LAPPEENRANTA UNIVERSITY OF TECHNOLOGY
Department of Industrial Engineering and Management

REQUIREMENTS TRACEABILITY

The subject of this thesis has been approved by the Council of the Department of Industrial Engineering and Management on the 29th of August, 2001.

Supervisor: Professor Tuomo Kässi
Instructor: Master of Science in Engineering Laura Nihti

Lappeenranta 10.10.2001

Mia Hokkanen
Ruotsalaisenraitti 2 B 31
53850 Lappeenranta
Tel. +358 (0)40 703 9647

ABSTRACT

Author: Mia Hokkanen

Name of thesis: Requirements Traceability

Department: Industrial Engineering and Management

Year: 2001 **Place:** Lappeenranta

Master's Thesis. Lappeenranta University of Technology.

84 Pages, 15 Figures, 7 Tables and 2 Appendices

Supervisor Professor Tuomo Kässi

Instructor Master of Science in Engineering Laura Nihti

Keywords: software engineering, requirements engineering, requirements traceability

Hakusanat: ohjelmistotuotanto, vaatimusmäärittely, vaatimusten jäljitettävyys

Requirements engineering is an important part of software engineering. Requirements traceability is a part of the requirements management process and it makes requirements management easier through the whole product development project. However, requirements traceability is very often ignored in software development projects.

The objectives of this thesis are to establish the importance of requirements traceability in software development projects and the methods to carry out requirements traceability information. Requirements traceability and implementation techniques are studied through the literature. The present state of requirements traceability in an organization is studied through the real process model and real projects.

As a result of the study there are reasons why requirements traceability information should be carried out in software development projects and procedures how the implementation could be started cost-effectively. There are also recommendations of the traceability strategies and methods. With minor corrections requirements traceability is easy to carry out in some level. Creating traceability matrixes is the biggest improvement proposal for the process model. Matrixes ensure traceability information in backward and forward directions. A proposed requirements management tool would help to maintain traceability information.

TIIVISTELMÄ

Tekijä: Mia Hokkanen

Työn nimi: Vaatimusten jäljitettävyys

Osasto: Tuotantotalous

Vuosi: 2001 **Paikka:** Lappeenranta

Diplomityö. Lappeenrannan teknillinen korkeakoulu

84 sivua, 15 kuvaa, 7 taulukkoa ja 2 liitettä

Tarkastajana professori Tuomo Kässi

Ohjaajana diplomi-insinööri Laura Nihti

Hakusanat: ohjelmistotuotanto, vaatimusmäärittely, vaatimusten jäljitettävyys

Keywords: software engineering, requirements engineering, requirements traceability

Vaatimusmäärittely on tärkeä osa ohjelmistotuotantoa. Vaatimusten jäljitettävyys on osa vaatimustenhallinta prosessia. Jäljitettävyystieto helpottaa vaatimusten hallintaa läpi koko tuotekehitys projektin. Hyvin usein vaatimusten jäljitettävyyttä ei kuitenkaan ole toteutettu ohjelmistokehitysprojekteissa.

Työn tavoitteena oli selvittää vaatimusten jäljitettävyyden tärkeyttä ohjelmistotuotannossa sekä kuinka jäljitettävyys voitaisiin toteuttaa ohjelmistokehitysprojekteissa. Vaatimusten jäljitettävyyttä sekä eri tekniikoita sen toteuttamiseksi on tutkittu kirjallisuuden avulla. Yrityksen vaatimusten jäljitettävyyden nykytilaa on selvitetty tutkimalla olemassa olevaa prosessimallia sekä todellisia tuotekehitysprojekteja.

Tuloksena esitettiin perusteluja, miksi jäljitettävyystieto pitäisi sisällyttää ohjelmistokehitysprojekteihin sekä menetelmiä, kuinka jäljitettävyystieto voidaan toteuttaa projekteissa kustannustehokkaasti. Työssä on esitetty strategiavaihtoehto ja menetelmät jäljitettävyyden toteuttamiseksi. Pienillä korjauksilla jäljitettävyys pystytään toteuttamaan kevyellä tasolla. Suurin parannusehdotus prosessimalliin on jäljitettävyysmatriisien luominen. Matriisien avulla pystytään projekteissa toteuttamaan jäljitettävyys sekä eteen- että taaksepäin. Vaatimustenhallintatyökalu helpottaisi jäljitettävyystiedon ylläpitoa.

ACKNOWLEDGEMENTS

The subject of this thesis was given by Sonera Service Software, which is Sonera's research and development unit.

I would like to thank my instructor Laura Nihti for the invaluable suggestions and guidance she gave during the study. She helped me to find the right focus for this study in the beginning and without her support I could not have finished this study so quickly. I am grateful to my supervisor, Professor Tuomo Kässi for his valuable instructions and dedication to instructing and supporting me. I would like to thank Jarkko Lehto for his valuable comments and for giving me time to concentrate on this thesis. I would also like to thank people in Sonera for their support. I especially thank all those people who I interviewed for their valuable input for the study. Special thanks to Janne for always supporting me.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

AMR	Analyzed Main Requirements
DP	Decision Point
FR	Functional Requirement
ID	Identification
MS	Mile Stone
O&M	Operation and Maintenance
Post-RT	Post Requirements Traceability
Pre-RT	Pre Requirements Traceability
RC	Requirement Catalog
RE	Requirements Engineering
RM	Requirements Management
RM-tool	Requirements Management tool
RS	Requirements Specification
RT	Requirements Traceability
RUP	Rational Unified Process
R&D	Research & Development
SRS	Software Requirements Specification
S3	Sonera Service Software
TI	Traceability Information
TIP	Technical Implementation Proposal
TS	Technical Specification
UC	Use Case

1 INTRODUCTION

1.1 Background

In a very competitive software development environment the quality and fastness of software development processes are remarkable competitive advantages. Development projects should be concluded on time, on budget and the end result should be the same as the customer is expecting. All these aspects measure the quality of projects but unfortunately many of the software development projects fail at least in one of these aspects. There are many reasons for these failures and results of the Standish Group International (2001) sample research show that four main reasons for project challenged factors are lack of user input, incomplete requirements and specifications, changing requirements and specifications, and lack of executive support. Thus, one way to increase the quality and fastness of the software development process is to improve requirements engineering (RE) and management processes. Requirements traceability (RT) is an important part of the requirements management (RM), but too often it is not carried out completely if at all in development projects.

In software development projects everything is very liable to changes and so iterative process models have become popular. The more iterative are the projects the harder it is to manage them. There will always be some changes in requirements during development projects. Traceability information (TI) shows which components have to be changed or held constant and which components will be affected by the changes. Shortage of the RT information may lead to higher costs, wrong or unnecessary changes, impossible reuse of components, harder fault tracking, frustrating waste of time and many other problems during a development project. If the TI is reliable, changes will be made correctly and completely during development project, which improves the productivity. The RT information helps to

manage requirements. Well-managed requirements improve the total quality of the development project.

Thus importance of RT and its benefits raise an interesting research topic. How could a company use the traceability information and for what? What are the benefits of the traceability information? What kind of traceability information is needed? How can a company get the traceability information? How can the traceability information be exploited? These are some of the questions for this thesis to approach through literature and real organizational settings in Sonera Corporation.

1.2 Arguments for Requirements Traceability

Why should requirements be traceable in software development projects? RT helps development projects succeed. With RT information requirements are easier to manage. When requirements management is easier the impact of the changes in development projects is also easier to evaluate. In addition, traceability information assists to verify that all requirements are implemented. All these issues make successful product development projects possible. Successful product development leads to better customer satisfaction. Thus RT finally enhances the competitive advantage of the company.

Traceability information is not freely available and some work has to be done to get it. The challenge is to get people convinced about the importance of the traceability information and to make them realize that it should be available in development projects as soon as possible. To convince people, some good reasons have to be given for the question why RT is so important that the firm should invest on it.

1.3 Objectives of the Study

Three objectives are identified to answer the research question of this thesis. *The first objective of the thesis is to identify the use of the requirements traceability information.* This is done through theoretical and empirical analysis. The aim is to

find answers for the questions how and what traceability information could be used. The thesis tries to create a clear picture of the use and advantages of the RT.

The second objective of the thesis is to find out what kind of information should be traced and how. Tracing requirements through the development project assemble additional expenses, which brings more challenges to the analysis of the needed information. The information that should be traced is identified through theoretical and empirical study. Empirical study will be used to create a picture of the present state of RT in Sonera and to identify what kind of requirements information could be useful for the company. As a result of the empirical study there will be an answer for the questions how RT information is implemented in the company's development projects, what information should be traced and how. The goal is to identify how RT can be carried out in a general level not in a detailed level. Anyhow always should be remembered that all necessary things should be traced but nothing more or less.

1.4 Research Methods

This thesis consists of a literature study and a case study. The concepts of software and requirements engineering are studied through literature. RT is studied through literature and empirical study. The aim is to define RT through literature, compare it to practice and thus give some recommendations and improvement proposals. Benefits, advantages and necessity of RT will be justified by combining the theoretical and empirical studies.

Empirical study is done through practical examples of the development projects in the company and in this way the current situation of RT is defined. This study is based on going through the material of the product development process model of the company and the material of the product development projects. The level of RT in the development projects is also analyzed by discussing and interviewing people involved in those software development projects. These discussions and

interviews are analyzed based on theoretical study. Proposal for the traceability information of requirements, which the company should have, is based on theoretical and empirical study. Empirical study identifies what kind of requirements should be traceable especially in the development projects of Sonera.

1.5 Scope of the Study

This thesis concentrates on reasoning the importance of RT in software development processes that concentrate on new products. The thesis argues and gives good reasons for a company to invest in requirements traceability information. RT is considered here only through one development project from requirements to testing. RT through different versions of the product is excluded from this study. At the end of the study there will be defined what information should be traceable because traceability information always costs and so all necessary things should be traced to reach majority of the benefits of RT.

The empirical study concentrates on Sonera's R&D unit Sonera Service Software (S3). The tracking of requirements in a real project is excluded from this study but the comparison between the theory and the present state of the requirements traceability in S3 is included. Recommendations of this study are directed to S3.

1.6 Structure of the Work

The structure of the study is illustrated in figure 1.

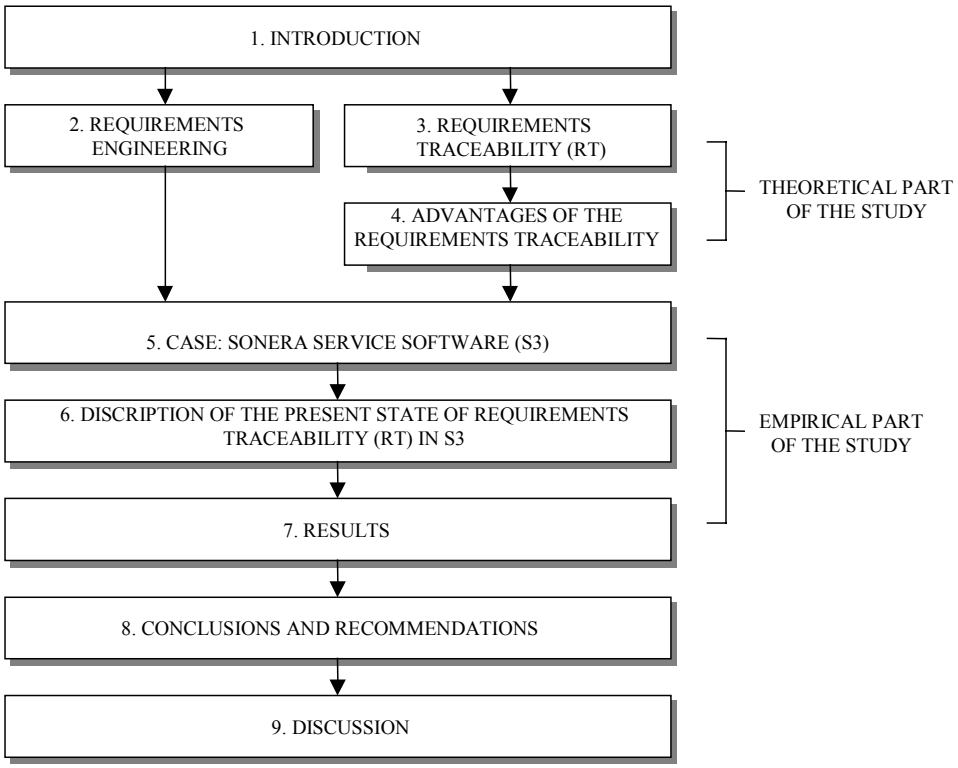


Figure 1. Structure of the study.

2 REQUIREMENTS ENGINEERING

The core of software engineering is constructing and maintaining computer-based systems. The first explicit model for the software development process was the "waterfall model", which cascades from one phase to another (Reinikainen 2000, p. 5). The waterfall model is illustrated in figure 2. However, nowadays many models exist for the software development life cycle. Thayer and Dorfman (1997, p. 8-11) mention five basic development models, which are the baseline management and waterfall models, the prototyping life cycle model, the incremental development model, the evolutionary development model, and the spiral model.

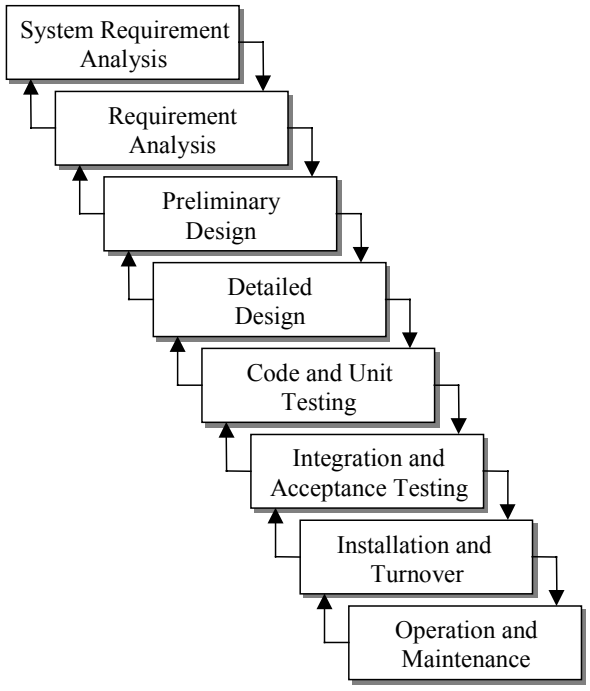


Figure 2. Illustration of a software development life cycle (Thayer & Dorfman 1997, p. 454).

Software engineering typically starts with Requirements Engineering (RE). The scope of the RE is to define the purpose of a proposed system and to outline its

external behavior. According to Sommerville & Sawyer (1998, p. 5) RE is a relatively new term which includes all of the activities involved in discovering, documenting, and maintaining a set of requirements for a computer based system. Traditionally RE was considered to be restricted to a particular phase of the software development life cycle. Normally this phase occurred before design, implementation, testing, and utilization. This view is based on the earlier mentioned waterfall model (figure 2). Nevertheless, this view has been changed during last decades and it is now generally accepted that RE process will continue through the whole software development process, as requirements are continually being refined throughout the life cycle. (Lauesen 2000, p. 10) (Goguen & Linde 1993, p. 152)

RE activities are often divided into five categories in literature. These five categories alternate between different writers but the contents are almost the same. Following will be presented common definitions for these five categories (Thayer & Dorfman 1997, p. 1) (Wiegers 1999, p. 43):

- *Requirements elicitation.* The process through which the customer and the developer of a software system discover, review, articulate, and understand users' needs and verifies user requirements through discussion. There are three common levels of requirements elicitation: business, user, and functional, which come from different sources. In this phase constraints are also set up on the software and the development activity.
- *Requirements analysis.* The process of analyzing the customers' and users' needs to achieve definitions of software requirements. All stakeholders have to understand the meaning of each requirement.
- *Requirements specification (RS).* The development of the document that clearly records each of the requirements of the software system, and possibly it is formalized to act as a basis for contractual purposes between the customer and the supplier of the product. In this phase requirements need to be documented in some consistent way and some standard convention must be established to uniquely identify each requirement.

- *Requirements validation/verification.* The process of ensuring that the software requirements specification is in compliance with the system requirements. Requirements' correctness (such as completeness and consistency) and feasibility properties (such as cost and resources needed) are evaluated and analyzed.
- *Requirements management (RM).* Assists in maintaining the requirements' evolution through the development project. RM consist of requirements elicitation, specification, analysis, and verification, and includes also planning, traceability, impact assessment of changing requirements and so on.

Wiegers (1999, p.268) points out that RM includes all activities that maintain the integrity and accuracy of the requirements agreement as the project progresses.

2.1 Requirements Engineering

Wiegers (1999, p. 19) defines RE activities as illustrated in figure 3. One minor difference for the definition depicted above is that Wiegers groups elicitation, analysis, specification and verification activities into one group named requirements development. This Wiegers definition of RE will be followed in this thesis.

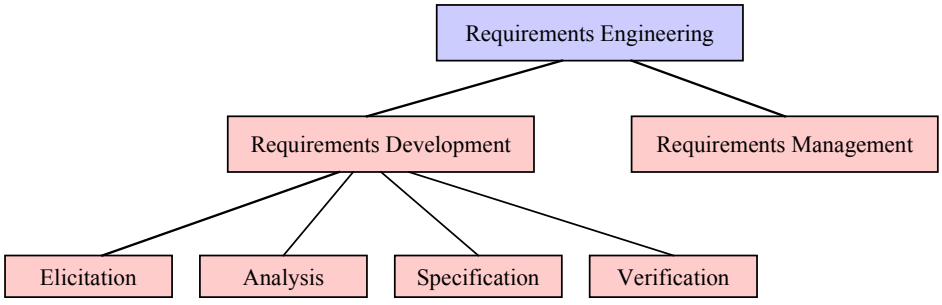


Figure 3. Hierarchical decomposition of the requirements engineering domain (Wiegers 1999, p. 19).

Wiegers (1999, p. 38) introduces some good practices for RE, which could help development teams do a better job on their requirements activities. These activi-

ties are roughly described in table 1. Good practices are grouped under same activities as in figure 3, but there is two additional groups, knowledge and project management, which have some important tasks related to RE process. Practices of these two groups will be discussed briefly because there are some tasks related to requirements management. Requirements management itself is discussed in more detail in chapter 2.2.

Table 1. Requirements Engineering Good Practices (Modified from Wiegers 1999, p. 38-39).

Knowledge	**Requirements Management**	**Project Management**
- Knowledge of requirements analyst - User reps and managers knowledge of requirements - Knowledge of the developers in application domain - Glossary for the terms	- Change management process - Requirements traceability - Versions control of requirements documents - Requirements status tracking - Measure requirements stability - Use a requirements management control	- Select appropriate life cycle - Base plans on requirements - Renegotiate commitments - Manage requirements risks - Track requirements effort

Requirements Development			
Elicitation	*Analysis*	*Specification*	*Verification*
- Vision and scope of the product - Define requirements development procedure - Establish focus groups - Identify use cases - Analyze user workflow - Define quality attributes - Reuse requirements	- Requirements' context diagram - Possible prototypes - Feasibility analysis - Requirements prioritization - Model the requirements - Create a data dictionary	- Adopt Software Requirements Specification (SRS) template - Sources of requirements - Requirements' labeling - Business rules recording - Requirements traceability matrix creation	- Requirement documents inspection - Test cases writing from requirements - User manual writing - Defining acceptance criteria

As Wiegers (1999, p. 42) points out, good *knowledge* of RE is important for people involved in RE process and it ensures better quality for the software development process. Requirements process is a key to success and so all project stakeholders should have a basic understanding of the rationale, importance, and practices of RE. People, who are primarily responsible for capturing, documenting, and analyzing user requirements should receive a large-scale training in these activities.

There are three levels of requirements *elicitation*: business, user, and functional. These come from different sources at different times during the project, have different audiences and purposes, and need to be documented in different ways. The business requirements cannot exclude any user requirements, and functional requirements should be traceable to user requirements. Nonfunctional requirements, such as quality attributes, should also be elicited from the most appropriate sources. (Wiegers 1999, p. 43) (Thayer & Dorfman 1997, p. 1)

Software *project management* approaches are intimately related to a project's requirement processes and especially to requirements management processes. Project plans should be based on the functionality that is to be built, and changes in requirements will affect on those plans. The project plans should therefore anticipate and accommodate expected changes in requirements and project scope. If the initial requirements are uncertain, a software development cycle that accommodates this uncertainty might be selected (e.g. the waterfall software development life cycle is appropriate only for fully defined initial requirements). (Wiegers 1999, p. 52-53)

According to Wiegers (1999, p. 148-149) a document referred to the Software Requirements Specification (SRS) states the functions and capabilities that a software system must provide and the constraints that it must respect. If the quality of this document is good, the rest of the development project is much easier to manage because the SRS is the basis for all subsequent project planning, design, and coding, as well as the foundation for system testing and user documentation.

Good quality of SRS means that it is understood and agreed by all stakeholders and everybody is committed to it. The SRS should also be consistent internally and with the existing business practice documents. Requirements in the SRS must be complete, implementation independent, unambiguous and consistent, precise, and verifiable (Dorfman & Thayer 1996, p. 91). The better the quality of the SRS the easier is the task for the requirements management and project management. Good quality of the SRS ensures that budget, resource, and lifecycle estimates will keep on. Likely there are also fewer changes to be made during development, which also helps the requirements management processing.

2.2 Requirements Management

According to Sommerville and Sawyer RM is concerned with all of the processes involved in changing system requirements. RM is a process, which support other requirements engineering activities and is carried out in parallel with them. Definition of RM pointed out by Leffingwell and Widrig (2000, p. 16) is as follows ”a systematic approach to eliciting, organizing, and documenting a process that establishes and maintains agreement between the customer and the project team on the changing requirements of the system.” The principal concerns of RM are:

1. Managing changes to agreed requirements
2. Managing the relationships between requirements
3. Managing dependencies between the requirement document and other documents produced during the systems and software engineering process. (Sommerville & Sawyer 1997, p.216)

In common sense RM is a critical activity – it ensures the voice of customer is heard throughout the development process and that RE is not restricted to a single phase in the lifecycle (Gotel 2000, p. 5). Good practices for RM were described in table 1. Wiegers (1999, p. 268) groups these practices into four main categories. Categories and activities of these categories are described in figure 4.

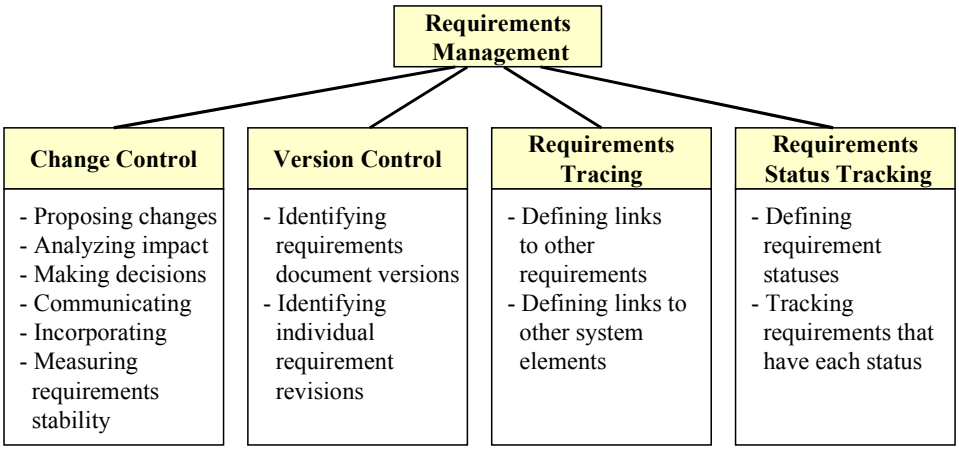


Figure 4. Major requirements management activities (Wiegers 1999, p. 268).

According to Sommerville and Sawyer (1997, p.217) to manage requirements, requirements traceability information need to be maintained. Traceability information helps to discover what other requirements might be affected by requirements changes. RM practices such as maintaining dependencies between requirements have long term benefits, which are better customer satisfaction and lower system development costs. These benefits will not appear immediately and so developers may consider RM as an overhead. Implementing RM in development projects may need some additional work at first, which makes it more difficult to convince people about its importance. However, the experience has shown that the investment in good requirements management processes is cost-effective.

3 REQUIREMENTS TRACEABILITY (RT)

The central task of RM is to assure requirements traceability, from the earliest elicitation activities through to system evolution and maintenance. Requirements traceability (RT) is at the heart of RM (Gotel 2000, p. 6) (Easterbrook & Nuseibeh 2000, p.6). Figure 5 depicts the role of the RT in the software engineering (SE) and shows that RT is in the middle of everything. If RT is implemented successfully it has positive impact on the whole project and improves the quality of the project and so it should be considered as an important part of the software engineering (Ramesh et al., 1995).

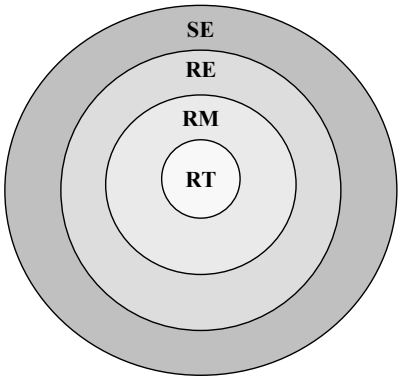


Figure 5. The location of the RT in the software development process.

According to Robertsons (1999, p. 265) a requirement is traceable if any part of the product that exists can be identified because of the requirement, and for any part of the product can be identified the requirement that caused it. Gotel & Finkelstein (1994, p.1) created a common definition for the term requirements traceability. This definition cited by many authors is as follows:

"Requirements traceability (RT) refers to the ability to describe and follow the life of a requirement in both a forwards and backwards direction (i.e. from its origins, through its development and specification, to its subsequent development and use, and through periods of ongoing refinement and iteration in any of these phases)."

RT is often thought as a concern in an increasing number of standards and guidelines for systems and software requirements engineering. This concern has occurred because of the variety of methods and tools have been developed to address RT issues and research interest in the area is growing. RT is a widely reported problem area in industry and the persistence of RT problems can be attributed to the lack of any thorough problem analysis (Gotel & Finkelstein 1994, p.1).

3.1 Classifications of RT

Traceability has been classified in many ways. The most common ways occurring in literature are the pre- and post –traceability (Gotel & Finkelstein 1994a, p.11), forward- and backward –traceability (Gotel & Finkelstein 1994a, p.10), and traceability types (Sommerville & Sawyer 1997, p. 226). It should be noted that any of these classifications do not close other classes out, in contrast they support and augment others.

3.1.1 Pre- and Post- Requirements Traceability

Pre- and post-requirements traceability classification divides RT into two basic types, which together encompass the whole area of RT. The following are the definitions for these terms defined by Gotel (1995, p. 78).

Pre- requirements traceability (pre- RT) refers to the ability to describe and follow those aspects of a requirement's life prior to its inclusion in the RS in both a forwards and backwards direction (i.e., requirements production and refinement).

Post- requirements traceability (post- RT) refers to the ability to describe and follow those aspects of a requirement's life that result from its inclusion in the Requirements Specification (RS) in both a forwards and backwards direction (i.e., requirements deployment and use).

There are two main phases of a requirement's life, its on-going production and its development. Pre- and post-RT are grouped based on these two basic lifecycle phases. Figure 6 shows a typical simplified setting of RT to illustrate the difference between pre- and post-RT. These two types deal with separate kinds of information, assist with different types of problem, and have different claims on potential support for traceability. (Gotel 1995, p. 78)

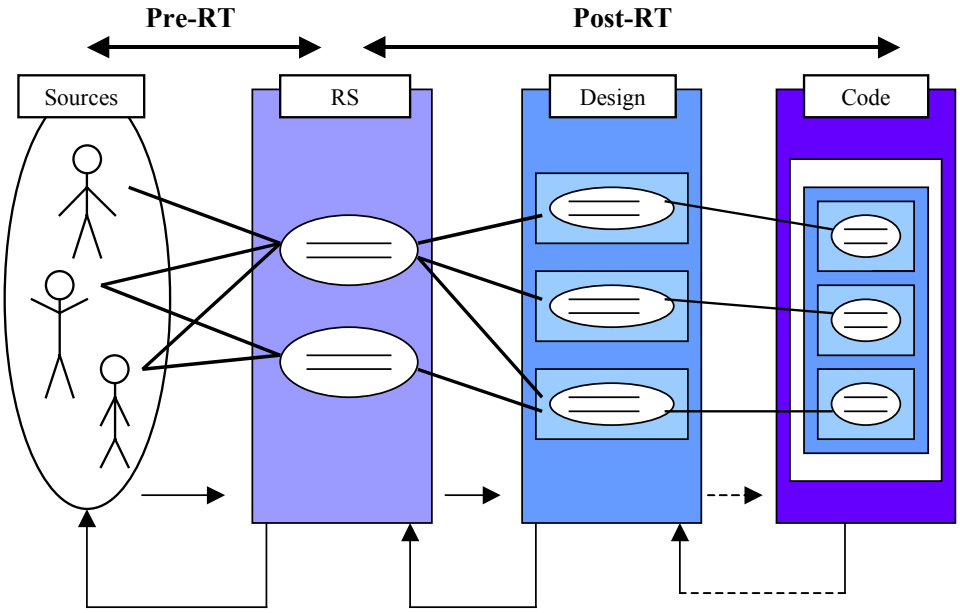


Figure 6. A simplified diagram to show the two basic types of RT, pre-RT and post-RT (Gotel 1995, p. 79).

Pre-RT is the requirements development part of RE (figure 2), which includes elicitation, analysis, specification, and verification. The output of pre-RT is the basis of the Software Requirements Specification (SRS) document, which includes traceability information before design phase. If this initial work of RT is well done, the quality of software development processes rises very likely. The more complete and consistent SRS set is the fewer and/or smaller changes will be made to the product during development. This is the reason why authors often emphasize the meaning of pre-RT and argue that post-RT have only limited affect on the quality of the development process.

Gotel (1995, p. 79) says that "Post-RT depends on the ability to trace requirements from and back to a relatively static baseline document, usually the RS, and through a succession of documents and products in which they are distributed." Post-RT helps to manage changes made to requirements baseline (SRS) through the chain of requirements distribution and helps to assess impacts of changes on the development project. Hence, even if the pre-RT is almost complete, the post-RT cannot be implemented without good instructions and planning. Post-RT needs to have some policies so that it could be implemented effectively. In the literature there are some common policies of RT that could be considered when companies own traceability policies are defined. Every company has to determine policies, which are appropriate to the development environment it is working on. This thesis considers traceability policies and manual in more detail later and it is considered as an important part of the empirical study.

Problems with post-RT occur mainly when the activities and frameworks of RE deviate from each other. This problem can be eliminated by formal development settings instead of informal development methods. Post-RT operates from baseline SRS and if this baseline is defective, difficulties might be assumed in post-RT too. Thus, post-RT cannot affect on the quality of pre-RT even though the quality of pre-RT affects on post-RT. Ultimate requirements sources need to be traceable to support such changes in the baseline and to assess their impact. (Gotel 1995, p. 79-80)

3.1.2 Forward- and Backward Traceability

One classification of RT is the forward- and backward RT, which means that a requirement can be followed from its origin through implementation. Figure 6 illustrates four types of forward- and backward links of RT. Customer needs are traced forward to requirements, so if needs change during development the links shows directly what requirements will be affected. This also proves that the specification has addressed all stated customer needs. Requirements are also backward traceable from requirement to its origin. (Wiegers 1999, p. 298)

The right hand side of the figure 7 addresses that, as system requirements flow into software requirements, design, code, and other artifacts during development, requirements can be traced forward by defining links between individual requirements and specific product elements. These types of links assure that every requirement has been satisfied because it can be shown which product components address each one. The fourth type of link traces specific work products backward to requirements and justifies why each item has been created. Most applications include code that is not related directly to user-specified requirements, but every line of code should have a reason to be implemented. (Wiegers 1999, p. 299)

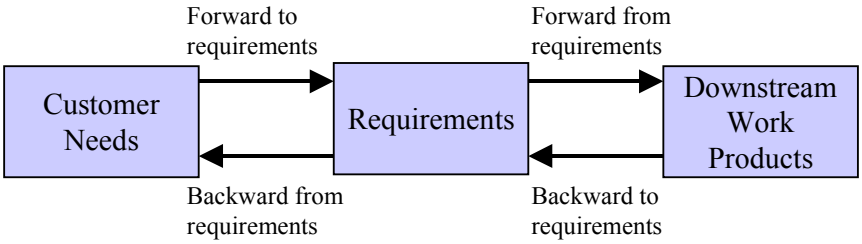


Figure 7. Four types of requirements traceability (Wiegers 1999, p. 299).

Basically forward- and backward traceability refer to the direction in which RT is carried out. RT can be described also in terms of horizontal and vertical RT. These terms refer to the phases of a requirement's life through which RT is carried out. Horizontal RT is traceability between versions and variants of a requirement within a particular phase of its life. Vertical RT illustrates traceability between previous or subsequent phases of a requirement's life in product development process. The distinction between horizontal, vertical, forward, and backward RT is described in figure 8. (Gotel 1995, p. 37)

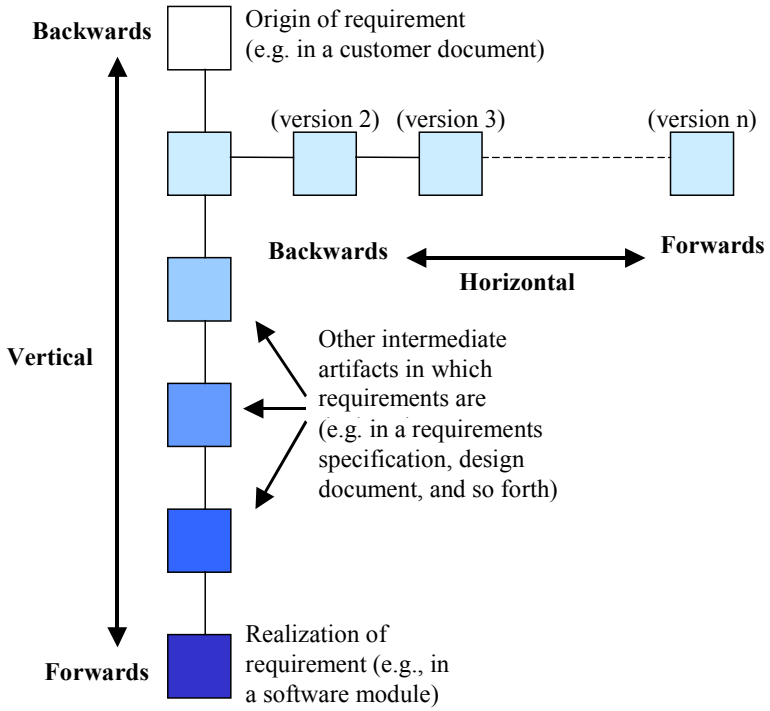


Figure 8. The distinction between horizontal, vertical, forwards, and backwards RT (Gotel 1995, p. 37).

3.1.3 Traceability Types

There are different types of traceability information, which might be maintained during the development project. Figure 9 illustrates eight types of traceability information. This classification classifies RT based on links between requirements and other entities. These types can be categorized into two above-mentioned groups of RT, pre- and post-RT. First four types can be included in pre-RT and last four in post-RT. All links consider forward- and backward traceability.

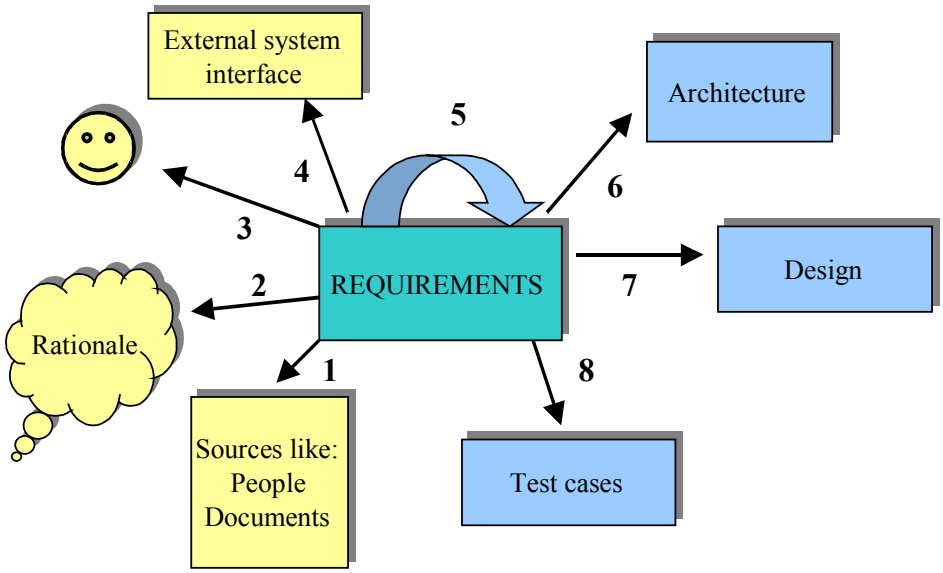


Figure 9. Traceability types (modified from Leino 2001, p. 8).

The eight traceability types presented in the figure are 1) requirements-sources, 2) requirements-rationale, 3) requirements-people involved, 4) requirements-interface, 5) requirements-requirements, 6) requirements-architecture, 7) requirements-design, and 8) requirements-test. These are explained in more detail next.

The requirements-sources (1) is a link to the information on which the requirement is based. A requirement source can be for example a stakeholder, standard, technical document, other document, other requirements or any combination of these etc. This link helps to analyze a requirement and to understand why the requirement exists. If the requirement changes the source can be found easily, which makes change management quicker. (Sommerville & Sawyer 1997, p. 75, 226)

The requirements-rationale (2) links the requirement with a description of why that requirement exists. The rationale associated with a requirement is a link between the problem and the requirements for the proposed problem solution. The rationale makes it easier for readers to understand the requirement and to assess the impact of changes to the requirement. Problem experts can use the rationale to

check if the requirement is consistent with the problem being solved. (Sommerville & Sawyer 1997, p. 87, 226)

The requirements-people involved (3) is a link to the people who have been specifying the requirement. If people involved information is available they are usually located and accessed rapidly when the information of requirements is needed. This gives the possibility to generate information lazily, in other words when it is needed. To record all the information related to the requirement is often unworkable and it is usually desirable to augment it with face-to-face communication. (Leino 2000, p. 14)

The requirements-interface (4) links requirements with the interfaces of external systems, which are used in the provision of the requirements. This link should be maintained where there is a high dependency on other systems. (Sommerville & Sawyer 1997, p. 226)

The requirements-requirements (5) links requirements with other requirements that are, in some way, dependent on them. This type of information should be always maintained (Sommerville & Sawyer 1997, p. 226). Different kinds of relations between requirements are defined in chapter 3.4.1.

The requirements-architecture (6) links requirements with the sub-systems where these requirements are implemented. This is particularly important where different sub-contractors develop different sub-systems. (Sommerville & Sawyer 1997, p. 226)

The requirements-design (7) links requirements with specific components in the system, which are used to implement the requirement. These may be hardware or software components. Requirements allocation to different components is important especially if the system is complex and critical or contains several components with not so fixed functionality. (Sommerville & Sawyer 1997, p. 226)

Requirements-component link can be used to ensure that all requirements are implemented in the system. This link is useful for management to allocate right people to do some critical components, which are defined easier with this link. Also the development schedule is easier to estimate. (Leino 2001, p. 12)

The requirements-test (8) link records which requirement the specific testing artifact is testing. The high-level customer requirements are usually linked to the acceptance tests and the system requirements to the system and integration tests. This link helps to prioritize tests because test can be derived from the priorities of the requirements. When the change to the system requirements takes place, the affected test cases can be identified with the links between the tests and requirements. (Leino 2000, p. 17-18)

3.2 Definitions of Requirements Traceability

There are many partial definitions of traceability. This is because there is a lack of common definition. These partial definitions are purpose-driven, solution-driven, information-driven, and direction-driven definitions or some combinations of these (Gotel 1995, p.71). These definitions are explained in the table 2.

Table 2. Definitions of requirements traceability (Gotel 1995, p. 71-72).

Definition type	RT defined in terms of	Example
Purpose –driven	What it should do.	RT is the ability to adhere to the business position, project scope and key requirements that have been signed off.
Solution –driven	How it should be implemented.	Traceability refers to the ability of tracing from one entity to another based on given semantic relations
Information –driven	The information that should be traced.	RT is the ability to link between functions, data, requirements and any text in the statement of requirements that refers to them.
Direction –driven	The direction in which it should be achieved. (forwards or backwards)	Traceability enables each requirement to be traced to its origin in other documents and to the software component(s) satisfying the requirement.

3.3 Types of Traceability Techniques

A number of basic techniques are identified through which RT can be achieved. Techniques can be fitted into three different types, which are cross reference-centered, document-centered, and structure-centered. These various types of techniques differ in the quantity and diversity of information they can trace between, in the number of interconnections they can represent between information, and in the extent to which they can adapt to reflect changes and so maintain RT throughout a project's life. (Gotel 1995, p. 42)

3.3.1 Cross Reference-Centered

According to Gotel (1995, p. 42) Cross reference-centered technique can be classified to simple cross referencing scheme and to more comprehensive cross referencing scheme. Simple cross referencing techniques are embedded in the main project documentation itself, as would be with phrases like "see Section x" or the repetition of terms and their explanation in a reference glossary. This kind of

technique can be automatically supported if the documentation is held in an online form by hypertextually linking the two ends of a cross-reference or the consecutive occurrences of a term. Examples of this scheme are those based on some form of explicit requirements tagging, numbering, or indexing, and those based on expressing and maintaining specified relationships between key phrases.

Gotel (1995, p. 42) continues that more comprehensive cross referencing schemes supplement the main project documentation, typically with the addition of specialized tables or matrixes, to specifically keep track of cross references. Examples for this scheme are widely used requirements traceability matrixes, as well as their extensions into matrix sequences.

3.3.2 Document-Centered

Document centered techniques ensure RT by dictating either all or part of the structure and content of the project documentation. Examples include those schemes that specify particular forms to fill in, as well as the use of hypertextual document templates within the Document Integration Facility. And there are also schemes, which use the notion of integration documents, or transformation documents, to store the links between documents created in different phases of development. (Gotel 1995, p. 42)

3.3.3 Structure-Centered

Structure-centered techniques enhance the project documentation to achieve RT by restructuring it in terms of an underlying network or graph. These techniques are particularly used when the focus of RT is the update and propagation of requirements changes. (Gotel 1995, p. 43)

3.4 Traceability Techniques

There are some basic techniques, which may be used to maintain traceability information. These are traceability tables, traceability lists, automated traceability links, general-purpose tools, and requirements management tools. Traceability tables are also called traceability matrixes. These techniques are discussed in the following.

3.4.1 Traceability Table

Traceability table is a cross-reference matrix where the entries in the table indicate some kind of traceability link between the items in the rows and the items in the columns. Table 3 describes one kind of traceability table or matrix in which links between use cases, functional requirements, design elements, code, and test cases can be seen.

Table 3. Other example of requirements traceability matrix (Wiegers 1999, p. 303).

Use case	Functional Requirement	Design Element	Code	Test Case
UC-28	Catalog.query.sort	Class catalog	Catalog.sort()	Search.7 Search.8
UC-29	Catalog.query.import	Class catalog	Catalog.import() Catalog.validate()	Search.8 Search.13 Search.14

Table 4 shows how each functional requirement is linked backward to a specific use case, and forward to one or more design, code, and test elements. Design elements can be objects in models such as data flow diagrams, tables in a relational data model, or object classes. Code references can be methods in a class, source code filenames, or procedures or functions within the source file. More columns can be added to extend the links to other work products, such as online help documentation. Traceability links can define one-to-one, one-to-many, or many-

to-many relationships between types of system elements. Table 3 makes it possible to add several items in each table cell and so these different relationships can be carried out.

Table 4. Requirements traceability matrix showing links between use cases and functional requirements (Wiegers 1999, p. 304).

Functional Requirement	Use Case			
	UC-1	UC-2	UC-3	UC-4
FR-1	↵			
FR-2	↵			
FR-3			↵	
FR-4			↵	
FR-5		↵		↵
FR-6			↵	

Table 4 illustrates a simple traceability table or matrix where use cases and functional requirements are linked together. This is a two-way traceability matrix. Each cell at the intersection of two linked components is marked to indicate the connection (Wiegers 1999, p. 304). Different symbols can be used to indicate different relations of connections. According to Sommerville & Sawyer (1997, p. 228) there might exist three different kinds of relations between requirements and these relations can exist also between other work products. These relations are described in the following.

- *Specifies / is-specified-by* relation indicates that some requirement B adds detail to another requirement A.

- *Requires / is-required-by* indicates that some requirement B requires the result provided by some other requirement A.

- *Constrains / is-constrained-by* indicates that some requirement B is constrained by some other requirement A.

When different kind of relations are used, the character and meaning of each requirement is easier to understand. These kinds of matrixes as illustrated in table 4 are more amendable to automated tool support than are single traceability matrixes illustrated in table 3 (Wiegers 1999, p. 304).

Traceability links should be defined by whoever has the appropriate information. Some typical sources of knowledge about links between various types of source and target objects are identified in table 5. The roles and individuals that can supply each type of traceability information in different projects should be defined.

Table 5. Likely sources for the information of traceability links (Wiegers 1999, p. 305).

Link Source Object Type	Link Target Object Type	Information Source
System requirement	Software requirement	System engineer
Use case	Functional requirement	Requirement analyst
Functional requirement	Functional requirement	Requirement analyst
Functional requirement	Software architecture element	Software architect
Functional requirement	Other design elements	Developer
Design element	Code	Developer
Functional requirement	Test case	Test engineer

3.4.2 Traceability Lists

Traceability lists are simplified from traceability table where, along with each requirement description, one or more lists of the identifiers of related requirements

are kept. Lists are more compact than tables and do not become as unmanageable with large number of requirements. Table 6 describes the traceability list, which is a simple list of relationships that can be implemented as text or as simple tables. This table describes dependencies between requirements. There might be several lists, one for each type of relationship such as requires, is-required-by, specifies etc., or a single list of related requirements may be kept. The disadvantage of these lists compared to traceability tables is that there is no easy way to assess the inverse of a relationship. It can be easily seen that R1 is dependent on R3 and R4 but the whole table must be looked through to see which requirements depend on it. (Sommerville & Sawyer 1997, p. 229-230)

Table 6. Traceability list (Sommerville & Sawyer 1997, p. 229).

Requirement	Depends-on
R1	R3, R4
R2	R5, R6
R3	R4, R5
R4	R2
R5	R6

3.4.3 Automated Traceability Links

Automated traceability links means that requirements are maintained in database and traceability links are included as fields in the database record. When requirements are managed in a database, it is easier to maintain links between individual requirements and to search for and abstract related groups of requirements. The appropriate way to implement requirements database depends on the type and the number of requirements, which must be managed and the particular ways of working in an organization. Issues that influence design choices are as follows.

- How are requirements expressed? Is it natural language, graphical models, mathematical expressions, etc., which will be used as forms of requirements?

- How many requirements are typically managed?
- Are the requirements always developed and managed by teams which work together at the same site and which use the same types of computers or is the access for requirements needed from different sites?
- Who will be responsible for database administration or is this a separate responsibility?

If the requirements are managed in a database, the database can be designed to include traceability information. So each requirement in the database should include at least two fields for traceability information. These fields should contain information about other requirements on which the requirement depends on, and requirements, which are dependent on that requirement. However, if there is a need to maintain other types of traceability information such as requirements-architecture links, database must include fields to record information about each different type of relationships. All above mentioned issues and costs need to be considered when an organization is making choices for database system. (Sommerville & Sawyer 1997, p. 227, 236-240)

3.4.4 General-Purpose Tools

Traceability tools do not do the work alone because verification of the requirements demands thinking. However, tools enable project members to inspect the traceability relationships to ensure that no verification relationships have been omitted and that no excess verification relationships are present (Leffingwell & Widrig 2000, p. 333). Leino (2001, p. 19) argues in her thesis that if tracing is implemented manually, it can result in most cases either not being done or performed poorly. General-purpose tools can be configured to support numerous activities and they support also those automated traceability links described in the previous chapter. General-purpose tools include hypertext editors, word processors, spreadsheets, database management systems, prototyping tools, etc. Although traceability is not a concern of such general-purpose tools they can be hand-configured to allow previously manual and paper-based RT tasks to be car-

ried out on-line. Generally this includes establishing cross-references between project documents, or document fragments, and placing conditions upon their automatic update. (Gotel & Finkelstein 1994a, p. 4-7) (Gotel 1995, p.45)

3.4.5 Requirements Management Tools

Complex systems need more dedicated tools than general-purpose tools for implementing RT information because of the volume of the data. Great amount of data makes RT implementation more challenging and more error-prone. Requirement management tools (RM-tools) generally supports RT documenting and other requirements management tasks. RM-tools should depict demanded attributes for requirements such as ID, version number, description, creation date, source, customer priority, and rationale. According to Leino (2001, p. 21) basic functionality of the RM-tools are:

- Documenting information to attributes of requirements
- Filtering and sorting to view requirements
- Importing and exporting requirements to and from the tool
- Documenting and using RT information
- Supporting configuration and change management tasks

Other characteristics for the RM-tools are graphical user interface, compatibility with other tools, and support for simultaneous users. The price of the RM-tool consist of licenses, consultation, training end-users, system maintenance, and integrating the tool to the current environment, and all these aspects should be considered when RM-tool is evaluated. An important factor is how compatible the RM-tool is with other tools used in the development environment. RM-tool should support other tools related to requirements such as creating use cases, documents, and even code. (Leino 2001, p.20-21) (Gotel & Finkelstein 1994A, p. 5-7)

3.5 Managing Requirements Traceability

Requirements management process includes requirements traceability. As mentioned earlier RT is at the heart of RM process. Thus there should be some policies for RM and RT. This thesis concentrates on RT policies and manual and excludes other RM policies. To manage requirements traceability there have to be some policies defined for it.

3.5.1 Traceability Policies

According to Sommerville and Sawyer (1997, p. 224) requirements management policy should define what traceability information should be maintained and how this should be represented. Traceability policies are written policies, which should define the following.

- The traceability information which should be maintained.
- The techniques, which may be used for maintaining traceability (described in detail in chapter 3.4).
- A description of when the traceability information should be collected during the RE and system development processes. The roles of the people should be defined also, such as traceability manager, who is responsible for maintaining the traceability information.
- A description of how to handle and document policy exceptions, that is, when time constraints make it impossible to implement the normal traceability policy. Realistically, there will always be occasions where changes have to be made to the requirements or the system without first assessing all change impacts and maintaining traceability information. The policy exceptions should define how these changes should be sanctioned. The traceability policies should also define the process used to ensure that the traceability information

is updated after the change has been made. (Sommerville & Sawyer 1997, p. 224-225)

In general traceability policies are independent of any particular system. As part of the quality planning process, the most relevant traceability policies should be selected and tailored to the specific needs of the system that are being specified. These should be specified in the system traceability manual. Maintaining traceability information is expensive because it involves managing large volumes of information. Thus traceability policies should be realistic and not too bureaucratic. Lightweight policies, which are followed, are better than comprehensive traceability policies, which are ignored. (Sommerville & Sawyer 1997, p. 225)

Traceability policies may be implemented by organizations at any level of RE process maturity. Simple traceability information is traceability of requirements to their sources and other requirements and this may be the starting point for RT implementation for the organization at the basic level in the RE process maturity model. In general this is a good way to start implementing traceability information. As organizational maturity increases, more complex traceability policies may be introduced. (Sommerville & Sawyer 1997, p. 225-226)

3.5.2 Traceability Manual

A traceability manual is the document used by requirements engineers and system developers and it is a supplement to the requirements document. It includes the specific traceability policies used in a project and all requirements traceability information. The traceability manual assures that project members can easily find the specific traceability policies for their project, and traceability information is in one place and easy to maintain. (Sommerville & Sawyer 1997, p. 232)

Thus, the traceability manual is a central record of the traceability policies and includes all relevant traceability information for a specific project. Projects have different characters and so each project have to evaluate what and how the traceability information should be represented. The traceability manual should also de-

termine a traceability strategy used for the project and the traceability information collection responsibilities. Different traceability strategies are discussed in the next chapter. The specific traceability policies for each project depend on a number of factors. Sommerville and Sawyer (1997, p. 233) groups these factors as follows.

1) *Number of requirements.* The greater the number of requirements, the more the need for formal traceability policies. In the case of a very large number of requirements, developers of the policy have to be realistic about what traceability policies can be implemented in practice. Complete requirements-design traceability is impractical for the most of large systems.

2) *Estimated system lifetime.* More comprehensive traceability policies should be defined for systems that have a long lifetime.

3) *Level of organizational maturity.* Detailed traceability policies are most likely to be cost-effective in organizations, which have a higher level of process maturity. Organizations at the basic maturity level should focus on simple requirements-requirements traceability.

4) *Project team size and composition.* The larger the project team is, the more formal and detailed traceability policies are needed. Basically, if the team is very small, the informal discussion between team members may be enough.

5) *Type of system.* Critical systems such as hard real-time control systems or safety-critical systems need more comprehensive traceability policies than non-critical systems.

The traceability manual is usually developed during the project matures. Traceability policies will be maintained first. Requirements dependencies will be added as soon as the requirements document is agreed. Design, document, etc. traceabili-

ties will be documented at the later stages of the development process. (Sommerville & Sawyer 1997, p. 233)

The most challenging task is to keep the traceability manual up to date. If the document is maintained on paper, there will always be a lag between changes made to the paper document and the document, which is used by the engineers maintaining the requirements and/or the system. Networked electronic document or RM-tool will decrease these problems, which should be considered while traceability manual will be created for the development projects. To ensure the traceability manual updating there should be a named person whose responsibility is to keep the manual up to date.

3.5.3 Traceability Strategies

The traceability strategy is an important part of traceability manual. The strategy defines implicit and explicit traceabilities that will be used in a specific project. Implicit traceability establishes the construction of mappings between the models and relationships between model item themselves. Figure 10 depicts these implicit relationships between models and shows a common classification for a requirement in different phases of its lifecycle. This is only one definition and so the names of the documents can vary in different companies but the content is the same. (Spence & Probasco 1998, p. 3-5)

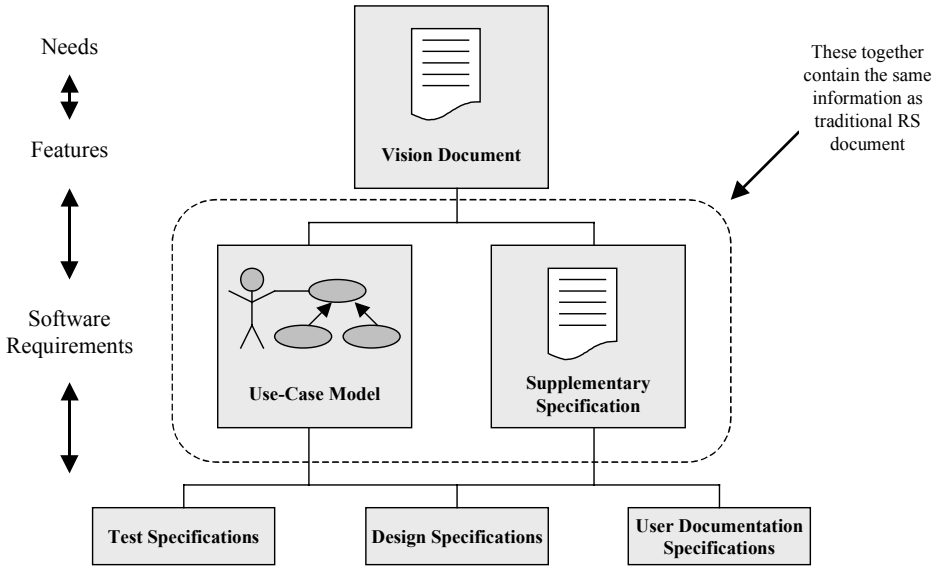


Figure 10. Sample requirements structure (Spence & Probasco 1998, p. 5).

Traceability strategy determines the level of explicit traceability that will be used in software development process. The level of traceability should be suitable for a project so that a return on investment will be got on any additional explicit traceability, which is maintained. Thus, the traceability strategy should be established and evaluated carefully before adopting it in a project. (Spence & Probasco 1998, p. 6)

There are two distinct approaches for strategies, which determines where software requirements are collected. One approach is that all requirements are gathered into the traditional software requirements specification (SRS) document and in the other approach requirements are defined in two places, use-case models and supplementary specification document. The later one is very common and it includes four different strategies how to manage requirements in these two different places and the connection between them. These four strategies are the following:

1) *Use-case model only.* This strategy uses only the use-case model as a statement of the system requirements. This strategy can be chosen only for the projects, which have close association and trust between customer and developer.

Basically the use-case model, glossary, and supplementary specification form the entire statement of the system's requirements. There are no additional definitions of needs, product features or software requirements.

2) *Features drive the use-case model.* In this strategy the use-case model and supplementary specifications form a complete software requirements specification as illustrated in figure 9. Features are documented in the vision document and are traced to use-cases or supplementary specifications. The Rational Unified Process (RUP) recommends this as a default strategy. In this case the use-case model acts as the main statement of the functional requirements.

3) *The use-case model is an interpretation of the SRS.* In this case the use-case model is an interpretation of a traditional SRS. This is used when traditional SRS is required due to regulatory or internal protocol and use-case model enables the practice of use-case driven development. Features are traced into a formal SRS document and software requirements are traced into the use-case model.

4) *The use-case model reconciles multiple sets of traditional software requirements.* The use-case model is the interpretation of a formal SRS from multiple sources and provides the specification of a single common system. In this strategy each stakeholder has their own set of product features and software requirements. These multiple viewpoints are reconciled within a single use-case model, which specifies what the system will do. (Spence & Probasco 1998, p. 6-7)

The strategy 2) features drive the use-case model, which is recommended as a default strategy by RUP and its traceability links are defined in figure 11.

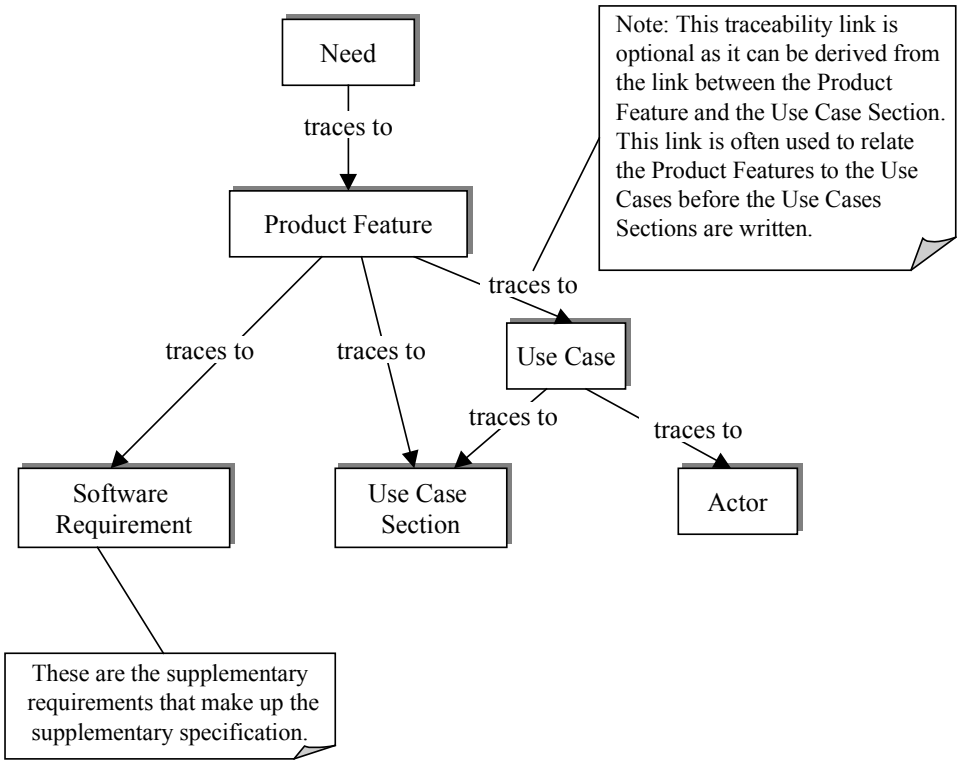


Figure 11. Traceability overview (Spence & Probasco 1998, p. 30).

3.6 Problems of Requirements Traceability

Basically even if the traceability policies and manual are defined clearly and effectively, there is no guarantee that traceability can be performed perfectly. This is because the quality of the traceability information depends on people‘s abilities to capture requirements information and other facts mentioned in figure 12, which illustrates the requirements traceability problem for provision. This means that to perform traceability information in development projects good quality of requirements is demanded.

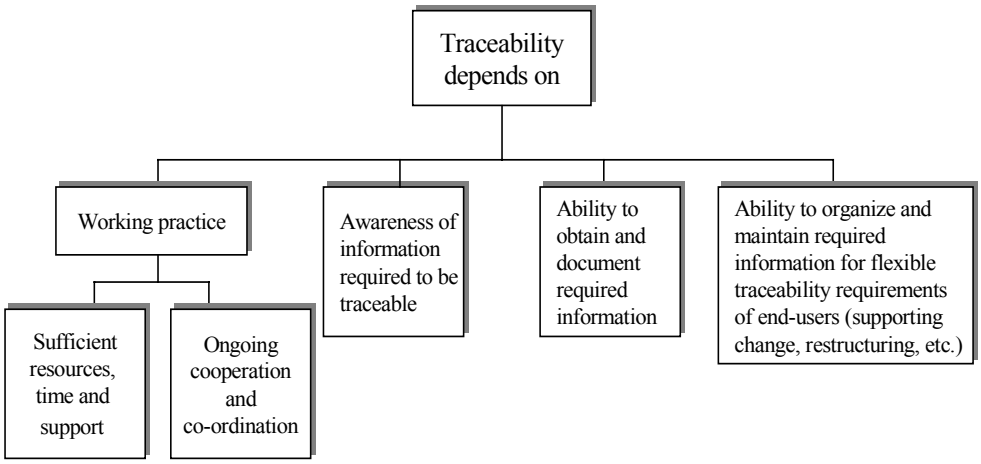


Figure 12. Deconstructing the requirements traceability problem for provision (Gotel & Finkelstein 1994a, p.14).

Another problem is to gather traceability information for end-user requirements, which are sometimes very ambiguous. Traceability information itself can be divided into two groups looking from the end-user point of view. These groups are of what and in what way (access to and presentation of information) the traceability information should be handled. In addition, these groups are dependent on who wants the information (user), why/when they want it (tasks), and what are the characteristics of the project. (Gotel & Finkelstein 1994a, p.15)

A big restriction of implementing requirements traceability is costs that it causes in the beginning when it is taken into use in an organization. When traceability is adopted in a development project, some additional costs exist because of the time consumed for its implementation at first. Implementation costs are also dependent on the technique used to RT. Later on, especially in the new product versions, costs and time consumed for RT will decrease and eventually there will be cost savings. Also the time used for the development project of the next release will decrease. Benefits of RT are long-term benefits and people who do the work cannot see benefits right away. Consequence of this is that people might be skeptical and not willing to implement RT. As mentioned earlier RT causes some costs and so it is very important to determine effective policies for RT and be very specific

of what should be traced. Sometimes if the schedule is very tight in a project RT might be ignored because of the lack of time.

3.7 Costs of Requirements Traceability

Requirements traceability costs consists of development of RT policies and methods, convincing and teaching the policies for developers, and maintaining of the traceability information. Maintaining costs are usually the larger ones, which makes it hard to convince people about the importance of RT. Maintaining costs might be considered as a wasted resources, which is true if the policies of RT are not planned carefully. Carefully planned traceability policies insure that maintaining costs will not increase above the benefits of traceability information.

Developing RT policies and methods adequate for the specific development environment, it should be remembered that development costs of the policies and methods is the minor part of the aggregate costs when maintaining traceability information in the software development environment.

Training developers to use and maintain traceability information in the software development processes causes a small part of the aggregate costs of the traceability information. Well-trained developers can maintain traceability information more effectively and appropriately, which will reduce the maintainability costs.

Requirements management tool helps to maintain RT information and links but it causes some costs too. RM-tool consists of costs of licenses, consultation, training end-users, system maintenance, and integrating the tool to the current environment (Leino 2001, p. 22). These issues should be considered while evaluating RM-tool for the use of an organization.

4 ADVANTAGES OF THE REQUIREMENTS TRACEABILITY

Wiegers (1999, p. 15) points out that organizations, which implement effective requirements engineering processes can enjoy multiple benefits. RT is at the heart of requirements management and so very important part of requirements engineering. RT cannot affect appreciably on the quality of requirements but it helps to verify that all requirements are implemented. Leino (2001, p. 24) argues that RT is worth of documenting if:

- Requirements are expected to change frequently
- It is important to see interrelations between documents
- Requirements or parts of the system are going to be reused
- Documentation is used to communicate between different parties
- The system is going to be maintained by a different company
- Project people frequently change

Above mentioned issues can be found almost from every software development project nowadays. So the conclusion would be that RT is worth of documenting in some level approximately in every software development project.

Requirements tracing provide a way to demonstrate compliance with a contract or specification at one level. At a more advanced level, RT can improve the quality of delivered products, reduce maintenance costs and facilitate reuse. However, RT is manually intensive task that requires organizational commitment. It demands discipline to keep link information current during the whole development project. If the traceability information becomes obsolete, you'll probably never reconstruct it. Following are some of the benefits of implementing RT in development projects (Wiegers, p. 301-302):

- *Certification.* Traceability information (TI) can be used for certification in safety-critical applications to demonstrate that all requirements were implemented.
- *Change impact analysis.* Without TI, there is a high probability of overlooking a system element that might be affected if you add, delete, or modify a particular requirement.
- *Maintenance.* Reliable TI facilitates making changes correctly and completely during maintenance, which improves productivity. If there is not TI for the entire system, it can be build one piece at a time as software surgery and enhancements are performed. Implementation of TI can be started by listing the requirements from the part of the system that have been worked on. The traceability links can be recorded from the current point downward.
- *Project tracking*. If the traceability data is recorded diligently during development, an accurate record of the implementation status of planned functionality will be available. Missing links indicate work products that have not yet been created.
- *Reengineering.* The functions in a legacy system can be listed and recorded where they were addressed in the new system's requirements and software components. Defining traceability links offers a way to capture some of what can be learned through reverse engineering of an existing system.
- *Reuse.* TI can facilitate the reuse of product components by identifying packages of related requirements, designs, code, tests and other artifacts.
- *Risk reduction.* Documenting the component interconnections reduces the risk if a key team member with essential knowledge about the system leaves the project.
- *Testing.* With the links between tests, requirements and likely parts of the code can be pointed to examine for a bug when a test fails to yield the intended result. Testers can also verify easily, which requirements are implemented.

Above-mentioned issues show that different stakeholders of the development project benefit in different ways from RT. Requirements engineer can manage requirements better with RT information. Requirements are also collected easier for the next release of the product if RT is done completely. Reusable parts of the product can be determined, which helps the work of requirements engineer, architect, designer and tester in the next release of the product. The impact of the change is easier to analyze, which helps project manager and architect to make correct workload estimations and so analyze impacts on project's costs and schedule. Testers can verify exactly which requirements are implemented; thus customer can be sure that they get what they wanted.

Good RT helps designers to code exactly what is needed but nothing unnecessary. This raises the quality of the product. RT helps to manage the whole development project and steering group can be sure what is the state of readiness of the product. Reusability of the requirements, components, and test cases makes next release development quicker. This is good while the fastness is one of the most important competitive advantages in today's software development markets. If costs and schedule estimations are followed, the product agrees with the plans and customer is satisfied the image of the whole organization will raise.

As it can be seen from this thesis that RT has many descriptions and it can be implemented in different ways. Different people have different opinions of how RT should be implemented. However, one important issue that RT ensures, no matter how it is implemented, is that the development project continues from one phase to another without breaks. This means that the next phase is based on the earlier phase and the workflow is fluent.

5 CASE: SONERA SERVICE SOFTWARE (S3)

Sonera Service Software (S3) is Sonera's corporate research and development unit, which specializes in research and development activities. The S3's focus areas are intelligent service technologies and multimedia technologies. The S3-unit supports Sonera's various business units in their goals to create new value in the changing telecom services market. (Sonera Corporation, 2001)

S3 supports the creation of flexible, synergetic production systems by developing production platforms and process interfaces. This aims to increase cost-efficiency in production and create new value from customer segment-based differentiation. S3 helps Sonera's business units to ensure continued competitiveness in the rapidly changing field of telecommunications by exploring new technologies and building new competencies. (Sonera Corporation, 2001)

5.1 R&D Process Model

Three different process levels are described for the R&D process in Sonera. These levels are general, applied, and adapted process levels. The general R&D process model does not describe activities in detail in areas that are specific to a given business division, unit, or technology program. Thus, there is a need for an applied R&D process, which combines the general process model with the routines and practices specific to a particular organization to meet the specific needs of a particular unit. An adapted process expresses how current development projects adapt and use the applied process model in practice. (Tolonen 1999, p. 3)

5.2 General R&D Process

Sonera's general process model consists of three main phases (prestudy, feasibility study, and project execution) and six decision points (DP). Figure 13 illustrates

these phases and decision points. A decision point is an external control point, with a standard agenda, which assures that some work is finished in some point. Milestones (MS) depicted in figure 13 are project specific internal control points of projects. The project group is responsible for making decisions about them. Figure 14 defines the main events of decision points.

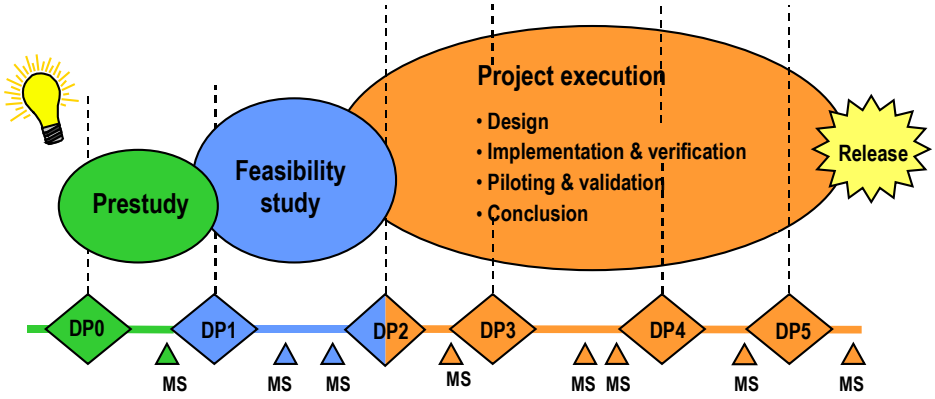


Figure 13. General R&D process model (Tolonen 1999, p. 4).

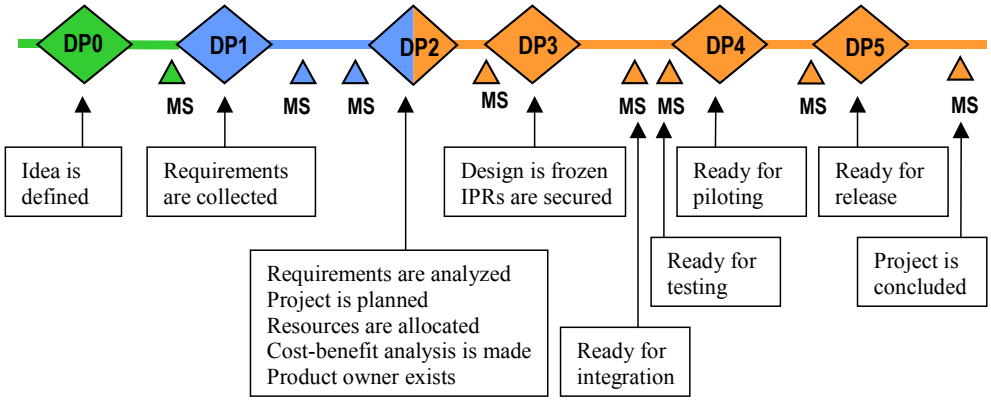


Figure 14. Main events of DPs (Tolonen 1999, p. 5).

The purpose of *prestudy phase* is to collect the main technical and commercial requirements of the product idea. These requirements are collected from customers. Rough schedule and workload estimates for the future project are also made during this phase. (Tolonen 1999, p. 9)

Feasibility study phase prepares a good basis for a successful execution of the project. This means that different realization alternatives and their potential consequences are analyzed, as well as their capacity to meet the requirements. In addition, project goals and strategies are defined, project plans are prepared, and risks are assessed. Contract negotiations with resource owners are initiated, and the project organization is defined at a comprehensive level. (Tolonen 1999, p. 11)

The purpose of *execution phase* is to carry out the project as planned in respect to time, costs, and characteristics, in order to attain the project's goals and meet the customer's requirements. The project execution phase has four subphases, which are design, implementation and verification, piloting and validation, and conclusion. (Tolonen 1999, p. 13)

5.3 Applied R&D Process in S3

S3 has created own applied R&D process, which is illustrated in appendix 1. This figure contains all sub-processes included in S3's product development process. In the following only those processes related to requirements traceability (RT) will be discussed, and they are requirement capture, system concept, design and implementation, and testing process. Other processes can also take advantages of RT, but they do not have direct impact on RT. Thus, those processes will not be concerned here. Present state of RT in S3's processes and product development projects is discussed in chapter 6.

5.3.1 Requirement Capture Process

S3 has not defined separate requirements management (RM) process but in practice requirements are managed in certain level in development projects. In the S3's development projects requirements are managed through these four processes mentioned earlier and change management procedure. The requirements

capture process starts requirements engineering in development projects by collecting customer requirements in the beginning. Technical requirements will be collected together with the system concept process. The output of requirements capture process is accepted requirement catalogue. Other possible outputs are some demo and short product description/introduction, but pre-RT is made available by requirements catalogue. S3's requirements catalogue template includes following fields for the requirements' information: group, identification (ID), description, date, source, customer priority, feasibility, agreed status of requirement, and notes. Requirements catalogue has the same purpose as the supplementary specification and software requirements specification (SRS) document, which are mentioned in the theoretical part of this thesis. (Leino K. 2000)

5.3.2 System Concept Process

The system concept process will be started during the requirements capture process. The purpose of system concept process is to analyze the feasibility of the software product based on proposed requirements. The process proposes appropriate architecture for the software product and analyzes both the interfaces to existing systems and the feasibility of the requirements using selected technology solutions. Analyzed main requirements (AMR) and technical implementation proposal (TIP) documents are the outputs of the process. AMR document prioritizes use cases and requirements from technical viewpoint, defines scenarios, conceptual model, roadmap, and additional requirements. TIP document observes non-functional requirements as capacity, technical risks, security issues and so forth. In addition, TIP determines configuration limitations and architecture, which includes system scope, system architecture in logical level, used standards, technical environment proposal, and limitations to the implementation of the system. (Kataikko 2000)

5.3.3 Design and Implementation Process

Before ending the system concept process starts the design and implementation process. The design and implementation process is divided into two phases: the design phase and the implementation phase. The purpose of the design phase is to carry out previously defined requirements, to produce sufficient documentation for the implementation of software, and to define tools and methods used during implementation. The goal is to extend the architecture description from system level to software level and define detailed technical specifications (TS) for each software module. Sufficient documentation means TS documents, product description, security plan, and user interface documents. In practice, projects' TS documents are the following: main TS, module TS, data storage TS, O&M (operation and maintenance) TS, and localization TS. During the implementation phase, software is implemented according to the outputs from the design phase. The implementation includes programming, module testing, system integration according to build plan, and necessary documentation. The process is also responsible for correcting faults detected during integration and system testing. Realization of the design is followed during the implementation. The specification documents are updated during the process and bigger changes are made through the change management procedure. Outputs of the implementation phase are build plan, build notes, module tested modules, updated TSs, and updated product description. (Ovaska 2000)

5.3.4 Testing Process

Testing process starts by defining the test strategy and test cases (test plan) and testing will be started as soon as the design and implementation process has something to deliver for testing. The purpose of the testing process is to prove that the specified, documented requirements have been fulfilled. Testing is based on inputs of other processes. These inputs are project plan, technical requirements, TIP, technical specifications, service concept descriptions, demonstrations and prototypes, build notes, and discussions with the project members. Outputs of the test-

ing process are test plan (test specifications, test case suites, test environment description) and test report (test case results, fault reports). Testing verifies that all requirements are implemented as planned. (Ylimäki 1999)

6 DESCRIPTION OF THE PRESENT STATE OF REQUIREMENTS TRACEABILITY (RT) IN S3

Present state of RT in the S3 is analyzed by researching the S3's product development process model and three different product development projects. The product development process is analyzed by checking out process descriptions and RT related templates. Three different development projects help to establish how RT is implemented in the real projects. The level of RT in the projects is analyzed by researching the related material and interviewing people in the projects. As mentioned in theoretical part one important element effecting on the level of RT is the size of the project. Thus, the projects that are considered here represents different sizes of the development projects in S3.

6.1 RT and the S3's Product Development Process Model

Output of the requirement capture process is the requirement catalog (RC) document, which includes all requirements. The RC template includes all relevant fields and gives relevant information about single requirement, which makes the realization of RT possible in some level. Requirements traceability is divided into pre- and post-RT in theory and the S3's requirement catalog ensures that pre-part of RT is almost complete. However, the RC does not enable forward traceability to use cases and modules. Requirement's rationale and version history information are missing from the template and the importance of those issues is analyzed later in this thesis.

The AMR document in the system concept process analyzes requirements roughly from system usage's point of view. The AMR template ensures RT by demanding to mention all requirements related to each use case. In that way use cases are traced backwards to requirements. The TIP document is a proposal, which will not be updated during the project and so RT does not have to be considered com-

pletely in this document even though some requirements might be mentioned in the TIP.

The design and implementation process should carry out RT information in the TS documents and code. Templates for the TS documents support RT by describing use cases and classes related to modules. However, class names are not defined in so that classes could be linked directly to the different use cases and so this traceability link is not perfect. A code is linked to classes by using same class names in a code as is used in the TS document. Until this phase of the project requirements are traced backwards in some level. Specific requirements' IDs are not included in the documents but different phases are linked together by referring to the earlier phases.

Test case suite is the document, which describes test cases in the testing process. Template for this document demands that the ID of the requirement, which the specific test case will verify, should be used as a header for that specific test case. Use cases are not mentioned in the template and so the test cases are traced backwards directly to the requirements in theory.

Basically, the requirements traceability information is executed roughly backward in the S3's product development process model. Single requirements are not traced from the requirement catalog through the whole process to the testing but the previous phase of the process is always linked to the earlier phase. Forward traceability is ignored and the meaning and importance of it for the S3 will be analyzed later.

6.2 RT in Real Projects

Traceability policies for the specific project depend on different kind of issues, which are mentioned earlier in the theory part (chapter 3.5.2). Thus, three different projects are used to analyze the present state of RT in the S3. The size of the

project affects most on the traceability policies and so different size projects are used to analyze RT in the S3. Instead of the real names the projects are cited as project A, project B, and project C. The size of the project can be defined through the size of the project group, duration time, and the amount of the requirements. The basic information of the projects is illustrated in table 7. The present state of RT in projects will be analyzed in the same order as the analysis of the S3's process model.

Table 7. Basic information of the projects.

	Project A	Project B	Project C
Duration of the project	> 1 year	~ ½ year	< ½ year
The size of the project group (people)	> 20	~ 10	< 10
Requirements	452	186	52
• Rejected/ postponed *	65 (14%)	85 (46%)	3 (6%)
• Implemented requirements	387	101	47

* Projects A & B have not been concluded and so these numbers may change before the projects ends.

6.2.1 Project A

Project A is in the implementation phase and it can be classified as a big project in the S3 and it has earlier releases. When the project has over 400 requirements it is not simple to get them traced, especially without tool support. At this phase of the project the requirement catalog (RC) is updated in some level but not completely. Missing information will be filled in while the catalog is updated.

There are eight different AMR documents because the project is so big. The quality of the traceability information varies between those documents. Some use cases have clear references to the related requirements and some do not have any

references to the requirements. Standardized IDs are not used in this project, which makes the implementation of the traceability information harder. Hence, backward RT is provided partly until this phase.

The TS documents, which describe design for the system and modules by use cases and other descriptions, do not have complete RT information in this project. The TS document should define use cases and contained classes. The level of described classes differentiates between documents. If the TS document determine classes in a very specific level, and if the same class names are used in code, then the requirements are traced backwards. However, in some TS documents the classes are described in a very common level and the designer have to do some additions. This is the very common situation when the module is defined from the scratch. Hence, the documents should be updated but in this project, the documents are updated late or not at all because of the tight schedule. This means that traceability links might break off partly.

Test case suite documents are used to determine the test cases. Each test case has information about the use cases and documents it is related to. In addition, traceability matrixes are used to link classes and use cases, and test cases and use cases. Designers map classes to use cases and testers map test cases to use cases, which ensures that backward and forward RT is fulfilled from classes to use cases and test cases. Matrixes are not filled completely yet but they will be filled in while the project matures.

RT is implemented partly in the project A. Requirements' IDs are not carried through the documentation but the traceability chain goes through the documents partly. With minor corrections backward traceability could be implemented perfectly. Forward traceability is carried out when the testing process begins and the traceability matrices are created. The proposals how to amend RT are discussed in the chapter 7.4. One big problem at the beginning of the project was that the project's scope changed a lot. This caused many changes and documentation updating became arduous and so RT was harder to carry out. Documents' updating is

the problem very often because tight schedules make it more important to get the product ready than update documents.

6.2.2 Project B

Project B is the medium size project in the S3. The budget and the schedule of the project will not fully meet the initial plans. RC updating was started during the implementation phase. Single requirements are not traced through the project. AMR document was partly updated. The architecture of the project changed a lot during the design phase and because of the tight schedule changes were not updated on the AMR document.

Classes and methods are described in the TS documents but references to the use cases, functional parts, requirements, and/or features do not exists. RT chain from requirements to testing ends to this point of development. Traceability from technical specifications to the code is quite clear because same headings are used for the classes and methods in the documents and code. The problem is that despite of good plans some changes are still made to classes and methods during coding. The TS documents are updated if there is time hence traceability is not complete and up-to-date to all classes.

Test cases are named based on the module and class names. This ensures backward traceability from the test cases to the classes and modules even though the test case IDs do not reflect to the use case IDs. The test case specification document is used to inform the implementation status of the test cases. Testing is partly executed as random testing. Random testing does not have direct traceability to the technical specifications and it does not show directly which requirements are implemented but it often bares even some critical bugs.

RT information chain is not complete in the project B. The chain of the backward RT breaks off before TS documents but continues after this break to test cases. As mentioned earlier, development proposals are discussed in the chapter 7.4.

6.2.3 Project C

Project C was the small project in S3. In small projects it is easier to estimate the budget and schedule and so this project met those estimates too. Even if the project was small the requirement catalog was not updated always on time. This is understandable because the document updating is usually done after the change is executed. The AMR document was not written in this project. The TIP document was provided and all requirements with "must" status in the RC were observed and fulfilled in this document. However, any references for the specific requirements were not marked on the document. The TS document continued the work done in the TIP and the traceability information was executed by using same names for the same functional parts. Same names for classes described in the TS were also used in the code, which ensures continuous traceability information chain backwards.

Testing was based on test case descriptions and requirement catalog. Test case names refer to the names of the specific functional parts described in the technical specification. With this traceability information implemented and tested features can be verified. While all requirements from the requirement catalog were checked it was noticed that one feature was not implemented. This shows that requirements can be verified completely only if every requirement can be somehow traced to the code level.

Even if requirements were not traced completely this project was able to verify the requirements which were implemented. The compact amount of people in the project group and the compact amount of requirements made this possible. The bigger the project the harder it is to verify non-traced requirements. However, this

project showed that in the small project almost complete RT chain is possible to carry out even if the traceability information is not documented clearly.

7 RESULTS

Backward traceability of the requirements is executed in some level in every project but forwards traceability is ignored. There are some minor problems, which could be corrected easily. Problems and benefits of the RT are discussed in the following. Also there is an analysis of what information should be traced and how it should be implemented.

7.1 Problems with RT in S3's Projects

One main problem is that the S3's product development process model does not support RT extensively. The quality of the requirements and features varies in different projects. One reason for the variable quality of requirements, which appeared in discussions, could be that planning of the new product is not always started from the needs and features and too specific requirements are described during the prestudy. If requirements are deficient the implementation of RT is more difficult because of many changes during the project. Templates support partly RT but there are still some flaws. The requirement catalog template does not determine standardized IDs for the requirements and so IDs varies in different projects. This may cause some problems in RT. For example R113 does not describe what kind of requirement is in question. In addition, there is not any information where the requirement will be concerned on and/or analyzed next (i.e. use case or another bigger entity) and so forward traceability is ignored. Across the board the problem is that the RC document is not updated often enough in the S3's development projects and the form of the catalog is not easy to handle and read. Improvements of the RC are discussed later.

The AMR document analyzes and groups the main requirements. However if the RC has a clear grouping system this grouping in the AMR may be useless. The AMR tries to link requirements to the bigger entities as product support, installa-

tion, charging and so on. Basically, requirements are listed under different headers especially in the chapters four and five. This type of expression is hard to read and update and some kind of matrix could be more useful. The importance of the chapters four and five (4. Requirements for development, 5. Requirements for deployment and runtime management) in the AMR should be analyzed and evaluated because the documentation always increases workload and costs and so everything which might be useless should be reevaluated. The AMR document is not used in the small size projects because it is the planning document and smaller systems may not need so much detailed and documented planning. The TIP document is ignored here because it does not have direct effects on RT.

The TS-main document template describes the overall system and it does not have to support RT very strongly. The TS -module document template supports RT by insisting references to the related use cases and classes of the module. However, the use case IDs are not standardized in the S3 and even projects does not standardize those IDs inside the project. Consequence of this is that the sub-use cases may have IDs, which are not directly linked to the IDs of the main use cases. Forward RT for the test cases is not required in the template and so it does not exist. Some testers in the project A have used traceability matrixes to link classes and use cases, and use cases and test cases. Thus, those matrixes describe forward traceability from the technical specifications to the code and test cases. Matrixes are not determined by the process description and they are not used widely in different projects.

The test case suite document describes test cases. The document template insists requirement IDs and names to be used as second headings but it may be hard to determine one specific requirement for each test case. In practice any of these three projects did not use requirements IDs and names as a header. Every project divided test cases according to functional entities as template forwarded but second headers were different in every project. Some test case suites used the use case IDs as headers and others had names, which were related to some part of the

functionality. Different kinds of names and not standardized use case IDs make the implementation of RT harder.

7.2 Usability and Benefits of RT Information for the S3

Many projects in the S3 cannot follow schedule and budget forecasts. RT could help with that problem. If the clear chain of requirements is created through the project from requirements to testing impacts of the changes are easier to evaluate. This helps architects and project managers to evaluate workload, budget, and schedule changes more accurately. Thus the whole change management becomes quicker and more effective. Effective change management ensures savings for the projects. Projects are also easier to manage if the RT information is available and project managers can more easily perceive what have been implemented and what still have to be done.

With the RT information steering group can easily assure customers that all features of the product are implemented. Customers understand better why some changes cannot be done in the present release if the impacts of the required change can be attested somehow. Budget, schedule and the quality of the product are the most important issues for the customer. If those issues could be better guaranteed the customer satisfaction would increase, which means longer customer relationships.

The RT information helps the support to track bugs and then resolve problems quicker, which makes the work of the support cost effective. Requirements, modules and test cases are also easier to reuse because the RT information determines which requirements are implemented in the module and what requirements are tested by the test case. Reusability of these issues helps product engineers', designers', and testers' work. While their work is easier the product development project of the next release will be quicker which saves time and costs. So the customer saves time and the S3 can allocate people more efficiently to the other projects. Basically, the work will be more cost-effective.

Testers benefit a lot from the RT information. Testers can verify that all requirements are implemented and bugs can be mapped easier. Testers can be sure that all requirements are tested. This also guarantees to the customer that every feature is included in the product. With the RT information the customer can better follow the implementation status of the project and they can see what they will get at the end of the project.

S3 does not have own products, which means that the next release of the product very often has different people involved in the development project. The good RT information from the previous release helps people's work in the next release. This may also lead to the quicker delivery time.

All these benefits of RT improve the quality and cost-effectiveness of the product, which leads to the better customer satisfaction. To achieve more benefits than costs the level of RT to be carried out in the S3's projects have to be determined carefully. This will be analyzed in the next chapter.

7.3 What Information Should Be Traced in S3

Benefits of the RT information are greater the more exact is the documentation of the RT information. At the same time the costs of RT will increase, which is the reason why the information that should be traced in different size projects in the S3 have to be analyzed carefully. This chapter analyzes what kind of requirements information should be traced in the S3 in general and the next chapter explains how the information could be traced.

In the literature there are a lot of discussions, which argues that every requirement should be traced to all phases of the project and all possible relations should be determined. That kind of traceability policy should be evaluated carefully because the costs may rise higher than the advanced benefits. It is not cost-effective at the moment for the S3 to implement perfect RT. This is because the quality of the re-

quirements varies and S3 have not had any kind of specific traceability policies yet. The S3's process development is not mature enough to adopt highly detailed RT yet. Important for S3 is to attain backward and forward traceability chain through the whole development project. After the level of RT is achieved the benefit of the more detailed RT can be evaluated again and the next step could be to maintain the requirements-requirements relation.

Backward traceability can be assured by making little improvements for the templates. Forward traceability needs some increments to the documents or the traceability matrixes. Backward-forward traceability ensures that whatever part of the development project have to be checked or analyzed the link for the previous and next phases is always available. Backward and forward traceability should ensure that at least the following links could be determined. People in brackets are responsible of creating those links:

- Functional requirements ⇔ Use cases (requirement analyst)
- Use cases ⇔ Modules (software architect or designer)
- Modules ⇔ Design classes (software designer)
- Design classes ⇔ Test cases (test engineer and designer)
- Test cases ⇔ Use cases (test engineer)

The chapter 7.4 determines how this kind of RT chain can be implemented.

In addition features-functional requirements link should be created. This could be implemented by giving clear IDs for the features and include those in the source field of the requirement catalog or create requirement IDs so that they directly link to the feature. This link would help to continue the work from features to requirements and the features would be used as the basis for the requirements.

The requirements-requirements link would also help the change management and as Sommerville and Sawyer (1997, p.226) note this link should always be maintained. However, if there are over a hundred requirements it is almost impossible and not cost-effective to keep those links updated without the requirements man-

agement (RM) tool. As Leino (2001, p. 48-49) states in her thesis the survey has shown that the RM-tool is necessary to document and visualize the requirement-requirement relation. S3 does not have the RM-tool or any kind of documentation management tool and so the requirement-requirement links should be considered later if the tool is available rather than now. However, small projects can try to trace requirements to other requirements manually with matrixes if it is considered to be useful and possible to do.

7.4 How to Implement RT in S3

Backward and forward RT can be implemented in S3 by improving the related document templates and creating at least two traceability matrices. Documents ensure backward traceability when correct IDs are used to the features, requirements, use cases, classes, and test cases. Traceability matrixes ensure also the forward traceability. Without matrixes forward traceability links should be added to the documents, which causes more work for documents updating. Matrixes are usually supported by the RM-tools and so they are easily adapted to the RM-tool.

7.4.1 Template Improvements for the Product Development Process Model

In the following the proposals are made what kind of changes and improvements could be done to the documentation to improve RT in the S3's process model and projects.

- *Product description:* The product description document should be created to describe the product in a common level and it should contain the product's features. With clear IDs, the features could be linked to requirements. Source field in the requirement catalog should include the information of the feature's ID the requirement is created from or the requirement's ID could directly link to the feature.
- *Requirement catalog (RC):* Links for the use cases should be added (e.g. notes field). Some standardized ID examples should be determined for the re-

quirements. Also standardized groups for requirements should be determined. Rational's FURPS (Functionality, Usability, Reliability, Performance, and Supportability) classes are already considered in S3 but not yet mentioned in the process description or RC template. Version history for the requirement is very useful but that information might be too arduous to carry out without the RM-tool.

- *Analyzed main requirements (AMR):* If requirements are grouped carefully in the RC, the chapters four and five in the AMR template are useless because listing requirements twice is frustrating and ineffective. Simply announced the AMR depicts use cases, which determine the functionality of the product. Use case IDs should be standardized so that the sub-use cases could be linked to the main-use cases directly by their names.

- *Technical specifications (TS):* The TS-main document describes overall architecture and all modules for the system. The TS-module document determines module's functionality by determining classes. Class IDs should be created so that the sub class name could be linked to the main class name. Same class names should also be used in the code as are used in the TS-module document.

- *Test case suite:* The document uses requirement ID as a header of the test case sets. Use case ID and/or name could be more useful for that purpose. The project A for example used the use case IDs as the headers in the test case suite documents, which appeared to be a good convention. However, related requirements should be mentioned in the test case if possible. The test case ID should be almost the same as the related use case ID. This ensures the direct traceability link from the test case to use case.

- *Traceability matrix:* Some kinds of traceability matrixes have already been used but those are not determined officially. The traceability matrixes should be defined for functional requirements ⇔ use cases, use cases ⇔ modules, modules ⇔ classes, and test cases ⇔ use cases relations. These traceability matrixes execute backward and forward RT through all phases from use cases to test cases. Appendix 2 depicts the developed traceability matrix for S3.

7.4.2 Traceability Strategy

Traceability strategies were discussed in the chapter 3.5.3. Figure 11 illustrates the strategy recommended by the Rational University. Figure 15 is almost the same and it depicts the possible RT strategy for S3. This strategy could be easily adapted to the S3's process model. The product description document is under development in S3. It is important to start the charting of the new product from the needs and features and bit by bit enlarge the description to the detailed requirements. So the product description document should describe the product in a general level and determine its features while the requirement catalog presents non-functional requirements for the product. The actual RT starts from requirements and the clearer is the product description for all stakeholders the easier it is to implement RT.

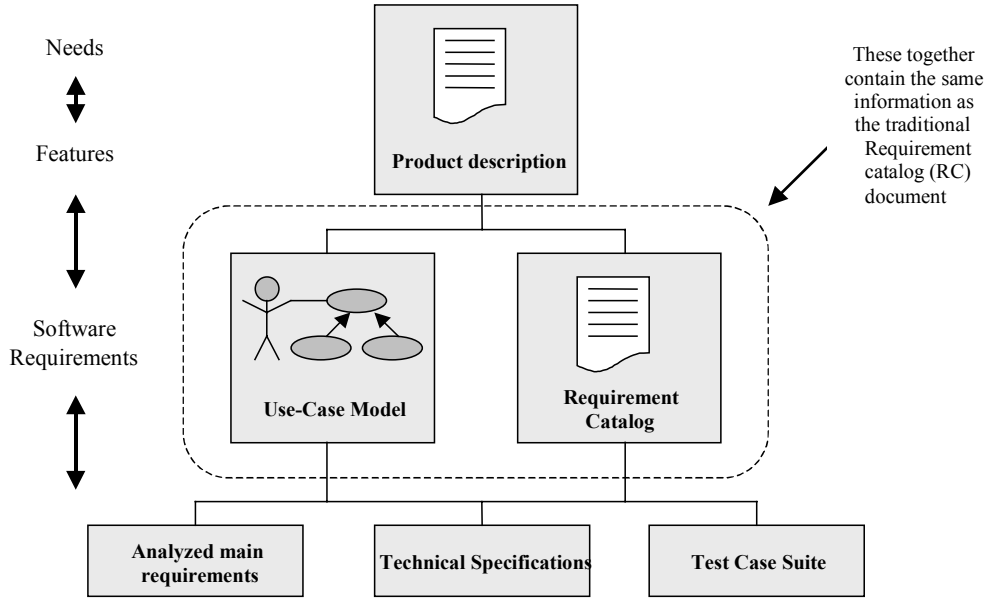


Figure 15. Proposal for the requirements structure in S3.

Figure 15 shows that requirements are stated by the use case models and the requirement catalog (RC) document. S3 has a pilot project for this kind of requirements description in which use cases tries to determine the functional require-

ments and the RC determines all non-functional requirements. The use case model includes use cases and more detailed text descriptions of the functional requirements. The use case model should have as detailed information of the functional requirements as the RC has from non-functional requirements. The use case model and the RC complements each other and IDs used in these documents should be congruent with each other.

After requirements are determined they will be analyzed and prioritized. Nowadays there is still the AMR document but the role of the AMR should be reconsidered if the use cases are already described in the use case models. After this phase traceability chain continues to the technical specifications and to the test cases with those standardized IDs as mentioned in the previous chapter.

The traceability matrix depicted in the appendix 2 should be used in the S3's product development projects. The matrix is created in the same way as the table 4 depicts. However the matrix of the table 3 could be considered in a small projects (100 requirements or less). Table 3 includes all relations from requirements to test cases but that kind of matrix would be laborious to use and update in larger projects.

Large and medium size projects, as the project A and B, should consider the documentation of the RT information seriously. Standardized IDs for the requirements, use cases, classes, and test cases are more important in large projects. Small projects, as project C, need to consider what information have to be documented and what issues are obvious for the project group even without documentation. Standardized IDs do not increase the amount of the work and could be useful also for the small projects. When the project group is small every one in the group knows the project very well, which enables lighter documentation. However, all RT issues, which are important for the possible next release of the product should be documented.

IDs for the needs and features are not used in S3 and not considered in this study. Later if there is a need for the IDs of the needs and features those should be created but at this point the most important issue is to get the RT information from requirements to test cases. Requirements are based on the needs and features, which need to be considered in the requirements eliciting phase.

8 CONCLUSIONS AND RECOMMENDATIONS

The scope of the study was to find reasons why requirements should be traceable in an organization. Two objectives were appointed for the study. The first objective was to identify the use of the requirements traceability information. While interviewing people the concept of the importance and need for RT strengthened. There have been some problems at the beginning of the software development projects in S3 basically in the requirements elicitation and those problems should be solved quickly. RT cannot help measurably the requirements elicitation but it helps to manage the rest of the project and to verify that all demanded features have been implemented. The RT information makes change management more effective because the impact of the change is easier to evaluate. RT brings most of the benefits to the next release of the product. With the RT information it can be seen what parts are reusable, what have to be changed or created.

The second objective was to find out what kind of information should be traced and how. In the literature requirements are often traced in very detail level. The methods performed in this thesis for providing the RT information are easy to carry out and not expensive. Basically, it is recommended to carry out RT by determining standard IDs for requirements, use cases, code classes, and test cases and creating traceability matrixes. Especially the relation between costs and benefits have been considered in this study. Costs/benefits relation is hard to determine before the recommended RT-level has been implemented in few projects. This is the one reason why this study recommends the implementation of RT in a very simple way at first. People are more easily committed into the implementation of the RT information if the method is simple. It should be remembered that the benefits of RT differentiate between the projects that have next releases and the projects that will end completely after the first release.

This study examined how RT has been implemented in the S3's projects by exposing to real projects. This view of how to implement RT in the S3's projects has been created by interviewing and discussing with people, and acquainting to the material of the projects and related literature. Opinions of the level of RT in the literature differentiate. Some authors accentuate punctual RT but mentions that RT can be implemented only in organizations whose processes are mature enough while others think that RT can be implemented in different level depending on the organizational maturity and other factors related to the development environment. However, this study concludes that RT can be started from the lower level in immature organization so that RT can be easily adapted into use. Also RT must not cause more costs than the attained benefits are.

S3 has some level RT already but the chain from requirements to test cases breaks off in some points. That is the reason why there is a need to create the RT policies. This study recommends what issues should be considered while creating RT policies for S3. Requirements management (RM) process is not determined separately in S3. Requirement capture process and requirements traceability policies are important parts of that process and so RM process should be determined to bind these two important issues together.

The traceability policies and strategy introduced in this thesis are based on standardized IDs and traceability matrixes. The recommendation is that IDs should be determined to cover all projects in S3. If standardized IDs are determined individually to each project people need always to study those IDs first while entering the new project, which is wasting of time. The more the projects have fixed conventions the easier it is for the people to become acquainted with new projects. However, standardized IDs should be adaptable because different projects have different demands for those IDs.

The traceability matrix template is introduced in the appendix 2. That template can be used without the RM-tool. Manual updating of the matrix might be arduous in large projects but not impossible. The RM-tool does not resolve any problems

alone but it would help and make the requirements management more efficient. For example database for reusable requirements could expedite the requirements elicitation phase. While RT is implemented manually its correctness cannot be assured. The traceability tool enables to inspect the traceability relationships to ensure that no verification relationships have been omitted and that no excess verification relationships are present (Leffingwell & Widrig 2000, p.333). This study recommends S3 to evaluate the RM-tools to find out which one could be the most useful for the S3's needs. If the RM-tool will be considered seriously in S3 the tool should integrate greatly with other the tools used in S3. However, it should be remembered that the tool alone cannot do the job; verification requires thinking (Leffingwell & Widrig 2000, p.333).

Requirement-requirement link could be executed with the same traceability matrix as illustrated in the appendix 2. However, the quality of the requirements in the projects of S3 varies much which makes creating of those links very hard. The Requirements-requirements links updating is also laborious without the RM-tool support. At this state the requirement-requirement link is not recommended for S3 even though it would help the change management a lot. If the recommended traceability strategy will be noticed to be useful for development projects and benefits are evident, the requirement-requirement link could be the next step to be implemented when developing the RT policies in S3.

While new traceability strategies are introduced to the organization it should always be remembered that different people have different ways to work. Thus, it is always challenging to get people convinced about new working methods. The RT strategy recommended by this study demand that all people in the project will be committed to follow the strategy before it works.

Other factors that affect on the implementation of the RT information are the awareness of the information required to be traceable, the ability to obtain and document the required information, and finally the ability to organize and maintain the required information. Thus, the traceability manual and clear strategies are

important to reduce people's confusion and people should be trained about the RT information.

9 DISCUSSION

This thesis focuses on the requirements traceability (RT), which is at the heart of the requirements management (RM). The central task of the RM is to assure RT, from the earliest activities through to system evolution and maintenance. Anyhow, too often RT is not carried out completely if at all in development projects. There will always be some changes in requirements during development projects and with the RT information those changes can be managed more effectively. Well-managed requirements improve the total quality of the development project.

The argument for the RT information is why should requirements be traceable in software development projects? The objectives of the thesis are to identify the usability of the RT information, and to find out what kind of information should be traced and how.

The concepts of the RT definitions and techniques are approached through the literature. The field of the determination of the RT definitions is vast. Basically RT is divided into pre-RT and post-RT, and backward and forward RT. Pre-RT concentrates on the requirements development phase of requirements lifecycle while post-RT concentrates on the requirements on-going production phase. Backward and forward traceability means that a requirement can be followed from its origin through the implementation to the verification and vice versa.

Different types of the RT links can be maintained during the development process. However, the main point is that RT should continue through the whole development project and the work should always be based on the previous phase. This RT chain can be implemented using different kind of techniques. Different types of traceability techniques are grouped as cross reference-centered, document-centered, and structure-centered. These illustrate different ways of documenting RT. Then there are also some basic techniques, which can be used to maintain the

traceability information. Techniques illustrated in this study are the traceability table or matrixes, traceability lists, automated traceability links, and requirements management tools.

These techniques and different types of the RT links should be considered while the company is creating own traceability manual. In addition each project should define own traceability strategies, which determines how the RT information will be maintained in a specific project. While creating these strategies the maintaining costs of the RT information should be considered carefully so that costs will not rise above the benefits of RT.

The RT information helps project managers to know the implementation status of the project and analyze the impact of the change request. The good RT information also reduces risks of the information gap if people changes in the project and more reusable components may appear. Also, testing and support benefits when bugs and other problems can be found quickly. These benefits are often long term benefits, thus people should be patient while carrying out the RT information.

The present state of RT in S3 is determined and analyzed. This is done by analyzing the S3's product development process model and by acquainting with three real development projects. This analyzing is made by acquainting to the related materials and by interviewing people in S3. The recommendations of the study have been created according to the literature and empirical study.

The RT strategies should be determined to S3 according to the recommendations made in this thesis. The RT strategy recommended in this study is a light RT strategy, which has been created especially for the S3's processes and so it is easy to adjust to the S3. The strategy helps people 's work because of cohesive working methods. The thesis also recommends S3 to seriously consider the RM-tool, which would help with requirements management and the RT information updating even though the RM-tool itself does not resolve any problems.

REFERENCES

Dorfman, M & Thayer, R. 1996. Software Engineering. The Institute of Electrical and Electronics Engineers, Inc. 21 p.

Easterbrook, S. & Nuseibeh, B. 2000. Requirements Engineering: A Roadmap. [Online]. <http://Citeseer.nj.nec.com/nuseibeh00requirement.html>. (Accessed 05.05.2001)

Goguen, J. & Linde, C. 1993. Techniques for Requirements Elicitation. Proceedings of International Symposium Requirements Engineering, The Institute of Electrical and Electronics Engineers, Inc. 13 p.

Gotel, O. 1995. Thesis: Contribution Structures for Requirements Traceability. London. Imperial College of Science, Technology and Medicine, University of London. 354 p.

Gotel, O. 2000. 7th International Summer School in Novel Computing: Systems Requirements Engineering, Unit 7. London. University College London. 35 p.

Gotel, O. & Finkelstein, A. 1994a. An Analysis of the Requirements Traceability Problem. London. Imperial College of Science, Technology and Medicine, University of London. 27 p.

Gotel, O. & Finkelstein, A. 1994b. Position Paper: Modelling the Contribution Structure Underlying Requirements. London. Imperial College of Science, Technology and Medicine, University of London. 7 p.

Kataikko, M. 2000. Sub-process: System Concept Process. Sonera Ltd. 10 p.

Lauesen, S. 2000. Software Requirements: Styles and Techniques. Frederiksberg, Denmark, Forlaget Samfundslitteratur. 191 p.

Leffingwell, D. & Widrig, D. 2000. Managing Software Requirements: A Unified Approach. 2nd Edition. California, United States of America, Addison Wesley Longman, Inc. 491 p.

Leino, K. 2000. Sub-process: Requirements Capture Process. Sonera Ltd. 9 p.

Leino, V. 2000. Types and Techniques of Requirements Traceability. Helsinki. Helsinki University of Technology. QURE Project. 27 p.

Leino, V. 2001. Master's Thesis: Documenting Requirements Traceability Information: A Case Study. Helsinki. Helsinki University of Technology. 54 p.

Ovaska, P. 2000. Sub-Process: the Design and Implementation Process. Sonera Ltd. 22 p.

Ramesh, B., Stubbs, C., Powers, T. & Edwards, M. 1995. Lessons Learned from Implementing Requirements Traceability. [Online]. <http://stsc.hill.af.mil/crosstalk/1995/apr/lessons.asp>. (Accessed 26.4.2001).

Reinikainen, L. 2000. Master's Thesis: Elicitation of Customer Requirements with Group Methods in Software Engineering. Lappeenranta. Lappeenranta University of Technology. 96 p.

Robertson, S. & Robertson, J. 1999. Mastering the Requirements Process. Harlow, England, ACM Press. 404 p.

Sommerville, I. & Sawyer, P. 1997. Requirements Engineering: A Good Practice Guide. West Sussex, England, John Wiley & Sons Ltd. 391 p.

Sonera Corporation. 2001. Sonera Service Software. [Online]. <http://www.sonera.fi/english/rd/organi.html>. (Accessed 12.06.2001)

The Standish Group. 1995. CHAOS: Sample Research Paper. [Online]. <http://www.standishgroup.com/visitor/chaos.htm>. (Accessed 11.04.2001).

Thayer, R. & Dorfman, M. 1997. Software Requirements Engineering. 2nd Edition. California, United States of America, IEEE Computer Society Press. 531 p.

Tolonen, A. 1999. Sonera R&D Process Model. Sonera Ltd. 35 p.

Wiegers, K. 1999. Software Requirements. Washington, United States of America, Microsoft Press. 350 p.

Ylimäki, Y. 1999. Sub-process: The Testing Process. 10 p.

Interviews

Verification Specialist of the project B. Sonera Corporation. Interview by email on July 6, 2001.

Account Manager of the project C. Sonera Corporation. Interview in Helsinki on June 25, 2001.

Project Manager. Sonera Corporation. Interview in Lappeenranta on May 7, 2001.

Project Manager of the project B. Sonera Corporation. Interview in Lappeenranta on May 7, 2001.

Department Manager. Sonera Corporation. Interview in Lappeenranta on June 8, 2001.

R&D Specialist of the project B. Sonera Corporation. Interview in Jyväskylä on July 3, 2001.

Verification Engineer of the project A. Sonera Corporation. Interview in Lappeenranta on May 4, 2001.

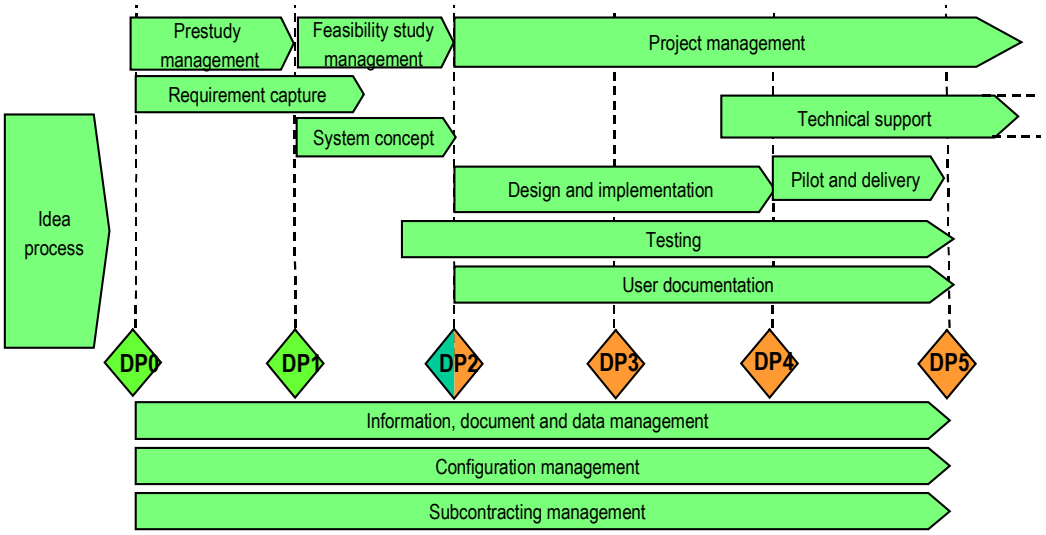
R&D Specialist of the project C. Sonera Corporation. Interview in Helsinki on June 25, 2001.

Project Manager of the project A. Sonera Corporation. Interview in Helsinki on June 25, 2001.

Chief Architect. Sonera Corporation. Interview by email on July 10, 2001.

APPENDIX 1.

S3's Product Development Process



APPENDIX 2.

Traceability Matrix Template