Requirements Engineering, Requirements Management, Information Managements, Requirements Attribute, Traceability, Dependencies between requirements and Requirements Change Managements

Acknowledgements

There are several persons I would like to thank for their support during this work. First, I would like to thank my supervisor, Benkt Wangler, for his valuable advises and patience during this work of this dissertation. I would also like to thank Anne Persson for her encouragement and support during the way towards an MSc degree, and Per Backlund for always been there during tough times. In addition, I thank my colleagues in the Information Systems Engineering research group for good discussion and valuable advises. Finally, I would like to thank and my husband Mats and my family, for just being there for me.

Table of Content

1 Introduction

The successes of information systems depend on their ability to meet the needs and expectations of their customers and users, and on their ability to fulfil the business needs of the organisation. Due to this, one of the central criticisms of today's information systems is that they do not meet these needs and expectations (Flynn, 1998). There are also many information systems development projects that are delivered late and over budget, or, in some cases, not at all. Many of these serious problems can be traced back to problems with requirements. (Loucopoulos and Karakostas, 1995; Willcocks and Lester, 1993; Leffingwell and Widrig, 2000; Karlsson, 1996). Requirements Engineering (RE) is hence a key issue in the information systems development process, in order to successfully develop an information system that meets the needs and expectation of the stakeholders.

RE is traditionally viewed as the iterative process of developing requirements, by analysing the problem, documenting the result in various representation formats, and checking the validity of the knowledge gained (Loucoucopoulos and Karakostas, 1995). However, RE has moved from supporting the early phases of individual projects, to supporting the whole lifecycle of information systems in a rapidly changing organisation (Jarke et al, 1994). RE is nowadays also concerned with maintaining the system’s requirements over time. One important aspect of this shift is the need of managing the large amount of requirements, and foremost, dealing with changes of the requirements information. RE is, hence, not just about eliciting, specifying and validating the requirements but also about managing the requirements throughout the whole lifecycle of an information system. This activity is called Requirements Management (RM)

There are two things that make management of the requirements problematic. First, creating and maintaining orderliness if there is large number of requirements to manage is not a trivial task (Leffingwell and Widrig, 2000). Second, changes to requirements are both difficult to handle and time consuming. The problem of changing requirements is well represented in the literature (Curtis et al, 1988; Leffingwell och Widrig, 1998; Lubars et al, 1993; Kotonya and Sommerville, 1998; Harker et al, 1993). RM addresses these problems. Today, there exists quite a lot of research within RM, but most of the

reports are at a very detailed level concerning specific, delimited issues within the area. It is quite difficult to gain an overall view of what RM is and what it is concerned with. In this work, the intentions are to make this overview of the area and to identify and describe the major activities and concerns of the area. The main aim is to identify problem areas where further research is needed.

1.1 Aim and Objectives

The aim of this work is to survey the area of Requirements Management. The survey will take a *life cycle perspective* and aims to serve as a basis for further research in the area. Therefore, an important task of this work will be to identify issues for further research.

In order to fulfil the aim of the dissertation, five objectives shall be obtained:

1. *Define the notions of 'Requirements' and 'Requirements Management'*
   Defining and describing the meaning of these two concepts are an essential foundation to the ongoing work. We will therefore start by discussing and defining these.

2. *Survey the area of Requirements Management from a lifecycle perspective*
   According to the definition formulated, a deeper study of RM will be made in order to achieve a greater understanding of the process. Major concerns and activities, performed within RM during the whole lifecycle of information systems, will be investigated.

3. *Identify how Requirements Management relates to Requirements Engineering and to the information systems life cycle.*
   It is also important to have a clear view of the relationship between RM and RE, since both processes are dealing with requirements. We will also relate RM to the information systems life cycle, i.e. where in the lifecycle RM is conducted.

4. *Investigate how Requirements Management is conducted in practice.*
   In order to get a broader view of RM, we will complement the work with a study of how RM is conducted in practice.

5. *Identify and discuss problem areas and problems within Requirements Management*
   We will identify relevant problems within the area, in order to find issues for further research.

1.2 Way of Working

The broad purpose of this research proposal is to gain a greater understanding of the area of RM, and to identify a number of issues for further research within the area. The work is, thus, of an explorative nature, and hence suitable for qualitative analysis. This means that we will gain a broad knowledge of RM, by extracting knowledge from different sources and analyse the information gained.

We will start by surveying the literature relevant for the work. Books as well as research reports contain valuable information and can contribute to establishing an understanding of the state of the art within the subject area.

Second, we will interview persons working with requirements, in order to achieve a broader understanding of how requirements are managed. The interview will serve as a complement to the literature survey and will be used to compare theory with practice. The interviews will also be used to identify relevant problems within RM.

1.3 Dissertation Outline

In chapter 2 the notion of requirements is looked into, both in general and by describing different classifications of requirements. The requirements chapter is ended with a brief presentation of the Requirements Document.

Chapter 3 involves a brief presentation or RE and places RM in a broader context. The chapter includes a presentation of requirements elicitation, specification and validation, as well as management. The chapter is concluded with a discussion of the relation between these activities. RM is presented in more detail in chapter 4, where the main activities of RM are looked into. The chapter starts with a discussion of how to define RM and several different definitions are compared. These are summarized and a

definition used in this work is presented. The two main activities of RM are described and the life cycle perspective of RM is discussed.

Chapter 5 presents an interview study describing RM in practice. The structure of the interview study is described and the companies and interviewees are briefly presented. The results of the interview are presented in three major categories: general description of the requirements work, management of requirements, and problems with RM. These findings are analysed and some tentative conclusions are made. In chapter 6 some RM issues are presented, based on the literature and interview study performed. Chapter 7 presents some concluding remarks about the work, which contains an evaluation of way of working as well as a description of future work.

2 Requirement

This chapter includes a presentation of the notion of requirements and serves as an important basis for describing RE and RM. The nature and elements of requirement are discussed and three different classifications of requirements are presented. This chapter also includes discussion about the level of detail of requirements and possible reasons for requirements change.

2.1 Requirement - A Definition

Requirements are important within the ISD process, because they set the criteria by which the acceptance, usefulness and success of the IS is measured (Loucopoulos and Karakostas, 1995). The success of IS is depending on their ability to meet the needs and expectations of their users and customers. If the requirements sets the criteria for success of the IS, they must, in some way, describe these needs and expectations. But it is not easy to define what a requirement is and there is no commonly used definition of requirements (Karlsson, 1998).

Generally speaking, requirements define what the system is supposed to do, i.e. they are specifications of the services that the system should provide. To view requirements only as descriptions of how the system should behave is rather simplistic. A requirement may also include information about the application domain, constraints or circumstances under which the system is required to operate, or description of properties or attribute of the system (Kotonya and Sommerville, 1998).

A large number of definitions of requirements have been proposed. One of the most notable definitions of requirement is IEEE Standard '610' (1990), which is used by many authors (Macaulay, 1996; Pohl, 1996; Loucopoulos and Karakostas, 1995). They define requirements as:

1. *A condition or capacity needed by a user to solve a problem or achieve an objective.*

2. *A condition or capacity 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 capacity as in 1 or 2.*

We have not found any description or discussion of this definition. The discussion below is therefore a presentation of different ways of interpreting the definition, as we see it.

The first part of the definition focuses on needs of the *users*. However, there are a lot of other actors that have needs that the system should fulfil, who are not end-users of the system (Karlsson, 1995). For example, people who are responsible for the maintenance of the system may have requirements for its development in order to make the system easy and/or cheap to maintain. Examples of other actors are customers, managers, and testers. Therefore, the term 'user' will be given a broad interpretation, also to include all people who are directly or indirectly affected by the system, i.e. all the stakeholders of the system.

According to IEEEs definition a requirement can be a condition or a capacity. A *condition* is a particular mode of being of the system or of the development of the system, a prerequisite for the system or development process. A *capacity* describes something that the system should be able to do, an ability that the system should have (Webster's, 1996). It is rather difficult to separate conditions and capacities. Whether a requirement can be considered as a condition or a capacity, highly depend on how the requirement is formulated. Our interpretation is that a requirement is something that the system shall be able to perform, an ability of the system, or something that the system shall be able to achieve, a prerequisite for the system. Examples of condition and capacity are: 'the system shall provide some control of access, to ensure data security', 'the development method used must be SSADM', 'the system shall be able to perform, i.e. start and complete, a customer search within 10 second', and 'the system shall be able to present a report of orders placed during the last month'.

The first sentence of the definition focuses on the users' *needs* to solve a problem or achieve an objective, normally in order to perform their work efficiently. This is a rather usual way of defining requirements, and it corresponds to many other definitions, e.g. in

Karlsson (1995). Examples of these kinds of requirements are: ‘the cost of developing the system shall not exceed 3 million US$’, ‘the system shall keep track of information about customers and their orders’ and ‘the system shall provide a function to search for available beds in the nearby hospitals’.

The second sentence of the definition indicates that a requirement may also be a condition or capacity that *must* be fulfilled by the system. Agreements such as contracts, standards, or specifications may force this condition or capacity on the system. Examples of such requirements are: ‘the system must follow the communication standard XXX’ and ‘the system must be able to present a report of the companies income per month, according to the Swedish accounting legislation’.

There may also be requirements that need to be fulfilled due to interactions with other systems. If the existing system is well documented, these kinds of requirements may be covered by the second sentence of the definition. However, there are a lot of systems that are not documented, and these may also affecting and constraining the development of the new system. These kinds of requirements are not fully covered by the definition above, given the interpretations discussed. Maybe it is not enough to broaden the first sentence to stakeholders, but also to other systems.

The third sentence states that a requirement can be a documented representation, e.g. a description, schemas or expression, of the parts of the definition discussed above. Our interpretation is that the definition separates the requirement and the representation of the requirement. They separate what is a need or requirements ‘inside peoples heads’ and what is documented or specified in the requirements document. This indicates that a requirement does not have to be documented to be a regarded as a requirement of the system. According to Karlsson (1995) requirements can be explicit or implicit. *Explicit* requirements are expressed by the stakeholders, and represent functionality or other requirements that the stakeholders clearly ask for. These expressions are preferably represented in the ‘requirements list’ in the requirements document. *Implicit* requirements can be represented in schemas or descriptions of different kinds, graphically or textually. It can also be requirements that the stakeholders do not explicitly ask for. Karlsson (1995) identifies two kinds of requirements, which the stakeholder may not ask for: expected and exciting requirements. Expected requirements are so fundamental or trivial for the stakeholders that they may not think

of mentioning them. These requirements are often very important for the success of the system. Exciting requirements are e.g. technical features that the stakeholders do not know of, and therefore cannot ask for. They are beyond the stakeholders' knowledge.

Karlsson (1996) defines a requirement as a *desirable* property, and states that there are few requirements that inevitably must be fulfilled. In this report we agree to that most requirement are negotiable, but it may also exist requirements that must be fulfilled, as discussed above. Furthermore, Karlsson (1996) discusses the nature of requirements, and state that every requirement has an *origin*, a *motive*, and a *realization object* (Figure 1).

A requirement always has an *origin*, usually one or more stakeholders who have proposed the requirement. Common stakeholders are: customer, customer's customer, end-user, developer and tester. This emphasises the result of the discussion above, there are other actors who may have requirements on the system, not only the end-users. But there are other origins of requirements. A requirement may be elicited from a document or from other requirements specified earlier in the process. It may also be essential due to the need to interact with existing system.

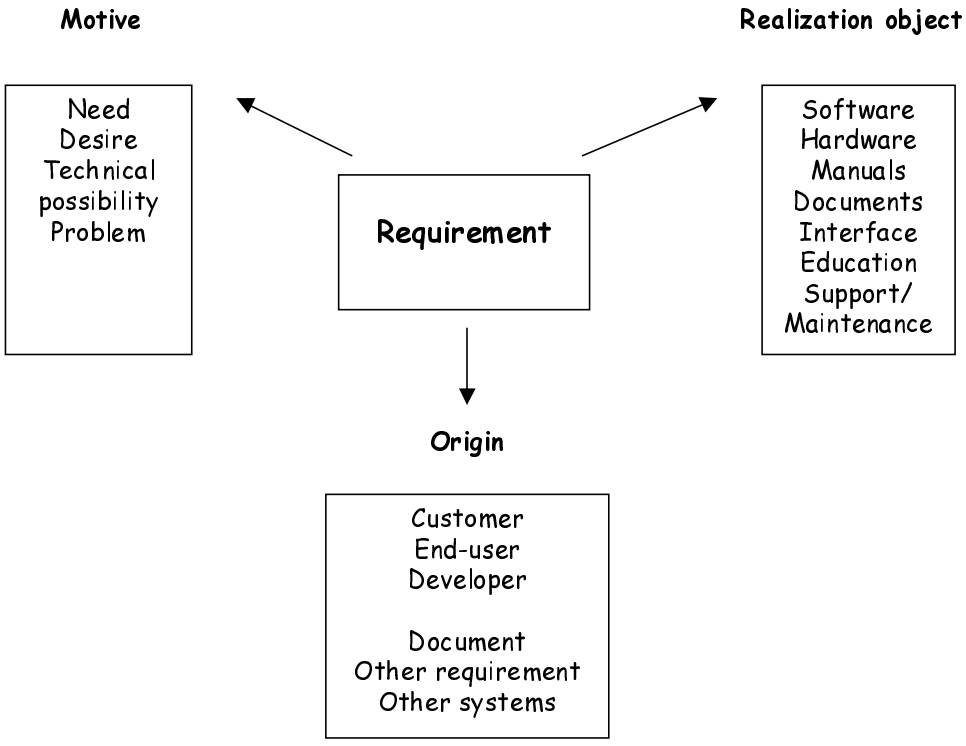

Figure 1: The elements of a requirement (from Karlsson, 1996)

Each requirement has some *motive*, which entitles the existents of the requirement. The motive is usually to achieve something for a special stakeholder, e.g. a functionality that the users need to perform their work or a technical possibility, which will make a certain task more efficient. The purpose of the system, described by the requirements, is to assist organisational-related activities in the customer's enterprise, i.e. there is a close relation between the system and the organisation of which it is a part (Flynn, 1998). Therefore, it is important that the system supports and meets the business strategies of the organisation (Loucopoulos and Karakostas, 1995). The requirements of the system must therefore be related to the business strategies of the organisation, and hence to the organization's objectives and mission.

Finally, a requirement has one or more *realization objects*, i.e. something that implements or fulfils the desirable property of the IS. The most usual realization object is software, but it may also be e.g. hardware, manuals, interface or documentation of the system. The realization object does not have to be chosen in the RE process. It is often more advantageous to handle this in the later phases, where the design-decisions are made.

To conclude and summarise the discussion about requirements, we can state that there exist no common and clear definition of what a requirements is. But some fundamental characteristics can be identified:

- Requirements describe various aspects of the hypothetical system. Requirements can describe the abilities of the system as well as, for example, prerequisites, constraints and domain information.
- A requirement can be either a desired property or a necessary property. Most requirements are negotiable, but there exist requirements that *must* be implemented due to standards, laws, or other formally imposed document.
- The source of a requirement is not only users, other stakeholders such as developers and managers may also have needs that ought to be fulfilled by the system. Requirements may also origin from document, existing systems, or other requirements.

- A requirement can be either explicit or implicit. The specified statement must be separated from the ‘thought’ or need inside of peoples head. Needs that are not expressed by the stakeholders, are also considered as requirements of the system.

- There are a close relationship between the organisation and the system to support it, and hence the requirements of the system. The requirements must be related to either the goals or strategy of the organisation as well as support the users’ working tasks.

Figure 3 states that a requirement has a source, a motive and a realisation object.

2.2 Classification of Requirements

There are many different ways to classify the requirements of an IS. The most well-known and traditional way is to divide them into two types: functional and non-functional requirements. Another way of looking at requirements is whether they directly originate from a stakeholder or if they have been derived from these into system requirements, i.e. the detail of the requirements. A third way of classifying requirements is into stable and volatile requirement. All these three classifications will be further described in the following sections.

2.2.1 Functional vs. Non-Functional Requirements

The *functional requirements* describe the services or, as the name implies, the fundamental functions of the IS, i.e. they describe what the system should do (Loucopoulos and Karakostas, 1995; Karlsson, 1996; Sommerville and Sawyer, 1997). The functional requirements are the core of the Requirements Document and should describe the relationship between all the combinations of input to the functions, with corresponding output (Karlsson, 1995). For example, a functional requirement may state that the system must provide some ability for the end-user to search for customer number in the customer register.

Non-functional requirements deal with the constraints and restrictions that can be placed on the IS, the development process, and the system environment, i.e. they are external constraint that the IS must meet (Loucopoulos and Karakostas, 1995; Kotonya and Sommerville, 1998). These affect the way in which the requirements can be

implemented (Sommerville and Sawyer, 1997). Karlsson (1996) has simplified the definition and categorizes all requirements that cannot be referred to as functional requirements as non-functional requirements. Non-functional requirements exist because stakeholders needs to achieve certain goals, e.g. concerning budget constraints, organisational policies, need to interoperate with other systems, or that a certain degree of security must be possessed (Kotonya and Sommerville, 1998). If there is a functional requirement concerning the possibility to search for the customers, a non-functional requirement may be that this search must take less than ten seconds. Examples of different kinds of non-functional requirements are: maintainability, usability, reliability, efficiency, performance and capacity of the system, process requirements, legal and economic constraints and interoperability requirements (Karlsson, 1995; Kotonya and Sommerville, 1998)

Both Karlsson (1995) and Kotonya and Sommerville (1998) describe a framework for classification of non-functional requirements, but the frameworks differ from each other. Karlsson (1995) divides non-functional requirements in quality requirements, constraints and other requirements.

- *Quality requirements*
  describe how well the system as a whole, rather than individual functions, is supposed to be perceived. Both users and developers can have quality requirements on the system, but with different focus. Developers are more concerned with the quality of the maintenance and the development, when the users often focus on the use of the system. Quality requirements may, for instant, condern; maintainability, usability, reliability and efficiency.

- *Constraints*
  state restrictions on the behaviour of the system by defining properties regarding how well a functional requirement must be performed. Examples of constraints are timing constraints such as response times, and accuracy constraints such as the correctness of computation.

- *Other requirements*
  are not directly concerned with or related to the system. These may be

requirements concerning scheduling, education, deliverables, develop time and cost, methods and standards etc.

Kotonya and Sommerville (1998) have classified non-functional requirements into products requirements, process requirements and external requirements.

- *Product requirements*
  focus on constraining the behaviour of the product, and therefore limit the design of the system. Product requirements may concern usability, reliability, safety, efficiency, performance or capacity of the system.

- *Process requirements*
  focus on constraining on how the development process can be performed, e.g. by development standards, using a certain method and having management reports that must be provided. Examples of process requirements are requirements concerning delivery, implementation and standards of the system development process.

- *External requirements*
  are requirements from the environment in which the system is developed, and focus either on the product or the process. Legal and economic constraints, and interoperability requirements placed upon the system belongs to this category.

Both frameworks classify approximately the same basic requirements, but they do it in different ways. Product requirements may either be comparable with quality requirements, if they are concerned with the system as a whole, or with constraints, if they are describing the behaviours of systems functions. Process requirements correspond to requirements in the ‘other requirements’ category. External requirements may, like the product requirements, be compared with requirements within quality requirements or constraints; depending on weather they are concerned with the whole system or different functions of the system. They can also be compared with ‘other requirements’ if they are concerned with e.g. economical constraints.

The first classification provides classes that are more clear and distinct. The distinction between product and process is clear, but external requirements are more difficult to

grasp. The first classification separates requirements that deals with the system as a whole from requirements that constraints the functional requirements. The category ‘other requirements’ includes requirements describing the development process and other aspects that are not direct statements about the system. We think that the first framework is a more useful, and easier to utilize.

To be of any use, a categorization of requirements must facilitate the communication within a project, be unambiguous and make the work in the RE-process more efficient (Karlsson, 1996). The distinction between functional and non-functional requirements is often not clear, and sometimes it is even unambiguous, which may cause less efficiency in the RE-process (Loucopoulos and Karakostas 1995). For example: a requirement can be seen as a functional or non-functional requirements depending on the degree of detail. This is shown by an example of Kotonya and Sommerville (1998), where the requirement: ‘the system shall ensure that data is protected form unauthorised access’ would mostly be considered a non-functional requirement. However, if the requirement is specified in some more detail, such as: ‘The system shall include a user authorisation procedure in which the users must identify themselves using a login name and password’ it looks more like a functional requirement. Because of this confusion, some authors claim that this categorisation of requirement should be avoided. Despite this there are some advantages with the categorisation. Most functional and non-functional requirements are rather different in their nature, and may therefore need different kind of documentation. Another reason is that non-functional requirements describe constraints on the system service, which make them critical important to the system ability to work in the actual environment in which it is to be installed. This causes that functional requirements may need to be sacrificed in order to make the system meet these constraints.

2.2.2 Stakeholder Requirements vs. System Requirements

Requirements can also be classified according to their degree of detail. Jarke and Pohl (1994) describe RE (Chapter 3) as the process of transforming the central system vision into a requirements document, which can be used as a basis for design and implementation. The central system vision is typically stated in a quite simple manner and gives an overall view of the hypothetical system. An example, given by Jarke and Pohl (1994, pp. 4), is John F. Kennedy's classical vision to *"send a man to the moon*

before the end of the decade". The requirements are hence specified in more and more detail during the RE process, from fuzzy initial ideas (the vision) about the system to a formal specification describing the system in detail.

Sommerville and Sawyer (1997) have identified two broad categories of requirements, based on their different degrees of detail; stakeholder requirements and systems requirements. The *stakeholder* requirements are abstract description of the system services needed by the stakeholder. The focus is to describe what the stakeholder needs and how the system should be integrated with the business processes. These types of requirements may also be referred to as user requirements (e.g. Stevens et al, 1998) but, as discussed before, requirements may not just come from the users of the system, and therefore, the more general term ‘stakeholder’ will be used in this work.

Systems requirements are a detail specification of system services, and constraints on the implementation of these services. These requirements should together be a complete description of the behaviour of the system.

Stevens et al (1998) also have a classification regarding the degree of detail of the requirements: user and system requirements. They describe *user* requirement as the first step towards a description of the system. User requirements define what result the system will provide to support the users. The user requirements can be used as a control instrument, with which the users can control the development of the system. User requirements are focused on the users need and must be understandable for the users. Therefore, user requirements shall be short, non-technical and easy to understand and correct, and they must be described with the users own words. This corresponds to the ‘stakeholder requirements’ presented by Sommerville and Sawyer (1997) above. As discussed, the term stakeholder requirements will be used for these kinds of requirements.

According to Stevens et al (1998, pp. 22) a *systems* requirement "*imposes requirements on an abstract model of the final system*". They are an establishment in more detail of what the system shall do. They are developed from the user requirements and define

what should be done to meet these high-level requirements. They are written for the developers and should describe their responsibility towards the user requirements[1].

To sum up, *stakeholder* requirements are:

- overall descriptions of the systems services, i.e. the result that the system will supply to the stakeholders (high level requirements)
- the first step towards a definition of the system
- can be used by the stakeholders to control the development process
- focused on the needs of stakeholders and must be described with stakeholders own words

System requirements are:

- detailed specification of the final system (low level requirements)
- evolved from user requirements
- focused on describing the final system to the developers

Stevens et al (1998) emphasize the need to separate these two types of requirements. If mixed it is not possible to control neither set of requirements for completeness. It also becomes more difficult for the user to understand the requirements, and thus, weakens their ability to participate and influence on the development of the system.

There are many approaches and procedures how to specify the requirements in more detail, i.e. evolve the central system vision into stakeholder requirements, and these into the more detailed system requirements (Hjelte et al, 1995). A well-known approach is Quality Function Deployment (QFD). QFD is a development method that aims at developing products that meets the customers' requirement, and it has been successfully used in the manufacturing industry (Karlsson, 1995). The principle behind QFD is to achieve this by focusing on issues that are most likely to achieve customer satisfaction. This is ensured by capturing the ‘voice of the customer’ early in the development process. After the needs, requirements, and problems are identified, these are translated

[1] Some authors use a different name, and call these kinds of requirements for software requirements e.g. Leffingwell and Widrig (2000). In this work, we decided to use the term system.

into a solution that is as optimal as possible (Nilsson, 1990). The similarities between the philosophy behind RE and QFD are quite apparent, and make it interesting to apply the concept of QFD to the RE process.

The ‘voice of the customer’ mostly expresses basic needs of the customer and as a result of this, they are often vague and not possible to verify (Karlsson, 1998). In QDF these requirements are called primary requirements. With the ‘voice of the customer’ as a basis, more detailed requirements of the system is derived. The primary requirements are normally expanded into more detailed secondary requirements, which describe the system more precisely. If needed, these are further expanded into even more detailed requirements, so called tertiary requirements. QDF are also concerned with relating the customer requirements to characteristics, which are measurable goals of the requirements and describes the properties of the product. This relationship is described in a matrix. The requirements are prioritised, to be able to select an optimal subset of requirements to implement. This prioritisation is also used as a basis for making trade-offs between requirements.

2.2.3 Stable Requirements vs. Volatile Requirements

As discussed above, there are a close relationship between information systems and the organisation, in which they are an integrated part. Business changes, such as changes to the company's mission, objectives, strategies, and business processes, affects the system development process, as well as the operation and maintenance of the implemented system (Patel, 1999). Changes in, for example, working procedures within the company must be reflected in the information systems supporting these activities and processes. Adjustments shall not only be undertaken during the development of an information system. During the operation and maintenance of information systems, it must be allowed to undertake changes in order to reflect business changes.

Patel (1999, pp. 80) has investigated the factors of organisational change in four organisations. The three most common factors were management decision, new or enhanced technology, and organisational processes or procedures. Other important factors mentioned are job description, organisational objectives, organisational task, and personnel. Many of these factors are rather fundamental for the organisations, and the information system(s) in the organisations shall support these factors, e.g. the business

processes and management decisions. If the organisation changes but not the systems, these are no longer supporting the organisation efficiently.

Dynamic organisations often has to deal with the problem that users at varying levels has widely different and extremely volatile views of the system's requirements, and hence what the system should do (Paul and Macredie, 1999). The maturity of the customer regarding using and, in some extent, developing information systems, affects the volatility of the requirements. An immature customer is more likely to discover new possibilities or requirements in the middle of a project (Karlsson, 1996).

It is not only changes in business that affects and causes changing requirements. There are a number of different other factors (Kotonya and Sommerville, 1998):

- Requirements errors, conflicts and inconsistencies
- The customer and end-user evolves their knowledge of the system during the lifecycle, and then change their requirements
- Changing customer priorities
- Problems with cost, schedule or technology, which makes the requirements non-feasible
- Changes in laws and regulations which are relevant to the system
- Changes in the environment in which the system is to be installed

A perfect view of the requirements is impossible to achieve. Clarification and modification of the requirements are inevitable during both development, and review and maintenance. New knowledge or understanding affects the view of the organisation, information system and it's requirements, and causes or highlights errors, feasibility problems. Requirements changes do not have to imply poor RE practice, although this may be the reason in some cases.

Although requirements change is inevitable, some requirements are more likely to change than others. Kotonya and Sommerville (1998) and Harker et al (1993) divide requirements into stable requirements and volatile requirements. *Stable* requirements are primarily high-level requirements, related to the technical core of the business and application domain. The business of the organisation is based on this core, e.g. produce meat, educate students, and lend books, and these requirements describe the core

functionality needed in the future system. They change too, but more slowly than volatile requirements. *Volatile* requirements are more specific, and relates to the *"instantiation of the system"*, installed in a particular environment and aim at a particular customer.

For example, stable requirements of a library system may be that the system should keep track of the objects you can borrow from the library, such as book, journals, and CDs. The system should also be able to store information about the borrowers, which objects that have been borrowed and by whom, and when the object should be returned. Volatile requirements of the system may be which information to be stored about the objects, how long the objects may be borrowed and the how the borrowing procedure is managed and which information that is stored about every single case. Thus, detailed information about how the library handles their business, lending objects such as books and journals.

Kotonya and Sommerville (1998) and Harker et al (1993) present five different types of volatile requirements:

- *Mutable requirements*
  These requirements are related to and affected by changes to the organisation's environment. This type of volatile requirements is highly represented in organisations with an environment that are dynamic.
  Examples of causes of changes are: market change, new means of production creates new opportunities, and government changes policies and legislations.
  Mutable requirements arise as a response to external pressure, i.e. demands outside the system and its development process
  Source: Dynamic and changing environment

- *Emergent requirement*
  These requirements evolve or emerge during the information system development process. Stakeholders often have problems with describing the intended system. It takes some time to identify requirements, goals to achieve, and to understand technical possibilities to serve these goals.
  Actions to develop the requirements and goals of the system must be taken.
  Emergent requirements are highly related to the means used to generate

requirements and are a direct outcome of the process of engaging the stakeholders in the development activities.
Source: Stakeholders engagement in developing the requirements

- *Consequential requirement*

  When a system is installed into an organisation, new ways of working, new tasks, and new insights about individual capabilities are likely to be discovered. This creates needs and pressure for an enhancement and upgrade of the system. Consequential requirements are usually identified after the system has been taken into use. But they can also be simulated by prototyping, using scenarios, or demonstrate the system. They are under these circumstances a kind of emergent requirements. Technology affects by nature the needs of an organisation and causes the requirements to evolve.
  Source: System installation and use

- *Adaptive requirements*

  Early system developed usually constrain the way users achieved their tasks. They determined, for example, the method of working, and the sequence of operations. These systems do not fit the variety of situation that the users have to deal with, and this were a reason for dissatisfactions among users. Systems must support a wide range of ways to operate, i.e. be flexible and adaptive to allow users to choose the most efficient way of performing their working tasks. Adaptive requirements are driven by a whish for increasing the adaptation and capability of the delivered system, i.e. that changes should be an on-going capability of the system.
  Source: Need for adaptation and flexibility in work

- *Migration requirements*

  During the process of improving and change an organisation, for example by a new information system, the core business of the organisation must still continuo. Revolutionary changes that threaten the execution of the core business should or even must be avoided. The organisation may choose to implement some requirements before others, so changes can be done orderly. Existing system and technology may then create constraints on requirements, and therefore changes the original set of requirements. The requirements that

are being implemented must fit with existing equipment, e.g. computers, platforms, or systems. Changes of the existing system and technology also affect the requirements of the system under development.
Source: Constraints of planned organisational development, and existing system and technology

Requirements may also, as discussed above, changes due to mistakes made during the development process, i.e. requirements errors, conflicts and inconsistencies, or problems with cost, schedule and technology.

2.3 Requirements Document

The Requirements Documents are official documents presenting a statement of the problem to be solved and the agreed requirement needed to solve it (Sommerville and Sawyer, 1997; Loucopoulos and Karakostas, 1995). It is in these documents that all the *agreed* requirements are specified.

A Requirements Document contains three parts, namely (Loucopoulos and Karakostas, 1995):

- Enterprise models
- Functional requirements
- Non-functional requirements

Enterprise models describe the environment where the system will be installed. To be able to develop a common understanding about the problem, the enterprise, in which the system will operate, must be defined. The purpose of the system is to be found in this description of the enterprise. Traditionally, the Requirements Document describes the functions of a system, i.e. the desired services of the system. This is the concern of the *functional requirements*, which describes the fundamental functions of the systems. The *non-functional requirements* describe the constraints placed on the system, the development process and the system's environment (see 0) and affect the design of the system. Both functional and non-functional requirements are normally presented in a list of explicit expression about the system, while the models should supports the understandings of the expression represented.

The traditional view of the Requirements Document is that the purpose is to serves as a basis for further development. In this work, the basic assumption is that the Requirements Document has other important functions in the information systems development process. These are (Loucopoulos and Karakostas, 1995):

- *Communication between the actors involved in the RE process*
  The Requirements Document provides a basis to create a common understanding of the domain, the business and the hypothetical system.

- *A contract between the customer and the developers*
  The Requirements Document is a contract of what system to be built. This is especially relevant when the developers are an external vendor, i.e. the system is not developed 'in house'.

- *A basis for evaluating the final product*
  The Requirements Document can be used for evaluating the system, for example in an acceptance test agreed between the customer and the developers.

The Requirements Documents has a central role within the development process. The quality of the Requirements Documents affects the quality and acceptance of the system that is developed. It is therefore important that the Requirements Document is of a high quality. According to IEEE standard 830 (1984), a good Requirements Document should be (the description is found in Macaulay, 1996):

- *Complete*
  A Requirements Document should include all the relevant and significant requirements of the system, both functional and non-functional.

- *Consistent*
  The consistency of the document should be ensured, for example regarding use of terminology and similar functionality e.g. the closure button should look the same in different views of the system.

- *Modifiable*
  The document should be organised in a coherent way that makes it easy to

change and use. Examples of such are table of contents, index and cross-referring between related parts.

- *Traceable*
  The traceability between related document should be ensured, e.g. back-ward to source and forward to design and code (see section 4.2.2).

- *Unambiguous*
  The statement of the requirements should be unambiguous, i.e. it should not be possible to interpret the statement in different ways. The same statement should not have more than one meaning.

- *Verifiable*
  It should be possible to verify the requirements, i.e. prove that the requirements is fulfilled. The requirements in the document should be measurable.

- *Usable during development, operation, and maintenance of the system*
  The document should be useable in the whole lifecycle of the information system, meaning that it should be designed in a way that it can serve as a basis for the work that is done during the development, operation and maintenance of an information system.

These characteristics are important so as to see that the Requirements Document can be used in the various situations needed in the development processes. The Requirements Document must reflect a complete and consistent view of the system, which cannot be interpreted in different ways.

3 Requirements Engineering

A common and concise definition of Requirements Engingeering (RE) is yet to be found. But it is widely agreed that RE is an attempt to understand the exact needs of the users of an information system, and to specify these needs into formal and unambiguous statement describing the hypothetical system (Loucopoulos and Karakostas, 1995). One definition, used by both Pohl (1996) and Loucopoulos and Karakostas (1995), describes RE as:

> *"… the systematic process of developing requirements through an iterative co-operative process of analysing the problem, documenting the resulting observation in a variety of representation formats, and checking the accuracy of the understanding gained."*

This definition emphasizes two important characteristics of the RE process. First, RE is a systematic process, which means that there is some regularity and methodically in the activities performed. Second, the activities of RE is performed iterative and the activities involved are preformed almost in parallel, i.e. the activities within RE can be viewed as ongoing micro processes.

The main purpose of RE is to "*transform a fuzzy initial idea into a precise system specification*" (Jarke and Pohl, 1994, pp. 1) which corresponds to the stakeholders needs. Pohl's (1996) three-dimensional framework can be used to describe the meaning of this statement. The framework presents three dimensions of the RE process, in which the work with the requirements proceeds (Figure 2). These dimensions corresponds to three main goals of RE. The aim of the *specification* dimension is to develop a specification of the system, which is as complete as possible. The aim of the *representation* dimension is to providing an integrated formal representation and to support the transformation between different representations. The third dimension is the *agreements* dimension, which aims at accomplish a common agreement among the stakeholders about the final specification.

In the RE process you strive to get from a point (at the lower left corner of the cube) that is characterised by a *personal view*, *opaque* system specification and an *informal*

representation to a point (at the upper right corner of the cube) characterised by a *common view*, *complete* system specification with a *formal* representation.

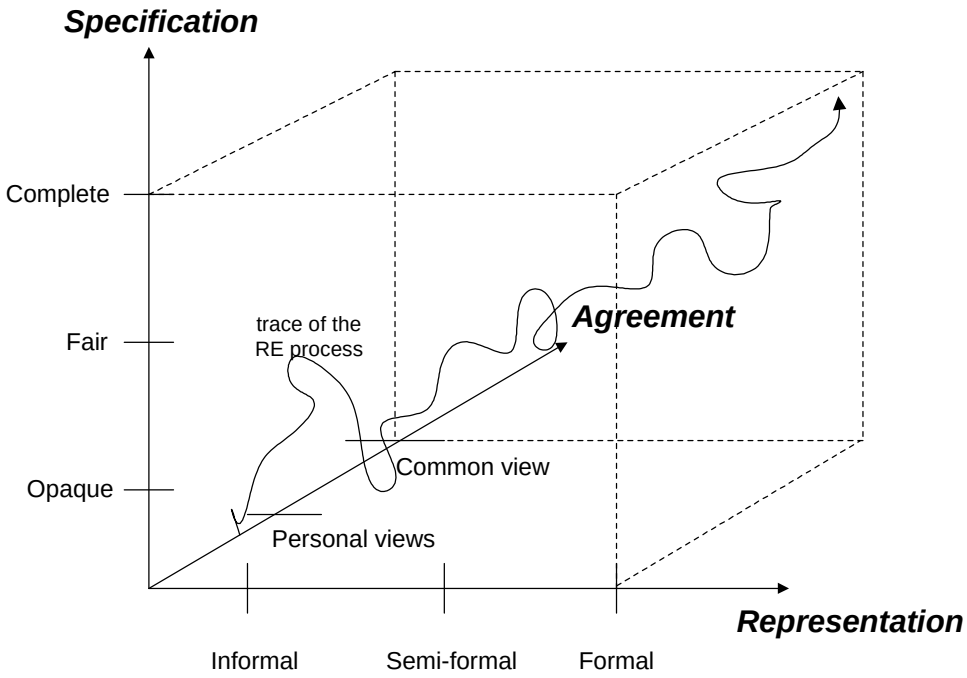

Figure 2 The Three-Dimensional Framework of RE (Pohl, 1996)

The *specification* dimension deals with, as mentioned above, the completeness of the Requirements Document. At the beginning of the process there is usually only a vague understanding of the system that shall be developed, and thus, the specification is more or less opaque. During the progress of the RE process the understanding of the system grows, as the people involved learns more about the organisation, the problem and the stakeholders needs. This also affects the completion of the Requirements Document, i.e. more and more requirements are specified and individual requirements are specified in more detail (see section 2.2.2).

The *representation* dimension is concerned with the language used to represent the requirements of the system. There are three categories of representation: informal, semi-formal, and formal. Informal representation is, like natural language, user-oriented and is used in everyday life. Examples are arbitrary graphic, description by examples and animations. Semi-formal representation is usually graphic languages, which has some form of syntax to be followed but they are still rather easy to understand. These representations are used to visualize certain features of the system. Examples are data flow diagrams and ER diagrams. Formal representation has a well-defined formal

semantics and is more system oriented, e.g. the specification languages VDM and Z. In the beginning of the RE process there are most common to use an informal representation, but as the work proceeds the specification becomes more and more formal, in order to develop a Requirements Document that is unambiguous and consistent.

The *agreement* dimension deals with the degree of agreement on the Requirements Document among the stakeholders. At the beginning of the RE process there are few requirements that are shared among the project team, and many individual requirements that exist only in the personal views of the stakeholders. During the RE process the requirements are analysed and negotiated, in order to develop a number of agreed requirements. The final goal is to have a common agreement on the final specification.

Traditionally, RE involves, at least, three different activities or sub processes: Requirements Elicitation, Requirements Specification and Requirements Validation. Many different authors have described these activities, for example Loucopoulos and Karakostas (1995), Pohl (1996), and Kotonya and Sommerville (1998. Generally, these authors have approximately the same basic view of RE, but the number and characteristics of the activities involved differs. The three activities above corresponds to the major concerns of RE; understanding the problem, formally describing the problem and attaining an agreement of the problem (Loucopoulos and Karakostas, 1995). These activities can therefore be considered as the main activities in RE.

RE has moved from supporting the early phases of individual projects, to supporting the whole lifecycle of information systems in a rapidly changing organisation (Jarke et al, 1994). RE is nowadays also responsible for maintaining the system's requirements over time. One important aspect of this shift is that the need of managing the large amount of requirements, and foremost, dealing with changes in the requirements information increases. Therefore, an additional main activity has been introduced into RE, namely Requirements Management (RM). Thus, RE consists of four different main activities:

- Requirements Elicitation
- Requirements Specification
- Requirements Validation
- Requirements Management

These activities are further described in the following sections.

3.1 Requirements Elicitation

Requirements elicitation is the first activity of RE and continuous during the whole RE process. The aim is to "*acquiring (eliciting) all the relevant knowledge needed to produce a requirements model of a problem domain*" (Loucopoulos and Karakostas, 1995, p.40). To be able to solve somebody else's problem, you first have to find out more about it. Thus, requirements elicitation is about exploring the problem domain and investigates all aspects of the problem and to grow an understanding of the software needed to solve the problem (Pohl, 1996).

Usually, software-related problems are complex enough to have knowledge distributed among many sources and represented in many different ways (Loucopoulos and Karakostas, 1995; Pohl, 1996). It is therefore essential to be able to identify relevant sources and to analyse them properly. Examples of sources are, the stakeholders, documents, standards and laws or the existing system. Not only the source of the knowledge and information vary, the representation does to. It can be, for examples, pictures, descriptions in natural languages, formal models of the domain, or even in mental models in people's minds. It is important that the hidden information and knowledge about the system is made explicit, and is expressed in way that everybody in the process can understand.

3.2 Requirements Specification

The main goal of RE is usually to produce a requirements document, which is as formal as possible. The purpose of this activity is to achieve that goal, and to produce a requirements document. The requirements document do not only include a list of requirements (see section 2.3), but also a set of models that (Pohl, 1996):

- shows the viewpoints of the various stakeholders of the system
- represent both the final requirements document and other result gained during the process
- are traceable and consistent

Thus, in the documentation activity, there exist a lot of different models expressed in various representation form and all of these models have to be kept consistent. In order to perform this, the requirements knowledge gathered during the requirements elicitation task are analysed and assimilated. The knowledge is then synthesized and organised into a coherent requirements model (Loucopoulos and Karakostas, 1995).

3.3 Requirements Validation

The purpose of this activity is to validate and verify the specified requirements, in order to ensure that they are consistent with the stakeholders´ intentions (Pohl, 1996; Loucopoulos and Karakostas, 1995). It is also ensuring that the right requirements are dealt with during the whole RE process. Validation is a cooperating activity and requires interaction between the analysts, the customer, the users and other stakeholders.

Requirements validation is an on-going activity, which is parallel with elicitation and documentation. The need for validation appears when new knowledge or a piece of information is assimilated in the Requirements Document or when information is integrated into a whole. Thus, validation is not only necessary to apply only on the final requirements document, but also at every new piece of information in the RE process, i.e both on the specification and on the intermediate results.

The validation phase can be divided in two main tasks (Pohl, 1996); at one hand, the specification can be validated together with the stakeholders, to ensure that the requirements are the right ones. On the other hand, the specification can be checked for internal consistency, to make sure that there are no ambiguous requirements and information

3.4 Requirements Management

RM is concerned with structuring and maintaining the large amount of requirements and information gathered during the RE process. The major concerns of RM are 1) organising and storing the requirements and 2) managing changes to the requirements. Organising the requirements, is concerned with structuring the requirements and storing additional relevant information about the requirements (Leffingwell and Widrig, 2000; Kotonya and Sommerville, 1998, Karlsson, 1996). One important aspect is to ensure

requirements traceability. Both traditional traceability, as described in Jarke (1998), and dependencies between requirements must be accomplished and administrated. It is of fundamental concern to trace dependencies between different requirements in order to estimate the effect of a proposed change (Kotonya and Sommerville, 1998). One way of storing needed information is by using a requirements repository. This repository may be useful during various phases in the lifecycle of an information system, e.g. as a basis for various decisions or for reuse of requirements (Robertson and Robertson, 1999).

Another important aspect of managing requirements is to manage and control changes to the requirements (Leffingwell and Widrig, 2000; Kotonya and Sommerville, 1998). Managing changes is concerned with making informed decisions whether to implement a requested change or not. The purpose is to ensure that changes contribute to fulfilling the business needs, and to ensure that changes are financially sound, i.e. the benefits of a change are greater than the cost to implement them. Change management of the requirements is also concerned with supporting the identification of which information, e.g. documents and other requirements, that is affected by the proposed change. The aim is to ensure that the requirements and additional information gathered is kept consistent when changes are implemented. This is, for example, important when maintaining the requirements repository to ensure that requirements and information stored in the repository is complete and consistent.

3.5 Discussion about Requirements Engineering

RE is not a linear and straightforward process. The iterations are very fast and the process can rather be seen as a micro process of parallel activities. RE may be considered as a knowledge acquisition process, in which the main problem is to obtain a good understanding of the problem domain, i.e. increase the knowledge within a certain domain (Loucopoulos and Karakostas, 1995). The users are the major source of information about the domain, which giver rise to several difficulties, for example the following:

- It is rather difficult to explain way of working and decision when solving a problem. The users possess a large amount of information that is hard to explain and document.

- Users and developers use different languages, which complicates the communication between these groups of people.
- The developers need to deal with a number of different users, which may have conflicting experiences and needs.

It may, hence, be rather difficult to establish a correct and complete view of the organization. As Pohl's (1996) three dimensions indicates, new information may cause a major decline in the understanding of the domain, e.g. because it is conflicting with previous information and assumptions made during the RE process.

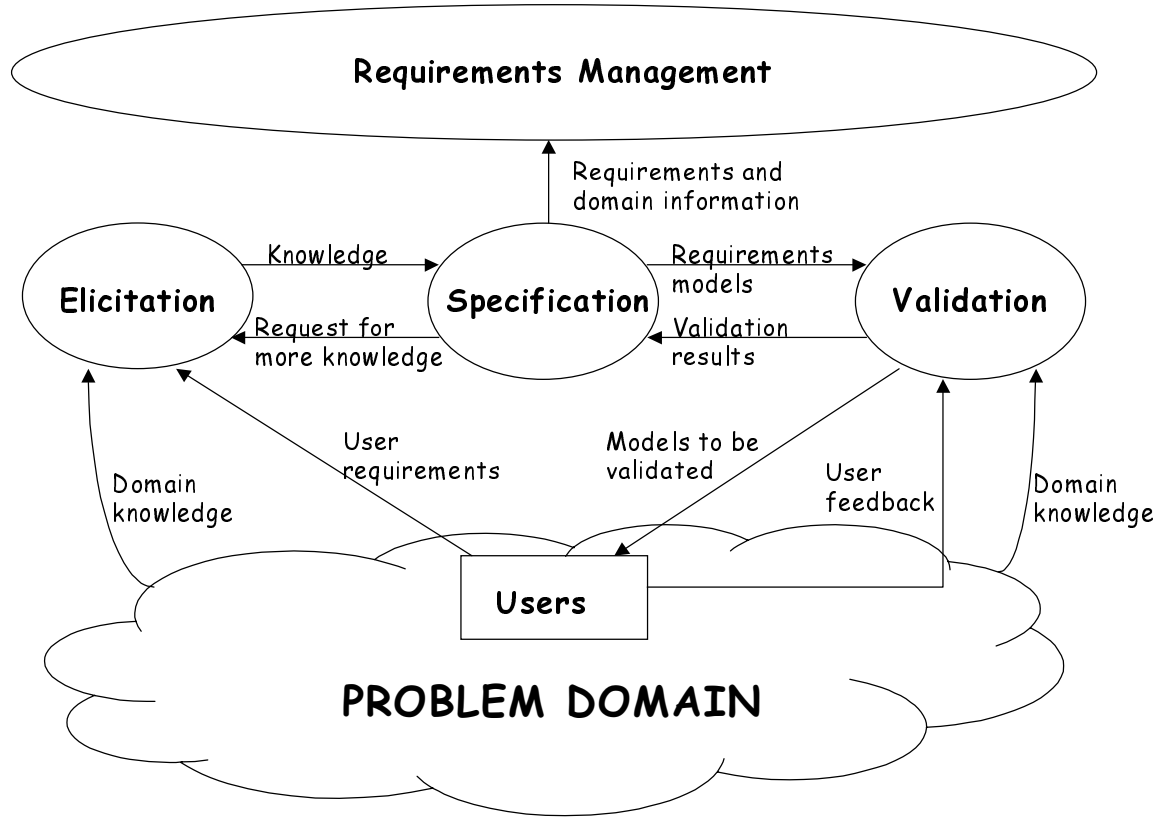

Figure 3: The RE process (from Loucopoulos and Karakostas, 1995)

The figure above describes the interaction between the activities within the RE process. As said before, elicitation, specification and validation are performed in a very fast iteration and can nearly be viewed as performed in parallel. However, before we can specify and validate a requirement, we must first elicit it. Within the elicitation activity, domain information and user requirements are acquisitioned from the problem domain. This knowledge is 'transferred' to the specification activity, where the knowledge is

translated into a representation form, which should be used in the Requirements Documents. The resulting models are then transferred to the validation activity where they are validated with the users. The users' feedback, as well as new domain information discovered is transferred back to the specification activity, where the new knowledge is specified. If more information is needed the elicitation activity begins all over again. The process continues this way until the requirements specification is considered as finished or when the project has to proceed on due to schedule or calculations.

As soon as the first piece of information is gathered the RM process starts, as the information is to be structured and organised. New information that are elicited during the process are organised in relation to existing information, to keep the information structured. Requirements are uniquely identified and listed, and relevant additional information is stored. Dependencies between requirements and other traceability information are documented. RM is concerned with managing the large amount of information gathered during the RE process.

4 Requirements Management

The purpose of this chapter is to make an overall presentation of Requirements Management (RM). First, a definition of RM is discussed, where several authors' definitions are taken into account. Second, the major activities or concerns of RM will also be discussed in more detail, organising requirements and requirements change management. Finally, the chapter is completed with a discussion of RM.

4.1 Requirements Management - A Definition

A common definition of RM is not easy to find. There are almost as many definitions of RM as there are authors discussing it. Fortunately, there is not so many of them. Leffingwell and Widrig (2000, pp. 16) defines RM as:

> *"… a systematic approach to eliciting, organizing, and documenting the requirements of the system, and a process that establishes and maintains agreement between the customer and the project team on the changing requirements of the system."*

This definition takes a rather broad view of the concept, and defines RM as the overall name of requirements work. This definitions of RM corresponds to the one used in the Capacity Maturity Model, described by Carnegie Mellon University Software Engineering Institute (1994). The main part of the activities mentioned in this definition corresponds to the activities within RE (see chapter 3), i.e. elicitation of knowledge and requirements, documentation of the information gained, and validation (*establish* an agreement)(Loucopoulos and Karakostas, 1995; Pohl, 1996; Kotonya and Sommerville, 1998; Macaulay, 1996; Karlsson, 1998; Karlsson, 1996). In this work, we will continue to use the term ‘RE’ to refer to these kinds of activities.

Compared with the definition of RE, this also covers administration of the requirements, namely ‘organising requirements’ and ‘maintain an agreement between the customer and project team’. To maintain an agreement, changes to the requirements must be managed, i.e. changes to the system should be based on an informed and well-founded decision whether to implement the change or not. This requires that relevant information about the requirements is stored and that traceability between relating

requirements is ensured, i.e. that the requirements are organised. We will use this part of the definitions when it is compare with other definitions of RM found.

Kotonya and Sommerville (1998, pp.114) define RM as:

> *"… the process of managing changes to a system's requirements."*

This is a rather narrow view of RM compared to the definition of Leffingwell and Widrig (2000), but it emphasise an important part of RM, managing requirements changes. This definition does not explicitly covers the problem with organising the large amount of requirements and information gathered. But, as mentioned before, this is a necessity to be able to manage changes effectively. Especially storing traceability information, are essential for managing changes to requirements and thus an important part of the RM process.

Karlsson (1996) believes that all activities that are needed to handle and manage the requirements during the whole development process, is a part of RM. This includes:

- Configuration Management
- Change management of requirements
- Influence analysis of changes

This definition is quite general, and do not give a clear view of what RM is, unless the examples are used as a part of the definition. A more clear and descriptive definition is needed to give a view of what RM really is. As the other two definitions, this also mentions change management and indicates that this is a fundamental concern of RM. In this definition Karlsson (1996) delimit RM only to the development process. The basic assumption in this work is that it is important to maintain the requirements during the whole lifecycle of an information system.

Karlsson (1996) also mentions Configuration Management, which is a well-known activity within the Software Engineering community. There exist similarities within these two areas, e.g. both are keeping track of changes to requirements, and which requirements that are implemented in a certain version of the system. But there are also activities/tasks in RM that are of no concern in Configuration Management and vice versa. The main difference is the main focus of the activities respectively. Configuration

Managements is mostly focused on the system as a whole, and to dealing with changes to whole documents e.g. the requirements documents. RM is focused on managing individual requirements and to ensure that all the requirements are stored and updated in case of change.

To summarize the discussion above, RM can be defined as:

> *The systematic process of organising and storing additional relevant information about requirements, while ensuring requirements traceability, and managing changes to these requirements during the whole lifecycle of the information system.*

From this definition, two main activities (or major concerns) of RM can be identified: organising the existing requirements and managing requirements change.

4.2 Organising Requirements

There are a lot of requirements and other information gathered during an information systems development process. When developing a small software product for purchasing there are approximately over 1000 requirements involved. To be able to get an overview and control of such a project, you must in some way organise the information gained. There is a need to manage these requirements and other relevant information gathered (Leffingwell and Widrig, 2000) to be able to keep a updated and consistent documentation of the requirements. This does not only involve structuring and storing additional information that descries the requirements, but also ensuring requirements traceability. One way of storing all this information is to use a requirements repository. This includes all the captured requirements and the additional relevant information, such as requirements attributes, documents, and models. Such a requirements repository may be useful during various phases in the lifecycle of an information system, e.g. in maintenance of the system or for reuse of requirements (Robertson and Robertson, 1999).

Organising requirements hence involves structuring the requirements information (this section mainly focus on requirements attributes), ensuring requirements traceability and the use of requirements repositories/databases.

4.2.1 Requirements Information

Additional information about requirements can be of various types: e.g. graphical models, text descriptions, and requirements attribute. This section focus most on the use of requirement attribute to describe single requirements in more detail. Requirements attribute are assigned to the requirements in order to give some guidelines for trade-offs made during the future development work (Dorfman and Thayer, 1990). The attributes are also used as basis for various decision made during the development and maintenance of the system.

To be able to store information about a single requirement, some kind of unique identification is needed for each requirement. This requires that the requirements are explicitly stated and represented. There are a lot of different approaches how to identify requirements, e.g. dynamic numbering, database record identification or symbolic identification (Kotonya and Sommerville, 1998). No matter what method that is used, it is important that the way of identify the requirements allows easy reorganisation and change of the order.

The attributes stored about the requirements differ from project to project, depending on what is relevant for a particular project. This is indicated in Carlshare and Regnell (2000), where two RE processes are described and compared: REPEAT and RDEM. The processes are state-oriented rather than traditionally phase-oriented. In both processes attributes are stored about the requirements, but they differ between the two models. The general differences between the models are the driving forces behind the processes and these affects the attributes stored. REPEAT is focused on ‘time-to-market’, while RDEM is more focused on quality. The attribute of REPEAT is mainly product-oriented and the attribute of RDEM has a stronger focus on quality issues. REPEAT also had fewer attributes, due to the ability to keep it simple and allow quick actions.

Kotonya and Sommerville (1998, see Appendix A), Leffinwell and Widrig (2000, see Appendix C), Stevens et al (1998, see Appendix B) and Karlsson (1996, see Appendix D) present a number of conceivable attributes of requirements. None of them claim to give a full set of attributes, always needed and valid for all situations. They give examples of typical attributes that they think are commonly attached to requirements.

Of course, some of the attributes correspond to each other, but it is surprisingly few. One reason for the different types of attribute presented in the literature used may be the different focuses of the sources.

Stevens et al (1998) are concentrated on the area of Software Engineering, and describe attributes mainly related to control design and implementation related issues. Examples are: performance, urgency and verifiability. Kotonya and Sommerville (1998) manly focus on describing RE and discuss the change management process also. The attributes discussed are mainly administrative or related to the change management process e.g. statement, rational, dependants and date_changed.

Non of the attributes presented in Appendix A-D are mentioned by all four sources (see Appendix E), but the most frequent mentioned attributes are:

- *Identifier*: Identifies the requirements uniquely.
- *Source*: The people or document that are the source of the requirement.
- *Status*: Present the current state of the requirement.
- *Priority*: Shows the importance of the requirement.
- *Stability*: The probability that the requirement will change.
- *Links*: Documents relation between requirements.

Other attributes (see Appendix E) also mentioned by more than one of the authors above are:

- *Statement*: Description of the requirement
- *Target release*: Shows which product version the requirement shall be implemented in.
- *Target group*: Lists the people needed the requirement.
- *Configuration*: Shows the change(s) that have been made to the requirement.
- *Verification*: Comprises different aspects concerning the verification of the requirement.

The most frequently mentioned attributes are mainly of an identifying and descriptive characteristic, i.e. general attributes. Only a few of the attributes above are more related

to special focuses of the author, for example stability and configuration. This is not a surprise, because the more general attributes forms a basis for understanding the nature of the requirements. More specialised attributes, related to the fundamental concerns of the organisation are often used as a complement to, or ‘on top of’, the more general attributes.

If there exist some general attributes needed in almost all projects are not investigated, nor have any investigation about this been found. It would be interested to perform such an investigation and extend it with trying to identify different factors of organisations that affects which attributes that are needed.

In an attempt to structuring the attribute presented in Appendix A-D, they have been summarised into three major categories. As Karlsson (1996) the attributes are categorised into administrative and technical attributes, but they are used in a slightly different way. In this work, the attributes used to manage changes are considered rather important. Therefore, attributes concerning changes and change management are divided into a separate category. (The category ‘administrative attributes’ are quite comprehensive and maybe it should be refined, e.g. in general and control attributes.)

- *Administrative attributes:*
  This kind of attribute identifies and describes the requirements. They may also be attributes that describe the progress of the project or is to be used as support for decisions making about the requirements. Examples of attribute are: identifier, statement, rational, status, priority, risk, assigned to, acceptance criteria, keywords and comments.

- *Change management attributes:*
  Supports the management of change to the requirements, including the traceability information, e.g. date_entered, date_change (configuration), source, effort, stability, target group, target release, dependants, is_dependent_on, model_links, and structure.

- *Technical attributes:*
  Concerns the implementation of the requirement, e.g. performance, urgency and verification.

Of course, these categories are not absolute, and there are attributes that may be related to several categories, e.g. source can be related both to administrative and change management attributes, and acceptance criteria to administrative as well as technical attributes. However, it provides an understanding of the different kinds of attributes that may be used in RM. Note that some of the requirements attributes mentioned are related to requirements traceability (see section 4.2.2).

This way of storing attributes describing the requirements are mostly focused on explicit requirements, where the requirements can be uniquely identified. How to manage models and descriptions are not dealt with in this work. Implicit requirements, e.g. information in models or text description, cannot be uniquely identified because the requirements are not explicitly stated. How to manage graphical models/schemas should be looked more deeply into. There is also a need to map explicit requirements with information or related requirements in models and text descriptions.

4.2.2 Traceability Information

Requirements traceability is a critical important for assessing the impact of a proposed change (Kotonya and Sommerville, 1998) and to ensure consistency within requirements document and system development process. Traceability is a prerequisite for efficient management of the system's requirements. Requirements traceability is defined as the "*ability to describe and follow the life of a requirement, in both forward and backward direct, ideally through the whole system life cycle*" (Jarke, 1998, pp. 32). Traceability information can be requirements sources, information about requirements dependencies, realization object and links to models (see section 4.2.1).

Four types of traceability links has been defined (see Figure 4) (Kotonya and Sommerville, 1998):

- *Backward-from traceability:*
  Links the requirements to the source in the domain. It can be to other documents or to people asking for the requirement. This can be crucial for validation of the requirement.

- *Forward-from traceability:*
  Links the requirement to the realization object, e.g. a design and implement

component, a document or a manual. The responsibility for achieving the requirements is assigned. This supports the estimation of effort if a change proposal of the requirement is analysed.

- *Backward-to traceability:*
  Links the realization object back to the requirement, which it realizes. Supports the verification of the system and helps avoiding implementation of object for which there are no requirements.

- *Forward-to traceability:*
  Links other documents and models which has precedes the specified requirements to the requirements respectively. Helps keeping track of changes in stakeholders needs, which, for example, can require reprioritisation of the requirements.

Kotonya and Sommerville (1998, pp. 130) also describe a number of more concrete traceability links between specific information. Examples of these links are: requirements-sources, requirements-rationales, requirements-architecture, requirements-design and requirements-interface.

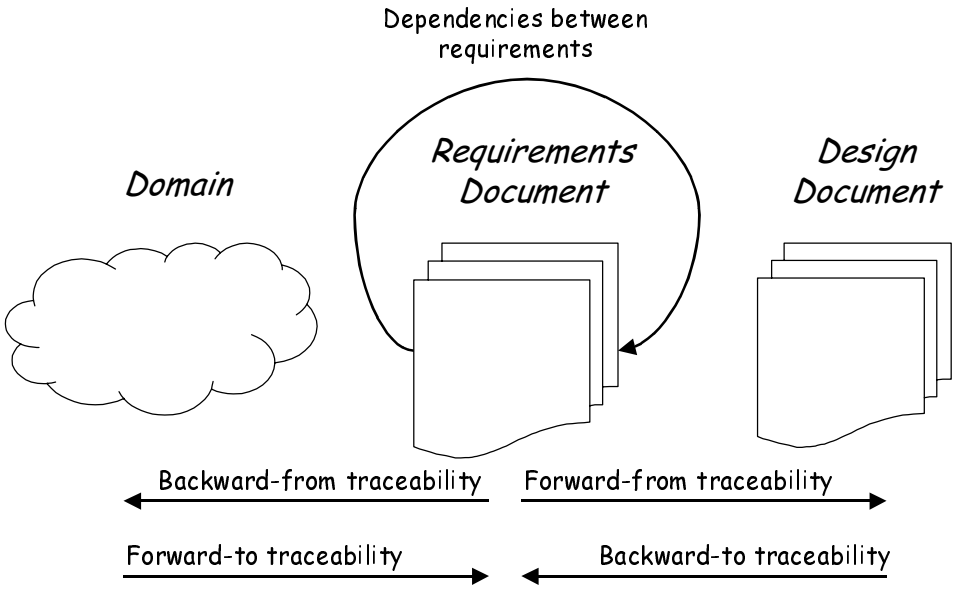

Figure 4: Different traceability links (from Kotonya and Sommerville, 1998)

The four links, described in Figure 6, do not cover one of the most important traceability information from a change assessment point of view (Kotonya and

Sommerville, 1998). It is also important to trace the dependencies between requirements, thou these have impact on how the requirements affect each other. This is of fundamental concern when an impact analysis of a proposed change is performed. Carlshamre and Regnell (2000) mention that these dependencies have an important functionality in connecting fragments of the collected information. They describe some different dependencies between requirements, and present a matrix, which classifies the dependencies in dimensions of scope and explicitness (see Figure 5).

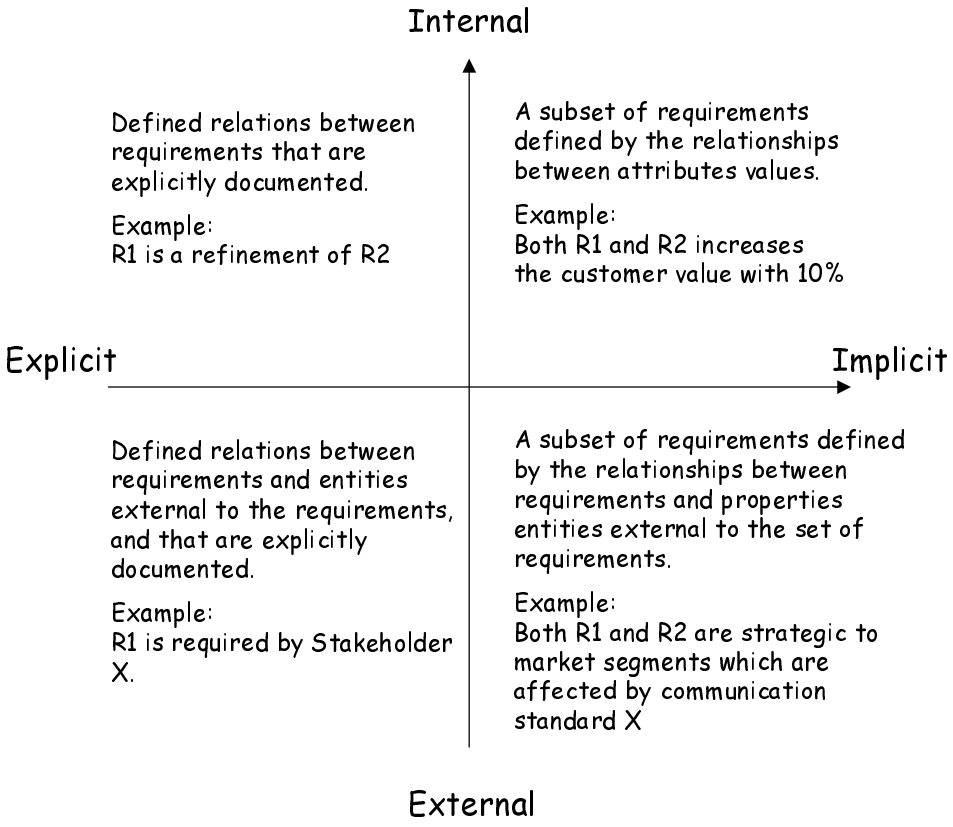

Figure 5: Two dimensions of dependencies between requirements (Carlshamre and Regnell, 2000)

The scope dimension is divided in internal and external dependencies. *Internal* dependencies are direct relation between requirements and *external* dependencies are between requirements and entities external to the set of requirements (e.g. stakeholders and document). The explicitness dimension is divided into explicit and implicit dependencies. Explicit dependencies indicate that there is a defined representation of the dependencies, i.e. that there is an explicit link between them. Implicit dependencies are dynamic and can only be referred from the additional information elicited and documented.

Carlshamre and Regnell (2000) also describe additional dependencies, such as: temporal (R1 must be implemented before R2), logical (R1 can not be implemented without R2), and value-related (the cost of R1 will increase linear with the cost of implementing R2). These are only some examples of dependencies, and there ought to exist more. Also in this area, more research is needed about which kinds of dependencies there is, and what aspects that affect which dependencies that are important in a certain situation. How these dependencies should be documented and maintain should also be looked into in more detail.

This traceability information described covers a very large volume of information. In practice, it is unfeasibly expensive to collect, store and manage all this kinds of information to ensure traceability. Especially the initial cost is very large, but the work with maintaining and updating the information is also expensive. Kotonya and Sommerville (1998) state that a project manager should define 'traceability policies', describing the essential traceability needed and that must be maintained. The policy should include, not only the information needed but also techniques used, descriptions of how to collect the information and ensure that the traceability information is updated when changes occurs.

4.2.3 Requirements Repository – A Database of Requirements

There exist different kinds of medium for storing the requirements. One usual way is to use a word processor to store the requirements in one or more files. The requirements are then stored together at one place, and are easy to access. New versions of the requirements document are easy to create. Most small and medium sized organisations store and maintain their requirements this way. However, there are several problems with this way of storing the requirements (Kotonya and Sommerville, 1998):

- There are several people involved in documenting the requirement, and therefore have to have access to the files. This might cause sharing violation problems, e.g. that two persons try to update the same document at the same time.
- Information concerning requirement dependencies and requirements traceability has to be externally maintained.

- It is not possible to electronically link requirements to other documents, like change proposals, meeting protocols, decisions and models.
- It problematic to search for certain requirement or to find sets of requirements with common characteristics. You are limited to the search facilities provided by the word processor.
- It is not possible to navigate automatically between related requirements and to documents related to the requirement e.g. models, decisions and change proposals.

An alternative way is to use a database to store and maintain information about the requirements. In a database, related requirements can be linked and it is often possible to formulate queries to identify sets of requirements. There is usually some form of access control, to keep the stored information consistent. From a database, reports of different kinds can be created e.g. Requirements Documents. Which approach to use depend on the situation, e.g. type of project, organization etc.

The requirements databases should, of course, contain all requirements elicited, but also the additional information that is elicited about the requirements. Figure 6 shows some different kinds of information that a requirements repository may include.

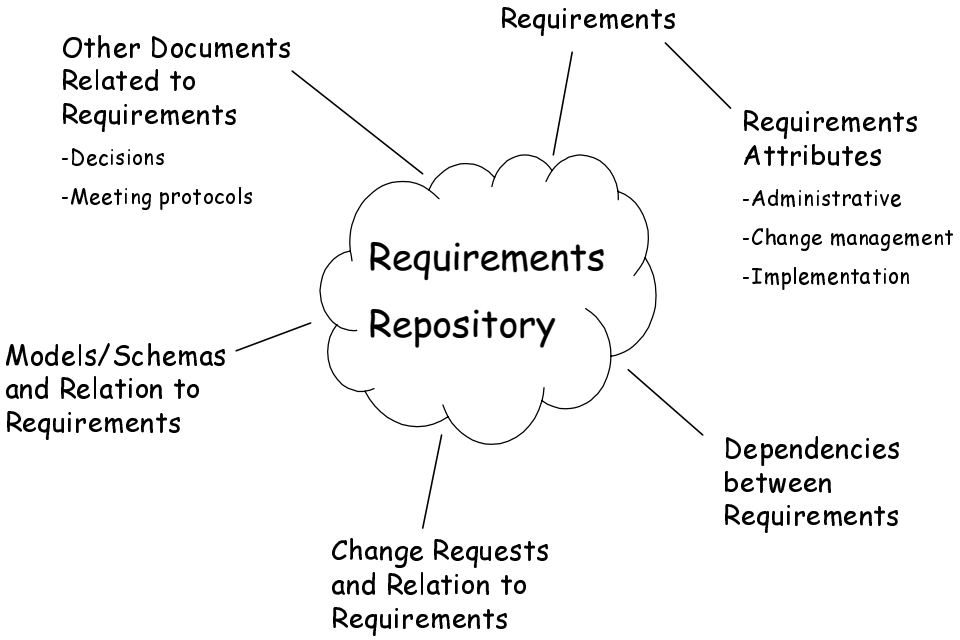

Figure 6: The different kinds of contents in a requirements repository

The *requirements* are uniquely identified and other related *attributes* are associated to each requirement. Which attribute to store are decided depending on what is relevant and important for the specific RE process of each organisation. Traceability information is partly included in the change management attribute group. However, the database may also include *dependencies between requirements*, which may be stored as requirements attribute or in specific requirement matrixes. This type of information is discussed in section 4.2.1 and 4.2.2 above.

There are other kinds of information gathered during the RE process. It may also be important to store and relate this information to the requirements. Proposed changes to requirements, i.e. *change requests* (see section 4.3), and which requirement(s) the request concerns may also be information that is important for some organisations to store. Many organisations develops *models/schemas* during the RE process that describes the current organisation and how the organisation should work when the new system is installed. The relations between these models/schemas and the requirements may be important to store in order to quickly identify what information to update according to a requested change and to keep the requirements information consistent. Some organisations may also find it important to store *other documents* such as decisions and meeting protocols and their relationship to the requirements, in order to be able to identify why certain decision about requirements may have been taken.

What information to store about the requirements differs between RE process and organisations (see section 4.2.1). The major part of the requirements information is used as a basis for further development work such as design and implementation, for example the requirements and models/schemas. The additional information stored is mainly used as a basis for decisions that is to be made during the RE process. Depending on what decision organisations have to take, different types of additional information are needed. Examples of decision are about requirements change (Kotonya and Sommerville, 1998) and within version planning (Carlshamre and Regnell, 2000). A good starting point for further research regarding additional requirements information needed is to investigate what decision that is made during the RE process and what information that is needed to make these decisions.

Today there exist some different tools on the market to support management of large amount requirements information gathered during the RE process. These tools provides

functionality for storing and retrieving information, as well as change management functionality to record change requests and link these to the affected requirements. A RM tool may also be able to keep track of the status of requirements and requirements history (Kotonya and Sommerville, 1998). Examples of RM tools are: RTM Workshop (Integrated Chipware, 2001), DOORS/ERS (Telelogic, 2001), and Requisite Pro (Rationale, 2001).

Figure 6 shows different kinds of requirements information that may be included in a requirements repository. Figure 7 shows the facilities of a requirements tool, i.e. what functionality that may be included into an RM tool.

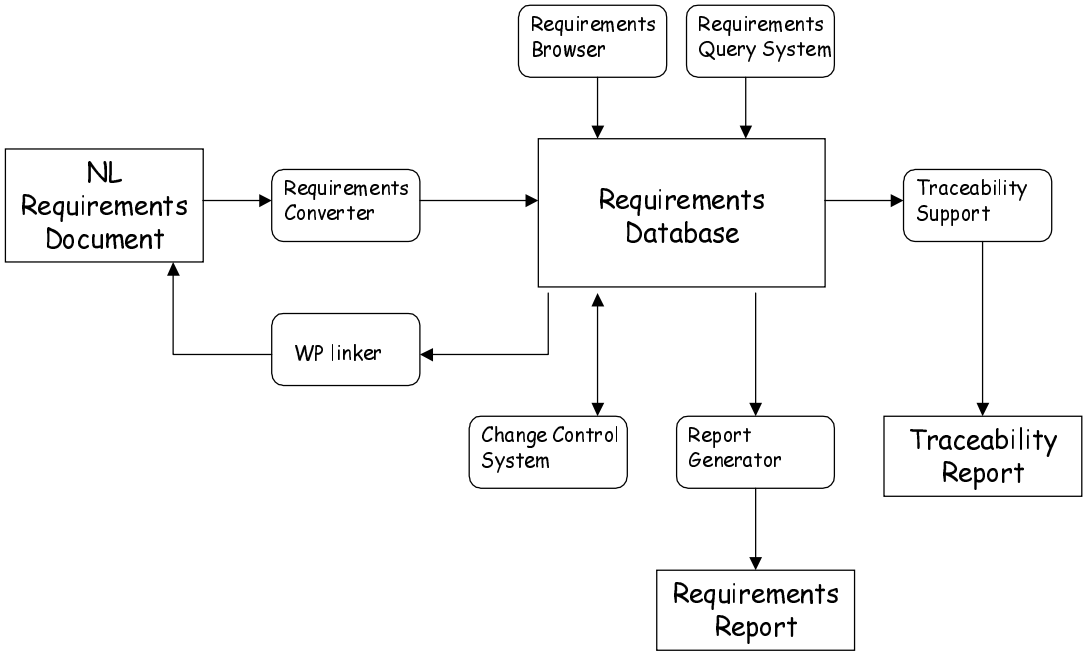

Figure 7: Facilities of RM tools (Leffingwell and Widrig, 1998)

A RM tool collects the requirements and additional information in database or a repository and provides several different facilities for accessing the information stored (Kotonya and Sommerville, 1998). Usually, these tools support the cooperation with word processing system. Meaning that they support the *conversion* of natural language (NL) requirements into a database format and maintenance of *links* between the database and NL representation of the requirements. RM tools may also includes the following facilities to access the data in the requirements database:

- *Requirements browser*: enables stakeholder and developers to browse the requirements database.

- *Requirements query system*: enables stakeholders and developers to retrieve specific requirements or, for example, relation between requirements.

- *Traceability support*: support generation of traceability information.

- *Report generation*: can generate various types of reports about the requirements, e.g. all requirements concerning a specific task.

- *Change control*: support management of requirement change, e.g. by identifying what requirements that are affected by a certain change request. It should also support the update of all affected information to keep the requirements information consistent (see section 4.3).

RM is not only concerned with organising and structuring the requirements information. An important part of RM, which is indicated in previous discussions, is to maintain this information during the whole lifecycle of an information system, while requirements changes occur. This is the concern of the second major activity within RM, requirement change management.

4.3 Requirements Change Management

There are few information systems development project, where no requirements change during the life cycle. These changes often reflect changes e.g. in the environment in which the system shall be, or is, installed, or in the users and customers needs. It is important to take these changes under consideration, because they may affects how well the system fulfils the needs and expectations of the users and customers. A systematic approach to manage changes is also important to maintain a consistent and complete documentation of the requirements during the whole life cycle of an information system.

To introduce new requirements or changes of existing requirements, a clear and straight decision process is required (Hjelte et al, 1995). The proposed changes should be discussed and a decision based on the cost and benefit of the proposed change should be

made. The risk otherwise is that low priority change are made instead of high priority changes, or that changes are made unnecessarily, in worst case, expensive changes. To be able to make efficient decisions about requirements changes, it is of fundamental concern that relevant and correct information about the requirements exist, and that the relation between requirements is documented.

The purpose of change management is to make sure that the changes contribute to the business needs of the organisation, and that they are economically sound, meaning that the benefits of the change are higher than the cost of implementing it (Kotonya and Sommerville, 1998). It ensures that for every change proposal the same decision procedures are used and that similar information is gathered. The organisation may use a change policy to ensure a consistent approach to managing requirements change. These should cover aspects like, the request process and information needed to be gathered, how to analyse the requests regarding cost and impact, who should make the decision and software support that should be used.

The requirements can be affected and changed in several different ways. New requirements may occur, improving suggestions can be received from the customer or trouble reports describing problems with the system or development of the system may arise (Karlsson, 1998). One important driver for adapting systems is new requirements, which requires new functionality or new quality characteristics that needs to be met (Galal, 1999). Chances to requirements can, hence, be that new requirements are discovered, or that existing requirements are changed. They can also be connected to the additional information stored in relation to requirements. Of course changes to requirements affects this additional information, but also other information documented, such as models/schemas, design documents and code files.

Both Kotonya and Sommerville (1998) and Leffingwell and Widrig (2000) describe a process for managing changes to requirement. However, the extent of these two processes differs from each other. According to Kotonya and Sommerville (1998) change management involves three tasks including documenting, reporting, analysing and implementing requested changes to requirements. The three tasks are the following:

- *Problem analysis and change specification*
  This task is initiated when a problem with requirements is discovered. The

problem could have been identified by analysis of the requirements, new customer demands found or operational problem with the system. This task involves analysing the identified problem and formulation of a change proposal.

- *Change analysis and costing*

  In this task the proposed changes are analysed and a rough estimation of the cost in time and money is made. First, the change proposal is checked to control that it is valid. Unnecessary changes, proposed because of misunderstandings, must be avoided. Second, the related requirements also affected are identified by the traceability information stored about the requirements. Third, with this information gathered as a basis the change proposal is further analysed. Consolation with the stakeholders might be necessary. Fourth, the cost of the proposed change is estimated. This estimation includes the effort to make the change and the time needed. The available resources for the work are also investigated. Finally, there is enough information about the change proposal to make a decision. This decision is made in cooperation with the customers, where the need and cost of the changes is discussed. I might be necessary to form other alternative changes to solve the requirements problem identified.

- *Change implementation*

  The agreed change proposal is implemented and affected documents and work is changed. This might affect the Requirements Document, the system design and even the finished system, depending on where in the lifecycle the change is managed. The implementation of the change must of course be validated using the normal instrument for quality control used in the project.

There are three stages in the process where a change may be rejected. *First*, during the validity check where the requirement might be found invalid. A normal reason for this is that the stakeholders might have misunderstand the requirements and therefore considered it wrong. The change is not really necessary. *Second*, the change proposal causes other changes, which the stakeholders finds unacceptable. If the change is implemented, other parts of the system must be changed, which causes more trouble for the users than the original requirements problem identified. *Third*, the cost of the changes is too high and/or takes to long time. The benefit of the changes is lower that the cost of implementing the proposed change.

Leffingwell and Widrig (2000) have defined a five-step process for managing changes to requirement. This process includes the following steps:

- *Recognize that change is inevitable, and plan for it*
  Changes will occur in the project, to some extent and the team should develop a plan for managing changes to requirements. Change request from the developer team are legitimated as well as change request form the users. The users have real needs that must be fulfilled by the system and the developer team has a deep understanding of the system.

- *Baseline the requirements*
  At the end of the RE process the development team should baseline all known requirements, i.e. the involved documents are put into version control. This means that the development team has a formal base for the remaining of the development process. This steps enables the development team to distinguish between know and existing requirements, from new requirements, or those being deleted or modified. When the baseline is established, changes to requirements can more easily be identified, analysed and managed. If a change is accepted, it is easier to manage the implementation of that change, from updating documents till changes source code.

- *Establish a single channel to control change*
  It is crucial that every requirements change go through one single channel, to be managed by the formal change control system defined. This channel may be a single person responsible for RM or a formal group of people with different views of the system, and who share the responsibility of managing requirement change. Changes to requirements must be properly managed, because seemingly simple changes can have unanticipated affects on the system. Changes to requirements may also affect the schedule and budget of the project and the decision to implement the requested change or no must be taken on the base of these facts.

- *Use a change control system to capture changes.*
  When a change occurs, an analysis of the change is required to identify the consequences of the change as well as the cost and benefit of the change. A

decision whether the change should be implemented or not must also be taken. Thus, the development team should implement a formal method for capturing all requested change to the system. This change control system works as a 'firewall' between the different stakeholders and developers, which may posing change request, and the documents produced during the development process, e.g. requirements specification, design document, and test cases. All change requests must be transmitted to the person/group responsible for the change management. They are responsible for making a decision whether to implement the change or not, to notify all people affected by the change, where the change should be inserted, and that the change are properly inserted.

- *Manage change hierarchically*
  A change to one requirement may have a 'ripple effect' on other requirements or documents produced during the development process. When a change shall be implemented, all effected requirements and documents must be updated according to the proposed change, in order to keep the information consistent. This means that the requirements information must be changes as well as design documents, code files and test cases. Changers should be implemented from 'the top' starting with the requirements and continuing with design, code and test in stated order.

The fact that we have a process to control changes, does not mean that a large number of changes can be implemented or that change management can be used as some form of compensation for an ill-performed RE process. However, it is important that the development team acknowledge that some changes may be important and beneficial and that changes must be allowed if they are. The problem is if the changes are implemented in an ad-hoc way without a proper analysis and proper implementation.

Both Kotonya and Sommerville (1998) and Leffingwell and Widrig (2000) emphasises that it is very important that changes to requirements are properly managed. Nevertheless, the two processes differ from each other, mainly in the extent of the process. Leffingwell and Widrig include activities such as planning, establishing change management team and a channel for change requests as a part of the change management process. Kontonya and Sommerville are more focused on the process of manage incoming change requests, e.g. analyse and decision-making. Their entire

process can be viewed as a more detailed description of step 4, and in some extent, step 5 in Leffingwell and Widrigs process.

In this work, we acknowledge that planning and other issues that enable a proper management of requested change is an important part of requirements change management. Leffingwell and Widrigs process is, hence, viewed as a more overall description of the whole change management process, and that Kontonya and Sommerville provides a more detail description of the change control system, i.e. more concretely how the requested changes and the impact analysis should be administrated. However, both this processes describe are rather general, and are more concerned with describing what should be done rather than how this should be accomplished.

Formal change management process may take a lot of time, but are necessary in order to ensure that changes corresponds to business needs and that they are economically sound. Ad hoc implementation of changes must be avoided. The processes presented above are rather general, and more detail information about how informed decision about changes can be made. This includes investigate how the process should be performed as well as what information that is needed. More research about how this process can be made more efficient is also needed.

4.4 The Life Cycle Perspective of Requirements Management

As stated in chapter 3, RE no longer performed only during the first phases of project, but an important activity that aims at supporting the development as well as the maintenance of information systems. Managing requirements during the whole lifecycle is an important part of that change. RE is now also concerned with maintaining the requirements during the whole development process as well as during operation and maintenance of the system. Allowing and managing changes to requirements is essential for adjusting the system to dynamic organisation and changing environments. If the system are not changed when the organisation changes, the system can no longer support the business processes of the organisation efficiently. To be able to manage changes to requirements in an efficient way, requirements and requirements information must be organised and maintained. RM enable the RE process to support the whole life cycle of an information system.

Generally speaking, the life cycle of an information system includes analysis, design, implementation and operation and maintenance (a simplification and summary of Avison and Fitzgeralds (1995) traditional Systems Development Life Cycle). In the *analysis* phase a thorough investigation of the problem domain is made, in order to increase the knowledge about the problem to be solved, and the solution. During this phase the requirements of the new information system are developed. The requirements are based on investigation of the organisation as well as current information systems and problems with those. During the *design* phase the solution are outlined in more detail. Input and output data of the system function, structure of the data files, and system testing and implementation plans etc. are developed based on the knowledge gained in the analysis phase. The *implementation* phase includes, for example, programming, software testing, and purchasing and installation of new hardware and software. One major aspect is quality control, all aspects of the system, manual as well as computerized, must be proven before the new system is taken into operation. Finally, the system is installed into the organisation. These three phases are part of the development process and is represented by the first box in the figure below.

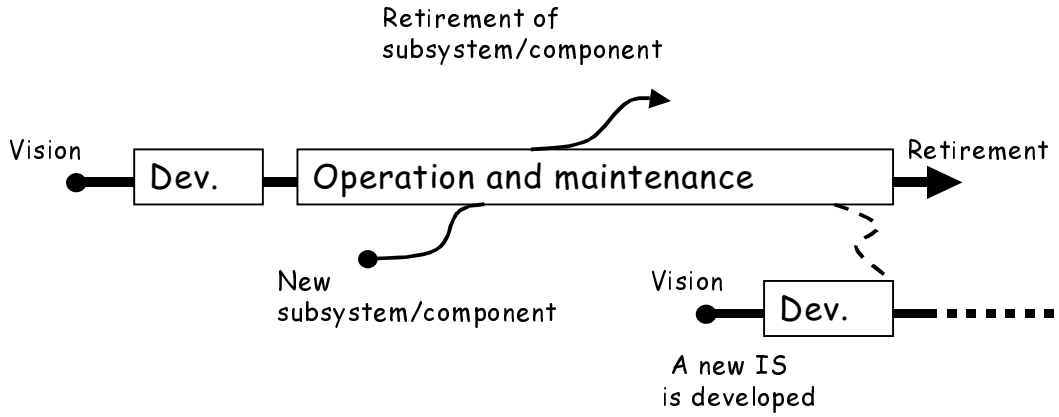

Figure 8: The life cycle of an information system.

When the development phase is finished and the system is installed in the organisation, the extensive *operation and management* phase begins. During the phase the system is continuously maintained to ensure that it runs efficiently. Changes may be necessary, in order to adapt the system to changes in the organisation and its environment, technological advantages, or rules or regulations. Part of the maintenance work is related to correcting errors made in the development process. Maintenance may also include development and installation of new subsystem/component or retirement of

existing subsystems/components. During this phase the system will also be reviewed to control that it still fulfils the requirements elicited during the development of the system. When the system no longer fulfils its purpose, the system is retarded and a new system is developed - the SDLC starts all over again.

RE should, hence support the system during the whole life cycle presented above. RM is mainly about organising and structuring the requirements, and manages changes to these, which includes both making informed decisions and update the effected information. Figure 9 illustrates this aim.

RM enables reuse of requirement both when new information system is developed and when new subsystem/components. Reuse of the existing requirements reduces cost and avoids specification errors (Pohl, 1996). The existing information system, or its lack of fulfil the requirements, is an important source to develop the requirements of the new information system. If the requirements specification shall be a useful source, it requires that it is consistent and unambiguous, and corresponds to the current state of the system.

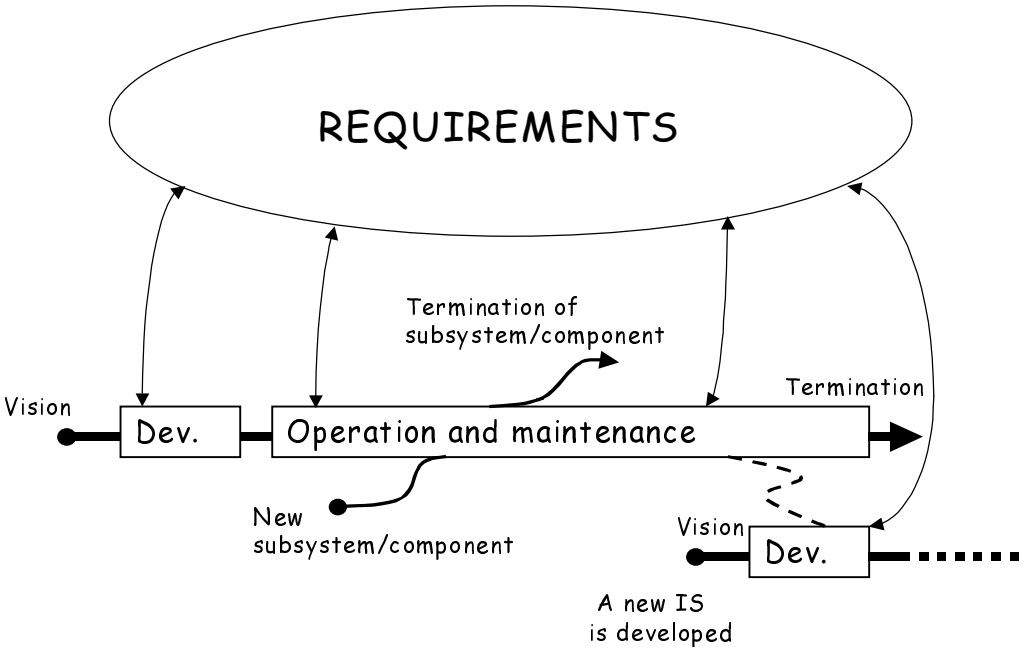

Figure 9: RM from a life cycle perspective.

RM also enables the maintenance team to make informed decisions about requested changes during the operation and maintenance phase, as well as for the development team during the development process. However, this requires that the information about requirements and traceability information is updated.

5 Requirements Management in Practice

The fourth objective of this work is to investigate how RM is conducted in practice. By making qualitative interviews with professionals within system development this issues has been looked more deeply into. Besides a summary of the interviews this chapter includes description of the interview structure, the companies, and the respondents. The chapter ends with an analysis of the interview material and conclusions.

5.1 Interview Structure

The purpose of the interview was to achieve an understanding of how system development practioners manage their requirements. In order to enable deep interviews of how each company and interviewee are working with their requirements a small number of interviewees were chosen. The criterions used to select interviewees were as follow.

a) The interviewee should work in a company that are dealing with some form of software system (e.g. information systems, COTS, or embedded software).

b) The interviewee should be involved in working tasks that includes dealing with requirements.

c) The companies should preferably perform different kinds of software development, in order to get a more widespread knowledge of the area.

The companies were identified mainly by connections of the author and/or colleagues of the author. This made it easier to ensure, in advanced, that the interviewees and the companies fulfilled the criterions above. Three different companies were chosen, and five people interviewed (for more information about the companies and interviewees, see section 5.2). The first company is configuring COTS system according to their customer's requirements, the second company develops products that include embedded software, and the last company develops software products within the ERP sector.

Since the interviews were intended to be of a qualitative characteristic, open questions which purpose were to enable discussions between the interviewer and the interviewees, were formulated. Since the purpose were to develop an understanding of how the

interviewee woks with requirements, the questions were of an ‘open’ characteristic in order to enable additional questions. The interviews were conducted in shape of a conversation between the interviewer and the interviewees, and the questions were adjusted to the interviewee and his/hers answers of the previous questions.

The questionnaire started with some personal questions in order to give a view of the competence and experience of the interviewee. The second part of the questionnaire was focused on more general questions about the requirements engineering process of the companies respectively. This part was followed by more specific questions about how requirements were organised and managed during the whole life cycle, and how changes were dealt with. The fourth part was focused on overall questions about configuration management and how it is related to requirements. The questionnaire was concluded with a question about the major problems with requirements work, according to the interviewee. The entire questionnaire can be found in appendix F.

The interviews were conducted during visits at the companies respectively. The discussions were recorded with a tape-recorded, to ensure the answers as a whole were captured. Each interview lasted for one till two hours.

5.2 Companies and Interviewees

The two following sections aim at providing an overview of the types of companies involved in the literature study and the competencies of the interviewees. The sections are, hence including brief descriptions of the three companies and the five interviewees.

5.2.1 A Brief Description of the Companies

Company X is a large international organisation within the manufacturing industry. It sells its products in over 100 different countries all over the world. Software is embedded in most of Company X’s products. Interviewees from two different departments have been interviewed. Two interviewees, A and C, work at a department that develop a certain part of Company X’s products. This part includes embedded software. The other department, to which interviewee B belongs, ensures the electricity supply between all parts of the product. This includes ensuring communication between the different parts of the product and software is an important support for this task.

Company Y is a software product development company, which aims at developing support for the business processes in different kinds of organisations. The company is one the world's tenth largest ERP vendor, and it has more than 3000 employees. Company Y is working with development of the software applications as well as implementation and adjustment of these packages to the customer's organisation.

Company Z is a large international organisation within the telecommunication industry. They develop everything from systems and applications to communication tools. Company Z is active worldwide and operates in more than 140 countries. It has approximately 5 000 employees. Interviewee E works at a department[2] that configures COTS system according to customer's needs. The COTS systems are bought from different vendors and Company Z are responsible for the adjustment of the system according to the customer's requirements.

5.2.2 A Brief Presentation of the Interviewees

Interviewee A is a project leader at Company X, within the same department as interviewee C. The projects at this department are approximately in size of 15 peoples, and are dealing with the whole development process, from idea and high-level requirements, to design and construction. Interviewee A has worked at Company X for 10 years, since interviewee A graduated from university.

Interviewee B is a 'group manager' within the Software Engineering group at Company X. The tasks are foremost management of their schemas, e.g. electricity schemas for the product as a whole, responsibility for communication busses, planning of the product's architecture and new functionality. Interviewee B has work at Company X since 1992.

Interviewee C is manager at a development department at Company X, where the control systems for an advanced part of Company X's products are developed. Interviewee C is responsible for the team within that department, but is also involved in the development process as well as the techniques used.

---

[2] When using the term "Company Z", it is this department of the company that is referred to.

Interviewee D is a chief architecture at Company Y and is responsible for process innovation and competence building within RM, architecture, and design. Process innovation means that interviewee D is responsible for improving Company Y's processes, and to combine them with RM. This includes working with people who are executing the RM process, in order to see what is working well and what needs improvement. The competence building includes ensuring that everybody in the process are properly trained and executes the processes according to the standard. Interviewee D is, hence, giving training courses and knowledge transfer sessions within RM. Before started at Company Y, Interviewee D worked at a university as a teacher and researcher within the area of information systems engineering.

Interviewee E has worked with software engineering since 1988, in different shapes and has dealt with issues such as project management, quality, business systems, programming, and system development. The interviewee has work at Company Z since 1996 as an enterprise developer, and during three of those four years, requirements engineering has been in focus. Interviewee E has developed and implemented a requirements engineering process at Company Z, but is now working as a system integrator. The current working task is to make a set-up for a Billing system for a large telephone company.

5.3 Results

This section presents the results of the interviews, compiled into three major tasks: a general description of the requirements work, management of requirements, and problems with RM. The answers of each interviewee is summarised under these three tasks.

5.3.1 General Description of the Requirements Work

Interviewee A

The working task for a new project comes from the product-planning department at Company X. This may be a very fuzzy description of the product, e.g. 'within three years you should develop a new version of Product A' and some general description of high-level properties of the new product. High-level requirements may also come from problems with old versions of the product. These requirements are rather general and

fuzzy and must be refined into more detailed requirements. However, there is no formal process of refining those high-level requirements into low-level, detailed requirements. The gap between the high-level and low-level requirements is too wide for defining a formal process, according to Interviewee A. A ‘middle-level’ is missing, and an intended area for improving the process is to develop a middle-level between the high-level and low-level requirements.

Low-level requirements are defined mainly due to the skills of single people in the project, and are based on discussion between project members. However, the main part of the project work lies in improving and further evolving the technical product. The project members are more or less experts on the product they are developing, and hence skilled on product details and have the competence to pose low-level requirements on the system. Old products and product descriptions are an important support for defining low-level requirements. According to Interviewee A, the intentions are to improve the process by using the product specifications as a basis for developing new versions in the future.

Low-level requirements can also be posed by other departments, e.g. the electricity construction department. The product that interviewee A’s department is developing is only a part of the overall products of Company Z, and the different parts must be able to communicate with each other to form the functionality of the overall product. Interviewee A’s department collaborates, for example, with electricity construction, and common requirements and how the parts shall work together is developed in cooperation with them. Because the parts of the overall products are developed in different departments, it is very important that specifications are shared with other departments that need the information. This makes it also important that the low-level specifications are kept updated and consistent.

The requirements are documented in so-called use cases, with plain text descriptions. No graphical models are used. The requirements specification for the product that interviewee A’s department develop is approximately in size of one and a half folder. The size of the development team is about 15 people.

Interviewee B

The high-level requirements or general description of the functionality can come from several different parties, e.g. by the product-planning department, which requests for new functionality, modification requests, inquiring from the market, and problems with old version of the product. A calculation of the maximum cost for the development or improvement of the functionality is also set based on the number of products to develop and sell. This calculation limits the resources available for the development work.

Based on the description of the functionality and the economical boundary, Interviewee B's departments starts with the so-called functional modelling. A group of people from different areas of responsibility is gathered to discuss the new functionality, in order to decide how the different control system involved should cooperate. This includes describing the functionality in more detail and to identifying which control systems that are affected. It is also important to decide how the messages between the different control systems should look like, in order to ensure that the different control systems are able to communicate. These control systems are often spread over different areas of responsibility and there should be a representative for each affected area. During this meeting, a protocol is written describing the functionality on a high level and how the different control system should cooperate.

Based on the functional model, each affected control system is developing their own specification of how to provide their part of the overall functionality i.e. how their part of the functionality should be achieved. This includes identify prerequisite for activating the functionality, which hardware to use, flows of messages within the control system and to other control system, etc. Old specifications are used to support this work. Because the different part of the overall product must be able to communicate and function together, it is very important that the departments share their specifications with the other departments needing the information.

The functional descriptions are documented and saved on a certain folder at a server, which all affected parties have access to. However, the requirements and other more general descriptions are harder to access. These documents exist, but there is no systematic way of how to distribute them.

Interviewee C

As for Interviewee A and B, the development process starts with a request from the product-planning department, describing rather general the new product/functionality. Examples of requirements may be to ‘develop a product X that is finished 2004 and that is better than all competitive products on the market at that point’. Within interviewee C’s department, this hardly gives any support and the more detailed requirements of the products are mainly set by the project members. The work begins at a system level, where an understanding of how the system shall be configured is developed. This usually means that the ‘hard’ parts of the product are specified first, and when all the ‘hard’ parts are in place they starts working with the electronics. For example: specifying the electronics in more detail, specifying the interface between the electronics, and functional requirements for the electronics.

The results of the work at the ‘system level’ are documented in a requirements specification. The specification is mainly developed by someone who is responsible for the documentation or for the requirements. At interviewee C’s department, this has lately been interviewee A, who documents the requirements based on his experience and knowledge. Interviewee A is also responsible for grounding the requirements that needs to be grounded. As soon as the system level of the product is set, the requirements are delegated to other groups that set the requirements of their part of the product.

The requirements are specified in pure text and in form of use cases. There exist graphical models when the more detailed functionalities of the product are described.

Interviewee D

Company Y has an extensive procedure for developing their software products, and also how to deal with their requirements. The RM process starts with collecting so-called market requirements from Company Y’s customers. These requirements can be of all kinds, small or large, futuristic or simple. All requirements that seem to be of interest are stored in a requirement database. The market requirements are then analysed in order to build Company Y’s own requirements. This is a process of reviewing what the customers ask for, what the market asks for, what the business analysts sees as the future, etc. Based on this information the so-called *business* requirements are formulated. These requirements are also included in the requirements database. The

business requirements are then selected into release projects, so for every new release a couple of requirements are selected and these are built into the system.

The persons responsible for creating the business requirements are called program managers, who is responsible for formulating business requirements for one of the thirty software products of Company Y. They analyse the market requirements and tries to group them together or generalise them, to form business requirements for future releases.

The requirements database also includes so-called customer commitments. These are a contractual commitment where Company Y guarantees to build some functionality into their products. The requirements work in Company Y also covers customer communication. This includes validating market requirements e.g. by asking customers if a certain functionality would be a desirable property in their system.

The requirements specifications consist of two parts. The first part is a form in the database where certain data has to be filled in. The other part is a so-called conceptual solution. This is a document describing the requirement in more detail, e.g. background, motivation, how the functionality shall be used in the business process, and eventually some ruff data model. The criterion for the conceptual solution document is that a software engineer should be able to create the design for the software without any additional document. They should hence get a full picture of the business background for a particular business requirement.

A project last for about 9-12 months, and about 150 requirements are included and are to be implemented in some of the 30 products of Company Y. A release project includes about 30 teams, which involves about 15 people each.

Interviewee E

Company Z has a well-defined process for Requirements Engineering (RE), which includes six activities. The fundamental philosophy of Company Z’s RE process is that it should start with an investigation of the customer’s organisation. This is done by modelling e.g. their business goals, organisational goals, and quality goals, in order to develop a general view of the enterprise. This information is then broken down into requirements that are a suitable basis for construction and test.

In the *first* activity of the RE process, the RE team is established. The RE team is composed by people from all the different disciplines involved in a project. This usually includes someone who is skilled at system issues and is well competent in the problem and solution domain. It also includes people from design, construction and test. It is important that the members of the RE team have different views of the system. This way of composing the team facilitates the document transference between different groups and phases in the project. The *second* activity is focused on planning of the RE process of the current project. This includes deciding what activities that should be performed during the development process in that particular project. Examples of other issues are: if any tools should be used, if a requirements database should be established, how the requirements should be documented, and what word processor to use.

When the first two activities are finished, the actual requirements work begins with the *third* activity, elicitation of requirements. Important customers and stakeholders are contacted and the requirements are developed in cooperation with these. The typical way of performing this activity is through workshops with the customer, where the system is discussed. This activity is performed in parallel with the *fourth* activity, specification of requirements. The requirements are normally specified in natural languages, but some projects may use sequence diagrams, state diagrams, and/or use cases to document the requirements. The result of these two activities is a requirements specification.

When the requirements specification is finished, the *fifth* activity, property checking begins, where the formulation and structure of the specification is checked. The RE team examines the specification using seven different properties, clarity, consistency (concerning the terminology), testability, traceability, validity, unambiguous and coherence. Properties can be added or removed from this set during the planning phase of each project. The *sixth* and final activity is requirements inspection, where the customer validates the requirements. Only the stakeholders that have been involved in the previous activities take part of this validation. After the validation phase is finished, the whole requirements specification is base-lined and the design phase of the project can begin.

This process contains general, overall activities describing what should be done. How these activities should be performed is decided in the planning phase. Which activities

that are to be a part of the process are set, as well as the method(s) to use. There are several different methods and techniques that may be applied, e.g. RUP, and which one(s) to use are decided on basis of the nature of the project.

The first contact with the customer is handled by the sales department, who are selling the system to the customers. They perform a feasibility study where a Request for Quotation or a Request for Proposal is written, which are a general description of the system. When the system is sold to the customer the request, including the deadline and the cost of the system, are sent to the development department. Usually, Interviewee E performs a second feasibility study to develop a requirements specification that is applicable as basis for design and construction of the reserved system.

A normal project last for about six month and includes approximately 300 requirements. Such a project involves about ten people.

5.3.2 Management of Requirements

Interviewee A

An extensive part of the requirements work is made manually, and is based on the knowledge and competence among the project members. There is no additional information stored about the requirements at all, and Interviewee A is not using any tool to support the organisation and maintenance of the information.

The requirements are not *prioritised*. The choice of which requirements that should be implemented in the product and which should not is based on a discussion between project members. Neither is the requirements *uniquely* numbered.

The *status* of the requirements is not documented and the project members manage the progress of the project manually. Company X only documents requirements that shall be implemented in the system. Which requirements that are to be included in the specification are decided by discussions between project members. Requirements that are cancelled are deleted from the specification or not documented at all. However, Interviewee A thinks that this may be a disadvantage. If the cancelled requirements could be saved until the next version, less time could be spent on identifying the requirements and on discussion. The requirements can be captured earlier in the process, and the risk to miss requirements decrease. There are no procedures of how to control

which requirements that are implemented and which are not. The project members manage this manually and the whole specification is normally implemented as a whole. However, this work is supported by a time schedule, which describes different subparts of the product and when these should be finished. These subparts are checked against the requirements of that specific part, but the overall product is not tested against every single requirement. The product is verified when it is implemented in the overall product, and tested in its real environment.

Conflicting requirements are dealt with by discussions between effected departments or people. If conflicting requirements are discovered as late as in the implementation phase the person who are responsible for the implementation of the requirements to implements the conflicting requirements as well as he/she can.

Company X is not storing any *traceability* information about the requirements. The project members control the correspondents between high-level requirements and low-level requirements manually during the process. The fulfilment of the high-level requirements is mainly checked when the product is tested by installing it in the overall product. The source of different requirements, i.e. where the requirements came from, is not documented. However, when requirements are posed from other departments, a specification is written for these requirements. This specification is saved, and it includes information about from whom the specification comes. Dependencies between requirements are also managed manually, and the projects members remember them by heart. A problem that may arise is that some dependencies are forgotten. However, if all dependencies where documented, the problem would instead be to keep the documentation updated and consistent, i.e. to deal with changes to the documented information.

There are two common reasons for requirements *changes.* These are that other departments adds functionality to their products, that requires information from interviewee A's department or that new functionality is added to the product that requires information from other department. How these requested changes are to be dealt with is decided in a discussion between the affected departments. If new requirements are added, the product planning department or managers must first approve them and add some more resources to the project to enable the implementation of the new requirement. According to Interviewee A, a more formal process, with more

concrete documented information as basis for the decisions, is likely to become unwieldy. It also requires someone who is responsible for and makes the decisions, and today there do not exist anyone who can take this role at Company X.

Interviewee B

Like Interviewee A's department, Interviewee B's department does not store any additional information about the requirements. The information is managed manually throughout the whole development process, and the work is based on the competence and knowledge among the project members. However, the overall functional descriptions, which can be compared with high-level requirements, are stored at a server where affected departments and people can access the descriptions. These overall descriptions are rather static.

Prioritisation is not explicitly made at interviewee B's department and the priority of each requirement is hence not documented. The project members manage the priority of the requirements manually, if the priority is needed to take into account.

The *status* of the requirements is not documented and the progress of the project is not documented in relation to the requirements. The main work with deciding which requirements that should be implemented is not decided by the development teams. This is settled on a higher level in the organisation e.g. product planning, product modification, or feature control, which all have some kind of responsibility for properties/functions. These decisions are mainly based on the economical calculation and on a request to the affected teams if they are able to develop the functionality within the limit of the calculation. If it is decided to implement the functionality, the actual development work is started for the affected teams. During the development process, all of these high-level requirements must be met.

Company X is not documenting which requirements that have been implemented or not. The controls of which requirements and functionality that are implemented are made by a systematic verification of the product. This is hence checked when the product is finished and there are no control of the coherence between the requirements or descriptions from the product-planning department and the functional description before implementation.

No *traceability* information is stored at Interviewee B's department. There exist no electronic relationship between the high-level requirements and descriptions and the more detailed specification written by each development teams. Dependencies and relationships between documents, requirements and other information are neither documented at Interviewee B's department. The different levels of the functional descriptions are managed by different departments, which may be one reason for no control between the levels. Interviewee B says that some kind of documentation management tool is needed to be able to have this kind of control. The Interviewee also states that it would be a gain to be able to identify how different functions affects each others, and which functions that are affected if we change a certain message, and vice versa.

Changes to the project scope, e.g. new requirements are dealt with in a similar matter as new functionality as described above. The change are sent to a committee, which makes a decisions whether to implement the requested change or not, based on the economical boundaries of the product. The changes are implemented if so is decided, and the system is then verified so as to see that the change worked. An approved change usually also includes more resources e.g. in terms of time and money. The project leader is responsible for dealing with new requirements or new ideas that occurs in the middle of a project. The decisions are made based on discussions between project members. Interviewee B says that if this should be handle in a more formal way, there must be at least a 100% more people involved in the project.

The overall functional descriptions stored at the server are rather static, and changes do not often occur. If changes do occur, there exist no automatic messages to every affected department and people that a change has been made in the functional descriptions at the server. The people who implement the requested change do this manually.

Interviewee C

Most of the requirements work is managed manually, and there is no additional information documented about the requirements. The requirements are not *prioritised* at interviewee C's department, because all the high-level requirements posed on the system is to be implemented.

The *status* of the requirements is not documented. What requirement that is to be implemented or not are managed by so-called work packaged with the length of 10 weeks. For each of these work packages there is a time schedule, describing what should be implemented when and what should be tested when, and what output that is expected. Every 10^{th} week a release is made and the result is followed up. If there are things that must be postpone due to a delay, the project leader decides what requirements to postpone, but in the end all requirements should be implemented in the system. There is no documentation of which requirements that are implemented or not. The person responsible for the system ensures that all requirements are implemented, and in order to check this, the system is tested and verified when it is finished.

No *traceability* information is stored at Interviewee C's department. The relationships between high-level and low-level requirements are kept track on by the person responsible for the documentation or requirements. No relationships or dependencies are documented between the different levels. This makes it more difficult to control the requirements and their relations, for example conflicting requirements, but interviewee C says that they have many competent co-workers that handles this. On the other hand, the relationship between the high-level and low-level requirements is not strong. The gap is to wide between the very general high-level requirements and the detailed low-level requirements. To document the relationship between these two levels some middle level is needed, and a tool is needed to support this. The requirements are also divided into isolated and independent areas of functionality. Different groups of people develop different functionalities, and the routines for specifying the interfaces between the functionalities are well described.

At interviewee C's department, some relationship between use cases are documented. If a use case includes parts that are also used by other use cases, these general use cases are documented separately at one place. The previous use cases are then coupled to this underlying use case. By this procedure, similar use cases or parts of use cases are not documented several times.

Interviewee D

Company Y has a well-defined process of how to manage their requirements. They have a RM-tool to manage the information gathered, and all requirements are stored in a requirements database.

In the requirements database a large amount of *additional information* is stored, about the market requirements as well as business requirements. Examples of additional information are: identity, creation date, request raiser, status, label, description, keyword, and capacity needed to build the requirements. The requirements are prioritised but not in relation to each other. The priority of the requirements are also stored and the categories are high, medium, and low. All this information is managed by the RM-tool and stored in the requirements database. The database also includes additional information about the conceptual documents and the functional components.

It is hence possible to documents the *status* of the requirements. However, the RM process is separated from the development process. The development is release-based, and in each release a new version of the products is developed. The requirements for a certain release is selected from the business requirements in the database, based on e.g. release themes. Once a new release project is started, the configuration management tool is managing the information, as well as the configuration of the products, and checking in and checking out processes. This tool is strictly separated from the RM-tool. The configuration management tool is also related to a test control system, which controls the testing of the products.

Company Y also stores an extensive amount of *traceability* information in their database. The business requirements are linked to the market requirements, which have been used as a basis for formulating the business requirements. The business requirements are also linked to the conceptual solutions document, which are describing the business requirement in more detail. If it is possible to identify the subfunctions in which the requirements are to be implemented, business requirements are also linked to functional components. Company Y is also storing dependencies between business requirements. These are:

- *Intergroup coordination*: Shows requirements that are split over several modules, where these modules are own by different team.
- *Parent-child*: Shows requirements that are contained in other requirements. The purpose of this relation is to allow grouping of requirements, but groping over different releases should be avoided.
- *Precedes*: Shows requirements that should be fulfilled before other requirements can be realised.

Interviewee D states that there are two kinds of *changes*: changes to the requirements, e.g. there must be some additional information or there are some errors to the requirement, and changes to the scope of the project, e.g. new requirements. The fist type is easy to manage, the information is changed and then the requirement is reviewed. For the second type, Company Y has a very formal decision-making process called Scope Management. This means that no one can just put a new requirement in the scope of a new release. The scope of the project is managed by a person in charge of these questions. When someone wants to get a new requirement into a release project, this person is the one handling the change requests. That person sends the request to the teams affected by it, and ask for the consequences or impact of the change. All the affected teams analyse this, and answer within a week the effort needed to implement the change and how much it will cost in terms of time. This is compiled and based on this information, the central release committee decides whether to accept this change or not. If it is accepted, all the teams must adept their project, e.g. their project plans and capacity plans. The affected business requirement and design documents must also be updated according to the change request.

Company Y has develop the RM-tool they use themselves. They did look at commercial RM-tools, and got some training in Doors and Requisite-Pro. However, Company Y had already developed their requirements database based on an access product. The decision where hence either to buy a commercial tool and adapt it to the Company Y way of dealing with requirements, or to add the needed functionality to Company Y's database, e.g. selection and prioritisation. It turned out to be easier and cheaper to adapt the database to the additional needs. Interviewee D says that their way of dealing with requirements should not be accommodated by a standard product.

Interviewee E

Company Z is using some simple techniques for basic management of the requirements. They use a word processor to store the requirements, and some few attributes and additional information is stored about the requirements. They also have a formal process of change management within the configuration management process.

The requirements specification is normally *structured* according to a certain pattern, e.g. IEEEs. The specification is divided into different sections describing various aspect of the system e.g. the stakeholders and their competence, the functions of the system. The

requirements describing similar aspects of the system are sorted into the same section. These sections are structured according to different viewpoints of the stakeholders, which normally describe different business processes of the organisation, or by different functionality, i.e. decomposing the system into subparts. For each of these section a rational is written, that motivates the requirements and introduces the reader into the background information of the requirements. All the requirements are uniquely numbered, in order to facilitate implementation, construction and change management.

There is no explicit *prioritisation* made of the requirements. The activity exist in the RE process, but this has also not been used so far. Which requirements that are important and less important are communicated to the constructors mutually by the members of the RE team, who have gained this knowledge during the requirements work. Besides rational, *source* is the only requirements attribute stored.

The RE process also includes a trade-off activity, but this has not been used so far. In this activity, an adjustment between the products requirements and the project requirements are done. This means that the developers' controls that the functionality of the system specified by the product requirements can be implemented within the boundaries in time and other resources specified by the project requirements. This has not been done explicitly so far, but it does happen implicitly. Requirements that are removed from the project are kept in old versions of the requirements specification. There exist not formal process of how to store and include these in the next version.

Within interviewee E's projects, there exist no explicit coupling between the organisation of the customer and the detailed constructional requirements documented in the requirements specification. This also means that there are no relationship documented between the high-level goals and requirements of the organisation and the requirements in the specification. However, the source of the requirements are documented, i.e. stakeholders, developers, or documents, such as the proposal from the customer. Quite many of the requirements are traced to project members, because these know by experience that certain requirements must be there and they must hence the source of the requirement. *Traceability* between test specification/cases and requirements are also documented, but not between requirements and design documents or code files. This requires too much time for maintenance, and the information is likely

to become inconsistent. Dependencies between requirements are not documented either. The RE team know this by heart.

Configuration management is an important part of the development process of Company Z and changes to requirements are dealt with within this process. When the specification is validated and approved, it is base-lined, which means that changes that occur must be dealt with using formal change management routines. For any *changes* to the requirements, a formal 'change request' must be written. This change request includes the configuration item that should be changed, e.g. a requirements specification, in which baseline the item exists, e.g. the version of the specification, a description of the change, motivation, and consequences. This changes request are then analysed by people that are affected by the changes, thus, when requested changes is about requirements, the RE team is analysing the change. However, other aspects of the change must also be taken into account, such as, market value, goodwill and resources. Therefore, people with various different views of the system should be involved in the decision process. When the change request is analysed, it is important to investigate which documents, e.g. design documents or code files that are affected and must be updated according to a certain change. It is different to change one design document and one code file compare with ten design documents and ten code files. Based on the result of the analysis a decision is made whether to implement the requested change or not. If it is agreed it is implemented in the system and document etc. are changed, and the result is verified. The change can also be rejected.

5.3.3 Problems with Requirements Management

Interviewee A

There are three major problems with RM according to interviewee A. *First*, the cooperation between different departments regarding exchange of requirements and information is problematic. The other departments must read the requirements specification on time and comment it, in order to get a complete set of requirements as soon as possible. All the products in the overall products must be able to communication and cooperate with each other to achieve the needed overall functionality. *Second*, changes of requirements and functionality from other departments as well as within interviewee A's department must be managed. The information must be updated and consisted and all departments must have read and use

the right version of the specification as a basis. *Finally,* one major problem is to decide what level of detail should the requirements be specified at. When is the specification clear and detailed enough? Interviewee A thinks that the requirements are over specified at this moment.

Interviewee B

The main problem with RM, according to interviewee B is distinctness. The responsibility for different task or areas of the development is explicitly alleged to people or groups of people in the organisation. It must be clearly stated who to turn to in different matters or with certain questions. Someone must be responsible for identifying different parts of the product, and stating the requirements for each area. Today, there are too much works that are done on behalf of people's own minds.

Better control over complex systems that are using several different control systems, in different parts of the overall product. Also here a more explicit distribution of the responsibilities are needed, on people as wells as groups of people. Requirements and functionality must be anchored at some economical department of the organisation. Interviewee B says that they need some kind of a documentation management tools to be able to manage the documentation. For example, the function descriptions, relation between the function and how they are affecting each other, in order to see how the functions are mutually related. This tool should also support change management. If a message is changed, which other are affected and which functionality and control systems are involved. The tool should somehow provide a map over the whole functionality of the system.

Interviewee C

There are several major problems with RM according to interviewee C. First, which level of detail is enough when the requirements are specified? Second, there are no person or group of people to control the more detailed requirements against. There is no way to control if the right product is developed until it is implemented and the testing has started. Third, there is no procedure how handle conflicting requirements, to control that the requirements are consistent and are not conflicting with each other.

Interviewee D

Interviewee D has identify several problems or rather research opportunities. First of all, the quality issue, what is a good requirement. How should requirements be formulated and what are the properties for good additional information to store? One other problem is also the granularity of the requirements – the abstraction level. Usually you cannot specify the requirement on a detailed level, because it takes too much resource to manage. The requirement are difficult to communicate to mangers, the scooping process becomes unwieldy etc. When have you specified your requirements detailed enough? One other issue is the bundling of requirements, for example be able to tell which requirements that are a part of another requirements, which are depending on, and which are conflicting.

The scooping process is a very lengthy process and you need about 30 man-days to analyse and make a decision whether to implement a requested change or not. How can you deal with situations when a requirement is inserted, changed, or removed and how can this process be performed more efficient? There exists very little empirical research on how to mange large volumes of requirements, both success stories and failure stories are needed. What is good practice within RM? The ability to track requirements forward to design, code, test, and documentation is also desirable.

Interviewee E

The most serious problem with RE according to interviewee E is to elicit the ‘right’ requirements and to validate the requirements. These activities are often performed on routine and not always in a correct way. Therefore, it is not sure that the system is implemented on basis of the ‘right’ requirements. Interviewee E has worked a lot with increasing the knowledge of how to formulate requirements and increasing the quality of the requirements specification. Interviewee E also states that all the people at the department must increase their knowledge within methods i.e. how to actually perform the activities efficient and correct. The quality and the productivity within the activities must be better.

Today, no dependencies between requirements and no relation between the requirements and the design documents and code files are documented. However, when a requested change is implemented, it is very important to identify what documents that must be updated according to a certain change. This information only exists in the heads

of people involved in the project, and there is a risk that dependencies and documents are overseen. Interviewee E said that there is a need to have traceability, but the time and effort to maintain it is to high.

5.4 Analysis of the Results

There were three companies involved in this interview study, and each of these companies represent different type of software development. It is hence, not possible to draw any absolute conclusions of how different kinds of development companies are dealing with their requirements or what differs between the types of companies. However, the interviews can give an *indication* of RM practice in different types of companies. This is also corresponding to the aim of the study: *understand* how companies are dealing with their requirements in practice.

Table 1 below briefly summarises the findings of the interview study. The first column shows the type of software development that each of the three companies are performing. Interviewee A, B, and C are working for the same company (X) that is developing an engineering product, which includes embedded software. Interviewee D is employed at a company (Y) in the development of a software product supporting enterprise resource planning (ERP), and interviewee E's company (Z) is configuring standard products according to the needs of specific customers. Company Y is hence market-driven, whereas Company Y is customer-driven, but they are both focused on developing software. Company X is focused on developing a mechanical product, and the software is only an embedded part of that product.

The amount of effort placed on management of requirements differs between the companies as can bee seen by studying the columns in Table 1. *Company Y* has the most extensive process concerning how to manage the requirements, and this process includes RE. There is a tool-support for administrating the whole set of information gathered, the requirements are stored in a database, they are using attributes to describe the requirements, and a large amount of traceability links are established and maintained. There is also a formal decision process for managing changes to the requirements. *Company Z* has a well-defined process for RE but not for managing the elicited and documented requirements. They are placing efforts on formulating a usable requirements specification, for construction as well as testing. They also have a certain

way of structuring their requirements specification and, in some cases, the source of the requirements and the rational of groups of requirements are stored. They have a formal process of dealing with changes in their configuration management system. However, they do not use any tool-support or repository to store all the information gathered. *Company X* is writing a specification of the product, but do not have a specified process of how to develop the requirements nor how to manage the requirements. They hardly store any additional information, only minor traceability information between some use cases. Neither is they formally managing changes to the requirements. However, they store some information, such as functional description and messages sent between control systems, in database/on servers. This information is shared between different development teams in the organisation and is fundamental to ensure that the different parts of the overall product are able to interact and communicate.

	Type of software dev.	Size of project	Specified RE/RM process	Requirem. attribute	Trace-ability	Change Managem.	Reposit-ory/Tool
A (X)	Embedded software	15 people per team, 300 pages of requirem., 3 years to produce	No/No	Non	Non	Informal decisions	No/No
B (X)	Embedded software	?	No/No	Non	Non	Informal decisions	Functional description/ No
C (X)	Embedded software	See A	No/No	Non	Between Use Cases at diff. levels	Informal decisions	Messages between control sys./ No
D (Y)	Software product developm. (ERP)	One release – 30 teams of 15 people each, 150 requirem., 6-9 months duration	Yes/Yes	Yes. Examples: id, priority, status, capacity to build, description, keyword	Yes. Business req. to: - market req - concept. solution - function. comp. Dependenc.	Scope management	Requirems, functional components, conceptual solution/ In-house built RM tool
E (Z)	Customer driven configurat. of COTS	10 people, 300 requirem., 6 months duration	Yes/No	Yes. - Source - Rational (for group of req.)	Yes. From req. to test cases	Configurat. Management (deals with changes to the whole specificat)	No/No

Table 1: Summary of the interview results

It is interesting to analyse the possible reasons for these discrepancies. Since Company Y has the most extensive RM process, we will use this case as a basis when discussing and comparing the differences between the three companies. Company Y is developing software products, which are improved and evolved into new versions of the products. There are several teams involved in developing each new release of the products and the amount of people and requirement involved is rather extensive. There is a need to control and maintain the information used and to reuse requirements and information from previous releases, as stated by Interviewee D. Company Z is developing a solution for a specific customer. The whole product is taken care of by one team, which configures the standard system according to the customer's requirements. There are no new versions developed of the systems, and the need for reuse information is minimal. Company Z explicitly stated that they are not considering the administration of the requirements as a problem and that information 'book-keeping' is not much beneficial for their work.

Both Company Y and Z have a mature RE process, but due to the differences in their software development, the focus of the RE process differs. In Company Z the main problem is to elicit the requirements of the customer and to validate these. They are also putting a lot of work into formulating the requirements and structuring the requirements specification. This work is mostly aimed at supporting the design and implementation of the system. For example, the traceability information stored is between the requirements and the test cases. For Company Y the main problem is to administrate the large amount of requirements that is collected and to prioritise them in order to choose which one's to implement in a certain version of the system. They are also, like Company Z, putting effort on formulating their requirements in a proper way. For Company Z, the customers in cooperation with the development make these types of decisions. In Company Y, this is in the hands of the development organisation, which are the owners of the system that is developed. For Company Z, a new system is developed in every project, while Company Y is developing a new version of the old product, and is interested in reusing requirements and knowledge developed during the work with the previous version.

The main differences between Company Y and Z is hence that Company Y has a market driven RE process, while Company Z has a customer driven RE process. For RM to be useful and beneficial in its whole extend, the companies must deal with some kind of

development of new versions, which influence the need to reuse requirements and knowledge of previous system.

Company X is also developing new versions of their product, as Company Y does. They are also having several teams developing different parts of the overall product, and lot of people involved in the development process. Nevertheless, they are not administrating their requirements, a part from documenting them in a specification. One way to increase the understanding of the possible reasons for this discrepancy is to compare the two companies. *First*, the products that the companies are developing differ. Company X is developing a mechanical product, where software is an embedded part, while the software is the product for Company Y. The method and techniques, as well as developing traditions may hence differ between the companies. Since the software is only a part of the product for Company X, the amount information to manage is rather extensive, and is not only about software. This may also affect the type of information to manage as well as how this is managed. Company Y is managing their information on a requirements level. Every single requirement are stored and managed separately. The information that Company X is storing and administrating is more on a documents level. Whole documents are administrated, e.g. specifications from other departments or the functional descriptions.

Second, the maturity of the RE process differs between the two companies. Company Z has a well-defined process of how to deal with the requirements, from elicitation to management. Company X started to write a requirements specification for their product in the current project, and is about to develop and formulate a process for the requirements work. This may also affect the interviewees' apprehension of their need for administrating requirements and other information. However, at Company X there is a tendency to store information in databases or at severs, when the information is of fundamental characteristic for several different teams, and it is important that these teams share this information. One example is how different control system, in the overall product should communicate.

It is also important to note that the interviewees at companies X and Y are at severely different levels in the organisation. The interviewees at Company X are project leaders or for a department developing a part of the overall product. The interviewee at Company Z is responsible for developing the processes of RM, design and

implementation. The interviewee has hence an overall view of the processes and the needs of each process.

To sum up, there are several factors that may affect RM in practice. These are:

- *Need of reusing requirements and knowledge*
  For RM to be important in its full extent, there must be some need to use and reuse the information stored in the repositories and databases.
- *Type of software*
  There seems to be a difference between organisations developing software where the software is the product, and organisations where the software only is a part of the product i.e. embedded software.
- *The size of the project*
  It is likely that organisations spend more resources on RM in such cases where there are several teams, involved in the development work, and where these interacts with each other.
- *The maturity of the RE/RM process*
  There is a tendency that organisations with a more mature RE process spend more effort on RM than organisations with an immature RE process.

The RM performed in organisation is usually affected by several of these factors. It is not likely, as can see by the analysis, that the possession of one of these factors mean that RM is an important process within that particular organisation. Usually it is a combination of different factors. For example: it is not certain that RM is an important issue because a company has large projects if there is no need to reuse requirements and knowledge.

It is also interesting to notice the importance of the skills and competences of the project members, despite the amount of RM performed in the organisation. In Company X, the requirements specification is formulated and developed based on the experiences of the person responsible for the document. Company Y has a program manager, who is responsible for formulating business requirements based on market requirements, business analysis etc. In Company Z a lot of the requirements are posed by the developers, based on their experience of developing these types of systems. As can be

seen in the interviews, the personal skills and competence are particularly important when formulating requirements of the system.

RM in practice is highly related to the type of project and software development performed within the company. The RM that are pursued according to the interviewees are rather similar to the RM described in chapter 4. The information and requirements gathered is *organised* in some way (section 4.2). The information may be *traceable* in some way, as described in section 4.2.2, and some type of *change management* (section 4.3) is used to deal with changes to the requirements. The results of the interviews are analysed for all of these three categories. For a more thorough description of the RM processes respectively and examples of RM in practice, the readers are referred to section 5.3 where the results of each interview are presented.

The companies involved in the interview study *organise* their requirements in fundamentally different ways. Both Company X and Z are using a word processor to store and maintain their requirements. However, at Company X the overall, common specifications are stored at a server, easy accessible for all involved departments. Company Y is using a RM tool with a requirements database to store the requirements and the additional information stored. Company X is not storing any additional information about their requirements, while Company Z is structuring their requirements specification in different sections of requirements. Each section starts with a rational that describe and motivates the set of requirements. In some cases both Company X and Z are storing the source of requirements. Company Z are doing it for some requirements and Company X for whole specifications from other departments. Company Y stores a large amount of additional information about their requirements, both attributes in a form in the database and documents describing the requirements in more depth. The attributes are similar to the once discussed in section 4.2.1, but there are attributes that differs. For example, creation date, label, keywords, and capacity to built. In practice, both word processor and requirements database is used to store and organise the requirements, as discussed in section 4.2.3. The amount of additional information store differs between different companies, but the attributes mentioned are similar to the once discussed in section 4.2.1.

The amount of *traceability* information stored also differs between the three companies. Both Company X and Z have minor traceability information stored. Company X stores

the source of specifications from other departments, and in some cases the relationship ‘part-of’ between use cases. Company Z also stores the source of some of their requirements, as well as the relation between requirements and test cases. The documentation of source for both companies can be categorised as ‘back-ward from’ traceability, in the framework discussed in section 4.2.2. Traceability of different levels of use cases can be seen as a dependency between requirements, and traceability between requirements and test cases can be both ‘forward-from’ and ‘backward-to’ traceability. Company Y has several different traceability links stored: from business requirements to market requirements, from business requirements to conceptual solution document, from business requirements to functional component. The first traceability link, from business to market requirements are a ‘backward-from’ traceability link, while the other are of the category ‘forward-from’ traceability. At Company Y some dependencies between requirements are also stored, intergroup coordination, parent-child, and precedes. In other words, the amount as well as the types of traceability information stored differs between the three companies. They have adjusted the amount of and the type of traceability to the needs of the companies respectively. The traceability links stored can easily be categorised according to the framework discussed in section 4.2.2, as shown above.

All three companies are dealing with *changes* to requirements during the projects. Company X do not have any formal decision process, and the changes that appear within the project are managed by the project leader in cooperation with project members and affected parties. Company X and Y differ between changes to the requirements within the limits of the project and changes to the scope of the project. Changes to the scope of the project are managed on a higher level in the organisation and not by the teams respectively. In both cases there are a committee that handles these types of errand.

Both Company Z and Company Y have formal decision process to manage changes to requirements. In literature, the change management process consists of five different steps: plan for change, baseline requirements, one channel to control changes, change control system, update information. Company Z has a configuration management system, which includes dealing with changes to the requirements specification. They are, hence, having a plan for dealing with changes. They are also baseline the

specification as well as a single channel responsible for managing changes to requirements (the RE team). The CM process includes a systematic process of analysing and making decision whether to implement a certain change or not. One part of this process is to analyse which documents that are affected and must be updated according to a certain change. The change management process of Company Z is, hence, well covering the change management process describe in literature. Company Y has a formal decision making process for managing changes to the project scope, e.g. new requirements. Company Y are hence planning for how to deal with changes, and within one person responsible for dealing with the change request that occur. The change requests are sent to the affected teams to be analysed and the impact of the change is estimated, which indicates that Company Y has a systematic process of control changes. If the change is accepted, all teams must update their affected information. The step of baseline the requirements is not explicitly mentioned as a step by interviewee D. However, the interview material indicates that the amount of requirements as well as the information describing the requirements are set before the project starts, because of the formal way of dealing with the project scope. We can hence conclude that the scope management process of Company Y is also well comparable with the change management process in literature.

Changes to requirements can hence be managed both informal, as Company X does, and with a formal decision-making process as Company Z and Y. Interesting to note is that the overall structure of the companies change management process are very similar, as well as the accordance with the overall process describe in literature. For a formal change management process, there is usually some form of systematic process defined, which includes person or group responsible for the managing changes, a systematic way of analysing the change and making a decision whether to implement the change or no, and a systematic way of implementing approved changes.

Each interviewee has also been asked to state the problems, which they consider as important within RM. The following problems where mentioned by the interviewees:

- *Information exchange and documentation control*
  Interviewee A stated that because all the parts of the overall products must be able to communicate to form the functionality of the overall product, it is important that the different departments share their requirements specification.

This cooperation between the departments could be improved. According to Interviewee B, also the control of the overall product should be better. It should be clear what complex systems that are using the different control systems and how the different control system should cooperate. Some form of documentation management tool is needed to be able to manage the documentation, e.g. functional description, relations between functions, and how these functions affect each other. Interviewee C also states that there is no way of control if it is the right product that is developed, and that there need to be some person or group of persons to control the detail requirements against.

- *Changes of requirements*

  Changes to requirements and functionality posed by other department must be managed, according to Interviewee A. The affected information must be updated and all the departments using the affected information must have the right version. Interviewee D and E are also mentioning dealing with changes of requirements as a problem. Interviewee D states that the scope management process is very lengthy and works should be done on how it can be performed more efficient. Today it takes about 30 man-days to make a decision whether to implement a change request or not. Interviewee E states that there is a need to analyse the impact of a certain change, i.e. what design- and code files, and other information that are affected and need to be updated. However, the effort to maintain the information and traceability is too high.

- *Clear distribution of responsibility*

  The responsibilities of different tasks in the organisation should clearly and explicitly alleged to a person or group of people. It must be clear whom to turn to in different matters and with certain questions. Especially concerning the different products and their requirements and other documentation.

- *Level of detail of requirements*

  Interviewee A, C, and D have all mentioned that there is a problem to decide the level of detail for the requirements. When are the requirements specified clearly and detailed enough?

- *Bundling of requirements*

  Interviewee D states that there are little know about how requirement can be and are bundled. You must be able to tell if a requirement is a part of another requirement, dependencies between requirements and which requirements those are conflicting. Interviewee C also mentioned conflicting requirements as a problem. There is no way of ensuring that the requirements are consistent with each other.

- *Eliciting the right requirements*

  According to Interviewee E, one of the major problems is to elicit the 'right' requirements in cooperation with the customers and to validate these. This work is often performed on routine and not always in a correct way.

- *Formulation of requirements*

  Interviewee E has worked a lot with how to formulate clear and useful requirements. The formulation of requirements affects the requirements specification and hence the material used as basis for further work. The quality and productivity will increase if this material is clear and consistent.

The problem mentioned differs in their focus, and some of the problems are clearly related to RM while others peripheral. The problems that are clearly related to RM are: information exchange and documentation control, changes of requirements, and bundling of requirements. Formulation of requirements, the level of detail of requirements, and eliciting the right requirements are more related to the RE activities elicitation and specification. Clear distribution of responsibility is more peripheral for this study. However, one of the major problems of RM are not mentioned in this part of the interviews but are instead indicated by some of the interviewees is the maintenance of the information developed and stored. It requires a lot of resources to keep the documentation updated and consisted, in order to make it usable. Interviewee A indicates this when talking about documenting dependencies, and Interviewee E mentions the maintenance in relation to ensuring traceability. Interviewee A and B also mention the working load as a problem with managing changes to requirements formally.

5.5 Conclusions of the Interview Study

The purpose of the interview study was to "*investigate how Requirements Management is conducted in practice"*. The intention was to increase understanding of how companies deal with requirements in order to compare the result with literature as well as to identify problems and problem areas. This section includes some tentative conclusions, which that are indicated by the interview material.

The effort put into RM differs in different organisations. The need to reuse requirements and knowledge about the old version of the system are important factors to describe this discrepancy. There is a tendency that market-driven organisations that develop software products are more likely to spend resources on RM, compared to customer driven organisations or organisations developing software products. The maturity of the RE and RM process also affects the efforts of administrating the requirements. The size of the development project may also affect the amount of RM performed.

RM in practice is rather similar to RM described in literature. All the three major areas of RM in literature can be recognised in the interviews.

- *Organising requirements:*
  The requirements are organised using word processor or a requirements repository/database. The attributes used in practice to describe the requirements are similar to the attribute discussed in literature. The attributes used are reflecting the focus of the organisation or project using the attributes.

- *Traceability information:*
  The amount of traceability differs between different organisation, according to the focus of their RE and development process. The traceability information stored where both in a backward and forward directions, i.e. from requirements back to source and from requirement forward to realisation object. There was, however, little traceability from other documents to the requirements.

- *Change management:*
  Both informal and formal change management where used in practice. In this particular case, the companies with formal change management process where the once with mature RE processes. The formal change management processes

described where severely similar to the process described in literature on an overall level. There is a systematic process of dealing with changes defined and this process includes a person or group of people responsible for the task, impact analysis, and update all affected information.

There are several problems mentioned by the interviewees, but the problems that are strongly related to RM are:

- *Information and document control and exchange:*
  Is mainly about structuring and administrating the large amount of information gathered, on a documents level as well as on a requirements level. The aim is e.g. to facilitate the reuse of requirements and knowledge and the information exchange between parts of the project.

- *Changes of requirements:*
  The problems are to identify the affected information in order to ensure that all affected information is updated and that a formal change management are very lengthy.

- *Bundling of requirements:*
  There is a problem to identify bundling of requirements, e.g. part-of relations, dependencies between requirement, and conflicting requirements.

One of the major problems with RM, which is indicated by some of the interviewees, is the maintenance of the information stored, i.e. the work with keeping the information updated and consistent and the formal process around this.

6 RM Issues

Based on the literature study and the interviews, important issues within RM have been identified. This chapter includes a presentation of the issues found. These issues have been divided into five problem areas:

- Requirement Change Management
- Dependencies between Requirements
- RM Tools
- Information Management

These areas are further described below.

6.1 Requirements Change Management

One of the most important aspects of RM is to manage changes to requirements. Some change management approaches have been described in this work, both from literature and from the interviews, but these are very general. They are more focused on what and less on how to do it (see section 4.3). More research is needed regarding various subjects within this area. How can changing requirements be managed? Success stories as well as failure stories are interesting to investigate in more depth to broaden the knowledge of how to manage changes. Which aspects do we need to take into consideration in order to make informed decision about changes? What information is needed as a basis for those decisions? How are the benefit/value of the change and the cost of implementing it decided? What support is needed to extract and compile this information and to identify the information that needs to be updated according to the requested change? One problem mentioned by the interviewees where to identify what information, e.g. design document and code files to update according to a certain changes (see section 5.4).

Formal processes, e.g. with the use of change requests are rather time consuming, but necessary to avoid ad hoc implementation of requirements changes (see section 4.3 and 5.4). This needs to be improved, but how can the decision process be performed more efficiently and what characterises and efficient decision process?

6.2 Dependencies between Requirements

Dependencies between requirements are of fundamental concern in order to estimate the impact of a proposed change (Kotonya and Sommerville, 1998). It is also important for other issues within information systems development, e.g. the release planning in market-driven RE where a subset of the requirements are selected for implementation (Carlshamre and Regnell, 2000).

Requirements can relate to each other in various ways. For example: when a new version of a software product is to be developed, there is a substantial difference if requirement A cannot be implemented without requirement B, compared with if the cost of implementing requirement A decreases if requirement B is also implemented. The effect of the choice not to implement requirement B is quite different in these two cases. To be able to make sound decisions, it must be possible to identify what kind of dependency there is between requirements, not only that there is a relationship. Today, rather little research is found concerning types of dependencies that may exist between requirements and how these affect the work with requirements (see section 4.2.2 and 5.4).

6.3 RM Tools

Management of requirements requires extensive work regarding e.g. ensuring traceability, managing changes, and these activities need some form of support. RM tools have been criticised for not supporting the functionality needed by RM practioners. Especially when people need more advanced support or more detailed information about the requirements. Is this an indication of poor understanding from RM tools vendors regarding the support needed within RM?

Acquisition of an RM tool is not unlike purchasing a COTS system supporting any business process in an organisation. The system must be flexible and configurable to be adjusted to the individual business process of the organisation, and to support the needs of the end-users (James, 1996). Software development organisations have the same needs. The systems and tools used to support the business process, i.e. the system development process used in that particular organisation, must be adjusted to fit that particular process.

How do current RM tools support the management of requirements, both regarding how to organise the requirements, and how to manage change to requirements? What is the required functionality of RM tools? The possibility to easily and quickly identify which information that must to be changed, according to a certain change request, is mentioned as a desired functionality of an RM tool (see section 5.4). But how can this functionality be provided?

6.4 Information Management

There is a large amount of information gathered during a development process that need be administrated. During the interviews some desires about keep track about document and other information, and relation between these where discussed. There was a need to have a better control over the whole product and the interaction between the different parts of the product. How shall such a support look like? Is an RM tool enough or is more support needed?

The additional information stored about the requirements is to be used as a support and basis for decision-making in requirements work. The information needed differ from project to project, depending on what information is relevant for the project. Which factors affect the selection of information to store about the requirements? Are there any general, overall requirements attributes usually employed in ‘all’ development projects? The RM tools available on the market should support these general attributes, but also provide the ability to specify additional attributes relevant for a particular business or project. If the additional information is used as basis for decision-making, there is also a need to identify what different kinds of decisions that are made during a development project, in order to identify the additional information needed.

The discussion above is mainly focused on explicit requirements, where the requirements can be uniquely identified. However, in a development process there are also implicit requirements, which are hidden in text descriptions and graphical models. Most RE researchers agree that to develop a system that corresponds to the needs and expectations of the users, these implicit requirements must be fulfilled as well. If they are implemented in the system, they are also affected if changes are made and must be taken into account e.g. when an impact analysis is performed concerning a requested change or to keep the implicit requirements consistent when a requested change is

implemented. How can dependencies between implicit requirements, and between implicit and explicit requirements, be managed in order to provided a complete basis for impact analysis? How can the implicit requirements, for example models, be kept consistent when changes are implemented? How can we store additional relevant information about implicit requirements to use as a basis for various decisions during the requirements work?

7 Concluding Remarks

This chapter includes a concluding discussion of RM and the results in this work. An evaluation of the way of working is performed and the chapter also includes a brief description of future work.

The aim of this work has been to give an overview of RM and to identify possible areas of further research. The work includes an overall description of RM and the various activities included in RM, a discussion about the relationship between RM and RE, and an interview study about RM in practice. Based on the literature and interview study issues within RM has been identified and areas, where further research is needed, are formulated.

Based on the definition of RM two main tracks can be identified: [1)] to organise the requirements and to store additional information about the requirements, which can be used as basis for various decision during the development work, and [2)] to manage changes to the requirements set and additional information in order to keep the information up to date and consistent. Both these activities require requirements traceability to be effectively performed. We have chosen to view RM as a support, not only for managing the requirements during the development of the system, but also for maintaining the requirements set during the whole life cycle of the information system.

The interview study showed that the resources spent on RM differs substantially between different organisations. The interview study also shows that the type of software development and the size of the project are affecting the RM work performed, as well as the need to reuse requirements and knowledge in subsequent versions of the system. There is a tendency that organisations that develop software products, which are evolved through releases of new versions, are more likely to spend resources on RM compared to organisation developing customer specific solutions. However, the maturity of the RE/RM processes and the role of the software in the development process may also affect the RM work.

Several problems with RM where discussed by the interviewees. One major problem concerns the fact that RM requires a lot of resources, especially to maintain the large amount of requirements and information stored, in order to keep it up to date and

consistent. Other problems mentioned are: information and document control and exchange, changes of requirements, and bundling of requirements.

7.1 Evaluation of the Way of Working

There was no common definition of RM found in literature and there are almost as many definitions as there are authors discussing RM. Unfortunately the definitions found were also rather divergent. RM can be defined as a ‘management’ part of RE, where the elicited and specified requirements are administrated, or as the overall term for all requirements work, including elicitation, specification, validation, and management of the requirements. The process of defining RM, based on the literature found, was rather time consuming and much effort was spent on identifying what RM is and the relationship between RM and RE. As a consequence, there is literature that has a different view of RM compared to the material presented in this work. There may also exist relevant literature that has been missed because it uses a different term for RM.

RM was also shown to be an extensive area, which includes a number of large sub areas. However, there was not much literature to be found about RM on an overall level. There exist much writing on a more detailed level e.g. in articles from journals and conferences, but it was not possible to investigate the sub areas respectively in such detail within the scope of this work. These problems made it difficult to find relevant literature for the work. As a consequence, some parts of the work are mainly based on the results in some few books and the number of references used to describe RM is rather small. Nevertheless, the work concludes the overall descriptions of RM and provides a wide-ranging view of this area, which can be used as a basis for further work. The result from the literature study is complemented by three interviews describing RM in practice. The results of the interviews agree well with the results of the literature study, and do hence, to some extent, validate the results. However, in future work, the sub areas that are related to the chosen research problem must be investigated in more detail.

Due to the overall and general character of the work, the type of information system or software development was not taken into consideration during this investigation. As can be seen from the result of the interviews these factors affect the RM work performed and should hence affect the description of RM. However, we think that this context

independent approach was interesting, as a start, to provide an overall view of RM. The result may also be used to delimit further work within the area, concerning type of information systems and/or software development (see section 7.2).

The interview study involved three different companies, which developed different types of software. It would have been interesting to interview several companies within in each category, in order to be able to identify patterns of how RM is performed within different types of organisations. The results of current interviews are more indications of the RM performed in practice. Nevertheless, the result of the interview study is a complement to the literature study and may also be used as a basis to identify relevant problems within the area.

The execution of the interviews may also be improved in some aspects. During the review of the interviews, statements or descriptions that were rather difficult to understand or to relate to previous parts of the interview were found. The routines concerning structuring the interviews and getting an overview may be improved, e.g. concerning what questions that have been answered, what have been said, and the relation between the answers. This can be accomplished for example, by drawing pictures presenting the processes described by the interviewees and validating these before moving on, and/or writing down documents and activities brought up to be able to discuss relations between concepts mentioned etc. These types of aspects were found to be rather difficult to manage during the interviews. It is also very important to have a clear idea of what knowledge to elicit during the interviews, what is relevant to know, and to assure that this has been answered before ending the interview.

During the interviews, the interviewees were asked to identify problems with RM according to their experience. The answer is presented in section 5.3.3. However, it is indicated both in the interviews and in the literature that the major problem with RM is the amount of extra effort needed to develop and to maintain the requirements and additional information. Especially when all affected information need be updated according to changes implemented during the various stages of the life cycle. Only to ensure traceability covers a large volume of information and linking, which shall be maintained as well. It is discussed in literature that traceability strategies should be developed which identify what type of information that need to be store, implying that only relevant traceability information should be stored. This indicates the need of a cost-

benefit approach to RM, i.e. evaluate RM in terms of benefits that are received and cost of implementing RM. It would also be interesting to investigate the cost of not implementing RM. The fact that the benefits and use of RM is not immediate and clearly visible makes it even more difficult to motivate customers and developers to put effort into this 'extra' work.

In the beginning of this work, the aim and objectives of this work were presented. The aim was to survey the area of RM from a life cycle perspective and to identify issues for further research (see section 1.1). In Table 2, the five objectives identified to fulfil the aim of the dissertation are listed, as well as the sections corresponding to the objectives respectively.

Objective	Chapter/Sections
1. Define the notions of 'Requirements' and 'Requirements Management'	In chapter 2 the notion and classification of requirements are discussed. Section 4.1 covers the notion of RM.
2. Survey the area of Requirements Management from a life cycle perspective	Chapter 4 presents RM and activities included in RM, whereas section 4.4 covers the life cycle perspective of RM.
3. Identify how Requirements Management relates to Requirements Engineering and to the information systems life cycle.	In chapter 3 the RE process are presented, which includes RM. Section 3.5 presents an overall figure describing RE and hence also RM.
4. Investigate how Requirements Management is conducted in practice	Chapter 5 presents the results, analysis and conclusion from a interview study of RM
5. Identify and discuss problem areas and problems within Requirements Management	Chapter 6 discuss issues within RM.

Table 2: Correspondence between Dissertation Objectives and Chapters/Sections

7.2 Future Work

Chapter 6 discusses a number of areas and problems within RM where further work is needed. This section will mainly focus on describing the future work planned by the author.

The problem area in focus for future research is relations or interdependencies between requirements (see 6.2). As have been found in this work, dependencies between

requirements are important to take into consideration in various decisions-making situations during the requirements work. Dependencies influence the cost of implementing changes to requirements as well as the selection of which requirements to implement in new versions of the products. Other situation where requirements dependencies may affect the requirements work is requirements allocation.

There are several important and interesting issues identified in relation to requirements dependencies. What types of requirements dependencies exists, and in what type of decision situations these are important to take into consideration? What characterises these types of decisions situations? What types of dependencies are particularly important or difficult to deal with in the types of decision situations respectively? An idea could be to prioritise the requirements dependencies in terms of importance for each type of decision situation.

There are also other more general questions concerning requirements dependencies: How can dependencies between requirements be documented and maintained? What existing RE problems can be traced to lack of knowledge of requirements dependencies? Taking a cost-benefit approach to requirements dependencies would also be interesting. What are the advantage and benefit of storing requirements dependencies, i.e. what are we improving, and what are the cost in terms of time and extra work.

According to Höst et al (2000), effective management of software requirements are an important success factor for the ability to meet the demands of the market. It is also indicated by the interviews, that management of the requirements are especially important in market-driven development organisations. In this type of software development requirements selection and release planning are crucial, in order to be able to deliver competitive software products. These activities are supported by prioritisation of the requirements, and maintenance of a requirements repository (database) including requirements that should be implemented in future releases. Future work within the area of RM will hence be focused on market-driven software development.

References

Avison, D. E. and Fitzgerald G. (1995) *Information Systems Development: Methodologies, Techniques and Tools*, McGraw-Hill.

Carlshamre, P. and Regnell, B. (2000) *Requirements Lifecycle Management and Release Planning in Market-Driven Requirements Engineering Processes*, Second International Workshop on the Requirements Engineering Process, Grenwich, London, September, 2000.

Carnegie Mellon University Software Engineering Institute (1994) *The Capability Maturity Model, guidelines for improving the software process,* Addison Wesley Longman, 1994.

Curtis, B., Krasner, H. and Iscoe, N. (1988) *A field study of the Software Design Process for Large systems*, Communications of the ACM, 31(11), pp. 1268-1286.

Dorfman, M and Thayer, R.H. (Eds.) (1990) Standards, Guidelines, and Examples on System and Software Requirements Engineering, IEEE Computer Society Press, Los Alamitos, California.

Flynn, D. (1998) *Information Systems Requirements: Determination and Analysis,* McGraw-Hill Publishing Company, London.

Galal, G. H. (1999) On the Architectonics of Requirements, *Requirements Engineering Journal*, Vol. 4, pp. 165-167.

Harker, S. D. P., Eason, K. D. and Dobson, J. E. (1993) The Change and Evolution of Requirements as a Challenge to the Practice of Software Engineering, Proc. of IEEE Symposium on Requirements Engineering, San Diego, Califonia.

Hjelte, M., Kranqvist, B., Nyman, U. and Oldgren, S. (1995) *Krav på krav - En studie av området kravhantering från kundbehov till färdig produkt vid utveckling av programvarubaserade system,* Sveriges Verkstadsindustrier.

Höst, M., Regnell, B., Natt och Dag, J., Nedstam, J. and Nyberg, C. (1998) Exploring Bottlenecks in Market-Driven Requirements Management Process with Discrete Event Simulation, In *Proc. of Software Process Simulation Modeling Workshop*, London, July

IEEE Standard 610.12 1990 (1990) *IEEE Standard Glossary of Software Engineering Terminology*, Institute of Electrical and Electronics Engineers, New York.

Integrated Chipware (2001) *RTM products: RTM Workshop - The Enterprise Engineering Solution*, http://www.chipware.com/sub1.php3?section=rtm, 2001-07-24.

James, L. (1996) What's Wrong With Requirements Management Tools, *Requirements Engineering*, Vol.1, No 3, p. 190-194, Springer-Verlagen, London

Jarke, M. and Pohl, K. (1994) Requirements Engineering in the Year 2001: On (Virtually) Managing a Changing Reality, *Software Engineering Journal*,

November, 1994

Jarke, M., Pohl, K., Dömges, R., Jacobs, S. And Nissen, H.W. (1994) *Requirements Information Management,* Special Issue on Requirements Engineering), Vol.2, No. 6, 1994

Jarke, M. (1998) Requirements Tracing, *Communication of the ACM,* vol. 41, No.12, December 1998, pp. 32-36.

Karlsson, J. (1995) *Towards a Strategy for Software Requirements Selection,* Licentiate Dissertation 513, Department of Computer and Information Science Linköping University.

Karlsson, J. (1996) *Framgångsrik kravhantering - vid utveckling av programvarusystem,* Sveriges Verkstadsindustrier.

Karlsson, J. (1998) *A systematic Approach for Prioritizing Software Requirements,* Doctorial Dissertation, Department of Computer and Information Science, Linköping University.

Kotonya, G. and Sommerville, I. (1998) *Requirements Engineering - Processes and Techniques,* John Wiley & Sons Ltd.

Leffingwell, D. and Widrig, D. (2000) *Managing Software Requirements - A Unified Approach,* Addison Wesley Longman, Inc.

Loucopoulos, P. and Karakostas, V. (1995) *System Requirements Engineering,* McGraw-Hill Book Company, London.

Lubars, M., Potts, C. And Richter, C. (1993) *A review of the state of the practice in requirements modeling,* Proc. of IEEE Symposium on Requirements Engineering, San Diego, Califonia.

Macaulay, A. (1996) *Requirements Engineering,* Springer-Verlag.

Nilsson, C. (1990) *Handbok I QFD. Kundorienterad produktutveckling,* Mekanikerförbundet, Mekanresultat 90002.

Patel, N. V. (1999) The spiral of Change Model for Coping with Changing and Ongoing Requirements, *Requirements Engineering Journal,* Vol. 4, pp. 77-84, Springer-Verlag London, Limited.

Paul, R. J. and Macredie, R. D. (1999) Guest Editorial: Managing Dynamic Requirements. *Requirements Engineering Journal,* Vol. 4, pp. 63-64, Springer-Verlag London, Limited.

Pohl, K. (1996) *Process-Centered Requirements Engineering,* Research Studies Press Ltd.

Rationale (2001) *Rational RequisitePro: Product Overview*, http://www.rational.com/products/reqpro/index.jsp, 2001-07-25.

Sommerville, I. and Sawyer, P. (1997) *Requirements Engineering - A good practice guide*, John Wiley & Sons Ltd.

Stevens, R., Jackson, K., Brook, P. and Arnold, S. (1998) *Systems Engineering - cooping with complexity*, Prentice Hall.

Telelogic (2001) *Telelogic DOORS/ERS*, http://www.telelogic.com/products/doorsers/, 2001-07-25.

Webster's (1996) Webster's Encyclopedic Unabridged Dictionary, Random House.

Willcocks, L. and Lester, S. (1993) How do organizations evaluate and control information systems investments? In Avison, D., Kendall, J. E. and DeGross, J. I. (eds), *Human, Organizational, and Social Dimensions of Information Systems Development*, Elsevier Science Publishers B.V, pp. 15-39.

Appendix A

Requirements Attribute Kotonya and Sommerville (1998):

- *Identifier*
  As mentioned above, to be able to store information about requirements, it is important to have some means to uniquely identify the requirement.
- *Statement*
  This is a statement or description of the requirement. It may be in natural language or a graphical description of some kind e.g. data flow model or timing model.
- *Date_entered*
  The date that the requirement where entered into the database.
- *Date_changed*
  The date that the last change where made to the requirement.
- *Source*
  This attribute contains one or more sources of the requirement. It is a support for making a decision when changes to the requirement are proposed.
- *Rational*
  This is a rationale explanation why the requirement exist and has been included in the database. The explanation may include text, diagrams, figures and photographs.
- *Status*
  This attribute presents the current state of the requirement. The various status categories may change between different project and/or organisation, depending on which status of the requirements are important to keep track of. Examples of status categories is; *proposed*, *under review*, *accepted* or *rejected*, and *implemented*. Rejected requirements should be maintained as well thou the may be proposed again in the future. This simplifies the process of handle new proposals.
- *Dependants*
  This is a list of requirements, which are dependent on the requirement, and are affected if the requirement is changed.
- *Is_dependent_on*
  This is a list of requirements on which this requirement is dependent. A change of some of these requirements is affecting this requirement.
- *Model_links*
  This links the requirement to one or more models and diagrams, which add details to the requirement.
- *Comments*
  This attribute gives the analyst the opportunity to add other information that may be useful. These general fields are often very useful, thou in practice it is almost impossible to cover everything with the defined attributes.

Appendix B

Requirements Attribute Leffingwell and Widrig (2000):

- *Status*
  Keep track of the progress of the specification of the requirement (see 'status' above).

- *Priority/Benefit*
  This attribute presents the priority of the requirement and shows the benefit or importance of the requirement. This attribute may be used to decide and managing scope of the project. Examples of priority categories are: Low, Medium and High, or 1 to 5 where 1 is low and high.

- *Effort*
  Contains the time, e.g. in person-weeks, or lines of code estimated for implemented the requirement. This gives a support to set what can be accomplishing within a certain time period.

- *Risk*
  This is the possibility that the requirement will cause problems like, overruns, schedule delays or higher cost for the project.

- *Stability*
  Is the probability that the requirement will change or that the developers or stakeholders view of the requirement will change. May be useful to see which requirements where additional knowledge needs to be elicited.

- *Target release*
  Shows which product version in which the requirement first shall be implemented.

- *Assigned to*
  Contains information about which team or person responsible for the design and implementation of the requirement.

- *Reason*
  This attribute is used to track the source of the requirements; which may be document as well as people (see 'source' above).

Appendix C

Requirements Attribute Stevens et al (1998):

- *Source*
  Is the person(s) asking for the requirement (see ‘reason’ and ‘source’ above).

- *Priority*
  Shows the importance of the requirements (see ‘priority/benefit’ above)

- *Performance*
  Contains information about how quickly the requirements must be met, i.e. how quick the systems perform the thing asked for in the requirement.

- *Urgency*
  Presents how soon the requirement is needed, i.e. how soon the requirement must be implemented in the system.

- *Stability*
  This attribute contains information about the stability of the requirements (see ‘stability’ above).

- *Verifiability*
  Is it possible to test the final product against this requirement, i.e. is the requirement verifiable?

- *Ownership*
  Lists the people(s) needed the requirement.

- *Acceptance criteria*
  Describes how the users acceptance of the requirements should be controlled.

- *Absolute reference*
  Uniquely identifies the requirement (see ‘identifier’ above).

Appendix D

Requirements Attribute Karlsson (1996):

Karlsson (1996) divides the attribute into two broad categories, administrative and technical. Administrative attributes are used to manage the requirements through the development process, and during review and maintenance. Technical attributes are used to describe the requirements. These are described as elements, defining groups of attributes.

Administrative attributes:

- *Identifier*
  Identifies the requirement uniquely (see 'identifier' and 'absolute reference' above).

- *Configuration*
  This attribute shows configuration of the requirements, which changes has been done to the requirement, who authorized them, why change, etc (see 'date_entered' and 'date_changed' above).

- *Version*
  It is important to know in which version of the system this requirements will be/is implemented in (see 'target release' above)

- *Target group*
  Describes the people/stakeholders that are interested in the requirement. The needs of the various stakeholders differ, e.g. the testers view of the requirements is rather different from the customers view. This affects how the requirement should be described (see 'ownership' above)

- *Links*
  In every developing project, there are relations between the requirements. This attribute should document such relations (see 'dependant' and 'is-dependent-on' above).

- *Status*
  Describes the maturity of the requirement, is it stated, under realisation, implemented, rejected, etc (see 'status' above)

- *Keywords*
  It is often useful to be able to identify the requirements with a couple of keywords. This helps when to identify sets of requirements.

Technical attributes:

- *Description*
  This attribute describes the requirements, mostly in natural language. This description should only cover one requirement, if not the requirement should be further expanded (see 'statement' above)

- *Properties*
  This element includes several different attributes that may be very useful in the development process, such as:
  - motive (see ‘rational’ above)
  - priority (see ‘priority/benefit’ above)
  - volatility (see ‘stability’ above)
  - validation

- *Verification*
  This element comprises different aspects concerning the verification of the requirement, i.e. how should the fulfilment of the requirements be controlled (see ‘verifiability’ above). Examples of attributes are:
  - accepted variances of the requirement
  - test methods
  - test plans

- *Structure*
  This element describes the structure on the requirement, and on the relation between different requirement, e.g. between high-level and low level requirements.

- *Comments*
  It there is any additional information not suitable for any of the other attributes (see ‘comment’ above).

Appendix E

Attribute described by more than one source

The table below describes which attributes that are mentioned by more than one author. We have used the description of the requirements as a basis, because the authors may have different names on similar attributes. Therefore, the attribute names mentioned by the authors respectively are also presented.

Attribute name	Kotonya and Somerville (1998)	Leffingwell and Widrig (2000)	Stevens et al (1998)	Karlsson (1996)
Identifier	Identifier	-	Absolute reference	Identifier
Statement	Statement	-	-	Description
Source	Source	Reason	Source	-
Rational	Rational	-	-	Properties (motive)
Status	Status	Status	-	Status
Comment	Comment	-	-	Comment
Priority	-	Priority/ Benefit	Priority	Properties (priority)
Stability	-	Stability	Stability	Properties (volatility)
Target release	-	Target release	-	Version
Target group	-	-	Ownership	Target group
Links	Dependants & Is-depentant-on	-	Structure	Links
Configuration	Date-entered/ Date-changed	-	-	Configuration
Verification	-	-	Verifiability	Verification

Appendix F – Questionnaire

Personal questions

1. Can you describe your what your working tasks are at your company?
2. How long, and with what have you worked within this area?

Requirements Engineering

1. What are the procedures to develop the requirements for your products?
2. What do you do if you think that the requirements are too fuzzy and high-level? What are your procedures to develop the underlying requirements?
3. How do you develop the requirements specification?
4. What are your routines for electing the requirements that should be included in the requirements specification?
5. How do you document the requirements?
   - text?
   - models?

Requirements Management

1. How do you control that low-level requirements (underlying) are consistent with the high-level requirements?
   (Documentation of relation between these? Consistency checking?)
2. How do you keep track of which requirements that are validated and agreed and that shall be included in the requirements specification?
3. How do you keep track of the prioritisation of the requirements? (How do you know which requirements that shall be implemented in the RS?)
4. How do you keep track of which requirements that are implemented and which is not?
5. How do you manage changes of requirements?
6. How do you keep track of the dependencies and relations between requirements?
7. How do you manage implicit requirements? How do you manage the models? How do you manage relationships between requirements and models?
8. How do you manage ambiguous requirements? How do you know which one to use?

Management of different versions

1. What are your procedures when you are developing a new version of a system?
2. What do you use as a base for the development of a new version?
   (Are you using old requirements specifications?)
3. What are the pros and cons with this?
   (How correct are the old RS?)

Problems within RM.

1. What do you se as the major issues and problems within RM?