MEASUREMENT BASED SOFTWARE PROCESS IMPROVEMENT

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

AYSUN ENER

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

NOVEMBER 2007

Approval of the thesis:

MEASUREMENT BASED SOFTWARE PROCESS IMPROVEMENT

submitted by **AYSUN ENER** in partial fulfillment of the requirement for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen _____
Dean, **Graduate School of Natural and Applied Sciences**

Prof. Dr. İsmet Erkmen _____
Head of Department, **Electrical and Electronics Engineering**

Prof. Dr. Semih Bilgen _____
Supervisor, **Electrical and Electronics Engineering Dept., METU**

Examining Committee Members:

Prof. Dr. Uğur Halıcı _____
Electrical and Electronics Engineering Dept., METU

Prof. Dr. Semih Bilgen _____
Electrical and Electronics Engineering Dept., METU

Asst. Prof. Dr. Cüneyt Bazlamaçcı _____
Electrical and Electronics Engineering Dept., METU

Dr. Ece Schmidt _____
Electrical and Electronics Engineering Dept., METU

Dr. Altan Koçyiğit _____
Informatics Institute, METU

Date: **30.11.2007**

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have cited and referenced all material and results that are not original to this work.

Name, Last Name: Aysun Ener

Signature:

ABSTRACT

MEASUREMENT BASED SOFTWARE PROCESS IMPROVEMENT

ENER, Aysun
M. S., Electrical and Electronics Engineering Department
Supervisor: Prof. Dr. Semih BİLGEN

November 2007, 115 pages

This thesis is a study on improving the software requirements management processes of embedded software department of a company. The literature on software process improvement and requirements engineering is reviewed. After determining the problems related to the current requirements management processes of the department, an improved process is proposed addressing these problems. The static process descriptions and the models of the current and improved requirements management processes are formed. A recently proposed pre-enactment model for measuring process quality is used for measuring the quality of the current and improved requirements management processes. Finally, the results of the process quality measurements are compared and evaluated.

Keywords: Software Process Improvement, Requirements Process Improvement

ÖZ

ÖLÇÜME DAYALI YAZILIM SÜRECİ İYİLEŞTİRME

ENER, Aysun

Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü

Tez Yöneticisi: Prof. Dr. Semih BİLGEN

Kasım 2007, 115 sayfa

Bu tez bir firmanın gömülü yazılım bölümünün gereksinim yönetimi süreçlerini iyileştirme üzerinedir. Yazılım süreci iyileştirme ve gereksinim mühendisliği üzerine olan literatür incelenmiştir. Bölümde yürürlükte olan gereksinim yönetimi süreciyle ilgili problemler belirlendikten sonra, bu problemleri ele alan iyileştirilmiş bir süreç önerilmiştir. Bölümde yürürlükte olan ve iyileştirilmiş gereksinim mühendisliği süreçlerinin statik süreç tanımları ve modelleri oluşturulmuştur. Yürürlükteki ve iyileştirilmiş gereksinim mühendisliği süreçlerinin kalitesini ölçmek için yakın bir zamanda önerilmiş olan süreç kalitesinin uygulama öncesi ölçülebilmesini sağlayan bir model kullanılmıştır. Son olarak, süreç kalite ölçümlerinin sonuçları karşılaştırılmış ve değerlendirilmiştir.

Anahtar Kelimeler: Yazılım Süreç İyileştirme, Gereksinim Süreç İyileştirme

To My Family & Emrah

ACKNOWLEDGEMENTS

I would like to express my sincere thanks to my supervisor Prof. Dr. Semih BİLGEN for his continuous support, patience and valuable guidance throughout this thesis.

I would like to thank to my husband, Emrah and my family for their encouragement, understanding and endless patience.

I also would like to thank to my colleagues in my department for their valuable contributions and support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

BSA:	Business Software Alliance
CMM:	Capability Maturity Model
CMMI:	Capability Maturity Model Integration
DoD:	Department of Defense
ESD:	Embedded Software Department
IDC:	International Data Corporation
INCOSE:	The International Council on Systems Engineering
IT:	Information Technology
KPA:	Key Process Area
NIST:	National Institute of Standards and Technology
PDD:	Product Description Document
RCRR:	Requirement Change Request Report
RE:	Requirements Engineering
REGPG:	Requirements Engineering – A Good Practice Guide
REPM:	Requirements Engineering Process Maturity
RM:	Requirements Management
ROI:	Return On Investment
RTM:	Requirements Traceability Matrix
SEI:	Software Engineering Institute
SME:	Small and Medium sized Enterprise
SPICE:	Software Process Improvement and Capability dEtermination
SRA:	Software Requirements Analysis
SRCM:	Software Requirements Change Management
SRCRD:	System Requirement Change Request Document
SRS:	Software Requirements Specification
SRTM:	Software Requirements Traceability Matrix
SysRD:	System Requirements Document
TSD:	Technical Specifications Document

CHAPTER 1

INTRODUCTION

In today's world software has become very pervasive across all businesses, industries and our everyday lives. Not only systems of many fields such as consumer electronics, military, telecommunications, medical and transportation depend on software, but also their development, production, and after-sales support depend on software. Software also facilitates research and leads to innovations in fields ranging from nanotechnology to human genetics.

All of these make software a very important factor affecting the global economy. According to the "Global Economic Impact Study" (2003) of Business Software Alliance (BSA) and International Data Corporation (IDC), information technology (IT) generates an estimated $420 billion in North America, $289 billion in Europe, $175 billion in Asia Pacific, $24 billion in Latin America, and $13 billion in the Middle East & Africa annually [1]. Since software is such a huge industry it is important to develop software in a cost-effective way.

On the other hand, according to a study (2002) commissioned by the US Department of Commerce's National Institute of Standards and Technology (NIST), software bugs, or errors, are so prevalent and so detrimental that they cost the US economy an estimated $59.5 billion annually, or about 0.6 percent of the gross domestic product[2]. The cost of software bugs is so high that it seems there is something wrong in the current way of developing software.

The survey conducted by The Standish Group in 1994 [3] is another widely cited source on the economic impact of software and its quality. A later report [4] by the same group has shown that by 2003, major improvement efforts have caused significant benefits to software users and industry.

Although there seems to be an improvement in the success of software projects in recent years, they are still suffering from cost and time overruns. Moreover, most of the time, the delivered software has a lot of bugs and lacks many of the specified features. All of these result in an increase in the number of unsatisfied customers, and accordingly the level of trust in software companies decreases.

Many software process models such as waterfall model, incremental model, iterative model, prototyping, spiral model [10] and the Unified Process [11] have been proposed in order to bring order to the chaotic way of software development [8]. Although the proposed process models have brought some order, the aforementioned survey results of The Standish Group [3, 4] and NIST [2] show that organizations' software development processes still need to be improved.

Since every organization has a different culture and develops different kinds of software such as embedded, real-time, commercial product or information systems in different domains, they can not just select a process model and apply it directly without any modifications. Organizations should first select the process that is most suitable for them and afterwards tailor the process to themselves. Then they should continuously improve their processes to produce high quality software on time and on budget.

There are many models and standards related to software process

improvement such as CMM [12], CMMI [13, 14], ISO/IEC 15504 [21], ISO 9001 [23], Six Sigma [27] and BOOTSTRAP [26]. CMM and CMMI present sets of recommended practices in a number of key process areas (KPA) that have been shown to enhance software-development and maintenance capability. On the other hand, ISO/IEC 15504 is a framework for the assessment of software processes. BOOTSTRAP is a combination of software process assessment and improvement methodologies. ISO 9001 standard is used for external quality assurance. Although ISO 9001 is not specifically developed for software quality assurance, ISO 9000-3 [24] provides guidelines for the application of ISO 9001 to the development, supply and maintenance of software. Last but not least, Six Sigma is a set of practices to increase customer satisfaction by systematically improving processes through prevention and elimination of defects. All of them can be used to improve software development processes, but they should also be tailored to the organization before using, to get maximum benefit.

In this thesis, a pre-enactment model for measuring process quality proposed by Güceğlioğlu [51] is used in improvement of software requirements management processes of Embedded Software Department (ESD) of the company in which the author is employed as a software engineer. To preserve confidentiality, the company is referred to as Company A. Since the duration of this thesis study is not long enough to put the improved processes into practice and then evaluate the results, Güceğlioğlu's model [51] is used to compare the quality of the improved processes to the currently applied ones before deploying the improved processes. The reason why Güceğlioğlu's approach has been applied is that, as its description with the term "pre-enactment" implies, it can be effectively applied purely on process models rather than observation and live evaluation of active processes. Naturally, while this is the reason for its appeal, the level of its realism may be considered questionable. An evaluation in this context is made within the scope of this thesis study.

In order to measure the quality of the current (AS-IS) processes, static process descriptions of the AS-IS processes have been formed. Also, in order to facilitate measurements, the AS-IS software requirements processes have been modeled. Descriptions of the related processes written in regulatory documents have also been found in order to use in measurements, and they are referred to as on-paper (in-theory) processes. Then using AS-IS and on-paper process descriptions, the quality of the current software requirements management processes have been measured by applying Güceğlioğlu's model. The problems of the current requirements management processes have been determined, and improved processes addressing that problems have been proposed. The quality attributes of the improved processes have been measured by applying Güceğlioğlu's model and using the static process descriptions and the models of the proposed processes. Then the quality of the current and proposed models are compared and evaluated.

The remainder of this thesis is organized as follows:

The literature on software process improvement, software requirements engineering and software requirements process improvement is given in Chapter 2.

Static descriptions and models of on-paper and AS-IS software requirements management processes of ESD are given in Chapter 3. The quality attribute measurements of the AS-IS processes, the problems related to current processes and solution suggestions to these problems are also included in Chapter 3.

Chapter 4 consists of the static descriptions and models of the improved processes. The quality attribute measurements of the improved processes and their comparison to AS-IS quality attribute measurement results are

also included in Chapter 4.

Chapter 5 consists of the evaluation and conclusion of the study including the limitations and future work.

CHAPTER 2

LITERATURE

2.1 Software Process Improvement

In his famous article “No Silver Bullet: Essence and Accidents of Software Engineering” [5] Brooks says that even there are some potentials, there is no silver bullet to make software costs drop as rapidly as computer hardware costs do. According to him complexity, conformity, changeability, and invisibility are the essential difficulties inherent in the nature of software systems.

Future software-intensive systems will be larger and more complex than today. Boehm’s “A View of 20th and 21st Century Software Engineering” [7] paper gives insight into history and future of software. According to him in the next decade the ability of organizations to survive will depend on their ability to integrate related software-intensive systems into systems of systems.

Kruchten [9] lists the differences between software engineering and the other engineering disciplines as follows:

- Absence of a fundamental theory,
- Ease of change,
- Rapid evolution of technologies,
- Very low manufacturing costs,
- No borders.

According to Kruchten, software industry has tried to apply the processes similar to the processes of other engineering disciplines, but failed because of the differences given above.

1960s, 1970s and 1980s were the years of software crisis. The symptoms of the crisis were as follows:

- Projects running over-budget,
- Projects running over-time ,
- Software was of low quality,
- Software often did not meet requirements,
- Projects were unmanageable and code difficult to maintain.

In order to overcome the problems related to software development and get rid of the crisis, a lot of research has been done. To produce high quality software meeting the specified requirements on time and on budget has been the main aim. In order to achieve this, many new software development methods and technologies are introduced up to 1980's. But the problems had not been solved and the developers realized that their fundamental problem was their inability to manage the software process [6]. Better tools and better methods couldn't do much in an undisciplined, chaotic process. So efforts to improve software process began.

In 1984, US Department of Defense (DoD) established the Software Engineering Institute (SEI) to resolve the software crisis. SEI developed the Capability Maturity Model (CMM) [12] in early 1990's. According to SEI, continuous process improvement is based on many small, evolutionary steps and the CMM provides a framework for organizing these evolutionary steps into five maturity levels [6]. The CMM presents sets of recommended practices in a number of key process areas (KPA) that have been shown to enhance software-development and

maintenance capability. As the aim of the current study is not a CMM-based improvement, further details of this approach will not be elaborated here.

SEI has also developed the IDEAL Model [15] to provide a usable, understandable approach to continuous improvement by outlining the steps necessary to establish a successful improvement program. The IDEAL model is named for the five phases it describes: initiating, diagnosing, establishing, acting, and learning.

- I – Initiating: Laying the groundwork for a successful improvement effort.
- D – Diagnosing: Determining where you are relative to where you want to be.
- E – Establishing: Planning the specifics of how you will reach your destination.
- A – Acting: Doing the work according to the plan.
- L – Learning: Learning from the experience and improving your ability to adopt new technologies in the future.

After CMM's successful adoption and usage in many domains, other CMMs were developed for other disciplines and functions such as systems engineering, people, integrated product development and software acquisition. In 2000, all of these CMM's are integrated into a common model framework named Capability Maturity Model Integration (CMMI) [13]. The latest version of CMMI (Version 1.2) [16] was released in August 2006.

According to the SEI Process Maturity Profile (2007) [18] CMMI has been applied in different industries by organizations of different sizes in approximately 50 countries. Although CMM and CMMI are very widespread, they are often criticized for being too big, complex and being

designed to operate in large organizations doing projects with very high budgets ([19], [20]).

There is a typical misconception that achieving a higher maturity level is the most important goal in the software process improvement effort. While achieving a higher maturity level is one of the objectives of an improvement effort, it should not be the end goal. Too often, organizations attempt to drive themselves to the next maturity level very quickly. This approach inhibits organizational understanding of why the improvement effort is beneficial, which results in loss of organizational support for the effort [17].

ISO/IEC 15504 [21] also known as SPICE (Software Process Improvement and Capability dEtermination) is a framework for the assessment of software processes. Since software process assessment can be a part of software process improvement or software capability determination, ISO/IEC 15504 can be used for both. ISO/IEC 15504 includes a reference process model which is used as the basis on which software process assessment can be executed. It has two dimensions, one of which is the process dimension and the other is the capability dimension.

ISO/IEC 15504 also provides a guide for performing an assessment and an assessment model which is the detailed version of the reference model [21]. Although ISO/IEC 15504 contains an assessment model, other models can also be used if they meet ISO/IEC 15504's criteria. For example SEI continues to work to make CMMI more compliant with ISO/IEC 15504. Model changes made in CMMI Version1.2 made it more compliant with ISO/IEC 15504 [22].

ISO 9000 is a series of standards dealing with quality systems that can be used for external quality assurance. ISO 9001 [23] deals with design,

development, production, installation and servicing. Therefore it is the one that is related to software development. In addition, ISO 9000-3 [24] provides guidelines for the application of ISO 9001 to the development, supply and maintenance of software.

The result of an ISO 9001 assessment can have only two possibilities of outcome: fail or pass. On the other hand, the result of a CMMI assessment gives the rating of maturity level of the organization in staged representation, or it gives the rating of capability level of the organization in a specific process area in continuous representation. Therefore CMMI provides much better measure of quality of processes, whereas ISO focuses more on whether an organization satisfies minimal requirements for a quality system or not [25].

Other approaches such as BOOTSTRAP [26] and Six Sigma [27] have also been proposed and applied in various industrial environments (e.g. [30], [31]).

In order to benefit from the proposed models, they should be tailored to the organization before being applied. Tailoring is especially important for small organizations. Since they can benefit from direct communication, processes with a lot of bureaucracy bring a lot of overhead to them. Also generally they do not have full-time staff that can be dedicated to software process improvement studies. SPI studies are generally carried out by senior developers as a part-time work. Since the majority of software organizations are small, it is important to find ways on how to apply software process assessment and improvement methods to small organizations.

According to a 2001 survey conducted in Brazil, only 8 percent of small software companies (1-49 employees) used ISO 9001, and only 2 percent used CMM. These results compare to 43 percent of large organizations

(100+ employees) using ISO 9001 and 11 percent of large organizations using CMM [28]. Wangenheim et al. [28] developed the MARES method, which is a set of well-structured guidelines for conducting ISO/IEC 15504-conformant software process assessments in small companies. Demirörs et al. [29] developed and implemented a work plan to provide an ISO 9001 compliant quality system in a small software organization. Both of these studies show that small organizations can also get benefit from software process assessment and improvement models and standards.

According to another survey [32] on the implementation of SPI in 85 UK companies, gaining management commitment to SPI, tailoring SPI to the needs of the organization and aligning SPI goals with organizational goals are found to be the most important success factors of SPI. 68% of companies consider SPI to have been successful whereas 23% consider SPI to have been less than successful. Only 29% measured the impact of SPI, whereas 54% measured effort spent for SPI. This shows that companies tend to measure the cost of SPI without considering its benefits and its return on investment (ROI).

2.2 Requirement Engineering Process Improvement

SPI need not be applied to the whole software development process; it may also be applied to one or more phases. The waterfall model of software development process is given in Figure 2-1. This is the basic model and it can be said that all other models are based on this model more or less. A typical software development process is composed of requirements, design, coding, testing and integration phases.

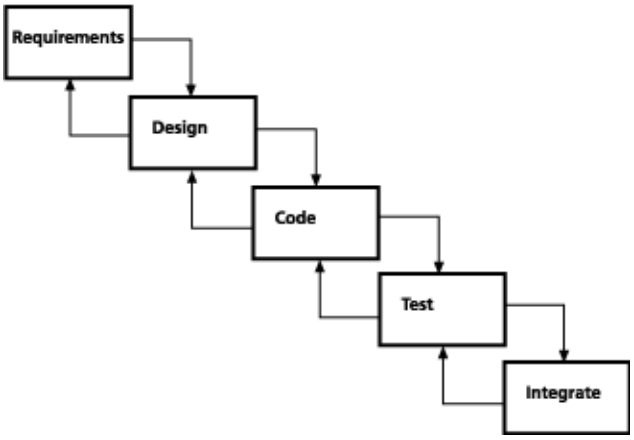

Figure 2-1 Waterfall Model

A software requirement is defined as [35]:

- A software capability needed by the user to solve a problem to achieve an objective
- A software capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documentation

Requirements engineering (RE) involves the systematic process of eliciting, understanding, analyzing, documenting and managing requirements throughout a product's life cycle [34]. RE activities can be decomposed into two as requirements development and requirements management. Requirements development process consists of eliciting, analyzing, documenting and validating requirements. Requirements management is the process of managing changes to software requirements. RE activities are given in Figure 2-2.

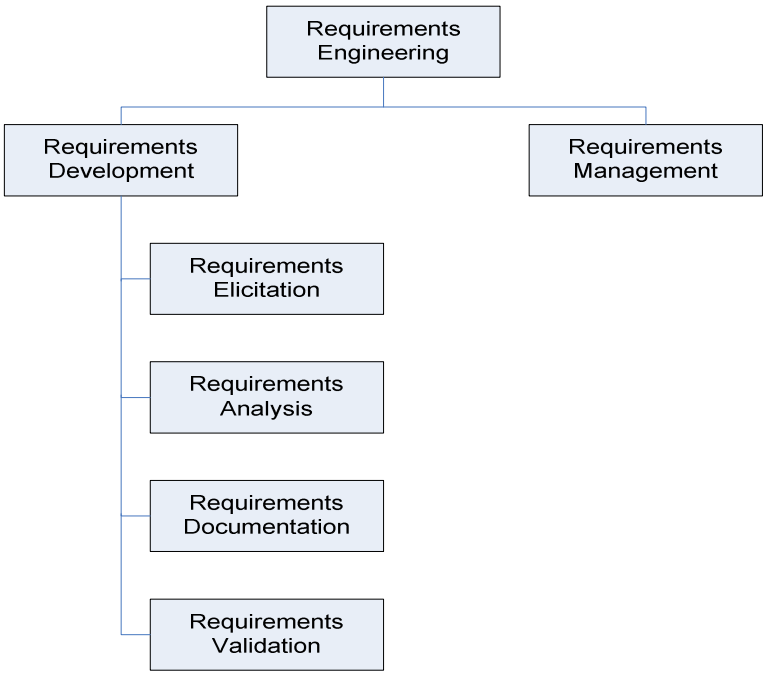

Figure 2-2 Requirements Engineering Activities

Figure 2-3 [39] shows the phases of software development life-cycle and related testing and acceptance phases. As it can be seen from the figure, software requirements are not only used in preliminary design, but also in planning software system test.

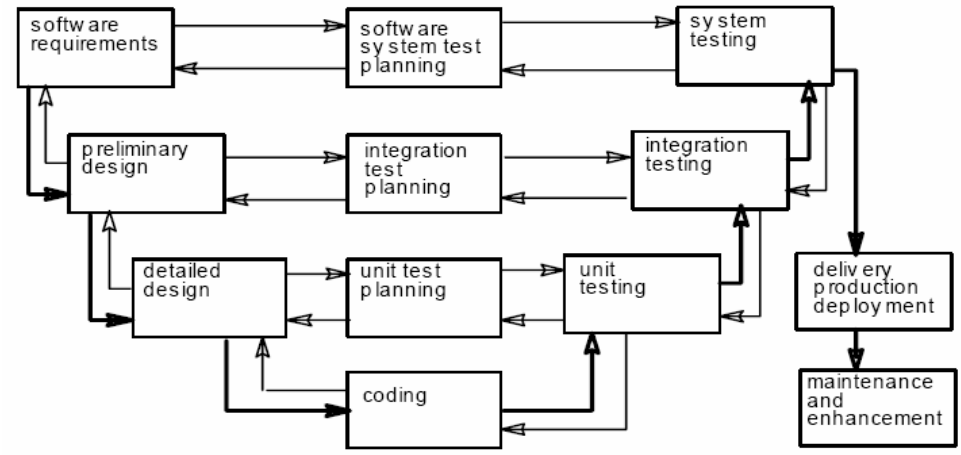

Figure 2-3 Software Development Life-Cycle [39]

In a large system development, the software requirements specification may play a variety of roles [33]:

- For customers, the requirements typically document what should be delivered and may provide the contractual basis for the development.
- For managers it may provide the basis for scheduling and a yardstick for measuring progress.
- For the software designers, it may provide the "design–to" specification.
- For coders it defines the range of acceptable implementations and is the final authority on the outputs that must be produced.
- For quality assurance personnel, it is the basis for validation, test planning and verification.

The purpose of requirements phase is always stated to be to define what to build without specifying how to build. Unfortunately the border between "what" and "how" is not so clear. Therefore there exists a "what versus how" dilemma which can be defined as "One person's *how* is another person's *what*" [42]. Davis [42] gives a detailed explanation of this

dilemma.

Brooks states the characteristics of requirements engineering in “No Silver Bullet” [5] very well as:
“The hardest single part of building a software system is deciding precisely what to build... No other part of the work so cripples the resulting system if done wrong. No other part is more difficult to rectify later.”

According to The Standish Group’s Chaos Report [3], lack of user input, incomplete requirements & specifications, and changing requirements & specifications are the most important factors that cause projects to be challenged, that is completed but over-budget, over the time estimate, and with fewer features than originally specified. Moreover it is found that incompleteness of requirements is the most important factor that causes projects to be cancelled. These results show the importance of requirements engineering activities in the software development process.

C. Jones [36] provided data regarding the likely number of potential defects introduced in various phases of a software development project and the typical efficiency with which a development organization removes those defects. The data is summarized in Table 2-1.

Table 2-1 Defect Summary

Defect Origins	**Defect Potentials**	**Removal Efficiency**	**Delivered Defects**
Requirements	1.00	77%	0.23
Design	1.25	85%	0.19
Coding	1.75	95%	0.09
Documentation	0.60	80%	0.12
Bad fixes	0.40	70%	0.12
Total	5.00	85%	0.75

As it can be seen from the table, one of every 5 defects is introduced in the requirements phase, and their removal efficiency is the lowest. Moreover one third of delivered defects are introduced in the requirements phase, and their cost is too high.

Relative cost to repair a defect at different lifecycle phases is given in Figure 2-4. If the cost of fixing a defect in the coding phase is one, its cost is between 0.1 and 0.2 in the requirements phase and it is 20 in the maintenance phase [35]. Therefore 200:1 cost saving results from finding errors in the requirements phase versus finding errors in the maintenance phase. The reason for this is that erroneous requirements cause erroneous design which in turn causes erroneous coding. So the amount of rework increases as defects are found later in the process.

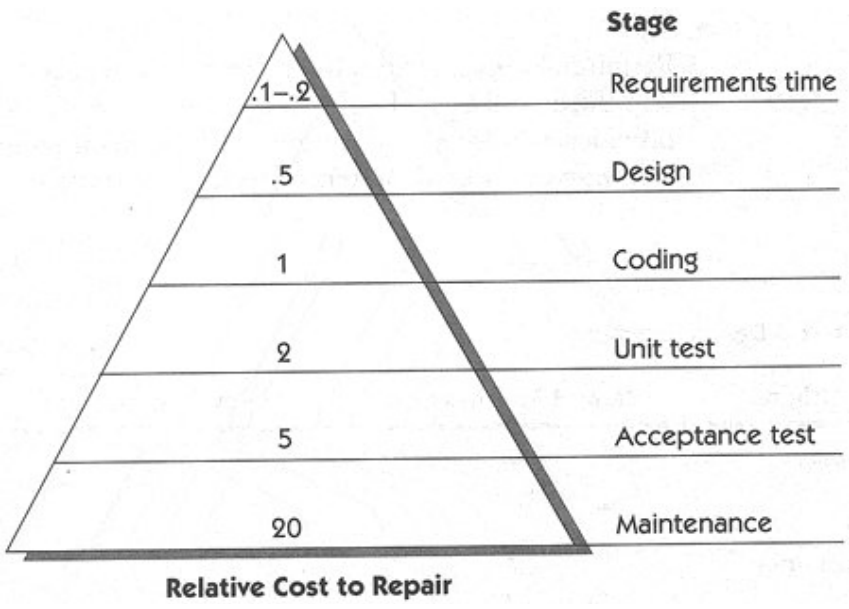

Figure 2-4 Relative Cost to Repair a Defect at Different Lifecycle Phases [35]

Although it is evident that requirements process improvement will provide great benefits to organizations, very few organizations have an explicitly-defined and standardized requirements engineering process [34]. According to Neill and Laplante's survey of requirements engineering's state of practice [37], 33 percent of respondent companies don't use any methodology for requirements analysis and modeling. 52 percent of respondents don't think that their company does enough requirements engineering whereas 29 percent are satisfied with the amount of their organization's requirement efforts.

Davis and Hickey [38] suggested that a reason why the results of requirements engineering research are not used in practice is that requirements engineering researchers do not practice what they preach: they do not analyze the problems of requirements engineering practice, and therefore their solutions do not address these problems. According to them if requirements engineering researchers would follow the first rule of any requirements engineer, i.e., 'Know the customer', more of the research would prove to be helpful in practice.

Being relatively a new field, there are some studies on requirements process improvement. Damian et al. [40, 41] present findings from a case-study of an organization that undertook a requirements process improvement initiative. Before the requirements process improvement initiative, the requirements were compiled in a document containing one-line description of the features. The requirements were not fully understood by developers and they were ambiguous in stating the required functionality. The requirements engineering process was revised to include requirements analysis sessions to refine the feature requests and derive more detailed functional requirements. After a full life-cycle of a project with improved requirements process, engineers unanimously agreed that the requirements process revealed further details, dependencies and complexities of features. 64 percent of respondents felt

that there had been less rework under the revised process and 90 percent thought that requirements artifacts had been helpful to validate the coverage of features. Also over 80% of respondents found that the thorough analysis of features was important in estimating effort required during design and implementation.

Sommerville and Sawyer developed a requirements process maturity model and an associated method for requirements process assessment which are documented in Requirements Engineering – A Good Practice Guide (REGPG) [44]. There are three levels of maturity in the model which are Initial, Repeatable, and Defined. These levels are similar to CMM levels. REGPG includes 66 good practice guidelines and a method of assigning scores to them. The process maturity assessment method assesses the use of RE practices in an organization and determines the maturity level of the organization according to the results.

Sommerville and Ransom [43] conducted an empirical study of industrial RE process assessment and improvement. They used Sommerville and Sawyer's requirements process maturity model [44] to assess the requirements process of 9 companies from different domains. According to the assessment results, they made proper improvement suggestions to the companies. After 10 months of process improvement efforts, the companies were reassessed. Reassessment results showed that companies had managed to improve their requirements processes as they increased the number and the usage level of requirements practices. Moreover some of the companies increased their requirements engineering process maturity levels.

The Requirements Engineering Process Maturity (REPM) [45] is a model that was developed in order to assess an organization's requirements process in a fast and cost-effective way. The model is especially targeted at Small and Medium sized Enterprises (SMEs) which lack the resources

to apply exhaustive assessments using other models like CMMI. It is a staged model with five levels. There are actions each of which is associated with a level from one to five and classified according to three process activities: Elicitation, Analysis and Negotiation, and Management. The results of the pilot study involving four SMEs indicate that the method yields useful results.

Davis and Zowghi, being devoted to the field of requirements as active researchers and practitioners, think that although there are many good requirements practices exist they are neither necessary nor sufficient for a project's success [46]. According to them, it is possible that you do a perfect job of requirements but if the subsequent design and coding stages introduce millions of errors it is clear that the project will not be successful. Although it is not very frequent, it is also possible for a project to be successful without applying good requirements practices.

2.3 A Pre-Enactment Model for Measuring Process Quality

ISO/IEC 9126 [47-50] standard provides a comprehensive model for evaluating quality of software products. It can be used for developing or selecting high quality software. The software product is evaluated for every relevant quality characteristics in the model by using validated and widely accepted metrics. There are six characteristics defined in the model which serve as the building blocks of software product quality. These characteristics are: Maintainability, Reliability, Functionality, Usability, Efficiency and Portability. Each of these quality characteristics is further refined into sub-characteristics. For example analyzability, changeability, stability, testability and compliance are the sub-characteristics of maintainability. ISO/IEC 9126 also defines one or more metrics to measure each of its sub-characteristics.

Güceğlioğlu [51] developed a model for measuring the software process quality. The model is based on the ISO/IEC 9126 Software Product Quality Model and adapts or redefines some of ISO/IEC 9126's software quality metrics to the process concept. It also defines some new metrics that can be used for measuring the process quality. Güceğlioğlu suggests using the model in software process improvement studies in order to measure impacts of the process improvement studies on process quality.

Güceğlioğlu [51] used the relationship between the software product and the process in his study. He claims that when logical structures of software product and process are compared, it can be recognized that "software product" logically matches with "process", and "function" of the software product with "activity" of the process. The relation between process and activity can be described as "activity is one of the subunits of the process and represents a logical completeness in its context." A similar relation exists between software product and function. Both of them constitute a part of the whole and have interactions with other parts. Moreover, high quality is of prime importance for both of them.

In order to measure the process quality two basic inputs are used. The first one is the static process definitions of processes ("AS-IS" in practice). The second one is available regulatory or guideline documents ("process in theory" or "on paper") about the processes in the organization where the processes are operated currently. Graphical modeling of both inputs facilitates the measurements, since interactions between processes and activities can be easily identified. Using the proposed model, "AS-IS" models of processes can be compared to the "on paper" models [51].

Instead of "AS-IS" modeling "TO-BE" modeling can also be used to compare "TO-BE" models of processes to processes in regulatory or guideline documents. Also both "AS-IS" models and "TO-BE" models can be used in order to find out the impacts of new arrangements on the

processes [51].

The model is designed with a four-leveled structure. The first level is called as category and there is one category as “quality” in the model. The second level is called as characteristic. Each category has its own characteristics. The quality category includes maintainability, reliability, functionality and usability characteristics. The third level is for sub-characteristics and finally, fourth level is for metrics for measuring the process quality attributes [51].

The characteristics, sub-characteristics and metrics for quality category are as follows [51]:

- Maintainability
    - Analyzability Metrics
        - Complexity
        - Coupling
- Reliability
    - Fault Tolerance Metrics
        - Failure avoidance
    - Recoverability Metrics
        - Restorability
        - Restoration effectiveness
- Functionality
    - Suitability Metrics
        - Functional adequacy
        - Functional completeness
    - IT Based Functionality Metrics
        - IT usage
        - IT density
    - Accuracy Metrics
        - Computational accuracy
    - Interoperability Metrics

        - Data exchangeability
    - Security Metrics
        - Access auditability
- Usability
    - Understandability Metrics
        - Functional understandability
    - Learnability Metrics
        - Existence in documents
    - Operability Metrics
        - Input validity checking
        - Undoability
    - Attractiveness Metrics
        - Attractive interaction

The “AS-IS” and/or “TO-BE” and “on paper” process models are used in measurements related to each metric. The results of the measurements can be used to find out the conformance of static process definitions (“AS-IS”) to processes in organization’s documents (“on paper”) or to compare “TO-BE” models of processes to processes in regulatory or guideline documents. Also both “AS-IS” models and “TO-BE” models can be used in order to find out the impacts of new arrangements on the processes. By this way, before putting “TO-BE” models into practice, an organization can measure its quality improvements. This method has the advantage of quantitatively comparing quality attributes of a proposed software process model to current “AS-IS” or “on paper” models before deploying the proposed model. This is expected to speed up and decrease cost of software improvement studies [51].

Sezer [52] conducted a study using Güceğlioğlu’s model [51] in order to find out the impacts of software design verification process improvement in one of the software engineering departments of a company. He concludes that there is a need for weighting the activities in the processes

according to their contribution or influence. In the current model, unimportant, straightforward activities and vital activities have the same contribution to the measurements. According to him, this may result in wrong conclusions about measurements.

Seçkin [53] also used Güceğlioğlu's model [51] in a study to evaluate a software requirements analysis and validation process. According to him the methodology can be used as a first step in software process improvement activities in order to decrease failure rate of software process improvement initiatives, but it is not sufficient for determining whether the improvement is applicable or not.

CHAPTER 3

CURRENT REQUIREMENTS MANAGEMENT PROCESSES

In this chapter, current on-paper and AS-IS software requirements management processes, their quality attribute measurements and the problems of the current processes are presented. Section 3.1 gives the on-paper definitions of the software requirements management processes of Embedded Software Department (ESD) of Company A. Section 3.2 gives the static process definitions and models of the current AS-IS software requirements management processes. In Section 3.3, quality attribute measurements of the AS-IS processes calculated using the metrics given in Güceğlioğlu's study [51] are given. Finally, in Section 3.4, the problems of the current software requirements management processes and improvement suggestions based on these problems are given.

There are two on-paper requirements management processes defined in ESD which are:

1. Software Requirements Analysis Process
2. Software Requirements Analysis Review Process

These two processes together are called as analysis in the regulatory documents and their descriptions are given in Section 3.1.

On the other hand, software requirements analysis review process which

is described in regulatory documents is not explicitly done in ESD, it is done as a part of the software requirements analysis process.

The AS-IS software requirements management processes applied in ESD are:

1. Software Requirements Analysis Process
2. Software Requirements Change Management Process (Does not have an on-paper definition)

The static process descriptions and models of AS-IS software requirements management processes are given in Section 3.2.

3.1 On-Paper Software Requirements Management Processes

In the analysis process, software requirements and constraints should be determined using system requirements. Review should be done in order to verify that software requirements are complete, consistent, traceable and acceptable. After review, software requirements should be approved by the authorities.

Project Design Manager, Analyst, Architect and Designer involve in software requirements analysis and review activities. Their roles are described as follows:

- Project Design Manager: Project Design Manager is responsible for planning and auditing the activities throughout the software development life-cycle.
- Analyst: Analyst is responsible for determining the software requirements using the system requirements and analyzing them.
- Architect: Architect is responsible for designing the architecture of software.

- Designer: Designer is responsible for designing the software module that is assigned to him. The design should meet the related requirements.

3.1.1 Software Requirements Analysis

Inputs:

- Product Description Document (PDD)
- Technical Specifications Document (TSD)
- System Requirements Document (SysRD)

To Do:

A1. Analyze requirements.

A2. Evaluate the feasibility of requirements. If needed request change and/or corrections.

A3. Determine software requirements.

A4. Prepare Software Requirements Specification (SRS)

A5. Prepare Software Requirements Traceability Matrix (SRTM)

Outputs:

- PDD and/or SysRD Change Report
- Software Requirements Specification (SRS)
- Software Requirements Traceability Matrix (SRTM)

Staff:

- Project Design Manager
- Analyst
- Architect

3.1.2 Software Requirements Analysis Review

Inputs:

- Product Description Document (PDD)

- System Requirements Document (SysRD)
- Software Requirements Specification (SRS)
- ESD Review Guide

To Do:

R1. Review SRS.

R2. Verify that SRS satisfies all of the requirements related to software in PDD and/or SysRD.

R3. Update SRS.

Outputs:

- Software Requirements Analysis Review Report
- Software Requirements Specification (SRS)

Staff:

- Project Design Manager
- Analyst
- Architect
- Designer

3.2 AS-IS Software Requirements Management Processes

In this section, the requirements management processes, as actually applied in Embedded Software Department (ESD) of Company A will be described. This description has been compiled based on the author's own observations and interviews with several design leaders and senior design leaders. The process model constructed in this way was then reviewed by a senior design leader.

The AS-IS software requirements management processes applied in ESD are:

1. Software Requirements Analysis Process
2. Software Requirements Change Management Process (Does not have an on-paper definition)

The organization of Embedded Software Department (ESD) of Company A is as follows: There exists a department manager and under his management there are some software development units. Each unit consists of approximately 6 developers and a unit leader who is a senior design leader. In every unit there are design leaders and software engineers each of which take the role of a software architect, a requirements analyst, a designer, a coder or a unit tester as necesary.

Static process definition of software requirements analysis process as actually applied in ESD is given in Table 3-1. Models of software requirements analysis process are given in Figures 3-1, 3-2, 3-3 and 3-4. Static process definition of software requirements change management process as actually applied in ESD is given in Table 3-2. Models of software requirements change management process are given in Figures 3-5 and 3-6.

Table 3-1 AS-IS Software Requirements Analysis Process

No	Activity Name	Activity Definition	Staff	Forms/ Documents/ Archival Records/ Tools/ Applications/ Other Medias
1	Allocation Meeting	System requirements allocated to software are identified in a series of meetings.	Project Manager, Department Managers, Design Leaders	Conversation, TSD, PDD, SysRD (Does not generally exist), Minutes of Allocation Meeting (Does not generally exist)
2	Send TSD, PDD and SysRD for review	ESD Manager sends TSD, PDD and SysRD to design leaders in ESD in order to ask their opinions on feasibility of system requirements allocated to software.	ESD Manager, Design Leaders	TSD, PDD, SysRD (Does not generally exist), E-mail
3	Review TSD, PDD and SysRD	Design leaders in ESD review TSD, PDD and SysRD and send their opinions on feasibility of system requirements allocated to software to ESD Manager.	ESD Manager, Design Leaders	TSD, PDD, SysRD (Does not generally exist), E-mail
4	Feasibility meeting	Collected opinions on feasibility of system requirements allocated to software are evaluated in the feasibility meeting.	ESD Manager, Design Leaders	Conversation, Minutes of Feasibility Meeting (Does not generally exist)
5	Decide whether TSD, PDD or SysRD (if exists) should be changed or not	ESD Manager decides whether TSD, PDD or SysRD (if exists) should be changed or not, according to the discussions in the feasibility meeting	ESD Manager	-
6	Send change requests	If ESD Manager thinks that TSD, PDD or SysRD should be changed, change requests are sent to Project Manager.	ESD Manager, Project Manager	E-mail

Table 3-1 Continued

No	Activity Name	Activity Definition	Staff	Forms/ Documents/ Archival Records/ Tools/ Applications/ Other Medias
7	High level software requirements meeting	High level software requirements are identified by ESD Manager and design leaders.	ESD Manager, Design Leaders	Conversation, Minutes of High Level Software Requirements Meeting (Does not generally exist), Undocumented Software Requirements
8	Decide whether all high level software requirements are clear enough to proceed with determining low level software requirements or not	High level software requirements meetings are repeated until ESD Manager thinks that all high level software requirements are clear enough to proceed with determining low level software requirements.	ESD Manager	-
9	Low level software requirements meeting	High level requirements are analyzed and elaborated to identify detailed low level software requirements.	Unit Leader, Developers	Conversation, Minutes of Low Level Software Requirements Meeting (Generally exists), Undocumented Software Requirements
10	Decide whether all low level requirements are clear enough to proceed with design or not	Low level software requirements meetings are repeated for every module until unit leader thinks that all low level requirements are clear enough to proceed with design.	Unit Leader	-
11	Hardware interface requirements meeting	Hardware interface requirements are determined in a meeting attended by hardware engineers, developers and the unit leader that is responsible for the software module interfacing hardware.	Unit Leader, Developers, Hardware Engineers	Conversation, Minutes of Hardware Interface Requirements Meeting (Does not generally exist), Undocumented Software Requirements

Table 3-1 Continued

No	Activity Name	Activity Definition	Staff	Forms/ Documents/ Archival Records/ Tools/ Applications/ Other Medias
12	Decide whether all hardware interface requirements are clear enough to proceed with design or not	Hardware interface requirements meetings are repeated for each module interfacing hardware until the related unit leader thinks that all hardware interface requirements for a module are clear enough to proceed with design.	Unit Leader	-
13	Meeting to determine software modules to be re-used	Software modules to be re-used are determined.	Unit Leader, Developers	Conversation, Minutes of Re-use Meeting (Does not generally exist)
14	Decide whether it is necessary to make some modifications to the modules to be re-used	Unit Leader decides whether it is necessary to make some modifications to the modules to be re-used.	Unit Leader	-
15	Reverse engineer modules to be re-used	If unit leader thinks that it is necessary to make some modifications to modules to be re-used, these modules are reverse engineered from code to design and then from design to requirements, since there are no formal software requirements specification or software design description documents and code is the only formal source.	Unit Leader, Developers	Undocumented Software Requirements
16	Decide whether a proof-of-concept (throw-away) prototype exists or not	ESD Manager decides whether a proof-of-concept (throw-away) prototype exists or not (Very simple decision but exists as an activity for the sake of completeness).	ESD Manager	-
17	Update software requirements according to evaluation of the prototype	If a proof-of-concept (throw-away) prototype exists, software requirements are updated according to customer's requests and evaluation of the prototype.	ESD Manager, Design Leaders	Customer's requests (Written or oral), Undocumented Software Requirements

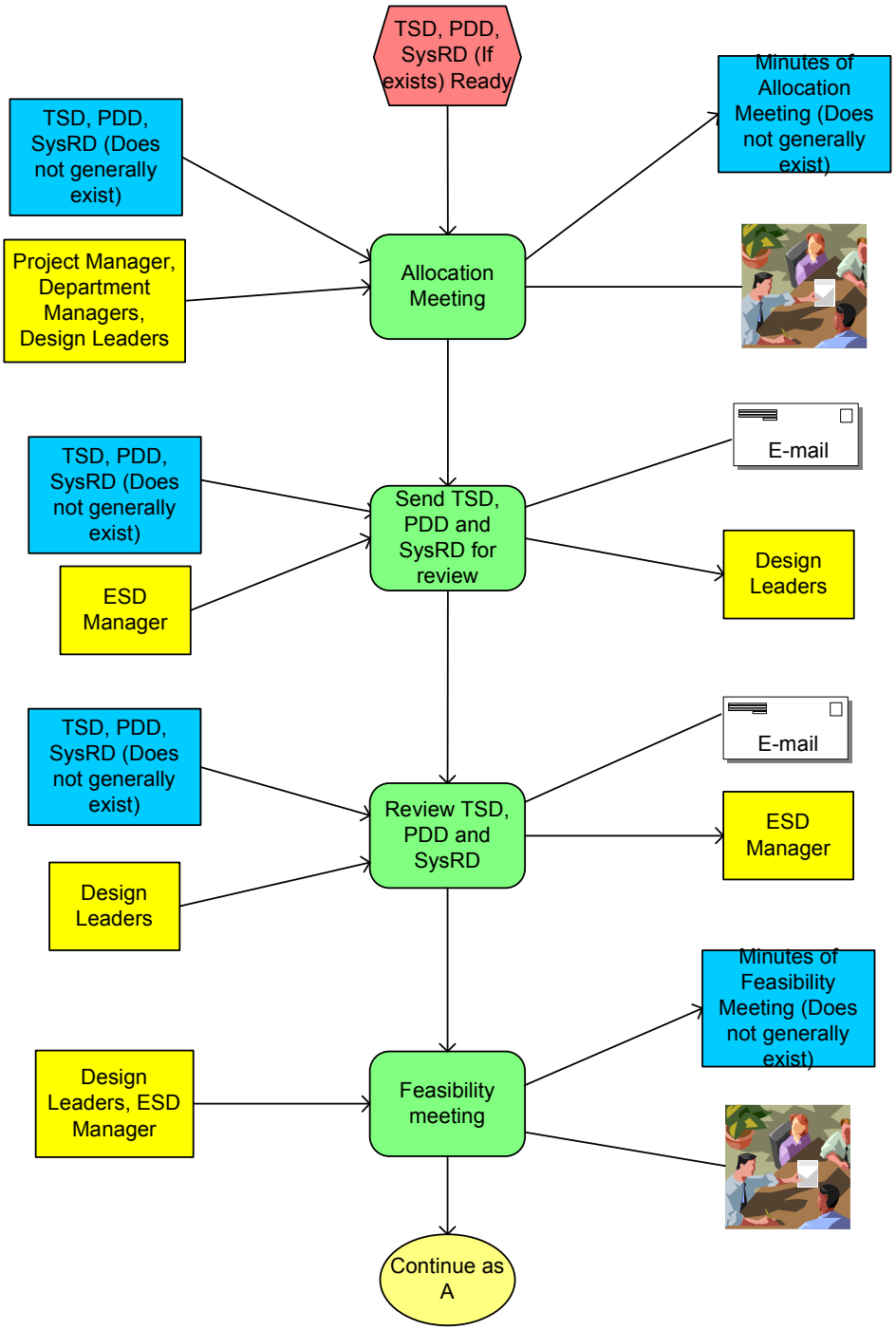

Figure 3-1 AS-IS Software Requirements Analysis Process Part 1

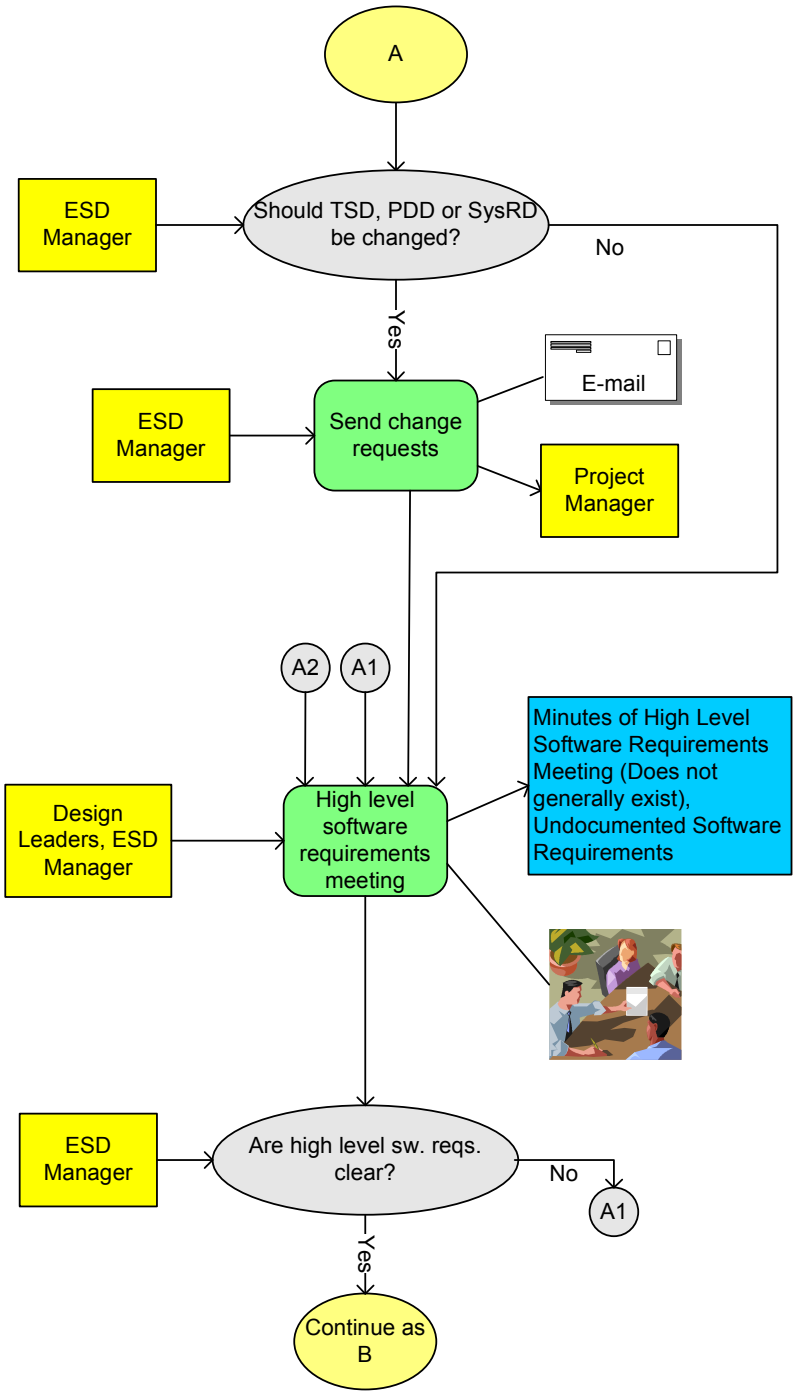

Figure 3-2 AS-IS Software Requirements Analysis Process Part 2

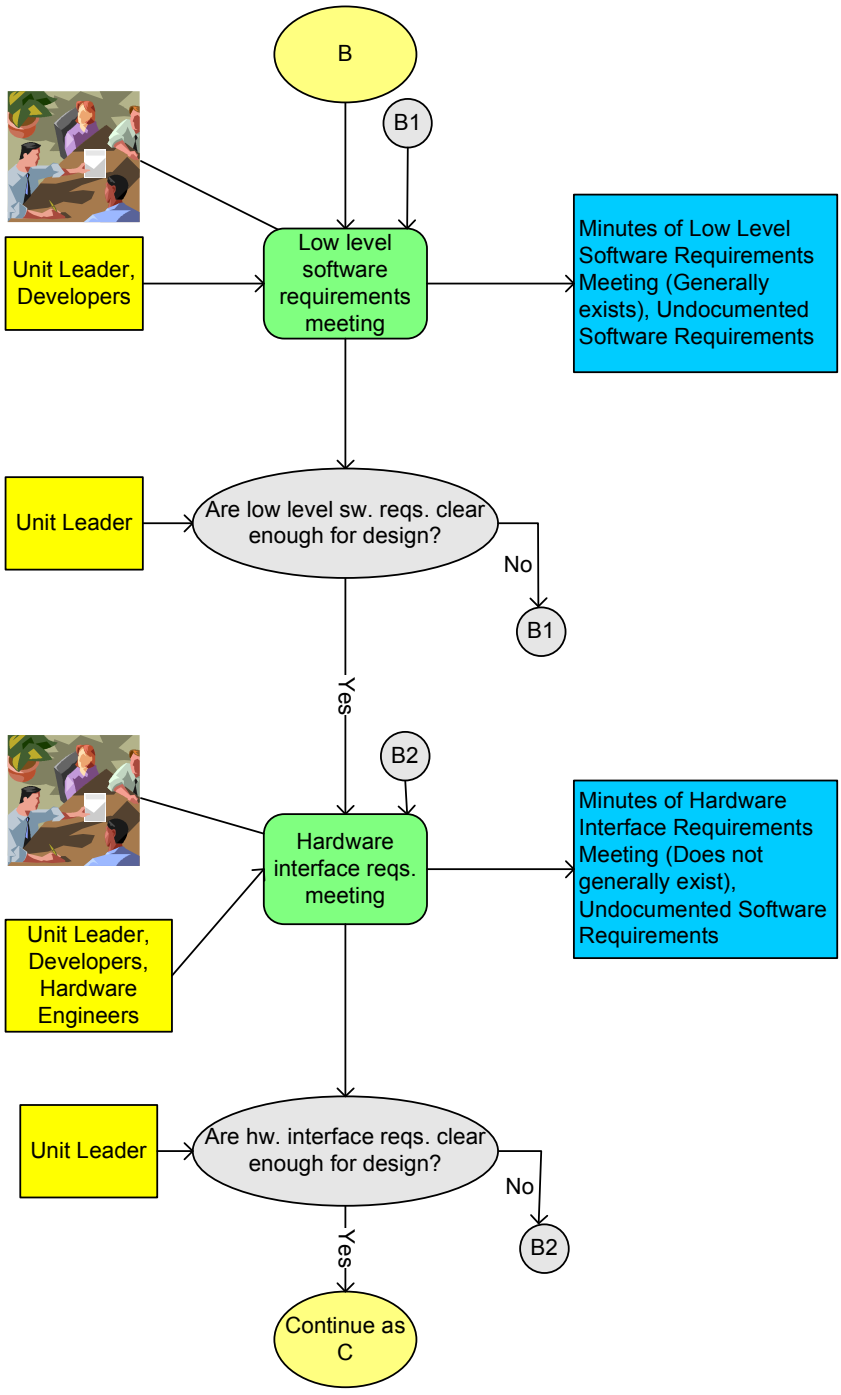

Figure 3-3 AS-IS Software Requirements Analysis Process Part 3

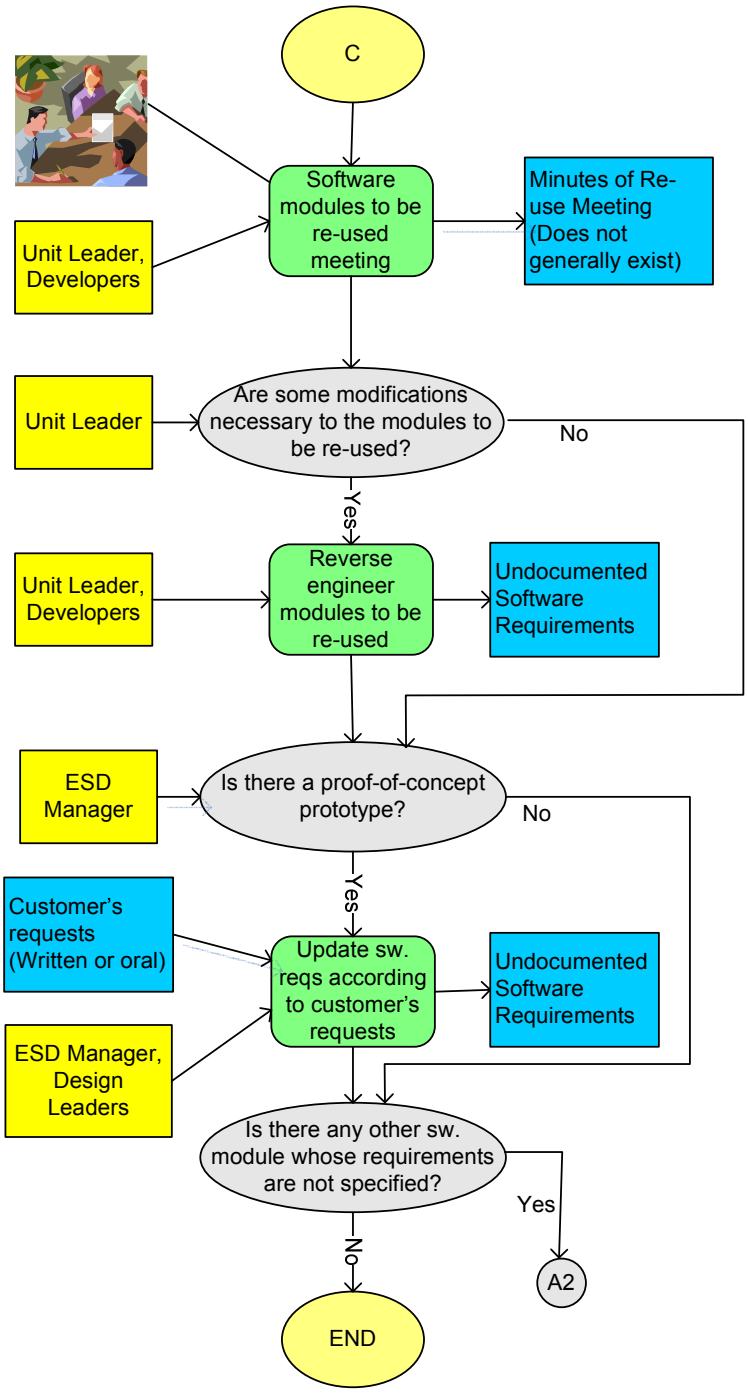

Figure 3-4 AS-IS Software Requirements Analysis Process Part 4

Table 3-2 AS-IS Software Requirements Change Management Process

No	Activity Name	Activity Definition	Staff	Forms/ Documents/ Archival Records/ Tools/ Applications/ Other Medias
1	Receive a requirement change request.	ESD Manager receives a requirement change request. It may be written or oral, and sourced by system engineers or customer.	ESD Manager	Requirement Change Request (Written or oral), E-mail, Telephone, Interview with customer
2	Meeting on feasibility of the requirement change request	Feasibility of the requirement change request is analyzed in a meeting attended by ESD Manager and related design leaders.	ESD Manager, Design Leaders	Conversation, Minutes of Feasibility Meeting (Does not generally exist), Requirement Change Request (Written or oral)
3	Decide whether the requirement change is acceptable, partially acceptable or not acceptable.	ESD Manager and design leaders decide whether the requirement change is acceptable, partially acceptable or not acceptable.	ESD Manager, Design Leaders	-
4	Accept the requirement change request	If ESD Manager and design leaders conclude that the requirement change request is feasible, ESD Manager accepts it by replying the source of change request via telephone or e-mail.	ESD Manager	E-mail, Telephone, Undocumented Software Requirements
5	Partially accept the requirement change request	If ESD Manager and design leaders conclude that the requirement change request is partially feasible, ESD Manager partially accepts it by replying the source of change request via telephone or e-mail.	ESD Manager	E-mail, Telephone, Undocumented Software Requirements
6	Reject the requirement change request	If ESD Manager and design leaders conclude that the requirement change request is infeasible, ESD Manager rejects it by replying the source of change request via telephone or e-mail.	ESD Manager	E-mail, Telephone

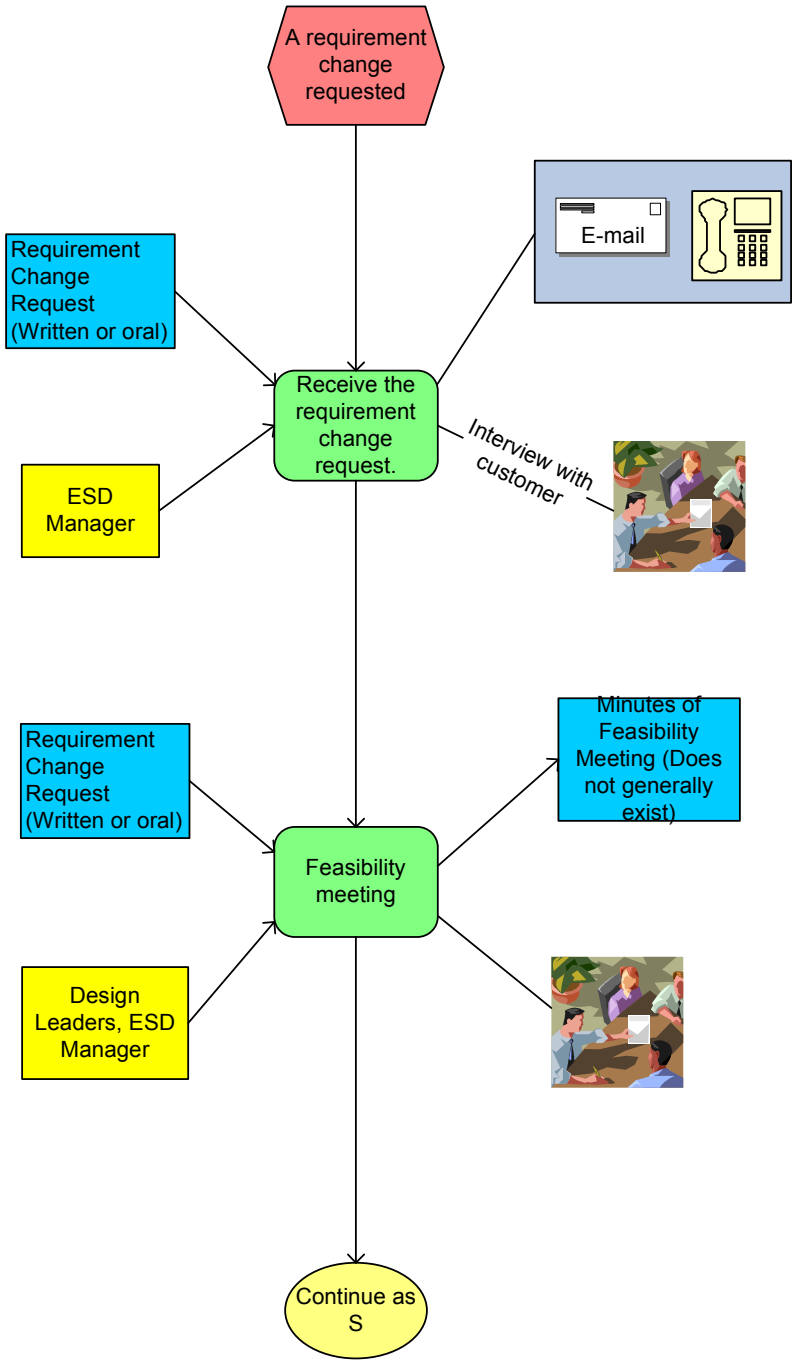

Figure 3-5 AS-IS Software Requirements Change Management Process Part 1

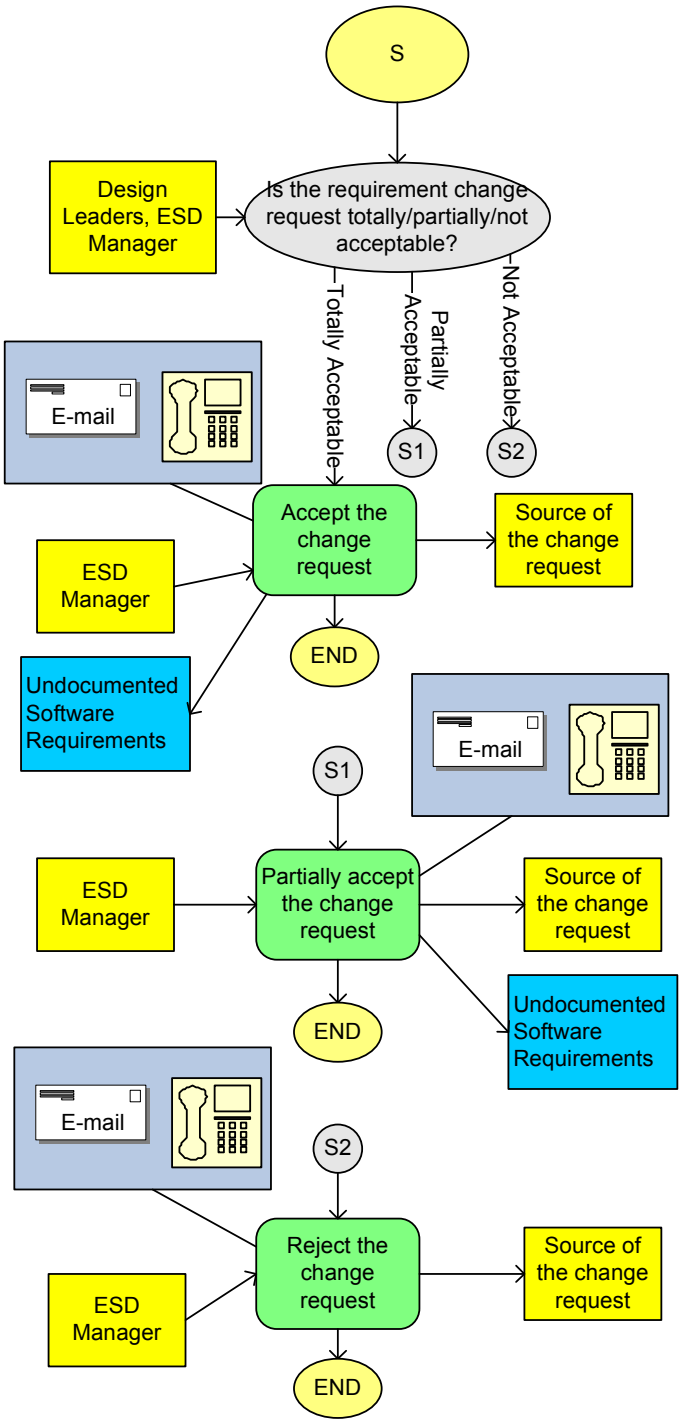

Figure 3-6 AS-IS Software Requirements Change Management Process Part 2

3.3 Measurement of AS-IS Software Requirements Management Processes

The quality of the AS-IS software requirements management processes are measured according to the method proposed in Güceğlioğlu's study [51]. Firstly, the process descriptions and the models of the software requirements management processes as actually applied in ESD, which are given in the previous section, are formed. Then measurements are performed on these process descriptions and graphical process models using the process quality metrics given in Güceğlioğlu's study.

The measurement details of AS-IS software requirements analysis process are given in Tables from A-1-1 to A-1-5, and the measurement details of AS-IS software requirements change management process are given in Tables from A-2-1 to A-2-5.

The summary of the results of the AS-IS process measurements is given in Table 3.3.

Table 3-3 AS-IS Measurement Results

Metrics	Software Requirements Analysis AS-IS Process (17 activities)	Software Requirements Change Management AS-IS Process (6 activities)
Complexity	X(1) = 1 / 17 = 0.059 X(2) = 5 / 17 = 0.294 X(3) = 0 / 17 = 0	X(1) = 0 / 6 = 0 X(2) = 1 / 6 = 0.167 X(3) = 0 / 6 = 0
Coupling	X = 4 / 17 = 0.235	X = 4 / 6 = 0.667
Failure Avoidance	X = 1 / 17 = 0.059	X = 0 / 6 = 0
Restorability	X = 4 / 17 = 0.235	X = 4 / 6 = 0.667
Restoration Effectiveness (Number of total activities is used in the formula)	X = 4 / 17 = 0.235	X = 4 / 6 = 0.667
Functional Adequacy	X = 17 / 17 = 1	Not applicable (On-paper process does not exist)
Functional Completeness (Number of on-paper activities is used in formula)	X = 1 - 5 / 8 = 0.375 (A1, A2, A3 are done, but A4, A5, R1, R2 and R3 are not done.)	Not applicable (On-paper process does not exist)
IT Usage	X = 4 / 17 = 0.235	X = 4 / 6 = 0.667
IT Density	X = 4 / 4 = 1	X = 4 / 4 = 1
Computational Accuracy	X = 0 / 5 = 0	X = 0 / 2 = 0
Data Exchangeability	X = 2 / 4 = 0.5	X = 3 / 4 = 0.75
Access Auditability	X = 4 / 4 = 1	X = 4 / 4 = 1
Functional Understandability	X = 3 / 17 = 0.176	X = 4 / 6 = 0.667
Existence in Documents	X = 0 / 17 = 0	X = 0 / 6 = 0
Input Validity Checking	X = 0 / 17 = 0	X = 0 / 6 = 0
Undoability	X = 4 / 17 = 0.235	X = 4 / 6 = 0.667
Attractive Interaction (Number of total activities is used in formula)	X = 0 / 17 = 0	X = 0 / 6 = 0

3.4 Problems of the Current Process and Suggestions

In this section, the problems that are related to the current software requirements management processes in ESD and solution suggestions to them are given. These problems and suggestions have been identified by the author according to her personal experience in Company A, and then discussed with several design leaders, and finally reviewed by a senior design leader.

The current software requirements management processes in ESD can be assessed as immature in many ways. This causes a lot of problems in both development and maintenance phases of software development.

The software requirements management related problems that are faced in ESD are given in the following sections.

3.4.1 No Documentation of Requirements

Problem: The first problem is that there is no documentation of requirements. In some of the meetings the decisions related to what to build, i.e. requirements, are taken, but no document related to requirements is prepared. Sometimes minutes of meeting is prepared, but even this does not always take place. Some of the undocumented requirements are written in minutes of meetings, some of them are written in some of the participants' notes, some of them exist in process participants' minds, but the others are forgotten.

Outcomes of the Problem: The results of not documenting requirements are very catastrophic.

1. Forgotten requirements cause producing software that does not meet customer's needs.

2. Incomplete requirements cause incomplete design, which in turn causes incomplete coding.
3. Not documenting requirements makes it impossible to review the requirements, and this causes a lot of misunderstandings. As a result, faulty and inconsistent designs come on the scene.
4. Since requirements are not documented, testing the final product to find out whether it meets the requirements or not is impossible. So this test is done later by the customer. When the customer finds out that some requirements are missing, a lot of rework has to be done to repair the faulty design and coding. Moreover, design is not documented, and this deteriorates the condition further.
5. Traceability links between the requirements are also not documented. This prevents taking the necessary steps when a requirement is changed. Since the traceability links between the requirements are not documented, the other requirements or the design blocks that should be changed when a requirement is changed are not known precisely. This leads to incomplete or inconsistent designs.
6. The other result of not documenting requirements is that it makes the company dependant on the developers to have information about the requirements. In case the developers having knowledge on requirements quit the job, the company will be in big trouble.

Example Situations: Many example situations can be given related to the problem of not documenting requirements. Two of them are as follows:

1. During the acceptance tests of project P1, the customers recognized that the product was not behaving as they wanted. Later it was found out that the requirements of a feature was misunderstood, and therefore implemented wrongly.
2. Since the requirements are not documented, test team does not know what to test in detail. During the acceptance tests of Project

P1, developers were invited to test a complex feature, since they were the only ones that knew the details of the feature and how to test it.

Suggestions: In order to eliminate the bad consequences of the problem of not documenting the requirements, following suggestions are made by the author and several design leaders:

1. **Requirements Management Tool:** Requirements management tools facilitate the requirements management processes by providing an IT based, easy to use interface to document the requirements. They provide traceability links not only between software requirements, but also between software requirements and design blocks, system requirements and test cases. Moreover measurements can be taken using the tool such as number of requirements covered in a release, number of changed requirements and so on. These measurements can be used to improve the process continually. DOORS[1] is a tool that has these features and can be used in ESD.
2. **Assigning Unique Numbers to Requirements:** Assigning unique numbers to requirements will facilitate communicating requirements and tracing the relationships between requirements.
3. **Software Requirements Specification (SRS):** The requirements stored in the requirements management tool database can then be imported to a Word Document. This document is the Software Requirements Specification (SRS) and includes all types of software requirements such as functional and non-functional. This document can be put under version control and can be used to communicate requirements with other departments.

[1] DOORS is used in other departments of Company A. There are some other RM tools such as RequisitePro, AnalystPro, ARTS and RTM that have similar features. INCOSE Requirements Management Tools Survey [54] provides a detailed comparison of the features of approximately 40 requirements management tools.

4. **Requirements Traceability Matrix (RTM):** The traceability links between the requirements can be documented in Requirements Traceability Matrix. This document can also be formed using the requirements management tool. RTM contains not only the relationships between requirements, but also between software requirements and design blocks, system requirements and test cases. When a requirement has to be changed, the other requirements that depend on the requirement to be changed can be found using RTM. Then, if necessary the dependent requirements can be changed, too.
5. **SRS & RTM Templates:** SRS and RTM templates can be used to facilitate preparing and managing SRS and RTM documents by providing a common format.
6. **Minutes of Meeting:** To document the decisions taken in the meetings and the rationales of them, it is decided to write minutes of meeting in all meetings.
7. **Review of Requirements and Requirements Review Guideline:** Since the requirements will be documented in the improved process, it will be possible to review them. Reviewing is necessary to ensure that the requirements are correct, unambiguous, complete, consistent, traceable and verifiable. Requirements Review Guideline will guide in reviews by providing the necessary steps to be taken to check whether the requirements are correct, unambiguous, complete, consistent, traceable and verifiable.

3.4.2 Not Having a Formal Requirements Change Management Process

Problem: The second problem of ESD's requirements process is not having a formal requirements change management process. When a requirement has to be changed, someone (customer or systems engineer or product engineer) tells this to a design leader or the manager, and the

task of implementing related things according to this change is given to a related developer. Again nothing is documented, and this information exists in a couple of people's minds convenient to be forgotten or misunderstood.

Outcomes of the Problem: Not having a formal requirements change management process has some costs.

1. Since nothing is documented about the change, and the information about the changed requirements exists in several people's minds, if these people quit the job, no one will be aware of this requirement change.
2. Sometimes requirement change requests are deferred to be accomplished later. In this case, since the requests are not documented, they may be forgotten or implemented incompletely.

Example Situations: Two experiences related to this problem are as follows:

1. The requirements of a service that is provided by project P1 was only known by a senior design leader. After that senior design leader quit the job, the customer requested a change about that service. But since the requirements are not documented and the only staff that had information on the requirements of the service quit the job, several developers worked hard to find out the design from the code, and then the requirements from the design to implement the required change.
2. In project P2, the customer requested a feature to be removed which was not needed by them anymore. The feature was removed but the removal request was not documented. During the integration tests, test engineers tried to test that removed feature and concluded that the test had failed. Later, it was understood that the feature had been removed and therefore its test is obsolete.

Suggestions: Following suggestions can help to improve the requirements change management process applied in ESD:

1. **Requirement Change Request Report (RCRR):** When a requirement change request is received from the customer, preparing Requirement Change Request Report will help to communicate the request in ESD. The report will contain the unique numbers of the requirements that are requested to be changed and the other requirements and software modules that depend on the requirements that are requested to be changed. After the feasibility meeting about the change request, RCRR will be updated to contain the decision taken related to the request.
2. **Requirements Management Tool:** Use of requirements management tool will facilitate tracing the relationships between the requirements to be changed and the requirements and software modules that depend on them.

3.4.3 Not Having a Requirements Re-use Mechanism

Problem: In most cases ESD does not start a project from scratch, but starts building a new project on previous ones. This type of development is very difficult to manage. This confuses the developers, since it is hard to remember undocumented requirements of different projects which are very similar but having some nuances. Actually, not starting projects from scratch is not a problem; it is a fact, but the way this fact is handled is problematic in the current situation.

Outcomes of the Problem: Some of the outcomes of this situation are as follows:

1. When building new projects on previous ones generally it is needed to reuse some of the software modules of the previous

projects. Sometimes a few modifications are needed to be done to the modules to be reused. Since there are no requirements or design documents, there are two ways to understand what a module does in detail. The first one is to find the developer related to this module and ask some questions to him. If you are lucky you can manage to find the developer worked on this module in the previous projects. Most of the time, this is not the case, and you work hard to get the design from the code and get the requirements from the design.

2. Building new projects on previous ones cause inconsistencies between projects. For example assume Project X is in maintenance phase, and a problem is reported by the customer. After the problem is solved by developers, the same problem continues to exist in several new projects built on Project X. There is not a document showing whether the requirement related to this problem is also a requirement for the new projects built on Project X or not. This is a very prevalent and detrimental problem in ESD.

Example Situations: Two experiences related to this problem are as follows:

1. Project P4 was based on Project P3. It had to cover all the features of Project P3 and besides some new features. Neither the requirements nor the designs and codes of Project P3 were re-usable and documented. Moreover the teams of Project P4 and Project P3 were different. So the project started from the requirements phase with a different team. Team of Project P4 arranged many meetings with the team of Project P3. These meetings wasted the time of both teams. At the end the customer was surprised, because the services that have the same names were showing different behaviors in Projects P3 and P4.
2. Project P5 which is in maintenance phase included the features of Project P2 which is in maintenance phase, either. Project P6 had to

include the features of Project P5. A problem about a feature of Project P2, which was also included in Projects P5 and P6, was reported by the customer. The developers solved the problem and corrected in Project P2, but the problem continued to exist in Projects P5 and P6. Later, the same problem is reported for Projects P5 and P6.

Suggestions: The following suggestions can be helpful to solve this problem.

1. **Requirements Management Tool:** All of the requirements of the projects should be stored in requirements management tool database with unique numbers. The traceability links between re-used requirements of different projects should be formed.
2. **Requirements Traceability Matrix (RTM):** The traceability links between re-used requirements of different projects formed in the requirements management tool should be documented using RTMs.

All of these problems cause chaos not only in the requirements management process, but also in the other software development processes. Moreover, all of these problems directly or indirectly cause wasting of time and accordingly most of the projects can not be finished on time and on budget.

One may think that how it is possible to produce acceptable products in spite of all of these problems. The answer is that the domain of the software projects developed in ESD has been approximately the same. The projects are similar to each other. Most of the time, new projects are developed by adding some improved features to old projects.

There are domain experts that have a good understanding of the domain of software projects. Therefore an intense elicitation process has not been

needed. There has not been a systematical process for determining and managing requirements. The undocumented knowledge of these domain experts have informally been communicated throughout the projects and by this way the department managed to produce successful products.

The projects are getting bigger and more complex in ESD. Also the domain gets wider and more people are joined to the development process. Under these circumstances, it is impossible for domain experts to handle all of the requirements in their minds. Therefore it is necessary to improve the ad-hoc requirements management process of ESD and maintain a systematical requirements management process.

CHAPTER 4

IMPROVED REQUIREMENTS MANAGEMENT PROCESSES

In this chapter, improved software requirements analysis and software requirements change management processes are given. The improvements are based on the solution suggestions to the problems given in Section 3.4. The improvements are proposed by the author according to interviews with several design leaders. The improvements are then reviewed by a senior design leader.

Section 4.1 presents the improved (TO-BE) software requirements management processes. Section 4.2 gives the results of the measurements calculated using Güceğlioğlu's [51] proposed method. Section 4.3 is composed of a discussion and comparison of the measurements of the AS-IS and TO-BE software requirements management processes of ESD.

4.1 TO-BE Software Requirements Management Processes

In this section, improved versions of software requirements processes are given. Table 4-1 gives the comparison of activities of AS-IS and TO-BE software requirements analysis processes. The changes between AS-IS and TO-BE processes and their rationales are specified in the table.

Figures 4-1, 4-2, 4-3, 4-4 and 4-5 present the model of TO-BE software requirements analysis process. The static process description of TO-BE software requirements analysis process is given in Table B-1-1, in Appendix B.

Table 4-2 gives the comparison of activities of AS-IS and TO-BE software requirements change management processes. The changes between AS-IS and TO-BE processes and their rationales are specified in the table. Figures 4-6, 4-7 and 4-8 present the model of TO-BE software requirements change management process. The static process description of TO-BE software requirements change management process is given in Table B-2-1, in Appendix B.

Table 4-1 Comparison of AS-IS and TO-BE Software Requirements Analysis Processes

AS-IS Activities	TO-BE Activities	Change & Rationale
1. Allocation Meeting: Using PDD, TSD and SysRD (if exists), system requirements allocated to software are identified in a series of meetings attended by Project Manager, Department Managers and Design Leaders.	**1. Allocation Meeting:** Using PDD, TSD and SysRD (if exists), system requirements allocated to software are identified in a series of meetings attended by Project Manager, Department Managers and Design Leaders. Minutes of Allocation Meeting and Allocation Document which specifies system requirements that are allocated to software are prepared.	**Minutes of Allocation Meeting:** To document the decisions and their rationales. **Allocation Document:** To clearly document which system requirements are allocated to software.
2. Send TSD, PDD and SysRD for review: ESD Manager sends TSD, PDD and SysRD (if exists) to design leaders in ESD in order to ask their opinions on feasibility of system requirements allocated to software.	**2. Send TSD, PDD, SysRD and Allocation Document for review:** ESD Manager sends TSD, PDD, SysRD (if exists) and Allocation Document to design leaders in ESD in order to ask their opinions on feasibility of system requirements allocated to software.	**Sending Allocation Document:** Allocation Document is sent to clearly specify which system requirements are allocated to software.
3. Review TSD, PDD and SysRD: Design leaders in ESD review TSD, PDD and SysRD (if exists) and send their opinions on feasibility of system requirements allocated to software to ESD Manager.	**3. Review TSD, PDD, SysRD and Allocation Document:** Design leaders in ESD review TSD, PDD, SysRD (if exists) and Allocation Document and send their opinions on feasibility of system requirements allocated to software to ESD Manager.	**Reviewing Allocation Document:** Allocation Document is reviewed to clearly understand which system requirements are allocated to software.
4. Feasibility Meeting: Collected opinions on feasibility of system requirements allocated to software are evaluated in the feasibility meeting attended by ESD Manager and design leaders.	**4. Feasibility Meeting:** Collected opinions on feasibility of system requirements allocated to software are evaluated in the feasibility meeting attended by ESD Manager and design leaders. Minutes of Feasibility Meeting is prepared.	**Minutes of Feasibility Meeting:** To document the decisions and their rationales.
5. Decide whether TSD, PDD or SysRD should be changed or not: ESD Manager decides whether TSD, PDD or SysRD (if exists) should be changed or not, according to the discussions in the feasibility meeting.	**5. Decide whether TSD, PDD or SysRD should be changed or not:** ESD Manager decides whether TSD, PDD or SysRD (if exists) should be changed or not, according to the discussions in the feasibility meeting.	No change
6. Send change requests: If ESD Manager thinks that TSD, PDD or SysRD (if exists) should be changed, change requests are sent to Project Manager via e-mail.	**6. Send change requests:** If ESD Manager thinks that TSD, PDD or SysRD (if exists) should be changed, he prepares System Requirement Change Request Document (SRCRD) and sends it to Project Manager via e-mail.	**Preparing and Sending System Requirement Change Request Document (SRCRD):** To document system requirement change requests.
7. High level software requirements meeting: High level software requirements are identified in a meeting attended by ESD Manager and design leaders.	**7. High level software requirements meeting:** High level software requirements are identified in a meeting attended by ESD Manager and design leaders. Minutes of High Level Software Requirements Meeting is prepared.	**Minutes of High Level Software Requirements Meeting:** To document the decisions and their rationales.

Table 4-1 Continued

AS-IS Activities	TO-BE Activities	Change & Rationale
Does not exist.	**8. Prepare/Update SRS and RTM:** Design leaders store identified high level requirements with a unique number in a database by the help of a requirements management tool. SRS and RTM are prepared/updated by the help of the requirements management tool and using SRS and RTM Templates.	**Prepare/Update SRS&RTM:** To document and uniquely identify software requirements. To trace relationships between requirements; and also between requirements and software modules. **Requirements Management Tool:** To facilitate recording, tracing and managing requirements. **SRS&RTM Templates:** To facilitate preparing and managing documents by providing a common format.
Does not exist.	**9. Review & Update SRS and RTM:** ESD Manager reviews SRS and RTM in order to ensure that the requirements are correct, unambiguous, complete, consistent, traceable and verifiable. ESD Manager uses Requirements Review Guideline and the requirements management tool while reviewing and updating SRS and RTM.	**Review & Update SRS and RTM:** To ensure that the requirements are correct, unambiguous, complete, consistent, traceable and verifiable. **Requirements Review Guideline:** To guide in reviewing SRS and RTM in order to ensure that the requirements are correct, unambiguous, complete, consistent, traceable and verifiable.
8. Decide whether all high level software requirements are clear enough to proceed with determining low level software requirements or not	**10. Decide whether all high level software requirements are clear enough to proceed with determining low level software requirements or not**	No change
9. Low level software requirements meeting: High level requirements are analyzed and elaborated by unit leaders and developers to identify detailed low level software requirements.	**11. Low level software requirements meeting:** High level requirements are analyzed and elaborated by unit leaders and developers to identify detailed low level software requirements. To capture requirements, use- case analysis is done by the help of a UML modeling tool. Minutes of Low Level Software Requirements Meeting is prepared.	**Use-case analysis:** To facilitate capturing functional requirements by clearly visualizing the interaction between the software and an external agent. **Minutes of Low Level Software Requirements Meeting:** To document the decisions and their rationales.

Table 4-1 Continued

AS-IS Activities	TO-BE Activities	Change & Rationale
Does not exist.	**12. Update SRS and RTM:** Developers store the identified low level requirements with a unique number in a database by the help of a requirements management tool. SRS and RTM are updated by the help of the requirements management tool and using SRS and RTM Templates.	**Update SRS&RTM:** To document and uniquely identify software requirements. To trace relationships between requirements; and also between requirements and software modules. **Requirements Management Tool:** To facilitate recording, tracing and managing requirements. **SRS&RTM Templates:** To facilitate preparing and managing documents by providing a common format.
Does not exist.	**13. Review & Update SRS and RTM:** Unit Leader reviews SRS and RTM in order to ensure that the requirements are correct, unambiguous, complete, consistent, traceable and verifiable. Unit Leader uses Requirements Review Guideline and the requirements management tool while reviewing and updating SRS and RTM.	**Review & Update SRS and RTM:** To ensure that the requirements are correct, unambiguous, complete, consistent, traceable and verifiable. **Requirements Review Guideline:** To guide in reviewing SRS and RTM in order to ensure that the requirements are correct, unambiguous, complete, consistent, traceable and verifiable.
10. Decide whether all low level requirements are clear enough to proceed with design or not	**14. Decide whether all low level requirements are clear enough to proceed with design or not**	No change
11. Hardware interface requirements meeting: Hardware interface requirements are determined in a meeting attended by hardware engineers, developers and the unit leader that is responsible for the software module interfacing hardware.	**15. Hardware interface requirements meeting:** Hardware interface requirements are determined in a meeting attended by hardware engineers, developers and the unit leader that is responsible for the software module interfacing hardware. Minutes of Hardware Interface Requirements Meeting is prepared.	**Minutes of Hardware Interface Requirements Meeting:** To document the decisions and their rationales.
Does not exist.	**16. Update SRS and RTM:** Identified hardware interface requirements are stored with a unique number in a database by the help of a requirements management tool. SRS and RTM are updated by the help of the requirements management tool and using SRS and RTM Templates.	**Update SRS&RTM:** To document and uniquely identify software requirements. To trace relationships between requirements; and also between requirements and software modules. **Requirements Management Tool:** To facilitate recording, tracing and managing requirements. **SRS&RTM Templates:** To facilitate preparing and managing documents by providing a common format.

Table 4-1 Continued

AS-IS Activities	TO-BE Activities	Change & Rationale
Does not exist.	**17. Review & Update SRS and RTM:** Unit Leader reviews SRS and RTM in order to ensure that the requirements are correct, unambiguous, complete, consistent, traceable and verifiable. Unit Leader uses Requirements Review Guideline and the requirements management tool while reviewing and updating SRS and RTM.	**Review & Update SRS and RTM:** To ensure that the requirements are correct, unambiguous, complete, consistent, traceable and verifiable. **Requirements Review Guideline:** To guide in reviewing SRS and RTM in order to ensure that the requirements are correct, unambiguous, complete, consistent, traceable and verifiable.
12. Decide whether all hardware interface requirements are clear enough to proceed with design or not	**18. Decide whether all hardware interface requirements are clear enough to proceed with design or not**	No change
13. Meeting to determine software modules to be re-used: Software modules to be re-used are determined in a meeting attended by the related unit leader and developers.	**19. Meeting to determine software modules to be re-used:** Software modules to be re-used are determined in a meeting attended by the related unit leader and developers. Minutes of Re-use Meeting is prepared.	**Minutes of Re-use Meeting:** To document the decisions and their rationales.
14. Decide whether it is necessary to make some modifications to the modules to be re-used	**20. Decide whether it is necessary to make some modifications to the modules to be re-used**	No change
15. Reverse engineer modules to be re-used: If unit leader thinks that it is necessary to make some modifications to modules to be re-used, these modules are reverse engineered from code to design and then from design to requirements, since there are no formal software requirements specification or software design description documents and code is the only formal source.	**21. Get the requirements of the modules to be re-used:** If unit leader thinks that it is necessary to make some modifications to the modules to be re-used, he gathers the requirements of the modules from the related RTM and SRS using the requirements management tool and prepares Requirements Re-use Report.	**Gather the requirements using the requirements management tool:** To facilitate gathering the requirements of the module to be re-used and the other requirements that depend on them. **Requirements Re-use Report:** To document the requirements related to the module to be re-used.

Table 4-1 Continued

AS-IS Activities	TO-BE Activities	Change & Rationale
Does not exist.	**22. Update SRS and RTM:** Requirements of the modules to be re-used are modified and stored with a unique number in a database by the help of a requirements management tool. SRS and RTM are updated by the help of the requirements management tool and using Requirements Re-use Report, SRS and RTM Templates.	**Update SRS&RTM:** To document and uniquely identify software requirements. To trace relationships between requirements; and also between requirements and software modules. **Requirements Management Tool:** To facilitate recording, tracing and managing requirements. **SRS&RTM Templates:** To facilitate preparing and managing documents by providing a common format.
Does not exist.	**23. Review & Update SRS and RTM:** Unit Leader reviews SRS and RTM in order to ensure that the requirements are correct, unambiguous, complete, consistent, traceable and verifiable. Unit Leader uses Requirements Review Guideline and the requirements management tool while reviewing and updating SRS and RTM.	**Review & Update SRS and RTM:** To ensure that the requirements are correct, unambiguous, complete, consistent, traceable and verifiable. **Requirements Review Guideline:** To guide in reviewing SRS and RTM in order to ensure that the requirements are correct, unambiguous, complete, consistent, traceable and verifiable.
16. Decide whether a proof-of-concept (throw-away) prototype exists or not	**24. Decide whether a proof-of-concept (throw-away) prototype exists or not**	No change
17. Update software requirements according to evaluation of the prototype	**25. Update SRS and RTM according to evaluation of the prototype:** If a proof-of-concept (throw-away) prototype exists, stored software requirements are updated according to customer's requests and evaluation of the prototype by the help of a requirements management tool. SRS and RTM are updated by the help of the requirements management tool and using Customer Prototype Evaluation Report, SRS and RTM Templates.	**Update SRS&RTM:** To document and uniquely identify software requirements. To trace relationships between requirements; and also between requirements and software modules. **Requirements Management Tool:** To facilitate recording, tracing and managing requirements. **SRS&RTM Templates:** To facilitate preparing and managing documents by providing a common format. **Customer Prototype Evaluation Report:** To have customer's feedback and requests about the prototype documented.
Does not exist.	**26. Review & Update SRS and RTM:** ESD Manager reviews SRS and RTM in order to ensure that the requirements are correct, unambiguous, complete, consistent, traceable and verifiable. ESD Manager uses Requirements Review Guideline and the requirements management tool while reviewing and updating SRS and RTM.	**Review & Update SRS and RTM:** To ensure that the requirements are correct, unambiguous, complete, consistent, traceable and verifiable. **Requirements Review Guideline:** To guide in reviewing SRS and RTM in order to ensure that the requirements are correct, unambiguous, complete, consistent, traceable and verifiable.

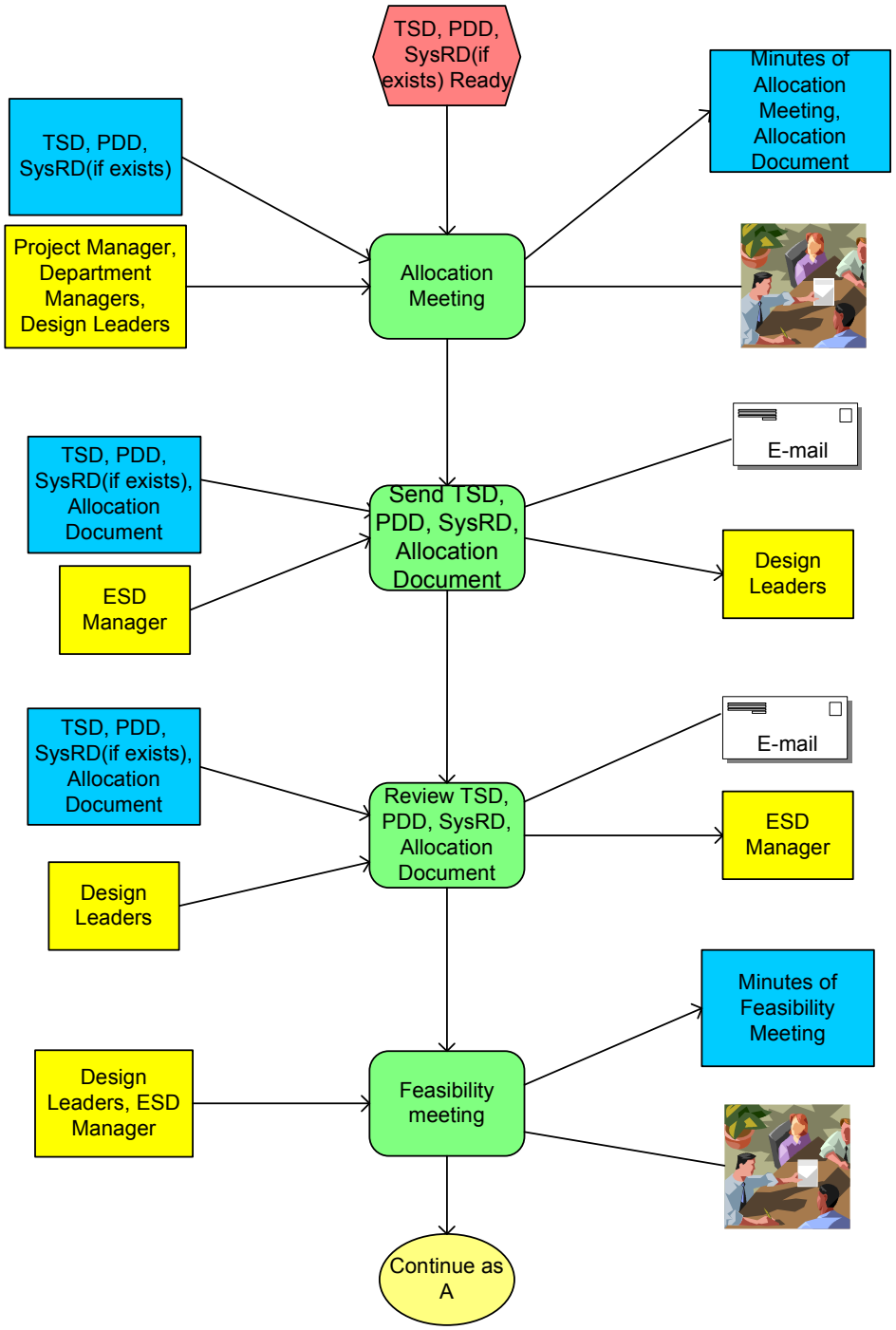

Figure 4-1 TO-BE Software Requirements Analysis Process Part 1

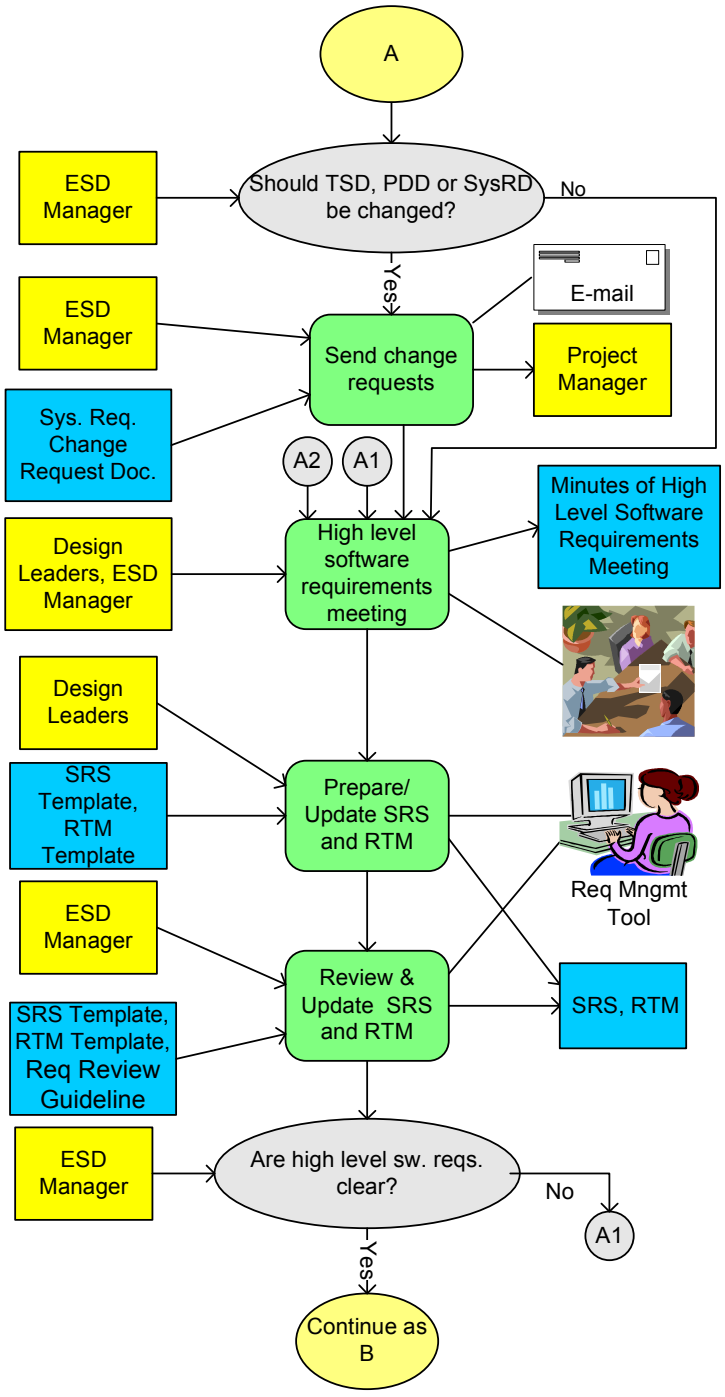

Figure 4-2 TO-BE Software Requirements Analysis Process Part 2

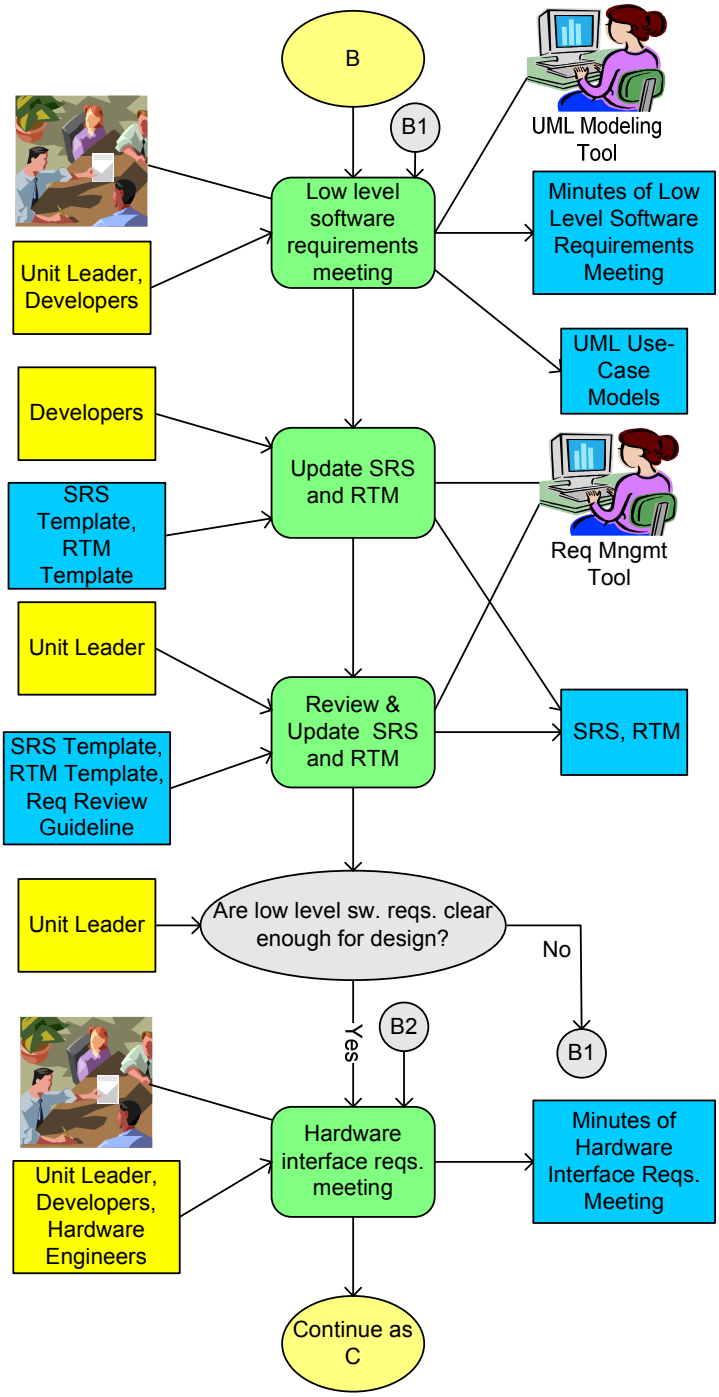

Figure 4-3 TO-BE Software Requirements Analysis Process Part 3

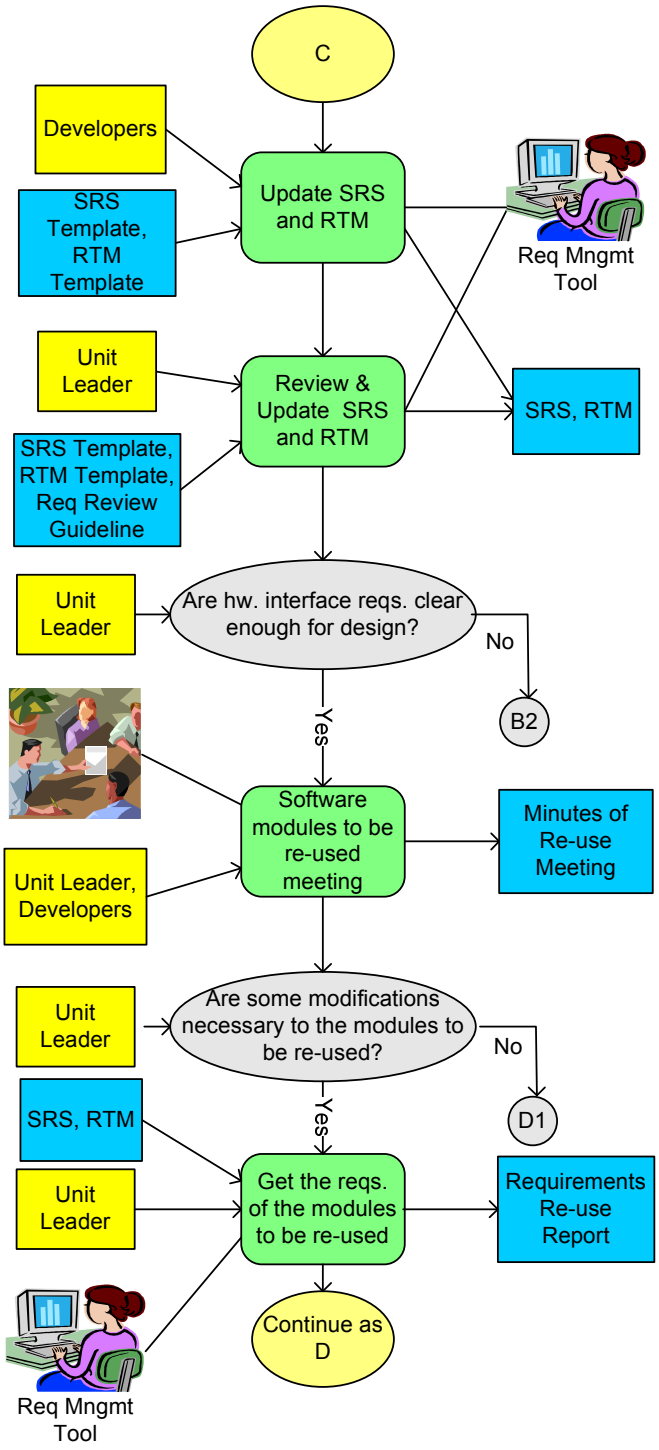

Figure 4-4 TO-BE Software Requirements Analysis Process Part 4

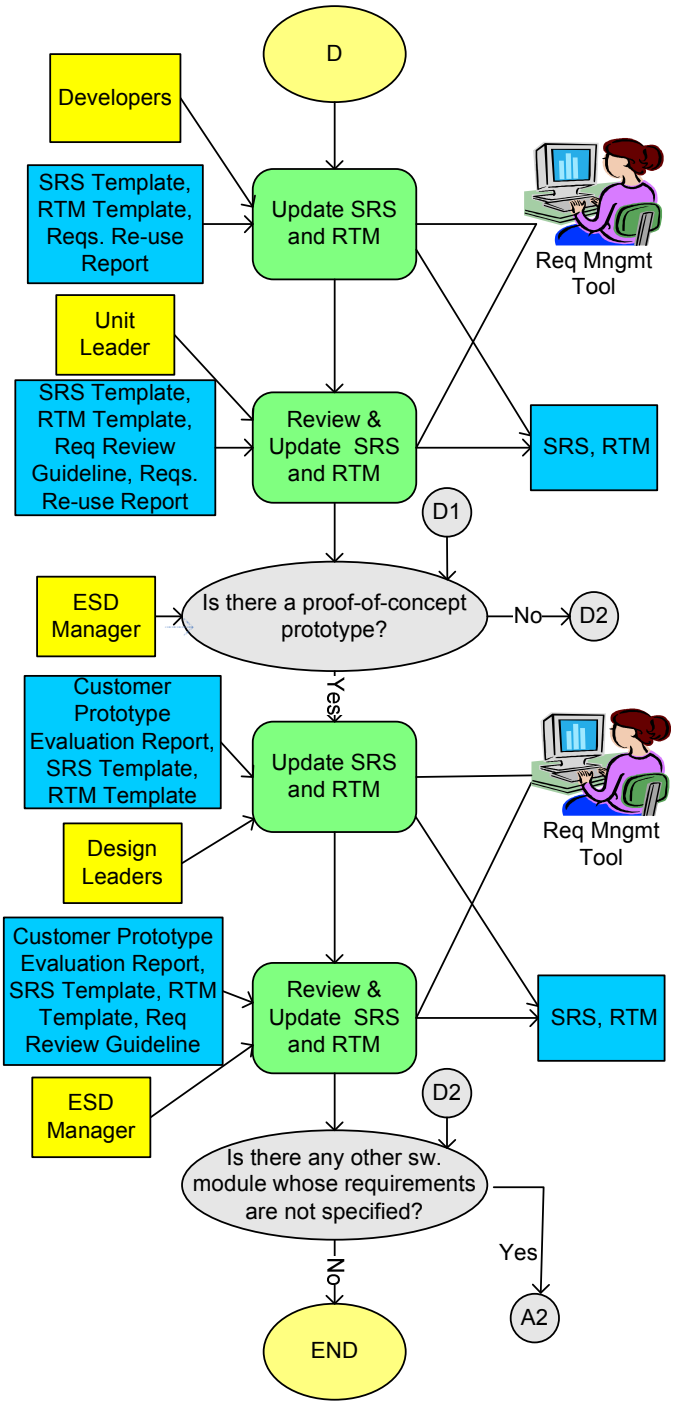

Figure 4-5 TO-BE Software Requirements Analysis Process Part 5

Table 4-2 Comparison of AS-IS and TO-BE Software Requirements Change Management Processes

AS-IS Activities	TO-BE Activities	Change & Rationale
1. Receive a requirement change request: ESD Manager receives a requirement change request. It may be written or oral, and sourced by system engineers or customer.	**1. Receive a requirement change request:** ESD Manager receives a requirement change request. It may be written or oral, and sourced by system engineers or customer.	No change
Does not exist	**2 Prepare Requirement Change Request Report:** A design leader assigned by ESD Manager prepares Requirement Change Request Report (RCRR), which contains the unique numbers of the requirements to be changed. The design leader also looks up the RTM using the requirements management tool to find out which requirements and which modules depend on the requirements to be changed, and adds the unique numbers of these requirements and modules to RCRR.	**Preparing Requirement Change Request Report:** To uniquely identify and document which requirements are requested to be changed. **Requirements Management Tool:** To facilitate recording, tracing and managing requirements.
2. Meeting on feasibility of the requirement change request: Feasibility of the requirement change request is analyzed in a meeting attended by ESD Manager and related design leaders.	**3. Meeting on feasibility of the requirement change request:** Feasibility of the requirement change request is analyzed in a meeting attended by ESD Manager and related design leaders using Requirement Change Request Report.	**Requirement Change Request Report:** To uniquely identify requirements that are requested to be changed. **Minutes of Feasibility Meeting:** To document the decisions and their rationales.
3. Decide whether the requirement change is acceptable, partially acceptable or not acceptable	**4. Decide whether the requirement change is acceptable, partially acceptable or not acceptable:**	No change
4. Accept the requirement change request: If ESD Manager and design leaders conclude that the requirement change request is feasible, ESD Manager accepts it by replying the source of change request via telephone or e-mail.	**5. Accept the requirement change request:** If If ESD Manager and design leaders conclude that the requirement change request is feasible, ESD Manager accepts it by replying the source of change request via telephone or e-mail. The decision of acceptance of the request is added to RCRR.	**Updating Requirement Change Request Report:** To document decisions and their rationales related to requirements that are requested to be changed.
5. Partially accept the requirement change request: If ESD Manager and design leaders conclude that the requirement change request is partially feasible, ESD Manager partially accepts it by replying the source of change request via telephone or e-mail.	**6. Partially accept the requirement change request:** If ESD Manager and design leaders conclude that the requirement change request is partially feasible, ESD Manager partially accepts it by replying the source of change request via telephone or e-mail. The decision of partial acceptance of the request is added to RCRR. Also, an explanation about the parts of the request that are accepted is added to RCRR.	**Updating Requirement Change Request Report:** To document decisions and their rationales related to requirements that are requested to be changed.
6. Reject the requirement change request: If ESD Manager and design leaders conclude that the requirement change request is infeasible, ESD Manager rejects it by replying the source of change request via telephone or e-mail.	**7. Partially accept the requirement change request:** If ESD Manager and design leaders conclude that the requirement change request is infeasible, ESD Manager rejects it by replying the source of change request via telephone or e-mail. The decision of rejection of the request is added to RCRR.	**Updating Requirement Change Request Report:** To document decisions and their rationales related to requirements that are requested to be changed.

Table 4-2 Continued

AS-IS Activities	TO-BE Activities	Change & Rationale
Does not exist.	**8. Update SRS and RTM:** A design leader assigned by ESD Manager updates stored software requirements according to Requirement Change Request Report by the help of a requirements management tool. SRS and RTM are updated according to the accepted change request using the requirements management tool and templates of SRS and RTM.	**Update SRS&RTM:** To document and uniquely identify software requirements. To trace relationships between requirements; and also between requirements and software modules. **Requirements Management Tool:** To facilitate recording, tracing and managing requirements. **SRS&RTM Templates:** To facilitate preparing and managing documents by providing a common format.
Does not exist.	**9. Review & Update SRS and RTM:** ESD Manager reviews SRS and RTM in order to ensure that the requirements are correct, unambiguous, complete, consistent, traceable and verifiable. ESD Manager uses Requirements Review Guideline and the requirements management tool while reviewing and updating SRS and RTM.	**Review & Update SRS and RTM:** To ensure that the requirements are correct, unambiguous, complete, consistent, traceable and verifiable. **Requirements Review Guideline:** To guide in reviewing SRS and RTM in order to ensure that the requirements are correct, unambiguous, complete, consistent, traceable and verifiable.

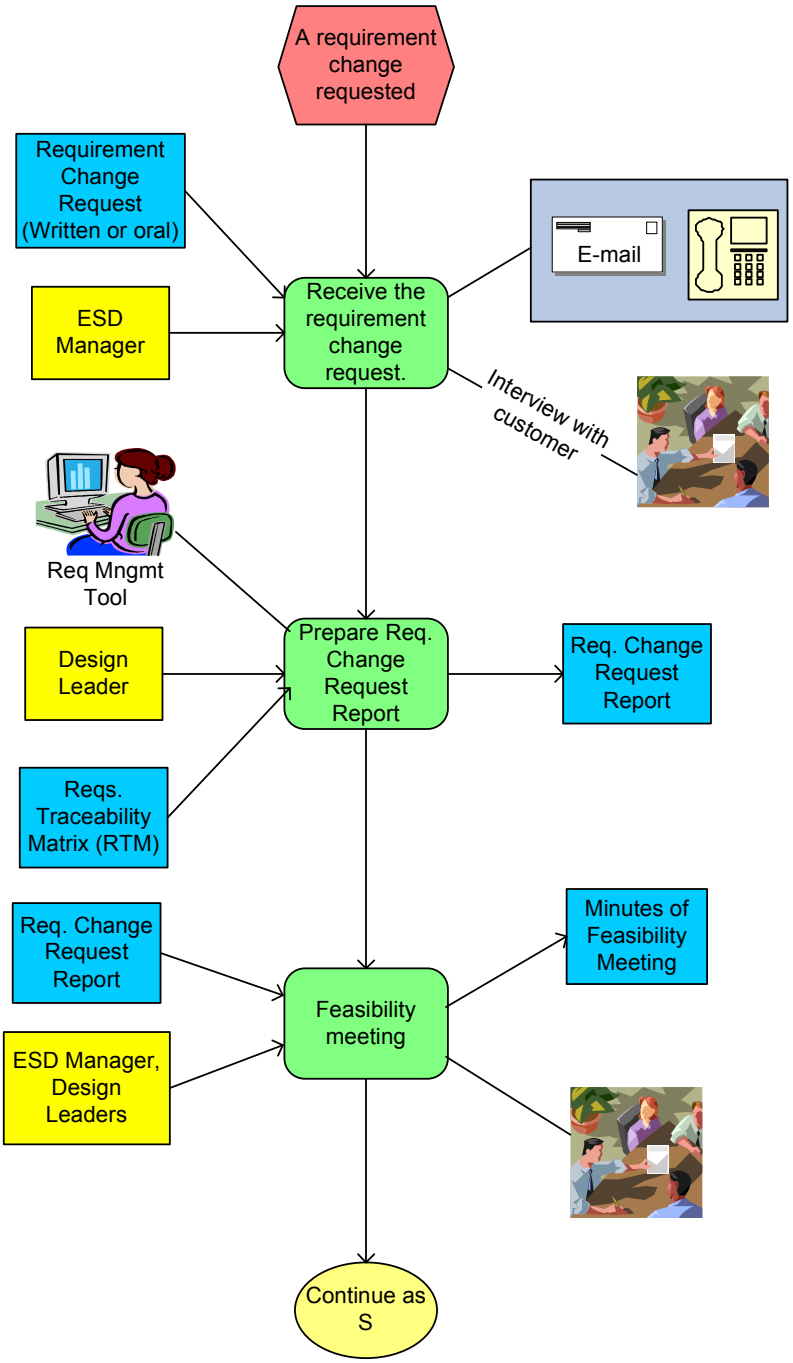

Figure 4-6 AS-IS Software Requirements Change Management Process Part 1

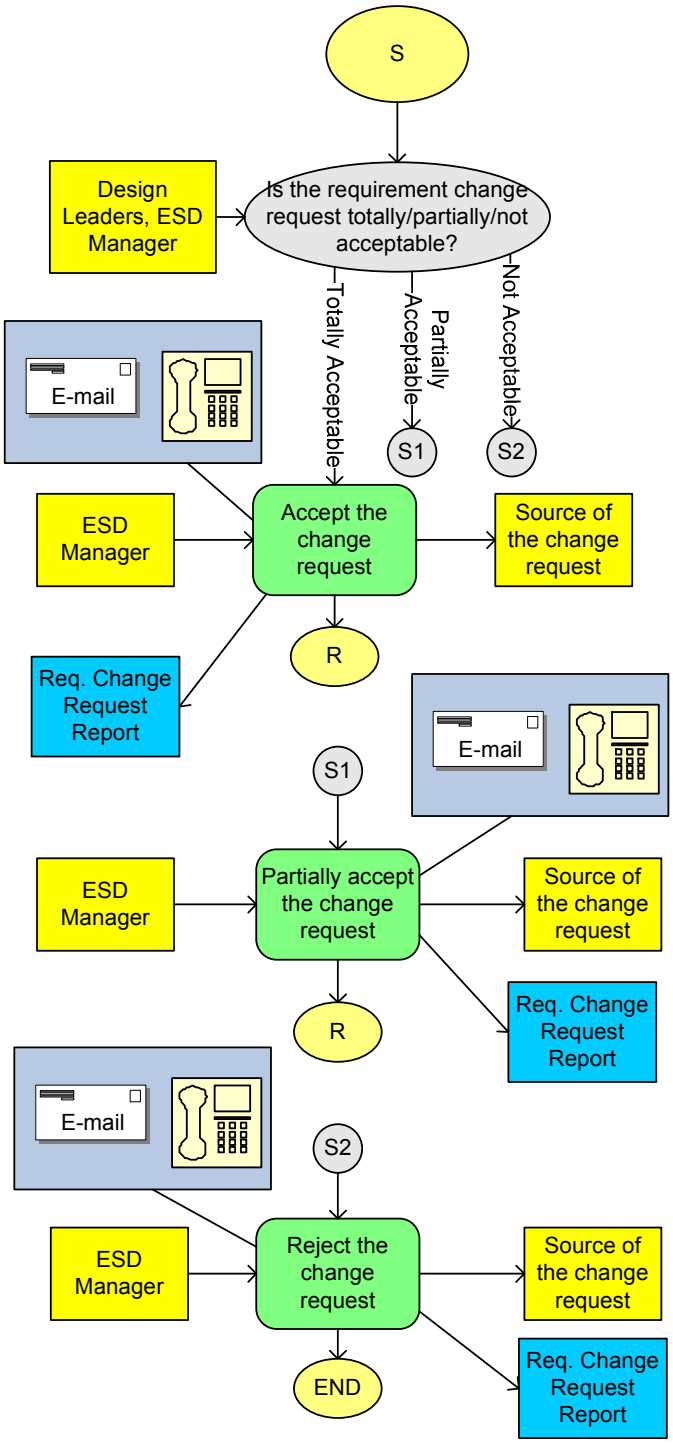

Figure 4-7 AS-IS Software Requirements Change Management Process Part 2

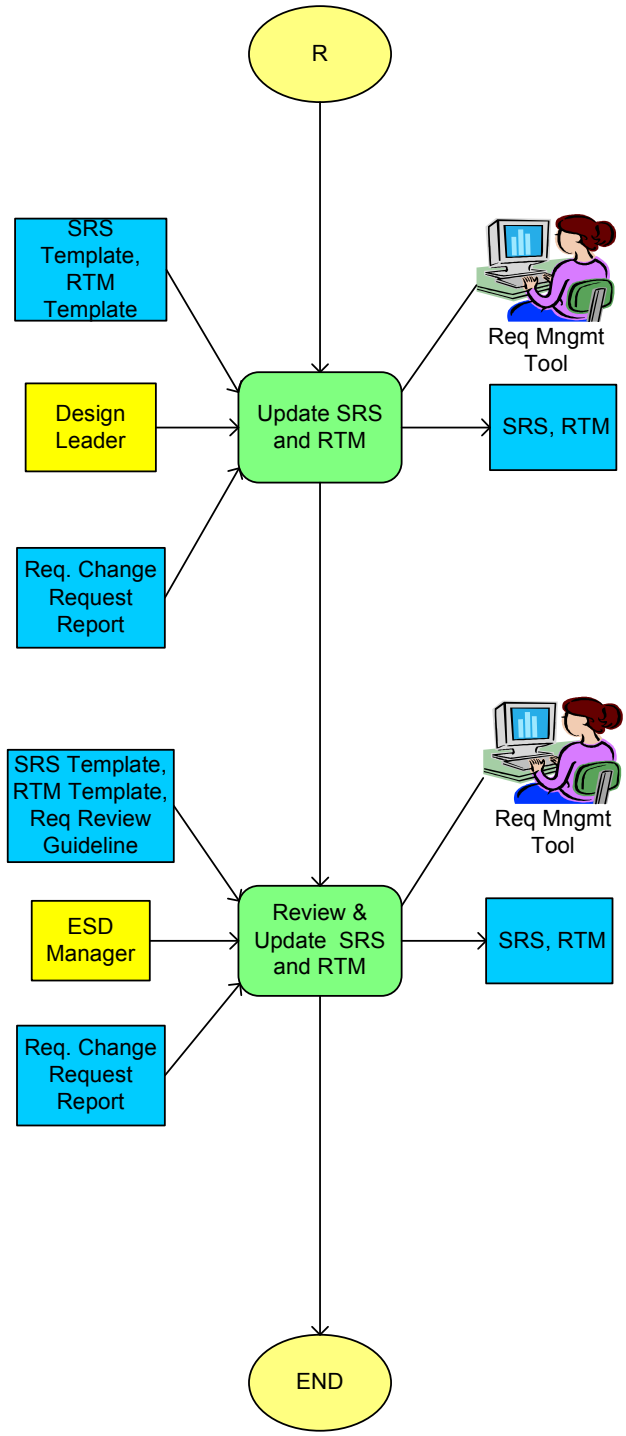

Figure 4-8 AS-IS Software Requirements Change Management Process Part 3

4.2 Measurement of TO-BE Software Requirements Management Processes

The quality of the TO-BE software requirements management processes are measured according to the method proposed in Güceğlioğlu's study [51]. Firstly, the process descriptions and the models of the improved (TO-BE) software requirements management processes, which are given in the previous section, are formed. Then measurements are performed on these process descriptions and graphical process models using the process quality metrics given in Güceğlioğlu's study.

The measurement details of TO-BE software requirements analysis process are given in Tables from C-1-1 to C-1-5, and the measurement details of TO-BE software requirements change management process are given in Tables from C-2-1 to C-2-5.

The summary of the results of the TO-BE process measurements is given in Table 4.3.

Table 4-3 TO-BE Measurement Results

Metrics	Software Requirements Analysis TO-BE Process (26 activities)	Software Requirements Change Management TO-BE Process (9 activities)
Complexity (X(1): Structured Decisions, X(2): Unstructured Decisions, X(3): Semi-strctured Decisions)	X(1) = 1 / 26 = 0.038 X(2) = 5 / 26 = 0.192 X(3) = 0 / 26 = 0	X(1) = 0 / 9 = 0 X(2) = 1 / 9 = 0.111 X(3) = 0 / 9 = 0
Coupling	X = 4 / 26 = 0.154	X = 4 / 9 = 0.444
Failure Avoidance	X = 11 / 26 = 0.423	X = 2 / 9 = 0.222
Restorability	X = 20 / 26 = 0.769	X = 8 / 9 = 0.889
Restoration Effectiveness (Number of total activities is used in the formula)	X = 20 / 26 = 0.769	X = 8 / 9 = 0.889
Functional Adequacy	X = 26 / 26 = 1	X = 9 / 9 = 1
Functional Completeness (Number of TO-BE activities is used in the formula)	X = 1 - 0 / 26 = 1	X = 1 - 0 / 9 = 1
IT Usage	X = 20 / 26 = 0. 769	X = 8 / 9 = 0.889
IT Density	X = 20 / 20 = 1	X = 8 / 8 = 1
Computational Accuracy	X = 5 / 5 = 1	X = 2 / 2 = 1
Data Exchangeability	X = 2 / 4 = 0.5	X = 3 / 4 = 0.75
Access Auditability	X = 20 / 20 = 1	X = 8 / 8 = 1
Functional Understandability	X = 17 / 26 = 0.654	X = 8 / 9 = 0.889
Existence in Documents	X = 26 / 26 = 1	X = 9 / 9 = 1
Input Validity Checking	X = 0 / 26 = 0	X = 1 / 9 = 0.111
Undoability	X = 20 / 26 = 0.769	X = 8 / 9 = 0.889
Attractive Interaction (Number of total activities is used in the formula)	X = 18 / 26 = 0.692	X = 7 / 9 = 0.778

4.3 Comparison of AS-IS and TO-BE Software Requirements Management Processes

In this section, the measurement results of AS-IS and TO-BE software requirements management processes are compared. Table 4-4 summarizes the comparison of the measurement results of AS-IS and TO-BE software requirements analysis (SRA) processes. Table 4-5 summarizes the comparison of the measurement results of AS-IS and TO-BE software requirements change management (SRCM) processes. The detailed discussion on values of metric calculations is given below with the short descriptions of metrics. The metric descriptions are based on the Güceğlioğlu's [51] study.

Complexity (0 < = X < = 1, The lower value of X (1), X (2), X (3), the better analyzability): Complexity is based on the ratio of activities including decision points. It has 3 types:

- Structured Decision (X(1)): Well-defined, programmable, repetitive decisions.
- Unstructured Decision (X(2)): Requires creativity, the situation is not clear and requires fuzzy logic.
- Semi-structured Decision (X(3)): May be repetitive and routine, but requires human intuition.

SRA: The values of X(1) and X(2) have been decreased in the improved process. This is caused by the increase in the number of activities in the TO-BE model. X(3) remains 0, since there are not any semi-structured decisions in AS-IS and TO-BE models.

SRCM: The value of X(2) has been decreased in the improved process. This is caused by the increase in the number of activities in the TO-BE model. X(1) and X(3) remain 0, since there are not any structured or semi-structured decisions in AS-IS and TO-BE models.

Coupling (0 < = X < = 1, The lower value of X, the better

analyzability): Coupling is based on the ratio of activities that include interactions with other processes.

SRA, SRCM: The value is decreased due to the increase in the number of activities.

Failure Avoidance (0 < = X < = 1, The higher value of X, the better failure avoidance): Failure avoidance is based on the ratio of activities that include reviews, checklists, templates etc.

SRA, SRCM : The value is increased, since the number of activities that include reviews, checklists and templates is increased.

Restorability (0 < = X < = 1, The higher value of X, the better restorability): Restorability is based on the ratio of activities that are recorded.

SRA, SRCM: The value is increased, since the number of activities that are recorded increased by documenting the requirements and minutes of meetings.

Restoration Effectiveness (0 < = X < = 1, The higher value of X, the better restorability effectiveness): Restoration Effectiveness is based on the ratio of activities that can be restored by using hard or soft back-up copies of documents.

SRA, SRCM: The value is increased, since the number of activities that are recorded in an IT based environment increased.

Functional Adequacy (0 < = X < = 1, The higher value of X, the better functional adequacy): Functional Adequacy is based on the ratio of activities that are adequate to the process descriptions in regulatory documents.

SRA: The value is 1 in both AS-IS and TO-BE models. Since the process descriptions are not very detailed in regulatory documents, the AS-IS activities are assumed to be implicit in on-paper process descriptions. All

TO-BE activities are adequate to the proposed process descriptions.

SRCM: Since SRCM process is not described in regulatory documents, the value can not be calculated for the AS-IS process. The value is 1 for the TO-BE process, since all TO-BE activities are adequate to the proposed process descriptions.

Functional Completeness (0 < = X < = 1, The higher value of X, the better functional completeness): Functional Completeness is based on the ratio of activities which are defined in regulatory documents but missed in practice.

SRA: The value is increased, since all proposed TO-BE activities are assumed to be put into practice.

SRCM: Since SRCM process is not described in regulatory documents, the value can not be calculated for the AS-IS process. The value is 1 for the TO-BE process, since all proposed TO-BE activities are assumed to be put into practice.

IT Usage (0 < = X < = 1, The higher value of X, the more IT usage): IT Usage is based on the ratio of activities in which IT applications are used.

SRA, SRCM: The value is increased, since requirements management tool is proposed to be used in many activities and documentation based on IT is increased.

IT Density (0 < = X < = 1, The higher value of X, the more IT density): IT Density is based on the ratio of documents in which IT applications are used to prepare, update and search documents.

SRA, SRCM: The value is not changed and it is 1 in both AS-IS and TO-BE processes. Although documentation is increased in the TO-BE process, the ratio of documents that are prepared, updated or searched with IT applications to all documents is 1 in both AS-IS and TO-BE processes. This is due to using IT applications whenever a document is prepared, updated or searched in both AS-IS and TO-BE processes.

Computational Accuracy (0 < = X < = 1, The higher value of X, the more accurate): Computational Accuracy is based on the ratio of activities in which accuracy requirements have been implemented as defined in the regulatory documents.

SRA: The value is increased, since reviews are included in the improved process.

SRCM: The value is increased, since reviews and preparing Requirement Change Request Report are included the improved process.

Data Exchangeability (0 < = X < = 1, The higher value of X, the more data exchangeability): Data Exchangeability is based on the ratio of activities in which no operation such as parsing or extracting is performed on the received data before using it.

SRA, SRCM: The value is the same since the data received from other processes remains the same.

Access Auditability (0 < = X < = 1, The higher value of X, the more auditable): Access Auditability is based on the ratio of activities in which there is access to data and the access can be audited.

SRA, SRCM: The value is not changed and it is 1 in both AS-IS and TO-BE processes. Although access to data is increased in the TO-BE process, the ratio of activities involving auditable access to data to all activities involving access to data is 1 in both AS-IS and TO-BE processes. This is due to using IT applications whenever data is accessed in both AS-IS and TO-BE processes.

Functional Understandability (0 < = X < = 1, The higher value of X, the more understandable): Functional Understandability is based on the ratio of activities that are easily understandable by the staff.

SRA, SRCM: The value is increased since documentation is increased in the improved process.

Existence in Documents (0 < = X < = 1, The higher value of X, the more complete documentation): Existence in Documents is based on the ratio of activities described in the available documents.

SRA, SRCM: The value is increased from 0 to 1 since the actual regulatory documents do not give details of the activities, but the activities of the TO-BE process is described in detail.

Input Validity Checking (0 < = X < = 1, The higher value of X, the better input validity checking): Input Validity Checking is based on the number of activities in which checking for valid data is provided for input parameters.

SRA: The value is the same and equal to 0 for AS-IS and TO-BE processes since no input validity checking takes place.

SRCM: The value is increased, since in TO-BE process, the requirements change request is checked to find out whether there exists a corresponding requirement or not.

Undoability (0 < = X < = 1, The higher value of X, the better better undoability): Undoability is based on the ratio of the recorded activities which can be undone after they are completed.

SRA, SRCM: The value is increased, since the number of activities recorded in an IT based environment is increased.

Attractive Interaction (0 < = X < = 1, The higher value of X, the more attractive interaction): Attractive Interaction is based on the ratio of activities which have attractive appearance and provide staff with easiness in preparation, deletion or updating documents.

SRA, SRCM: The value is increased, since the usage of templates and tools is increased.

Table 4-4 Comparison of AS-IS and TO-BE Software Requirements Analysis Process Measurement Results

Metrics	Software Requirements Analysis AS-IS Process (17 activities)	Software Requirements Analysis TO-BE Process (26 activities)
Complexity (X(1): Structured Decisions, X(2): Unstructured Decisions, X(3): Semi-strctured Decisions)	X(1) = 1 / 17 = 0.059 X(2) = 5 / 17 = 0.294 X(3) = 0 / 17 = 0	X(1) = 1 / 26 = 0.038 X(2) = 5 / 26 = 0.192 X(3) = 0 / 26 = 0
Coupling	X = 4 / 17 = 0.235	X = 4 / 26 = 0.154
Failure Avoidance	X = 1 / 17 = 0.059	X = 11 / 26 = 0.423
Restorability	X = 4 / 17 = 0.235	X = 20 / 26 = 0.769
Restoration Effectiveness (Number of total activities is used in the formula)	X = 4 / 17 = 0.235	X = 20 / 26 = 0.769
Functional Adequacy	X = 17 / 17 = 1	X = 26 / 26 = 1
Functional Completeness (Number of TO-BE activities is used in formula)	X = 1 - 5 / 8 = 0.375 (A1, A2, A3 are done, but A4, A5, R1, R2 and R3 are not done.)	X = 1 - 0 / 26 = 1
IT Usage	X = 4 / 17 = 0.235	X = 20 / 26 = 0. 769
IT Density	X = 4 / 4 = 1	X = 20 / 20 = 1
Computational Accuracy	X = 0 / 5 = 0	X = 5 / 5 = 1
Data Exchangeability	X = 2 / 4 = 0.5	X = 2 / 4 = 0.5
Access Auditability	X = 4 / 4 = 1	X = 20 / 20 = 1
Functional Understandability	X = 3 / 17 = 0.176	X = 17 / 26 = 0.654
Existence in Documents	X = 0 / 17 = 0	X = 26 / 26 = 1
Input Validity Checking	X = 0 / 17 = 0	X = 0 / 26 = 0
Undoability	X = 4 / 17 = 0.235	X = 20 / 26 = 0.769
Attractive Interaction (Number of total activities is used in formula)	X = 0 / 17 = 0	X = 18 / 26 = 0.692

Table 4-5 Comparison of AS-IS and TO-BE Software Requirements Change Management Process Measurement Results

Metrics	Software Requirements Change Management AS-IS Process (6 activities)	Software Requirements Change Management TO-BE Process (9 activities)
Complexity (X(1): Structured Decisions, X(2): Unstructured Decisions, X(3): Semi-strctured Decisions)	X(1) = 0 / 6 = 0 X(2) = 1 / 6 = 0.167 X(3) = 0 / 6 = 0	X(1) = 0 / 9 = 0 X(2) = 1 / 9 = 0.111 X(3) = 0 / 9 = 0
Coupling	X = 4 / 6 = 0.667	X = 4 / 9 = 0.444
Failure Avoidance	X = 0 / 6 = 0	X = 2 / 9 = 0.222
Restorability	X = 4 / 6 = 0.667	X = 8 / 9 = 0.889
Restoration Effectiveness (Number of total activities is used in the formula)	X = 4 / 6 = 0.667	X = 8 / 9 = 0.889
Functional Adequacy	Not applicable (On-paper process does not exist)	X = 9 / 9 = 1
Functional Completeness (Number of TO-BE activities is used in formula)	Not applicable (On-paper process does not exist)	X = 1 - 0 / 9 = 1
IT Usage	X = 4 / 6 = 0.667	X = 8 / 9 = 0.889
IT Density	X = 4 / 4 = 1	X = 8 / 8 = 1
Computational Accuracy	X = 0 / 2 = 0	X = 2 / 2 = 1
Data Exchangeability	X = 3 / 4 = 0.75	X = 3 / 4 = 0.75
Access Auditability	X = 4 / 4 = 1	X = 8 / 8 = 1
Functional Understandability	X = 4 / 6 = 0.667	X = 8 / 9 = 0.889
Existence in Documents	X = 0 / 6 = 0	X = 9 / 9 = 1
Input Validity Checking	X = 0 / 6 = 0	X = 1 / 9 = 0.111
Undoability	X = 4 / 6 = 0.667	X = 8 / 9 = 0.889
Attractive Interaction (Number of total activities is used in formula)	X = 0 / 6 = 0	X = 7 / 9 = 0.778

CHAPTER 5

EVALUATION AND CONCLUSION

5.1 Evaluation

In this thesis, improving the software requirements management processes of Embedded Software Department (ESD) of Company A is studied. Güceğlioğlu's method [51] is decided to be used to evaluate and compare the qualities of actual (AS-IS) and improved (TO-BE) processes. This method is expected to speed-up process improvement studies, since it allows to quantitatively compare quality attributes of a proposed software process model to actual (AS-IS) process model before putting the proposed model into practice.

In order to evaluate the quality attributes of the current (AS-IS) software requirements analysis and change management processes, the static process descriptions and the models of the current processes were constructed. This was a very exhaustive work, since the requirements management activities in ESD were ad-hoc and not carried out in a pre-determined way. For example, although most of the time, minutes of meetings are not prepared, they are prepared in some of the meetings. Similarly, System Requirements Document exists in some projects, and does not exist in many others. There are many examples like that, and there is not a specific reason behind doing or not doing an activity in many cases.

Güceğlioğlu's model does not specify how to model the processes that are not always operated in the same way. In this thesis, the activities or documents that take place most of the time, are accepted as parts of the AS-IS process. The activities that are operated differently in different times make it difficult to calculate the metric values, since most of the metric values are defined as the ratio of activities done in a specific way to number of all activities.

The other difficulty experienced in this study related to metric calculations is not having a detailed definition of the processes in regulatory documents. The on-paper processes were not detailed and therefore it was difficult to calculate the values of the metrics that depend on on-paper processes. For example Functional Adequacy metric is defined as the ratio of activities that are adequate to the regulatory documents to total number of activities. If the on-paper processes are not detailed, and contain only a few sentences on how to accomplish a task, how can this metric be calculated? Since the details of the activities are not given, most of the AS-IS activities that have the purpose of accomplishing the tasks written in the on-paper process descriptions can be assessed as adequate.

Güceğlioğlu's model assumes that regulatory documents of the organization define the processes in detail by giving information on activity flow, staff and documents. The model does not tell how to calculate the metric values when the organization does not have a defined process.

Generally the organizations that have a defined process are more mature than the organizations that do not have a defined process. Güceğlioğlu's model will be more easily applied by immature organizations, if it is updated to include guidance on applying the model to organizations that do not have defined processes. Also if the method is updated to include

guidance on applying the model to organizations that have ad-hoc, indeterminate AS-IS processes, it will be easier for them to apply the model.

Some metric definitions in the model may also be updated. Data Exchangeability is defined as the ratio of the number of activities in which no change is performed on the received data before using it to the number of activities which have interactions with other processes. This definition assumes that if there is an interaction with a process, data is received from the interacting process. However, this may not be case, and data may be sent to interacting processes without receiving any data. So the definition of Data Exchangeability metric should be updated as follows:

> **Data Exchangeability:** X = A / B
> A: The number of activities in which no change is performed on the received data before using it
> B: The number of activities in which data is received from other interacting processes

Restorability metric is defined as the ratio of the number of activities which are recorded to total number of activities. The quality of recording is not measured with this definition. For example, if the output data of an activity is sent via e-mail to a related person, this activity can be assessed as recorded, since e-mails are saved in a backed-up environment. However, when it is needed to access the recorded data, it will be hard to search and find the required data. Therefore, the author thinks that a new metric should be added to the model that calculates the quality of documents. Documents with unique numbers or identifiers, documents that are under version control and documents that have templates are easier to access and update, and therefore are of high quality.

IT-Density metric is defined as the ratio of the number of documents in which IT is used in preparing, updating, or searching the documents to total number of documents. The quality of the IT usage is not measured with this definition. For example, if the output data of an activity is sent via e-mail to a related person, this activity can be assessed as using IT in preparing the output data. However this type of IT usage does not facilitate preparing the data so much. Therefore, the author thinks that a new metric should be added to the model that calculates the quality of IT usage. Commercial or in-house IT applications that are specifically constructed for process activities facilitate the process a lot, and therefore are of high quality.

Another suggestion is on the name of Computational Accuracy Metric. The source of this metric is ISO/IEC 9126 Software Product Quality Model. Software products may have "computational" accuracy requirements. However, the definition of this metric in Güceğlioğlu's model is based on the activities that have any kind of accuracy requirements, not only computational. Therefore the name is misleading, and should be corrected as Accuracy Metric.

Even though there are some difficulties in modeling the ad-hoc AS-IS activities and calculating the values of metrics that depend on on-paper processes, which are not detailed, Güceğlioğlu's model is clear and easy-to-implement. It helps to gain an insight into the effects of applying a proposed process model before putting it into practice.

The comparison of the calculations of AS-IS and TO-BE processes revealed that the quality attribute values of the improved processes are better than the current processes. TO-BE processes are more reliable, functional and usable than AS-IS processes according to the results of the measurements. Increased reliability of requirements management processes will facilitate finding and fixing defects in the process which, in

turn, helps to develop software products with less defects. Increased functionality of requirements management processes will reduce the ad-hoc characteristics of the current processes. And finally, increased usability will help to easily operate the processes.

The improved process models are evaluated in interviews with several design leaders and senior design leaders. They think that the improved processes are applicable to ESD and can address the problems related to requirements management processes of ESD. According to them, documenting requirements, using a requirements management tool and tracing the relationships by using a Requirements Traceability Matrix are the most important improvements.

The model helps to compare the quality attributes of the AS-IS and TO-BE processes, however it does not take into account the side effects of the improvements. For example it can be said that approximately no documentation takes place in AS-IS processes. On the other hand, a lot of documentation takes place in TO-BE processes. This increase in documentation may increase the time that is spent for requirements processes. And that result may found to be unacceptable by the Project Manager when the improved process is put into practice.

Therefore, although Güceğlioğlu's model helps to measure the quality attributes of an improved process, it can not be used alone to decide whether an improved process is feasible or not. However it can be used in feasibility studies to guide in whether the quality attributes of a proposed process are better than the present process applied.

5.2 Conclusion

This study aimed to improve the software requirements analysis and software requirements change management processes of Embedded

Software Department (ESD) of Company A. Since the duration of this thesis study is not long enough to put the improved processes into practice and then evaluate the results, a pre-enactment method that is recently proposed by Güceğlioğlu [51] for measuring process quality is used to compare the quality of the improved processes to the current ones.

In order to measure the quality of the current processes as actually applied in ESD, the static descriptions and models of the current (AS-IS) processes are constructed. The regulatory document that describes the processes of the organization is inspected. Then static process descriptions, AS-IS models and on-paper description of the processes written in the regulatory document are used in calculating the metrics of quality attributes defined in Güceğlioğlu's model.

The problems of the current process and solution suggestions addressing the problems are discussed by the author and several design leaders. These problems and solution suggestions are then reviewed by a senior design leader. Based on these solutions, improved (TO-BE) software requirements analysis and software requirements change management processes are constructed. The static process descriptions and models of the improved processes are formed. These descriptions and models are then used to calculate the quality attribute metrics of the improved processes.

The comparison of the calculations of AS-IS and TO-BE processes revealed that the quality attribute values of the improved processes are better than the current processes. TO-BE processes are more reliable, functional and usable than AS-IS processes according to the results of the measurements.

The improved process models are evaluated in interviews with several

design leaders and senior design leaders. They think that the improved processes are applicable to ESD and can address the problems related to requirements management processes of ESD.

While conducting this study, two main difficulties are faced. The first one is that the AS-IS software requirements management processes of ESD are not operated in the same way in all times. The processes are ad-hoc and undeterministic. This situation made it difficult to model the AS-IS processes and calculate the metric values associated to them.

The second difficulty faced is not having detailed on-paper process descriptions in regulatory documents. The descriptions are very rough and this situation made it difficult to calculate the metric values that depend on on-paper processes.

As a future work, this method can be applied to other software development processes of ESD. Also, applying this model to processes in ESD that have detailed on-paper process descriptions and determinate AS-IS processes will yield interesting results to compare with this study.

If another study is conducted on improving the requirements management processes of ESD, it will also be interesting to calculate its quality attributes and compare them with the ones in this study.

REFERENCES

1. J. Horwitz, “Human Capital & Software Sector Growth: A First Look at the Role of Labor Markets in Low and Middle Income Countries”, Carnegie Mellon University, April 21, 2004
2. M. Newman, “Software Errors Cost U.S. Economy $59.5 Billion Annually”, National Institute of Standards and Technology (NIST), http://www.nist.gov/public_affairs/releases/n02-10.htm, June 28, 2002, Last date accessed: August 10, 2007
3. The Standish Group, "The Standish Group Report-Chaos", 1994
4. The Standish Group, “Chaos Chronicles Version 3.0”, 2003
5. F. P. Brooks, “No Silver Bullet: Essence and Accidents of Software Engineering”, IEEE Computer, 10-19, April1987
6. M. C. Paulk, B. Curtis, M. B. Chrissis, C. V. Weber, “Capability Maturity Model, Version 1.1”, IEEE Software, July 1993
7. B. Boehm, “A View of 20th and 21st Century Software Engineering”, ACM, ICSE’06, May 20–28, 2006
8. R. S. Pressman, “Software Engineering-A Practitioner’s Approach”, Fifth Ed., McGraw-Hill , 2001
9. P. Kruchten, “Putting the “Engineering” into “Software Engineering””, Proceedings of the 2004 Australian Software Engineering Conference (ASWEC’04), IEEE 2004
10. B. W. Boehm, "A Spiral Model of Software Development and Enhancement," IEEE Computer, Vol. 21, 1988, pp. 61-72
11. P. Kruchten, “The Rational Unified Process: An Introduction”, 3 Ed., Addison-Wesley, Boston, 2004.
12. M. C. Paulk, B. Curtis, M. B. Chrissis, C. V. Weber, “Capability Maturity Model for Software, Version 1.1”, Software Engineering

Institute, CMU/SEI-93-TR-24, February 1993.

13. SEI, “Capability Maturity Model Integration (CMMI), Version 1.1 CMMI for Software Engineering (CMMI-SW, V1.1) Continuous Representation”, CMU/SEI-2002-TR-028, August 2002
14. SEI, “Capability Maturity Model Integration (CMMI), Version 1.1 CMMI for Software Engineering (CMMI-SW, V1.1) Staged Representation”, CMU/SEI-2002-TR-029, August 2002
15. B. McFeeley, “IDEAL: A User’s Guide for Software Process Improvement”, SEI, CMU/SEI-96-HB-001, February 1996
16. CMMI Product Team, “CMMI for Development, Version 1.2”, SEI, CMU/SEI-2006-TR-008, ESC-TR-2006-008, August 2006
17. S. Boycan, J. Mattingly, M. Lewis, “Air Force Software Process Improvement”, STSC CrossTalk, February 1995
18. “SEI Process Maturity Profile CMMI v1.1 SCAMPI v1.1 Class A Appraisal Results 2006 End-Year Update”, SEI, March 2007
19. B. Pierce, “Is CMMI Ready for Prime Time”, STSC CrossTalk, July 2000
20. L. Heinz, “CMMI Myths and Realities”, STSC CrossTalk, June 2004
21. ISO/IEC IS 15504 – 2: Software Engineering – Process Assessment – Part 2: Performing an Assessment, International Organization for Standardization & International Electrotechnical Commission, 2003.
22. SEI, “ISO/IEC IS 15504 “, http://www.sei.cmu.edu/cmmi/faq/15504-faq.html, Last date accessed: August 13, 2007
23. ISO 9001 Quality systems – Model for quality assurance in design/development, production, installation, and servicing, First edition, International Organization for Standardization, March 1987
24. ISO 9000-3. Quality management and quality assurance standards – Part 3: Guidelines for the application of ISO 9001 to the development, supply, and maintenance of software, First edition, International Organization for Standardization, June 1991
25. S. Godfrey, J. Andary, L. Rosenberg, “Using Pilots to Assess the

Value and Approach of CMMI Implementation", Goddard Space Flight Center, SEPG 2003 2/03.

26. P. Kuvaja, J. Simila, L. Krzanik, A. Bicego, G. Saukkonen, "Software Process Assessment and Improvement", The BOOTSTRAP Approach. Blackwell Publishers, 1994
27. J. Siviy, M. L. Penn, E. Harper, "Relationships Between CMMI and Six Sigma", Technical Note CMU/SEI-2005-TN-005, December 2005
28. C. Gresse, A. Anacleto, C. F. Salviano, "Helping Small Companies Assess Software Processes", IEEE Software January/February 2006
29. E. Demirörs, O. Demirörs, O. Dikenelli, B. Keskin, "Process Improvement Towards ISO 9001 Certification in a Small Software Organization", 20th International Conference on Software Engineering (ICSE'98), 1998
30. M. Diaz, J. Sligo, "How Software Process Improvement Helped Motorola", IEEE Software Vol 14 No 5, September/October 1997
31. B. C. Hardgrave, D. J. Armstrong, "Software Process Improvement: It's a Journey, Not a Destination", Communications of the ACM Vol 48 No 11, November 2005
32. T. Hall, A. Rainer, N. Baddoo, "Implementing Software Process Improvement: An Empirical Study", Software Process Improvement and Practice 2002 – 7
33. R. H. Thayer, M. Dorfman, "Software Requirements Engineering" Second Edition, IEEE Computer Society Press, 1997
34. G. Kotonya, I. Sommerville, "Requirements Engineering: Processes and Techniques", John Wiley & Sons Ltd., 1998
35. D. Leffingwell, D. Widrig, "Managing Software Requirements: A Unified Approach", Addison Wesley, 1999
36. C. Jones, "Revitalizing Software Project Management", American Programmer 6, 7, June 1994
37. C. J. Neill, P. A. Laplante, "Requirements Engineering: The State

of the Practice", IEEE Software November/December 2003

38. A. M. Davis, A. M. Hickey, "Requirements Researchers: Do We Practice What We Preach?", Requirements Engineering Journal 7:107–111, 2002
39. IEEE Recommended Practice for Software Requirements Specifications, IEEE Std 830-1998 (Revision of IEEE Std 830-1993), IEEE, New York,1998
40. D. Damian, D. Zowghi, L. Vaidyanathasamy, Y. Pal, "An Industrial Case Study of Immediate Benefits of Requirements Engineering Process Improvement at the Australian Center for Unisys Software", Empirical Software Engineering 9, 45–75, 2004
41. D. Damian, J. Chisan, L. Vaidyanathasamy, Y. Pal, "An Industrial Case Study of the Impact of Requirements Engineering on Downstream Development", Proceedings of the 2003 International Symposium on Empirical Software Engineering (ISESE'03), IEEE 2003
42. A. M. Davis, "Software Requirements (Revised): Objects, Functions, and States", Prentice Hall, New Jersey, 1993
43. I. Sommerville, J. Ransom, "An Empirical Study of Industrial Requirements Engineering Process Assessment and Improvement", ACM Transactions on Software Engineering and Methodology, Vol. 14, No. 1, January 2005, Pages 85–117
44. I. Sommerville, P. Sawyer, "Requirements Engineering – A Good Practice Guide", John Wiley, 1997
45. T. Gorschek, M. Svahnberg, K. Tejle, "Introduction and Application of a Lightweight Requirements Engineering Process Evaluation Method", Proc. Requirements Engineering Foundations for Software Quality '03 (REFSQ'03), Klagenfurt/Velden, Austria, 2003
46. A. M. Davis, D. Zowghi, "Good requirements practices are neither necessary nor sufficient", Requirements Eng. 11: 1–3, 2006
47. ISO/IEC, "ISO/IEC 9126-1 Software engineering - Product quality - Part 1: Quality model", 2001

48. ISO/IEC, "ISO/IEC 9126-2 Software engineering - Product quality - Part2: External metrics", 2002
49. ISO/IEC, "ISO/IEC 9126-3 Software engineering - Product quality - Part3: Internal metrics", 2002
50. ISO/IEC, "ISO/IEC 9126-4 Software engineering - Product quality - Part4: Quality In Use metrics", 2002
51. S. Güceğlioğlu, “A Pre-Enactment Model for Measuring Process Quality”, Ph.D. Thesis, Middle East Technical University, Department of Information Systems, 2006
52. B. Sezer, “Software Engineering Process Improvement”, M.Sc. Thesis, Middle East Technical University, Department of Electrical and Electronics Engineering, 2007
53. H. Seçkin, “Software Process Improvement Based On Static Process Evaluation”, M.Sc. Thesis, Middle East Technical University, Department of Electrical and Electronics Engineering, 2006
54. The International Council on Systems Engineering (INCOSE), “INCOSE Requirements Management Tools Survey”, http://www.paper-review.com/tools/rms/read.php, Last date accessed: November 1, 2007

APPENDIX A

AS-IS PROCESS MEASUREMENT DETAILS

A.1 AS-IS Software Requirements Analysis Process Measurement Details

Table A-1-1 AS-IS Analysis Process Metrics 1-3

Activity Number	Complexity (1)	Coupling (2)	Failure Avoidance (3)
1	No decision	There is an interaction with the software project management and systems engineering processes. TSD, PDD and SysRD (if exists) should have been prepared before the allocation meeting. Project manager as well as managers and design leaders from all related departments attend the allocation meeting.	No review, inspection, checkpoint or similar techniques
2, 4, 7, 9, 13, 15	No decision	No interaction	No review, inspection, checkpoint or similar techniques
3	No decision	No interaction	Design leaders in ESD review TSD, PDD and SysRD (if exists).
5	Unstructured Decision: Deciding whether TSD, PDD or SysRD should be changed or not is a complex task.	No interaction	No review, inspection, checkpoint or similar techniques
6	No decision	There is an interaction with the software project management process. If ESD Manager thinks that TSD, PDD or SysRD should be changed, he sends change requests to Project Manager.	No review, inspection, checkpoint or similar techniques
8	Unstructured Decision: Deciding whether all high level software requirements are clear enough to proceed with determining low level software requirements or not is a complex task.	No interaction	No review, inspection, checkpoint or similar techniques

Table A-1-1 Continued

Activity Number	Complexity (1)	Coupling (2)	Failure Avoidance (3)
10	Unstructured Decision: Deciding whether all low level requirements are clear enough to proceed with design or not is a complex task.	No interaction	No review, inspection, checkpoint or similar techniques
11	No decision	There is an interaction with the hardware engineering process. Hardware engineers attend the interface requirements meeting.	No review, inspection, checkpoint or similar techniques
12	Unstructured Decision: Deciding whether all hardware interface requirements interfacing hardware are clear enough to proceed with design or not is a complex task.	No interaction	No review, inspection, checkpoint or similar techniques
14	Unstructured Decision: Deciding whether it is necessary to make some modifications to the modules to be re-used is a complex task.	No interaction	No review, inspection, checkpoint or similar techniques
16	Structured Decision: Deciding whether a proof-of-concept (throw-away) prototype exists or not is a very simple decision. It simply exists or not.	No interaction	No review, inspection, checkpoint or similar techniques
17	No decision	There is an interaction with the customer's evaluation of the proof-of-concept prototype process. Customer's requests resulted from the evaluation of the prototype are used to update software requirements.	No review, inspection, checkpoint or similar techniques

Table A-1-2 AS-IS Analysis Process Metrics 4-5

Activity Number	Restorability (4)	Restoration Effectiveness (5)
1, 4, 7, 11, 13	Not recorded: There is not a formal document prepared after the meeting, also minutes of meeting does not generally exist.	No restoration
2, 3, 6	Recorded: E-mail is stored in the network.	Restorable
5, 8, 10, 12, 14, 15, 16, 17	Not recorded	No restoration
9	Recorded: There is not a formal requirements document prepared after the meeting, but minutes of meeting is generally prepared and stored as an archive file in the network.	Restorable

Table A-1-3 AS-IS Analysis Process Metrics 6-9

Activity Number	Functional Adequacy (6)	Functional Completeness (7)	IT Usage (8)	IT Density (9)
1, 4, 5, 7, 8, 10, 11, 12, 13, 14, 15, 16, 17	Adequate: On-paper process descriptions are not very detailed, but this activity is implicit in on-paper process descriptions.	This metric is measured using on-paper process definitions.	No IT usage	Generally no formal documents are prepared.
2, 3, 6	Adequate: On-paper process descriptions are not very detailed, but this activity is implicit in on-paper process descriptions.	This metric is measured using on-paper process definitions.	IT usage in sending e-mail.	E-mail is sent with Microsoft Outlook.
9	Adequate: On-paper process descriptions are not very detailed, but this activity is implicit in on-paper process descriptions.	This metric is measured using on-paper process definitions.	IT usage in preparing minutes of meeting and storing it.	Minutes of meeting is prepared with Microsoft Word and stored with PVCS.

Table A-1-4 AS-IS Analysis Process Metrics 10-13

Activity Number	Computational Accuracy (10)	Data Exchangeability (11)	Access Auditability (12)	Functional Understandability (13)
1	No accuracy requirements defined in the regulatory documents	No change is performed on TSD, PDD and SysRD (if exists) before using them.	No access to data	Difficulties and misunderstandings in allocating system requirements to software requirements, since SysRD does not always exist and PDD is used instead
2	No accuracy requirements defined in the regulatory documents	No interaction	Auditable: E-mail	No difficulties or misunderstandings
3	No accuracy requirements defined in the regulatory documents	No interaction	Auditable: E-mail	Difficulties and misunderstandings in evaluating the feasibility of system requirements allocated to software, since there is not a document describing which system requirements are allocated to software.
4	No accuracy requirements defined in the regulatory documents	No interaction	No access to data: Minutes of meeting does not generally exist.	Difficulties and misunderstandings in evaluating the feasibility of system requirements allocated to software, since there is not a document describing which system requirements are allocated to software.
5	No accuracy requirements defined in the regulatory documents	No interaction	No access to data	Difficulties and misunderstandings in deciding whether TSD, PDD or SysRD should be changed or not, since there is not a document describing which system requirements are allocated to software.

Table A-1-4 Continued

Activity Number	Computational Accuracy (10)	Data Exchangeability (11)	Access Auditability (12)	Functional Understandability (13)
6	No accuracy requirements defined in the regulatory documents	There is an interaction, but no data is received from other processes, data is sent to other processes.	Auditable: E-mail	No difficulties or misunderstandings
7	According to on-paper process, software requirements should be reviewed in order to verify that they are complete, consistent, traceable and acceptable. This requirement is not implemented.	No interaction	No access to data: Minutes of meeting does not generally exist.	Difficulties and misunderstandings in determining high level software requirements
8	No accuracy requirements defined in the regulatory documents	No interaction	No access to data	Difficulties and misunderstandings in deciding whether all high level software requirements are clear enough to proceed with determining low level software requirements or not
9	According to on-paper process, software requirements should be reviewed in order to verify that they are complete, consistent, traceable and acceptable. This requirement is not implemented.	No interaction	Auditable: PVCS	Difficulties and misunderstandings in determining low level software requirements
10	No accuracy requirements defined in the regulatory documents	No interaction	No access to data	Difficulties and misunderstandings in deciding whether all low level requirements are clear enough to proceed with design or not
11	According to on-paper process, software requirements should be reviewed in order to verify that they are complete, consistent, traceable and acceptable. This requirement is not implemented.	Data received from hardware engineers is converted into software requirements.	No access to data: Minutes of meeting does not generally exist.	Difficulties and misunderstandings in determining hardware interface requirements
12	No accuracy requirements defined in the regulatory documents	No interaction	No access to data	Difficulties and misunderstandings in deciding whether all hardware interface requirements are clear enough to proceed with design or not
13	No accuracy requirements defined in the regulatory documents	No interaction	No access to data: Minutes of meeting does not generally exist.	Difficulties and misunderstandings in determining modules to be re-used

Table A-1-4 Continued

Activity Number	Computational Accuracy (10)	Data Exchangeability (11)	Access Auditability (12)	Functional Understandability (13)
14	No accuracy requirements defined in the regulatory documents	No interaction	No access to data	Difficulties and misunderstandings in deciding whether it is necessary to make some modifications to the modules to be re-used
15	According to on-paper process, software requirements should be reviewed in order to verify that they are complete, consistent, traceable and acceptable. This requirement is not implemented.	No interaction	No access to data	Difficulties and misunderstandings in reverse engineering modules to be re-used.
16	No accuracy requirements defined in the regulatory documents	No interaction	No access to data	No difficulties or misunderstandings
17	According to on-paper process, software requirements should be reviewed in order to verify that they are complete, consistent, traceable and acceptable. This requirement is not implemented.	Customers' requests are converted into technical software requirements.	No access to data	Difficulties and misunderstandings in converting customers' requests into technical software requirements.

Table A-1-5 AS-IS Analysis Process Metrics 14-17

Activity Number	Existence in Documents (14)	Input Validity Checking (15)	Undoability (16)	Attractive Interaction (17)
1, 4, 7, 11, 13	Not Described	No input validity checking	Not recorded: There is not a formal document prepared after the meeting, also minutes of meeting does not generally exist.	Not recorded: There is not a formal document prepared after the meeting, also minutes of meeting does not generally exist.
2, 3, 6	Not Described	No input validity checking	Undoable: E-mail can be called back.	Not attractive interaction: It is an ordinary e-mail; there is not a template prepared for this activity.
5, 8, 10, 12, 14, 15, 16, 17	Not Described	No input validity checking	Not recorded	Not recorded
9	Not Described	No input validity checking	Undoable: Minutes of meeting is prepared with Microsoft Word and stored with PVCS.	Not attractive interaction: There is not a formal requirements document prepared after the meeting; there is a template for minutes of meeting, but later it is hard to search for the minutes of meeting that contains the low level software requirements.

A.2 AS-IS Software Requirements Change Management Process Measurement Details

Table A-2-1 AS-IS Change Management Process Metrics 1-3

Activity Number	Complexity (1)	Coupling (2)	Failure Avoidance (3)
1	No decision	There is an interaction with the systems engineering process. Requirement change request may be sourced by system engineers. It may also be sourced by customers; they may request some changes during development or maintenance phases.	No review, inspection, checkpoint or similar techniques
2	No decision	No interaction	No review, inspection, checkpoint or similar techniques
3	Unstructured Decision: Deciding whether the requirement change is acceptable, partially acceptable or not acceptable is a complex task.	No interaction	No review, inspection, checkpoint or similar techniques
4, 5, 6	No decision	There is an interaction with the systems engineering process. The reply to the requirement change request is sent to the source of the request which may either be the systems engineers or the customer.	No review, inspection, checkpoint or similar techniques

Table A-2-2 AS-IS Change Management Process Metrics 4-5

Activity Number	Restorability (4)	Restoration Effectiveness (5)
1, 4, 5, 6	Recorded: E-mail is stored in the network.	Restorable
2	Not recorded: There is not a formal document prepared after the meeting, also minutes of meeting does not generally exist.	No restoration
3	Not recorded	No restoration

Table A-2-3 AS-IS Change Management Process Metrics 6-9

Activity Number	Functional Adequacy (6)	Functional Completeness (7)	IT Usage (8)	IT Density (9)
1, 4, 5, 6	Inadequate: There is not any defined on-paper software requirements change management process in regulatory documents.	This metric is measured using on-paper process definitions.	IT usage in sending e-mail.	E-mail is sent with Microsoft Outlook.
2, 3	Inadequate: There is not any defined on-paper software requirements change management process in regulatory documents.	This metric is measured using on-paper process definitions.	No IT usage.	Generally no formal documents are prepared.

Table A-2-4 AS-IS Change Management Process Metrics 10-13

Activity Number	Computational Accuracy (10)	Data Exchangeability (11)	Access Auditability (12)	Functional Understandability (13)
1	No accuracy requirements defined in the regulatory documents	Requirement change request is converted into technical software requirement.	Auditable: E-mail	Difficulties and misunderstandings in requirement change requests
2	No accuracy requirements defined in the regulatory documents	No interaction	No access to data: Minutes of meeting does not generally exist.	Difficulties and misunderstandings in evaluating the feasibility of requirement change requests
3	No accuracy requirements defined in the regulatory documents	No interaction	No access to data	Difficulties and misunderstandings in deciding whether the requirement change is acceptable, partially acceptable or not acceptable.
4, 5	According to on-paper process, software requirements should be reviewed in order to verify that they are complete, consistent, traceable and acceptable. This requirement is not implemented.	There is an interaction, but no data is received from other processes, data is sent to other processes.	Auditable: E-mail	No difficulties or misunderstandings
6	No accuracy requirements defined in the regulatory documents	There is an interaction, but no data is received from other processes, data is sent to other processes.	Auditable: E-mail	No difficulties or misunderstandings

Table A-2-5 AS-IS Change Management Process Metrics from 14-17

Activity Number	Existence in Documents (14)	Input Validity Checking (15)	Undoability (16)	Attractive Interaction (17)
1, 4, 5, 6	Not Described	No input validity checking	Undoable: E-mail can be called back.	Not attractive interaction: It is an ordinary e-mail; there is not a template prepared for this activity.
2	Not Described	No input validity checking	Not recorded: There is not a formal document prepared after the meeting, also minutes of meeting does not generally exist.	Not recorded: There is not a formal document prepared after the meeting, also minutes of meeting does not generally exist.
3	Not Described	No input validity checking	Not recorded	Not recorded

APPENDIX B

TO-BE STATIC PROCESS DEFINITIONS

B.1 TO-BE Software Requirements Analysis Process

Table B-1-1 TO-BE Software Requirements Analysis Process

No	Activity Name	Activity Definition	Staff	Forms/ Documents/ Archival Records/ Tools/ Applications/ Other Medias
1	Allocation Meeting	System requirements allocated to software are identified and documented in Allocation Document in a series of meetings.	Project Manager, Department Managers, Design Leaders	Conversation, TSD, PDD, SysRD (if exists), Minutes of Allocation Meeting, Allocation Document
2	Send TSD, PDD, SysRD and Allocation Document for review	ESD Manager sends TSD, PDD, SysRD (if exists) and Allocation Document to design leaders in ESD in order to ask their opinions on feasibility of system requirements allocated to software.	ESD Manager, Design Leaders	TSD, PDD, SysRD (if exists), Allocation Document E-mail
3	Review TSD, PDD, SysRD (if exists) and Allocation Document	Design leaders in ESD review TSD, PDD, SysRD (if exists) and Allocation Document and send their opinions on feasibility of system requirements allocated to software to ESD Manager.	ESD Manager, Design Leaders	TSD, PDD, SysRD (if exists), Allocation Document, E-mail

Table B-1-1 Continued

No	Activity Name	Activity Definition	Staff	Forms/ Documents/ Archival Records/ Tools/ Applications/ Other Medias
4	Feasibility meeting	Collected opinions on feasibility of system requirements allocated to software are evaluated in the feasibility meeting.	ESD Manager, Design Leaders	Conversation, Minutes of Feasibility Meeting
5	Decide whether TSD, PDD or SysRD (if exists) should be changed or not	ESD Manager decides whether TSD, PDD or SysRD (if exists) should be changed or not, according to the discussions in the feasibility meeting.	ESD Manager	-
6	Send change requests	If ESD Manager thinks that TSD, PDD or SysRD (if exists) should be changed, he prepares System Requirement Change Request Document (SRCRD) and sends it to Project Manager via e-mail.	ESD Manager, Project Manager	System Requirement Change Request Document, E-mail
7	High level software requirements meeting	High level software requirements are identified by ESD Manager and design leaders.	ESD Manager, Design Leaders	Conversation, Minutes of High Level Software Requirements Meeting
8	Prepare/Update SRS and RTM	Identified high level requirements are stored with a unique number in a database by the help of a requirements management tool. SRS and RTM are prepared/updated using the tool.	Design Leaders	Requirements Management Tool, SRS Template, RTM Template, SRS, RTM
9	Review & Update SRS and RTM	ESD Manager reviews SRS and RTM in order to ensure that the requirements are correct, unambiguous, complete, consistent, traceable and verifiable. ESD Manager uses Requirements Review Guideline and the requirements management tool while reviewing and updating SRS and RTM.	ESD Manager	Requirements Management Tool, Requirements Review Guideline, SRS Template, RTM Template, SRS, RTM

Table B-1-1 Continued

No	Activity Name	Activity Definition	Staff	Forms/ Documents/ Archival Records/ Tools/ Applications/ Other Medias
10	Decide whether all high level software requirements are clear enough to proceed with determining low level software requirements or not	High level software requirements meetings are repeated until ESD Manager thinks that all high level software requirements are clear enough to proceed with determining low level software requirements.	ESD Manager	-
11	Low level software requirements meeting	High level requirements are analyzed and elaborated to identify detailed low level software requirements. To capture requirements, use-case analysis is done by the help of a UML modeling tool.	Unit Leader, Developers	Conversation, Minutes of Low Level Software Requirements Meeting, UML Modeling Tool, UML Use-Case Models
12	Update SRS and RTM	Identified low level requirements are stored with a unique number in a database by the help of a requirements management tool. SRS and RTM are updated using the tool.	Developers	Requirements Management Tool, SRS Template, RTM Template, SRS, RTM
13	Review & Update SRS and RTM	Unit Leader reviews SRS and RTM in order to ensure that the requirements are correct, unambiguous, complete, consistent, traceable and verifiable. Unit Leader uses Requirements Review Guideline and the requirements management tool while reviewing and updating SRS and RTM.	Unit Leader	Requirements Management Tool, Requirements Review Guideline, SRS Template, RTM Template, SRS, RTM
14	Decide whether all low level requirements are clear enough to proceed with design or not	Low level software requirements meetings are repeated for every module until unit leader thinks that all low level requirements are clear enough to proceed with design.	Unit Leader	-

Table B-1-1 Continued

No	Activity Name	Activity Definition	Staff	Forms/ Documents/ Archival Records/ Tools/ Applications/ Other Medias
15	Hardware interface requirements meeting	Hardware interface requirements are determined in a meeting attended by hardware engineers, developers and the unit leader that is responsible for the software module interfacing hardware.	Unit Leader, Developers, Hardware Engineers	Conversation, Minutes of Hardware Interface Requirements Meeting
16	Update SRS and RTM	Identified hardware interface requirements are stored with a unique number in a database by the help of a requirements management tool. SRS and RTM are updated using the tool.	Developers	Requirements Management Tool, SRS Template, RTM Template, SRS, RTM
17	Review & Update SRS and RTM	Unit Leader reviews SRS and RTM in order to ensure that the requirements are correct, unambiguous, complete, consistent, traceable and verifiable. Unit Leader uses Requirements Review Guideline and the requirements management tool while reviewing and updating SRS and RTM.	Unit Leader	Requirements Management Tool, Requirements Review Guideline, SRS Template, RTM Template, SRS, RTM
18	Decide whether all hardware interface requirements are clear enough to proceed with design or not	Hardware interface requirements meetings are repeated for each module interfacing hardware until the related unit leader thinks that all hardware interface requirements for a module are clear enough to proceed with design.	Unit Leader	-
19	Meeting to determine software modules to be re-used	Software modules to be re-used are determined.	Unit Leader, Developers	Conversation, Minutes of Re-use Meeting
20	Decide whether it is necessary to make some modifications to the modules to be re-used	Unit Leader decides whether it is necessary to make some modifications to the modules to be re-used.	Unit Leader	-

Table B-1-1 Continued

No	Activity Name	Activity Definition	Staff	Forms/ Documents/ Archival Records/ Tools/ Applications/ Other Medias
21	Get the requirements of the modules to be re-used	If unit leader thinks that it is necessary to make some modifications to the modules to be re-used, the requirements of the modules are gathered from the related RTM and SRS using the requirements management tool.	Unit Leader	Requirements Management Tool, SRS, RTM, Requirements Re-use Report
22	Update SRS and RTM with the modified requirements of the modules to be re-used	Requirements of the modules to be re-used are modified and stored with a unique number in a database by the help of a requirements management tool. SRS and RTM are updated using the tool.	Developers	Requirements Management Tool, Requirements Re-use Report, SRS Template, RTM Template, SRS, RTM
23	Review & Update SRS and RTM	Unit Leader reviews SRS and RTM in order to ensure that the requirements are correct, unambiguous, complete, consistent, traceable and verifiable. Unit Leader uses Requirements Review Guideline and the requirements management tool while reviewing and updating SRS and RTM.	Unit Leader	Requirements Management Tool, Requirements Review Guideline, SRS Template, RTM Template, SRS, RTM
24	Decide whether a proof-of-concept (throw-away) prototype exists or not	ESD Manager decides whether a proof-of-concept (throw-away) prototype exists or not (Very simple decision but exists as an activity for the sake of completeness).	ESD Manager	-
25	Update SRS and RTM according to evaluation of the prototype	If a proof-of-concept (throw-away) prototype exists, stored software requirements are updated according to customer's requests and evaluation of the prototype by the help of a requirements management tool. SRS and RTM are updated using the tool.	Design Leaders	Customer Prototype Evaluation Report, Requirements Management Tool, SRS Template, RTM Template, SRS, RTM

Table B-1-1 Continued

No	Activity Name	Activity Definition	Staff	Forms/ Documents/ Archival Records/ Tools/ Applications/ Other Medias
26	Review & Update SRS and RTM	ESD Manager reviews SRS and RTM in order to ensure that the requirements are correct, unambiguous, complete, consistent, traceable and verifiable. ESD Manager uses Requirements Review Guideline and the requirements management tool while reviewing and updating SRS and RTM.	ESD Manager	Requirements Management Tool, Requirements Review Guideline, SRS Template, RTM Template, SRS, RTM

B.2 TO-BE Software Requirements Change Management Process

Table B-2-1 TO-BE Software Requirements Change Management Process

No	Activity Name	Activity Definition	Staff	Forms/ Documents/ Archival Records/ Tools/ Applications/ Other Medias
1	Receive a requirement change request	ESD Manager receives a requirement change request. It may be written or oral, and sourced by system engineers or customer.	ESD Manager	Requirement Change Request (Written or oral), E-mail, Telephone, Interview with customer
2	Prepare Requirement Change Request Report	A design leader assigned by ESD Manager prepares Requirement Change Request Report (RCRR), which contains the unique numbers of the requirements to be changed. The design leader also looks up the RTM using the requirements management tool to find out which requirements and which modules depend on the requirements to be changed, and adds the unique numbers of these requirements and modules to RCRR.	Design Leader	Requirements Management Tool, RCRR, RTM
3	Meeting on feasibility of the requirement change request	Feasibility of the requirement change request is analyzed in a meeting attended by ESD Manager and related design leaders.	ESD Manager, Design Leaders	Conversation, RCRR, Minutes of Feasibility Meeting
4	Decide whether the requirement change is acceptable, partially acceptable or not acceptable.	ESD Manager and design leaders decide whether the requirement change is acceptable, partially acceptable or not acceptable.	ESD Manager, Design Leaders	-

Table B-2-1 Continued

No	Activity Name	Activity Definition	Staff	Forms/ Documents/ Archival Records/ Tools/ Applications/ Other Medias
5	Accept the requirement change request	If ESD Manager and design leaders conclude that the requirement change request is feasible, ESD Manager accepts it by replying the source of change request via telephone or e-mail. The decision of acceptance of the request is added to RCRR.	ESD Manager	E-mail, Telephone, RCRR
6	Partially accept the requirement change request	If ESD Manager and design leaders conclude that the requirement change request is partially feasible, ESD Manager partially accepts it by replying the source of change request via telephone or e-mail. The decision of partial acceptance of the request is added to RCRR. Also, an explanation about the parts of the request that are accepted is added to RCRR.	ESD Manager	E-mail, Telephone, RCRR
7	Reject the requirement change request	If ESD Manager and design leaders conclude that the requirement change request is infeasible, ESD Manager rejects it by replying the source of change request via telephone or e-mail. The decision of rejection of the request is added to RCRR.	ESD Manager	E-mail, Telephone, RCRR
8	Update SRS and RTM	A design leader assigned by ESD Manager updates SRS and RTM according to the accepted change request using the requirements management tool.	Design Leader	Requirements Management Tool, RCRR, SRS Template, RTM Template, SRS, RTM
9	Review & Update SRS and RTM	ESD Manager reviews SRS and RTM in order to ensure that the requirements are correct, unambiguous, complete, consistent, traceable and verifiable. ESD Manager uses Requirements Review Guideline and the requirements management tool while reviewing and updating SRS and RTM.	ESD Manager	Requirements Management Tool, Requirements Review Guideline, RCRR, SRS Template, RTM Template, SRS, RTM

APPENDIX C

TO-BE PROCESS MEASUREMENT DETAILS

C.1 TO-BE Software Requirements Analysis Process Measurement Details

Table C-1-1 TO-BE Analysis Process Metrics 1-3

Activity Number	Complexity (1)	Coupling (2)	Failure Avoidance (3)
1	No decision	There is an interaction with the software project management and systems engineering processes. TSD, PDD and SysRD (if exists) should have been prepared before the allocation meeting. Project manager as well as managers and design leaders from all related departments attend the allocation meeting.	No review, inspection, checkpoint or similar techniques
2, 4, 7, 11, 19, 21	No decision	No interaction	No review, inspection, checkpoint or similar techniques
3	No decision	No interaction	Design leaders in ESD review TSD, PDD and SysRD (if exists).
5	Unstructured Decision: Deciding whether TSD, PDD or SysRD should be changed or not is a complex task.	No interaction	No review, inspection, checkpoint or similar techniques
6	No decision	There is an interaction with the software project management process. If ESD Manager thinks that TSD, PDD or SysRD should be changed, he sends change requests to Project Manager.	No review, inspection, checkpoint or similar techniques
8, 12, 16, 22	No decision	No interaction	SRS and RTM are prepared using SRS and RTM Templates
9, 13, 17, 23, 26	No decision	No interaction	SRS and RTM are reviewed using the Requirements Review Guideline, SRS Template and RTM Template

Table C-1-1 Continued

Activity Number	Complexity (1)	Coupling (2)	Failure Avoidance (3)
10	Unstructured Decision: Deciding whether all high level software requirements are clear enough to proceed with determining low level software requirements or not is a complex task.	No interaction	No review, inspection, checkpoint or similar techniques
14	Unstructured Decision: Deciding whether all low level requirements are clear enough to proceed with design or not is a complex task.	No interaction	No review, inspection, checkpoint or similar techniques
15	No decision	There is an interaction with the hardware engineering process. Hardware engineers attend the interface requirements meeting.	No review, inspection, checkpoint or similar techniques
18	Unstructured Decision: Deciding whether all hardware interface requirements are clear enough to proceed with design or not is a complex task.	No interaction	No review, inspection, checkpoint or similar techniques
20	Unstructured Decision: Deciding whether it is necessary to make some modifications to the modules to be re-used is a complex task.	No interaction	No review, inspection, checkpoint or similar techniques
24	Structured Decision: Deciding whether a proof-of-concept (throw-away) prototype exists or not is a very simple decision. It simply exists or not.	No interaction	No review, inspection, checkpoint or similar techniques
25	No decision	There is an interaction with the customer's evaluation of the proof-of-concept prototype process. Customer's requests resulted from the evaluation of the prototype are used to update software requirements.	SRS and RTM are prepared using SRS and RTM Templates

Table C-1-2 TO-BE Analysis Process Metrics 4-5

Activity Number	Restorability (4)	Restoration Effectiveness (5)
1	Recorded: Minutes of Allocation Meeting and Allocation Document are prepared.	Restorable: Word documents are stored in PVCS Database
2, 3	Recorded: E-mail is stored in the network.	Restorable: E-mail is stored in the network
4	Recorded: Minutes of Feasibility Meeting is prepared.	Restorable: Word documents are stored in PVCS Database
5, 10, 14, 18, 20, 24	Not recorded	No restoration
6	Recorded: System Requirement Change Request Document and e-mail are stored in the network.	Restorable: Word documents are stored in PVCS Database, e-mail is stored in the network
7	Recorded: Minutes of High Level Software Requirements Meeting is prepared.	Restorable: Word documents are stored in PVCS Database
8, 9, 12, 13, 16, 17, 22, 23, 25, 26	Recorded: SRS and RTM are updated.	Restorable: SRS and RTM are stored in Requirements Management Tool Database.
11	Recorded: Minutes of Low Level Software Requirements Meeting is prepared.	Restorable: Word documents and UML Models are stored in PVCS Database
15	Recorded: Minutes of Hardware Interface Requirements Meeting is prepared.	Restorable: Word documents are stored in PVCS Database
19	Recorded: Minutes of Re-use Meeting is prepared.	Restorable: Word documents are stored in PVCS Database
21	Recorded: Requirements Re-use Report is prepared.	Restorable: Word documents are stored in PVCS Database

Table C-1-3 TO-BE Analysis Process Metrics 6-9

Activity Number	Functional Adequacy (6)	Functional Completeness (7)	IT Usage (8)	IT Density (9)
1, 4, 7, 15, 19	Adequate	This metric is measured using on-paper process definitions.	IT usage in preparing and storing documents.	Documents are prepared with Microsoft Word and stored in PVCS Database
2, 3	Adequate	This metric is measured using on-paper process definitions.	IT usage in sending e-mail.	E-mail is sent with Microsoft Outlook
5, 10, 14, 18, 20, 24	Adequate	This metric is measured using on-paper process definitions.	No IT usage	No documents
6	Adequate	This metric is measured using on-paper process definitions.	IT usage in preparing, storing documents and sending e-mail.	Documents are prepared with Microsoft Word and stored in PVCS Database; e-mail is sent with Microsoft Outlook
8, 9, 12, 13, 16, 17, 22, 23, 25, 26	Adequate	This metric is measured using on-paper process definitions.	IT usage in preparing and storing documents.	SRS and RTM are prepared and stored in Requirements Management Tool Database.
11	Adequate	This metric is measured using on-paper process definitions.	IT usage in preparing and storing documents.	Documents are prepared with Microsoft Word, UML Use-Case Models are prepared with UML Modeling Tool and all of them are stored in PVCS Database
21	Adequate	This metric is measured using on-paper process definitions.	IT usage in searching, preparing and storing documents.	Requirements to be re-used are found using Requirements Management Tool, documents are prepared with Microsoft Word and stored in PVCS Database

Table C-1-4 TO-BE Analysis Process Metrics 10-13

Activity Number	Computational Accuracy (10)	Data Exchangeability (11)	Access Auditability (12)	Functional Understandability (13)
1	No accuracy requirements defined in the regulatory documents	No change is performed on TSD, PDD and SysRD (if exists) before using them.	Auditable: PVCS Database	Difficulties and misunderstandings in allocating system requirements to software requirements, since SysRD does not always exist and PDD is used instead
2, 3	No accuracy requirements defined in the regulatory documents	No interaction	Auditable: E-mail	No difficulties or misunderstandings
4, 21	No accuracy requirements defined in the regulatory documents	No interaction	Auditable: PVCS Database	No difficulties or misunderstandings
5, 20, 24	No accuracy requirements defined in the regulatory documents	No interaction	No access to data	No difficulties or misunderstandings
6	No accuracy requirements defined in the regulatory documents	There is an interaction, but no data is received from other processes, data is sent to other processes.	Auditable: PVCS Database, E-mail	No difficulties or misunderstandings
7	No accuracy requirements defined in the regulatory documents	No interaction	Auditable: PVCS Database	Difficulties and misunderstandings in determining high level software requirements
8, 12, 16, 22	No accuracy requirements defined in the regulatory documents	No interaction	Auditable: Requirements Management Tool	No difficulties or misunderstandings
9, 13, 17, 23, 26	According to on-paper process, software requirements should be reviewed in order to verify that they are complete, consistent, traceable and acceptable. This requirement is implemented.	No interaction	Auditable: Requirements Management Tool	No difficulties or misunderstandings
10	No accuracy requirements defined in the regulatory documents	No interaction	No access to data	Difficulties and misunderstandings in deciding whether all high level software requirements are clear enough to proceed with determining low level software requirements or not
11	No accuracy requirements defined in the regulatory documents	No interaction	Auditable: PVCS Database	Difficulties and misunderstandings in determining low level software requirements
14	No accuracy requirements defined in the regulatory documents	No interaction	No access to data	Difficulties and misunderstandings in deciding whether all low level requirements are clear enough to proceed with design or not

Table C-1-4 Continued

Activity Number	Computational Accuracy (10)	Data Exchangeability (11)	Access Auditability (12)	Functional Understandability (13)
15	No accuracy requirements defined in the regulatory documents	Data received from hardware engineers is converted into software requirements.	Auditable: PVCS Database	Difficulties and misunderstandings in determining hardware interface requirements
18	No accuracy requirements defined in the regulatory documents	No interaction	No access to data	Difficulties and misunderstandings in deciding whether all hardware interface requirements are clear enough to proceed with design or not
19	No accuracy requirements defined in the regulatory documents	No interaction	Auditable: PVCS Database	Difficulties and misunderstandings in determining modules to be re-used
25	No accuracy requirements defined in the regulatory documents	Customers' requests are converted into technical software requirements.	Auditable: Requirements Management Tool	Difficulties and misunderstandings in converting customers' requests into technical software requirements.

Table C-1-5 TO-BE Analysis Process Metrics 14-17

Activity Number	Existence in Documents (14)	Input Validity Checking (15)	Undoability (16)	Attractive Interaction (17)
1, 4, 7, 11, 15, 19	Described in documents	No input validity checking	Undoable: Word and PVCS usage	Attractive Interaction: There is a template for Minutes of Meeting.
2, 3	Described in documents	No input validity checking	Undoable: E-mail can be called back.	Not attractive interaction: It is an ordinary e-mail; there is not a template prepared for this activity.
5, 10, 14, 18, 20, 24	Described in documents	No input validity checking	Not recorded	No interaction with documents or archival records.
6	Described in documents	No input validity checking	Undoable: Word, PVCS and e-mail usage.	Attractive Interaction: There is a template for System Requirement Change Request Document
8, 9, 12, 13, 16, 17, 22, 23, 25, 26	Described in documents	No input validity checking	Undoable: Requirements Management Tool usage.	Attractive Interaction: SRS and RTM are updated using templates and Requirements Management Tool.
21	Described in documents	No input validity checking	Undoable: Requirements Management Tool, Word and PVCS usage.	Attractive Interaction: Requirements Management Tool is used.

C.2 TO-BE Software Requirements Change Management Process Measurement Details

Table C-2-1 TO-BE Change Management Process Metrics 1-3

Activity Number	Complexity (1)	Coupling (2)	Failure Avoidance (3)
1	No decision	There is an interaction with the systems engineering process. Requirement change request may be sourced by system engineers. It may also be sourced by customers; they may request some changes during development or maintenance phases.	No review, inspection, checkpoint or similar techniques
2, 3	No decision	No interaction	No review, inspection, checkpoint or similar techniques
4	Unstructured Decision: Deciding whether the requirement change is acceptable, partially acceptable or not acceptable is a complex task.	No interaction	No review, inspection, checkpoint or similar techniques
5, 6, 7	No decision	There is an interaction with the systems engineering process. The reply to the requirement change request is sent to the source of the request which may either be the systems engineers or the customer.	No review, inspection, checkpoint or similar techniques
8	No decision	No interaction	SRS and RTM are prepared using SRS and RTM Templates
9	No decision	No interaction	SRS and RTM are reviewed using the Requirements Review Guideline, SRS Template and RTM Template

Table C-2-2 TO-BE Change Management Process Metrics 4-5

Activity Number	Restorability (4)	Restoration Effectiveness (5)
1	Recorded: E-mail is stored in the network.	Restorable
2	Recorded: Requirement Change Request Report (RCRR) is stored in PVCS.	Restorable
3	Recorded: Minutes of Feasibility Meeting is stored in PVCS Database.	Restorable
4	Not recorded	No restoration
5, 6, 7	Recorded: E-mail is stored in the network, RCRR is stored in PVCS Database.	Restorable
8, 9	Recorded: SRS and RTM are stored in Requirements Management Tool Database.	Restorable

Table C-2-3 TO-BE Change Management Process Metrics 6-9

Activity Number	Functional Adequacy (6)	Functional Completeness (7)	IT Usage (8)	IT Density (9)
1	Adequate	This metric is measured using on-paper process definitions.	IT usage in sending e-mail.	E-mail is sent with Microsoft Outlook.
2	Adequate	This metric is measured using on-paper process definitions.	IT usage in preparing and storing documents.	RCRR is prepared using Requirements Management Tool and Microsoft Word. RCRR is stored in PVCS Database.
3	Adequate	This metric is measured using on-paper process definitions.	IT usage in preparing and storing documents.	Minutes of Meeting is prepared with Microsoft Word and stored in PVCS Database.
4	Adequate	This metric is measured using on-paper process definitions.	Not recorded	No documents
5, 6, 7	Adequate	This metric is measured using on-paper process definitions.	IT usage in sending e-mail, preparing and storing documents.	RCRR is updated using Microsoft Word and stored in PVCS Database. E-mail is sent with Microsoft Outlook.
8, 9	Adequate	This metric is measured using on-paper process definitions.	IT usage in preparing and storing documents.	SRS and RTM are prepared and stored in Requirements Management Tool Database.

Table C-2-4 TO-BE Change Management Process Metrics 10-13

Activity Number	Computational Accuracy (10)	Data Exchangeability (11)	Access Auditability (12)	Functional Understandability (13)
1	No accuracy requirements defined in the regulatory documents	Requirement change requests are received from system engineers or customers. The requests can be written or oral. Data conversion is done in the next activity.	Auditable: E-mail	Difficulties and misunderstandings in requirement change requests
2	The unique numbers corresponding to the requirements that are requested to be changed should be found. Also, the unique numbers of requirements and modules that depend on the requirements to be changed should be found. These are done in the activity.	The unique numbers corresponding to the requirements that are requested to be changed are found.	Auditable: Requirements Management Tool, PVCS	No difficulties or misunderstandings
3	No accuracy requirements defined in the regulatory documents	No interaction	Auditable: PVCS	No difficulties or misunderstandings
4	No accuracy requirements defined in the regulatory documents	No interaction	No access to data	No difficulties or misunderstandings
5, 6, 7	No accuracy requirements defined in the regulatory documents	There is an interaction, but no data is received from other processes, data is sent to other processes.	Auditable: E-mail, PVCS	No difficulties or misunderstandings
8	No accuracy requirements defined in the regulatory documents	No interaction	Auditable: Requirements Management Tool	No difficulties or misunderstandings

Table C-2-4 Continued

Activity Number	Computational Accuracy (10)	Data Exchangeability (11)	Access Auditability (12)	Functional Understandability (13)
9	According to on-paper process, software requirements should be reviewed in order to verify that they are complete, consistent, traceable and acceptable. This requirement is implemented.	No interaction	Auditable: Requirements Management Tool	No difficulties or misunderstandings

Table C-2-5 TO-BE Change Management Process Metrics from 14-17

Activity Number	Existence in Documents (14)	Input Validity Checking (15)	Undoability (16)	Attractive Interaction (17)
1	Described in documents	No input validity checking	Undoable: E-mail can be called back.	Not attractive interaction: It is an ordinary e-mail; there is not a template prepared for this activity.
2	Described in documents	Input validity checking: Checking whether there exist requirements corresponding to the change request.	Undoable: Requirements Management Tool, PVCS, Word usage	Attractive interaction: Requirements Management Tool, PVCS and Word usage
3	Described in documents	No input validity checking	Undoable: PVCS, Word usage	Attractive interaction: PVCS and Word usage
4	Described in documents	No input validity checking	Not recorded	No interaction with documents or archival records.
5, 6, 7	Described in documents	No input validity checking	Undoable: E-mail can be called back; PVCS and Word usage	Attractive interaction: PVCS and Word usage.
8, 9	Described in documents	No input validity checking	Undoable: Requirements Management Tool usage	Attractive Interaction: SRS and RTM are updated using templates and Requirements Management Tool.