Software Process Assessment & Improvement in Industrial Requirements Engineering

Tony Gorschek

Blekinge Institute of Technology Licentiate Series No 2004:07
ISSN 1650-2140
ISBN 91-7295-041-2

Kaserntryckeriet AB
Karlskrona, Sweden

Jacket Illustration: moebius © 2004 Brent Dearth [a6@aural-6.com]

Software Process Assessment & Improvement in Industrial Requirements Engineering

Tony Gorschek

Tony Gorschek

Digitally signed by Tony Gorschek
DN: CN = Tony Gorschek, C = SE, O = BTH, OU = APS
Reason: I am the author of this document
Date: 2004.10.22 13:17:49 +02'00'

School of Engineering
Department of Systems and Software Engineering
Blekinge Institute of Technology
Sweden

to my beloved family, all four…

Abstract

Requirements Engineering (RE) is a crucial part of any product management and product development activity, and as such deficiencies in the RE process may have severe consequences. There are reports from industry that point towards inadequate requirements being one of the leading sources for project failure.

Software Process Improvement (SPI) is generally seen as the main tool to address process deficiencies in general and within RE. Assessments lead to establishing plans for improvements that are subsequently implemented and evaluated, and then the SPI cycle starts again, in an optimal case being incremental and continuous.

Most well known SPI frameworks, e.g. CMM, CMMI, SPICE and QIP, are based on these general principles. There are however several factors that can have a negative impact on SPI efforts in general, and in the case of SPI targeted at RE in particular. *Time* and *cost* are two fundamental factors that can effectively "raise the bar" for SPI efforts being initiated at all. This is the particular case for Small and Medium sized Enterprises (SMEs) with limited resources, and a limited ability to wait for the return on their investment. Other issues include commitment and involvement in the SPI work by the ones affected by the changes, coverage of the RE area in SPI frameworks, and the ability to focus improvements to areas where they are needed the most.

The research presented in this thesis is based on actual needs identified in industry, and all of the proposed solutions have also been validated in industry to address issues of applicability and usability. In general, the goal of the research is to "lower the bar", i.e. enabling SMEs to initiate and perform SPI activities. It is accomplished through the presentation and validation of two assessment methods that targets RE, one aimed at both fast and low-cost benchmarking of current practices, and the other designed to produce tangible improvement proposals that can be used as input to an improvement activity, i.e. producing a relatively accurate assessment but taking limited time and resources into account.

Further, to offer a structured way in which SMEs can focus their SPI efforts, a framework is introduced that can be used to package improvement proposals with regards to their relative priority taking dependencies into account. This enables SMEs to choose what to do first based on their needs, as well as a way to control time to return on their investment by controlling the size of the undertaking.

As a result of industry validation of the assessment method and packaging framework, several improvement proposals were identified and prioritized/packaged. As a part of a process improvement effort (based on an improvement proposal package) an RE model was developed that was appropriate for SMEs faced with a market-driven product centered development situation. The model, called Requirements Abstraction Model (RAM), addresses the structuring and specification of requirements.

The main feature of the model is that it not only offers a structured way in which requirements can be specified, but it also takes a requirement's abstraction level into account, using abstraction for the work-up instead of putting all requirements in one repository independent of abstraction level. The RAM was developed to support primarily the product management effort, recognizing that RE from this perspective is not project initiated but rather project initiating. The model assists product managers to take requirements on varying abstraction levels and refining them to the point of being good-enough to offer decision support for management, and at the same time being good-enough for project initiation.

The main contribution of the thesis is to present SMEs with "tools" that help them commit to and perform SPI activities. Moreover, the thesis introduces the RAM model that was developed based on needs identified in industry, and subsequently piloted in industry to assure usability.

Acknowledgements

First and foremost I would like to extend my sincere gratitude to my supervisor and collaborator Professor Claes Wohlin. Without his guidance and advice this endeavor would not have been such a great experience.

The research presented in this thesis was conducted in close and fruitful cooperation between academia and industry. I would like to thank everyone involved in the process improvement work at Danaher Motion Särö AB (engineers as well as management) for their commitment and tireless work, in particular Stefan Börlin, Per Garre and Arne Hedström.

I would also like to thank all participants of the process improvement work conducted at ABB Automation Technology Products, in particular Cecilia Bergström and Stefan Forssander for their support.

The industry cooperation mentioned here has been an invaluable learning experience, and I would like to thank all involved for their help, patience, and enthusiasm for new ideas as well as seeing possibilities and not obstacles.

The work environment at Blekinge Institute of Technology is inspiring and heartfelt, and is nourished by all colleagues (researchers, teachers and administration). With this in mind I would like to thank all my colleagues for never being too busy, always lending a helping hand. I would also like to thank the SERL group for being a central part of creating this environment.

This work was partly funded by The Knowledge Foundation in Sweden under a research grant for the project "Blekinge - Engineering Software Qualities (BESQ)" (http://www.ipd.bth.se/besq).

Last but not least I would like to thank my beloved ones for putting up with all the work. You are the best.

Contents

List of Paper Co-Authors

Professor Claes Wohlin
Blekinge Institute of Technology, School of Engineering
claes.wohlin@bth.se

Dr. Mikael Svahnberg
Blekinge Institute of Technology, School of Engineering
mikael.svahnberg@bth.se

Kaarina Tejle
Mandator
kaarina.tejle@mandator.com

Chapter One

Introduction

Chapter One

It is not necessary to change. Survival is not mandatory.
- W. Edwards Deming (1900-1993)

1. Introduction

During the past decade Requirements Engineering (RE) emerged into its own field of study within software engineering. This was witnessed by the commencement of an international journal, 'Requirements Engineering Journal' published by Springer, and a conference and symposium sponsored by IEEE. By the late 1990's the field had grown into being able to support several other conferences and meetings discussing requirements engineering.

RE involves the systematic process of eliciting, understanding, analyzing, documenting and managing requirements throughout a product's life cycle [1]. Requirements from a software engineering perspective are the descriptions of what services and under what constraints a system should operate [2].

As requirements are direct products of the requirements engineering process inadequacies herein can have severe consequences. These consequences are mainly due to the fact that problems in requirements filter down to design and implementation [2]. Davis published results indicating that it could be up to 200 times as costly to catch and repair defects during the maintenance phase of a system, compared to the requirements engineering phase [3], and several other sources indicate that inadequate requirements are the leading source for project failure [4-9].

If RE is studied in the context of a market-driven incremental product development situation, which is becoming increasingly commonplace in software industry [10, 11], requirements engineering is also a prerequisite for release planning activities, i.e. the decision about which customers get what features and quality at what point in time, which in itself is a major determinant of the success of a product [12].

The importance of having an adequate RE process in place that produces good-enough requirements can be considered as crucial to the successful development of products, whether it be in a bespoke [2] or market-driven development effort. There are however clear indications that requirements engineering is lacking in industry since inadequacies in requirements is a major determinant in project failure [4-9]. As such the increased attention given to the field of RE in research is understandable and crucial.

The increased focus on requirements engineering can be seen in the area of Software Process Improvement (SPI) as well through the incorporation of more RE specific practices in recent SPI frameworks (e.g. CMMI [13]) compared to the practices devoted to RE in previous ones (e.g. CMM [14]).

SPI is aimed at helping organizations in industry improve the software engineering and management practices in order to enable an organization to produce products of

high-enough quality, and at a low-enough cost, to be competitive [2, 15]. It is only logical to conclude that RE improvements is a part of this work.

However, many SPI frameworks are often too large and bulky to get an overview of and to implement [16-18], and if implemented return on investment can take anything from 18 to 24 months [15]. This makes it difficult for organizations, in particular small and medium sized enterprises (SMEs), to initiate and perform improvement activities as cost and time are crucial considerations [16-18]. In addition to this there are several other critical factors, both RE specific and general, that can have an adverse effect on SPI efforts [19-23].

The goal of the research presented in this thesis is to assist SMEs to perform assessments and improvements aimed towards requirements engineering. This is accomplished through the presentation of assessment and improvement methods which were designed with RE and SMEs in mind, as well as taking the critical factors mentioned above under consideration. In addition an RE model is presented. It was designed and validated in industry as a result of a process improvement activity.

The methods presented, as well as the RE model, originate from explicit needs identified in industry, and subsequent to their development all parts were also validated in industry. The reasoning in using industry as a laboratory was to ensure that state-of-practice was not ignored which has been considered a threat against practical applicability evaluation of research results as well as successful technology transfer [24, 25]. The research partners cooperated with in the research presented in this thesis are Danaher Motion Särö AB - Primary research partner, and ABB Automation Technology Products AB - Secondary research partner.

The introduction of this thesis (Chapter One) is structured as follows.

Section 1.1 aims to give an introduction to SPI concepts and a brief overview of some of the general SPI frameworks in use today. This information serves as a background primarily for the critical SPI success factors presented in Sections 1.2 and 1.3.

Section 1.2 gives an overview of several critical factors that can be crucial for SPI success in general and the factors are discussed in the context of the SPI frameworks presented previously.

Section 1.3 gives an overview of additional critical SPI factors, but from the perspective of process improvement targeted at RE. The factors are discussed in the context of the SPI frameworks presented previously, and to some extent related to previously presented general critical factors.

Section 1.4 gives the outline and contribution of the research presented in this thesis by presenting four papers. The contribution of each paper is compared to the critical factors presented in Sections 1.2 and 1.3. This is done to clarify the contribution as well as place the research in a relevant context.

Section 1.5 gives an overview of the research approach used presenting scope, research questions and methods/tools used to answer the research questions.

Section 1.6 elaborates on future work and **Section 1.7** presents the conclusions drawn based on the research presented in the thesis.

1.1. Software Process Assessment & Improvement – General Concepts

There exist several well-known and established SPI frameworks used for process assessment and improvement. Most of them are based on a general principle of four fairly straightforward steps, "evaluation of the current situation", "plan for improvement", "implement the improvements", "evaluate the effect of the improvements", and then the work takes another cycle (see Figure 1-1, inspired by [16]).

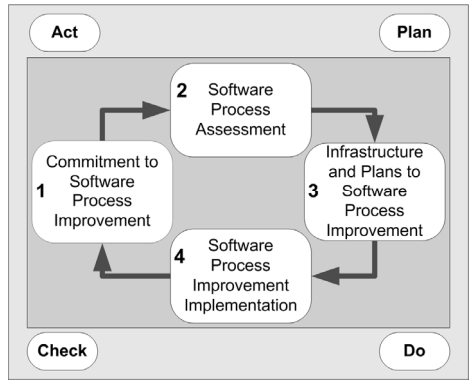


Figure 1-1. Generic process improvement

A classic example of this continuous and in theory never ending cycle of improvement is seen in Shewart–Deming's PDCA (Plan-Do-Check-Act) paradigm, which embraced the necessity for a cyclic and continuous process improvement as early as 1939 [26].

However, there is a clear and distinctive difference between frameworks and how they approach process improvement. A rudimentary division of SPI frameworks can be done based on whether or not they are bottom-up (inductive) or top-down model-based (prescriptive) in nature. Below inductive and prescriptive SPI frameworks are characterized through exemplification of well-known examples of each.

1.1.1. Inductive Frameworks

Basili's well-known QIP (Quality Improvement Paradigm) [27] is based on a bottom-up approach that is inductive in nature, i.e. what is to be performed in terms of improvements is based on a thorough understanding of the current situation (processes) [28]. QIP proposes a tailoring of solutions based on identification of critical issues identified in the project organization. Subsequently the solutions are evaluated in pilot projects before a official change is made to the process [29].

The idea is to use experiences from executing processes in projects to base improvements on, i.e. there is no general initial assessment like performing a "baselining" against a pre-defined set of practices. Rather, as illustrated in Figure 1-2, inspired by [30], quantifiable goals are set, based on this improvements are chosen (which can be anything from new processes, methods, techniques or tools). The improvement is then tested in a project (A through C), the metrics gathered are analyzed and compared to prior experiences, and last the new experiences are packaged and stored, i.e. in essence incorporated in the total experience repository of the organization [31].

GQM (Goal-Question-Metric) can be used for the gathering of metrics (as a way to measure impact of an improvement). Basically GQM is centered on a goal-oriented measurement of projects [31, 32].

Strictly speaking QIP does not require any specific method (e.g. GQM) be used but can work with any measurement/analysis methodology. However, GQM is well suited for the task as both GQM and QIP are based on setting goals (measurable), choosing and executing a change (improvement) and then carefully analyzing the results of that change before any decision is made as to keep the change or not. Either way the experience of a cycle (see Figure 1-2) is packaged and stored, in effect adding to the knowledge of the organization whether an improvement was successful or not.

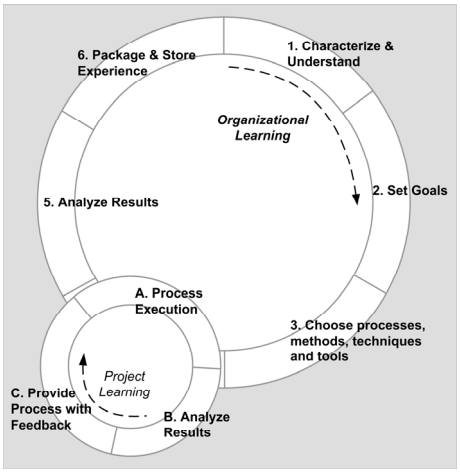


Figure 1-2. Quality Improvement Paradigm.

1.1.2. Prescriptive (Model-based) Frameworks

In contrary to inductive frameworks model-based process improvement is a prescriptive approach which is based on a collection of best practices describing how e.g. software should be developed. The prescriptive nature of such models lay in the fact that *one* set of practices is to be adhered to by all organizations. No special consideration is taken as to an organizations situation or needs other than how the development process (at the organization subject to SPI) is in comparison to the one offered through the framework [15, 33]. A general trait common for most model-based frameworks is that assessments are performed as a benchmarking against the set of practices advocated by the model in question, i.e. interviews, questionnaires, and so on used as tools of the assessment are designed towards benchmarking.

1.1.2.1 Capability Maturity Model

The Software Engineering Institute's CMM (Capability Maturity Model) [34-36] is probably one of the most well-known model-based SPI standards. It is centered on standardizing the contents of processes according to a predefined number of practices. It generally follows the steps described in Figure 1-1, i.e. starting with assessment of the organizational maturity (benchmarking against CMM practices). The process assessments generically associated with CMM are called CMM-based appraisals (CBAs). They are based on CMM's practices and an SEI (Software Engineering Institute) questionnaire. An assessment can be performed with the assistance of SEI professionals or as a self-assessment (which requires training of in-house personnel by SEI) [15].

Subsequent to the assessment there is planning and preparation for introducing relevant practices (or changing present ones) to conform to CMM. This concerns both what practices are used, and in what order they are implemented.

After the execution of the improvement a new assessment is performed to evaluate the success of the improvement as well as prepare for the next interaction [37].

The practices of CMM are organized into Key Process Areas (KPA), e.g. Requirements Management and Software Quality Assurance (CMM V1.1).

Each KPA is placed on a certain level of maturity spanning from 1 to 5. The maturity levels are presented in Figure 1-3 where the bottom most level (1: Initial) is the lowest level of maturity where practices are undefined, the process is ad-hoc and non-repeatable (CMM does not have any KPAs on this level). As the maturity of an organization matures (Level 2 and upwards) KPAs (with relevant practices) are introduced for every step in a predefined order.

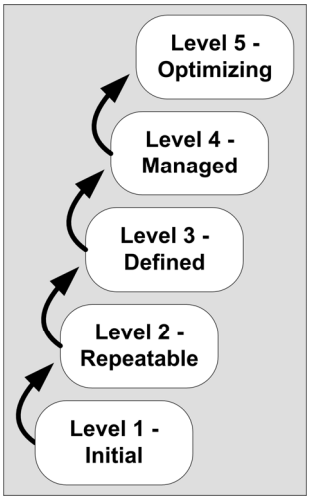


Figure 1-3. CMM maturity levels.

As the original CMM was not aimed at any specific discipline (e.g. software development), but rather general in nature (suitable for any development organization) there was a need to introduce several other versions of the CMM framework which were aimed at more specific disciplines. As a result several "specific" CMM versions emerged, e.g. SW-CMM (CMM for Software) [14] and IPD-CMM (Integrated Product Development CMM) [38]. In addition several other standards, largely based on or inspired by CMM, emerged. Some of the more well-known are CMMI (Capability Maturity Model Integration) [36], ISO/IEC 15504 (a.k.a. SPICE – Software Process Improvement & Capability dEtermination) [39], and BOOTSTRAP [15].

1.1.2.2 Capability Maturity Model Integration

CMMI is the newest of these and is largely based on SW-CMM (V.2.0C) and IPD-CMM (V0.98A). Like the name implies "Integration" stands for the fact that CMMI is an integration of several CMM versions. The necessity of integration came from the impracticality of using several CMM versions in the same organization in order to cover the entire spectrum needed. CMMI was developed as "the one model" to replace use of multiple versions [40].

CMMI comes in two basic versions *Staged* and *Continuous Representation*. Both are based on the same KPAs (i.e. same content), but they are represented differently and thus address SPI in different ways. *Staged Representation* is aimed towards assessing and improving overall organizational maturity, i.e. following the maturity levels and implementing practices (KPAs) as indicated to achieve an overall organizational maturity increase. *Continuous Representation* is adapted towards assessing and improving individual process areas, e.g. if RE practices are considered lacking in an organization, CMMI Continuous Representation can be used to address the specific process areas as relevant.

However, it should be noted that even if CMMI allows for targeted improvements it still guides priorities, i.e. what practices should be improved/added and in what order, as it still is prescriptive (model-based) in nature [40].

The appraisal methodology that is a part of CMMI is based on several appraisal requirements called ARC (Appraisal Requirements for CMMI). These requirements are a basis on which appraisals can be developed, i.e. of primary interest for development of new appraisal methods. The official implemented appraisal method for CMMI is called SCAMPI (Standard CMMI Appraisal Method for Process Improvement) and is developed to meet all requirements described in ARC as well as compliant to ISO/IEC 15504 (SPICE) [40].

In general CMMI supports three classes of appraisals [41, 42]:

- **Class A:** Full and comprehensive method. Covers the entire CMMI model and provides a maturity level of the organization as a whole. SCAMPI (V1.1) is a Class A assessment method. The effort needed for completing a SCAMPI assessment is considerable, i.e. ranging from 800 to 1600 person hours.
  Tools: E.g. Interviews, Document Reviews, Questionnaires or other instrument.
- **Class B:** Less in depth than Class A. Concentrated on areas that need attention and gives no overall maturity rating. It is considered as beneficial as an initial assessment method. The effort needed for completing a Class B assessment can range from 80 to 640 person-hours. It should be observed that several Class B assessments may be needed to completely assess an area.
  Tools: E.g. Group interviews and document reviews.
- **Class C:** This is often called "a quick look" at specific risk areas. The effort needed for completing a Class B assessment can range from 20 to 60 person-hours.
  Tools: E.g. Self assessment.

It should be noted that the estimations stated above are of effort pertaining to the assessment group (which can range in size), and that the effort of other personnel, e.g. the ones being interviewed/filling out questionnaires and so on are not included. Parts of the assessment group in the case of Class A and B appraisals have to be trained and certified, Class C appraisals require little training.

1.1.2.3 Software Process Improvement & Capability dEtermination

ISO/IEC 15504, or as it is often referred to, SPICE, is influenced by CMM, but if compared SPICE is closer to CMMI. It should be noted that SPICE was developed in 1993, before CMMI, and that some parts of CMMI were designed to conform to ISO/IEC 15504 and not the other way around [40, 43]. This in addition to the fact that both were originally influenced by CMM makes for relatively easy mapping of contents (process areas) between SPICE and CMMI. There are however some differences, e.g. process areas that are present in one, but not in the other. A fundamental difference is that SPICE has only Continuous Representation (not Staged).

ISO/IEC 15504 assessments[1] are very similar to SCAMPI (CMMI Class A) as mentioned above, i.e. both have similar requirements. A fundamental difference is that while CMMI can use both internal (if trained) or external (SEI) assessment group members SPICE demands that an external assessor be head of the assessment [43, 44].

1.2. General SPI - Critical Factors

There is no shortage in SPI experiences reported from industry assessment and improvement efforts, based on both inductive and prescriptive frameworks, like the ones described above. Several factors (issues) have been identified during the practical application of these frameworks as determinants of success. Some of these factors (relevant to the research presented in this thesis) are presented in a summarized form below, and to some extent exemplified with the use of the frameworks described above.

1.2.1. SPI Initiation Threshold

The initial critical success factor is of course that an SPI initiative be adopted in the first place. *The threshold for initiating and committing to an SPI* effort is often high due to the associated resources that have to be committed. An assessment-improvement cycle is often rather expensive and time consuming [45]. A typical SPI cycle using e.g. CMM can take anything from 18 to 24 months to complete and demand much resources and long-time commitments in order to be successful [15].

In addition the threshold is not lowered by the fact that many view extensive SPI frameworks, e.g. CMMI and SPICE as too large and bulky to get an overview of and to implement [16-18]. This is in particular the case for small and medium sized enterprises (SMEs) (e.g. less than 250 employees) [46] where time and resources always are an issue both regarding assessment and improvement [16-18].

The problem of SPI frameworks being too large, costly and running over extended periods of time (long time until return on investment) is confirmed by some initiatives in research to develop SPI frameworks of a lightweight type. Examples of this can be seen through the IMPACT project [47] where a QIP inspired framework is presented. Adaptations and versions of prescriptive frameworks like CMM has also been presented, see e.g. IDEAL [48] and Dynamic CMM [49].

1.2.2. Commitment and Involvement

Assuming that there is a genuine desire and need for SPI in an organization there has to be *commitment from management*, which is considered one of the most crucial factors for SPI to be successful. SPI efforts need to be actively supported and management needs to allow for resources to be dedicated to the SPI effort. An example of a reoccurring problem is assuming that SPI work will be accomplished in addition to the organizations regular work-load [19-23, 50]. Management commitment is of course to some extent

[1] It should be noted that SPICE assessments referred to here is the type associated with SPI (improving the organization), i.e. not capability determination (assess capability of e.g. suppliers) or self assessment (assess capability of accepting certain projects).

connected to cost and resource issues presented above, as management is less likely to commit to an SPI effort if it is very costly and time consuming.

Commitment from management is of course not enough to ensure success. There has to be *commitment and involvement by management, middle management and the staff e.g. developers* (i.e. other parties that are involved in the SPI work and that are influenced by the outcome). It is a genuinely good idea to let the ones working with the processes every day be actively involved in the improvement work [19-23, 51-53]. One reason for this is that people that are a part of the organization often have insights and knowledge about what areas are in need of improvement, and this knowledge often becomes explicit during an assessment activity [54].

The use of inductive SPI frameworks is based on collecting and using experiences as a basis for all SPI work, which speaks to the advantage of e.g. QIP in this regard as the work is based on the experience of coworkers. However as there is no set of best practices (i.e. a predefined set of process areas like in CMMI or SPICE) QIP improvements might be limited in an organization with low maturity (inexperienced). CMMI could provide structure and a well defined roadmap to the SPI activity. On the other hand, CMMI might force practices on e.g. developers that they do not consider relevant or necessary, or miss issues that are important to the organization [55].

1.2.3. Goals and Measurement

Irrelevant of SPI framework there should be *clear and well defined goals* pertaining to the SPI activity. This is achieved through preparation and planning, but also through having clear focus on what needs to be done (what is important to improve and why) [19-23, 56]. As improvements should be performed in a continuous manner, taking small evolutionary steps, prioritization of what to improve and in which order is implied [34, 50, 52]. This priority is a given when using prescriptive SPI frameworks as practices are predefined and so is the order of implementation. Using inductive frameworks like QIP this is left up to the SPI work group as the situation (current state) in the organization steers improvements as well as priority.

A crucial part of any SPI activity is the ability to *measure improvement effect* in order to learn whether or not an improvement is successful [32, 50, 56, 57]. QIP propagates stringent adherence to collection (during pilot project) and analysis of metrics through e.g. GQM in order to ascertain if an improvement has been successful (compared to the goals set up). Prescriptive model-based SPI frameworks measure e.g. adherence to a set of predefined practices primarily, although measurement methods can be used as a part of this.

1.2.4. Summary of General SPI - Critical Factors

A summary of the factors presented above gives the following list.

- **SPI Initiation Threshold** can be broken down into two main factors:
  - F1: Time to return on investment (long-term work, long-term gain for an SPI cycle).

- F2: Cost/Resources/Size (the nature of large SPI frameworks implies commitment of much resources over an extended period of time regarding both assessment and the actual improvement).

- **Commitment and Involvement** can be broken down into two factors:
  - F3: Commitment to SPI by management.
  - F4: Commitment to SPI and Involvement in the SPI work by coworkers.
- **Goals and Measurement** can be broken down into two factors:
  - F5: Focus (on the SPI effort with clear and well-defined goals).
  - F6: Measurability of improvement effects.

F1 through F6 are based on experiences of general SPI activities performed in industry, and can be considered as universal factors influencing the success rate of an SPI activity.

1.3. Requirements Engineering SPI - Critical Factors

In addition to the general SPI factors presented above there are experiences in industry from SPI efforts conducted with focus on RE in particular. Some of these are presented below in order to address specific issues pertaining to RE, and to complement the general SPI factors presented above.

1.3.1. Coverage

One of the main issues regarding SPI targeted at specifically RE is *coverage*. Using prescriptive frameworks with a set number of practices there is a risk that the area of RE is covered in an unsatisfactory manner, i.e. only addressed in very broad strokes [58]. This is of course not surprising as many prescriptive model-based frameworks like the ones described earlier in Section 1.1.2 are general in nature, having to cover a wide area of practices of which RE is only a small part. The situation is somewhat better looking at CMMI. First, there is a possibility to target specific areas explicitly, e.g. RE (this applies to SPICE as well). Second the coverage of RE is improved compared to e.g. CMM and SPICE [36, 40, 59]. However, compared to RE specific best practice models like e.g. the ones presented by Sommerville & Sawyer in their practice guide [9, 60], the area is still marginalized [58].

There has been RE specific prescriptive SPI frameworks presented as a way to address the problem of coverage, the most well-known one probably being a part of the REAIMS project [9, 61].

Using inductive frameworks, e.g. QIP, the coverage is based on experiences of the organization (its constituents) and the SPI work group, thus the knowledge of these people is the basis for the coverage.

1.3.2. Requirements Engineering and Project Focus

An underlying issue complicating SPI efforts targeted towards RE is the fact that RE more and more transcends projects in an organization as market-driven product development is becoming increasingly commonplace in software industry [2, 10, 11].

This basically implies that the situation is far more complex than the one presented by the classical bespoke [2] development situation where the development activity was initiated by e.g. an order, which generally resulted in a project (maybe preceded by a pre-study) and then RE was initiated through this project. As RE in this case is project initiated most RE work is performed within the project (barring e.g. a pre-study).

As illustrated in Figure 1-4, the situation in a market-driven development situation is different since the requirements flow is not limited to a development instance (e.g. a project), but rather continuous in nature, and the requirements themselves act as the catalyst for initiating development.

The nature of a process area (RE) impacts on SPI as many SPI frameworks are project centered in nature. This can be seen in the case of e.g. QIP, where projects are the main testing ground for improvements. Prescriptive SPI frameworks, e.g. CMM and CMMI are also focused on assessing and improving the development process (which mainly takes place within projects) [62].

Requirements engineering is not limited to projects but also a part of e.g. product management activities [63, 64].

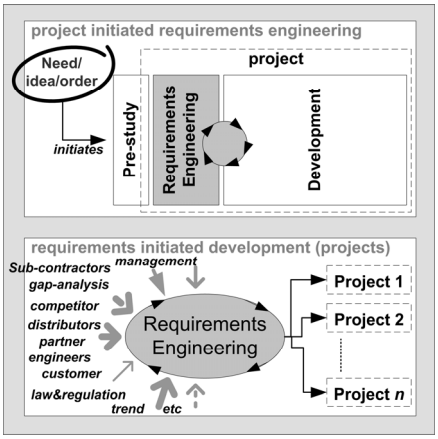


Figure 1-4. Requirements' role in the development process.

1.3.3. Commitment and Perceived Benefit

The perceived benefit of an SPI activity influences the outcome. If the people affected by the improvement perceive the changes as meaningless or unnecessary, i.e. they do not see the value, they probably will not support the work being conducted. Thus effectively lowering the chance for the improvement to actually become a part of a new and improved process [51, 53]. This also impacts the willingness to commit and participate in an SPI activity [9].

This is connected to the general SPI factor described in Section 1.2.2 (F4) as involvement of the people affected by the improvements in the SPI work probably will increase the chances of seeing the improvements as beneficial as the coworkers are a part of suggesting them. The reasoning of increasing perceived value through involvement in the SPI process seems applicable if using an inductive framework like QIP, but prescriptive frameworks are another matter. As prescriptive frameworks imply adopting certain predetermined practices the ability of involved coworkers to influence the improvements is limited. The benefit of involvement in this case is more of an implicit

nature, i.e. relying on that if people understand the improvements they will be more inclined to se them as beneficial.

1.3.4. Quantification and Measurement

As stated in Section 1.2.3 the ability to measure SPI success is considered important. Looking at requirements engineering quantification of process improvement results is difficult [9, 16, 51]. This in particular applies in the case of SMEs as having good long-term measurement programs in place is costly [50]. Using e.g. GQM (as a part of QIP) would therefore demand additional resources and time be committed to the SPI activity, a proposition that effectively would raise the SPI initiation threshold described in Section 1.2.1.

In the case of prescriptive frameworks the "measurement" is not reliant on metrics to the same extent, but on compliance comparisons (although compliance to the practices can be measured using metrics).

1.3.5. Summary of Requirements Engineering SPI - Critical Factors

A summary of the requirements engineering specific factors gives the following list.

- REF1: Coverage (how well covered is the area of RE in SPI frameworks).
- REF2: Project transcendence (RE is neither a project initiated nor a project limited activity, nor is it just a part of the development).
- REF3: Commitment and Perceived benefit.
- REF4: Quantification and Measurement.

1.4. Contribution and Outline of the Thesis

The research in this thesis is presented through four papers, each presented in a separate chapter. In this section each paper is presented briefly and related to the previous sections, in attempt to clarify the contribution of each paper. (Information pertaining to authors and publication can be viewed initially in each chapter).

It is important to realize that the research presented through the papers below is not intended to replace or even compete with any existing SPI framework. The motivation is to complement existing frameworks regarding assessment and improvement. As such no complete new SPI framework is presented, but rather complementary parts that can be used as a part of an SPI effort. The contribution is mainly in the fact that the methods below are developed from the perspective of requirements engineering, taking both general and RE specific SPI critical factors into account.

Chapter Two (Paper I) - Model-based Process Assessment

Introduction and Application of a Lightweight Requirements Engineering Process Evaluation Method

Paper I introduces a prescriptive (model-based) process assessment framework inspired by the structure of CMM, and RE specific process areas identified in literature (e.g. the REAIMS REPG [9]). The assessment framework is called REPM (Requirements Engineering Process Maturity). The process areas contained in the REPM model are a collection of best practices for requirements engineering and they are the basis for the assessment methodology.

The reasoning behind the REPM model was centered on offering a "snapshot" of the RE status in projects through an assessment method that was easy-to-use and low-cost, e.g. comparable to CMMI assessment Class C (see Section 1.1.2.2) but targeted at RE specifically. The aim was to give practitioners in industry a "tool" that could give an overview of the RE process pertaining to a certain project with a minimum of effort and time, the reasoning was that it was better to perform fast (but maybe not completely accurate) assessments than no assessments at all.

The product of an REPM model assessment is a maturity measure of the RE process in a specific project graded on a scale from 1 to 5 (like CMM/CMMI Staged Representation). The results of a snap-shot assessment can be used as decision support material for further assessment and/or improvement activities.

A fundamental difference between the REPM model and e.g. CMM is the REPM models built-in *model-lag* feature. Model-lag was devised as a way to take model inapplicability into account (if practices in the model did not make sense in a certain project this would be taken into consideration). This effectively made the model (under certain circumstances) adapt to the situation at hand in the project being assessed, as opposed to the normal situation of model-based assessments were the practices of the model are non-negotiable.

The REPM model was validated in industry through an evaluation, and subsequently through four industry trials (assessments) conducted in Sweden (two companies) and in Ireland (two companies).

The second objective of the paper was to give an idea of the problem scope pertaining to requirements engineering practices in industry, i.e. get an indication of the state-of-practice in industry to some extent. A fast and low-cost assessment framework like the REPM model offered the possibility for this. The results from the four assessments indicate a lacking in process maturity in primarily the area of Requirements Management in the assessed projects.

Paper I - main contributions in comparison to critical factors:

- Collection of best practices for RE (primarily bespoke development). Addressing primarily:
  - REF1: Coverage (how well covered is the area of RE in SPI frameworks).
- Development and validation of an assessment (process maturity measurement) framework which is easy to use and low-cost. Addressing primarily:
  - F2: Cost/Resources/Size (the nature of large SPI frameworks implies commitment of much resources over an extended period of time regarding both assessment and the actual improvement).

- F3: Commitment to SPI by management.

Chapter Three (Paper II) - Inductive Process Assessment

Identification of Improvement Issues Using a Lightweight Triangulation Approach

Paper II introduces an inductive assessment framework that uses data point triangulation to identify potential improvement suggestions (issues that are lacking/need to be addressed in the process). In contrast to the REPM model (Paper I) the assessment approach is aimed at finding improvement proposals, i.e. prepare for a process improvement activity, not give a maturity measurement of the RE process according to a predefined set or practices.

Data point triangulation in the context of the assessment implies use of multiple data sources and methods (interviews and documentation) from multiple views (project and line organization) in order to identify and confirm (triangulate) improvement proposals. The result of the assessment is an in-depth evaluation of the current state-of-practice regarding RE in an organization, as well as a list of tangible improvement proposals which can be used as input to an improvement activity. In addition, these proposals are not based on solely project assessment (line organization is also targeted).

As the assessment is inductive in nature the knowledge, views and experiences of the constituents (e.g. management, developers and so on) are used as a basis for the assessment. I.e. the people in the organization whose RE process is to be improved are involved and active in the assessment and formulation of the improvement issues.

The assessment framework was used in industry on two separate occasions to identify improvement issues, of which one is presented in the paper. In spite of the in-depth nature of the assessment the resources needed for an assessment was relatively low, e.g. 180 person-hours in total. The price-tag was an issue as the assessment framework was created with primarily SMEs in mind.

Paper II - main contributions in comparison to critical factors:

- The use of multiple data sources from multiple views. Addressing primarily:
  - REF2: Project transcendence (RE is neither a project initiated nor a project limited activity, nor is it just a part of the development).
- The inductive nature of the assessment. Addressing primarily:
  - REF3: Commitment and Perceived benefit.
  - F4: Commitment to SPI and Involvement in the SPI work by coworkers.
- Triangulation produces confirmed tangible improvement issues as input to an improvement effort. Addressing primarily:
  - F5: Focus (on the SPI effort with clear and well-defined goals).
- Development and validation of an assessment framework which produces input to improvement efforts using a moderate amount of resources. Addressing primarily:
  - F2: Cost/Resources/Size (the nature of large SPI frameworks implies commitment of much resources over an extended period of time regarding both assessment and the actual improvement).

- F3: Commitment to SPI by management.

Chapter Four (Paper III) - Process Improvement (Pre-) Planning

Packaging Software Process Improvement Issues – A Method and a Case Study

Assessment is one important step in process improvement. However, given that a list of improvement proposals has been derived, it is often very important to be able to prioritize the improvement proposals (obtained from an assessment, see Paper II) and also look at the potential dependencies between them. Paper III presents a method, called DAIIPS (Dependency Adherent Improvement Issue Prioritization Scheme), intended for prioritization and identification of dependencies between improvement proposals. The prioritization part of the method is based on a multi-decision criteria method and the dependencies are identified using a dependency graph.

The developed method has been applied in industry, where people with different roles applied the method. The paper presents both the method as such and the successful application of it in industry.

Studying DAIIPS the first and foremost motivation for the development of the scheme was to give organizations with limited resources for SPI (e.g. SMEs) a chance to choose what to do first based on their needs. This is made explicit as the personnel affected by the subsequent improvements prioritize the improvement proposals.

The dependency mapping is performed in order to ascertain important dependencies between improvements proposals as this in combination with the priorities is the basis for implementation order.

DAIIPS assists in structuring improvement proposals into SPI packages. As these packages represent a delimited and prioritized number of proposals this helps focus the SPI effort on a delimited number of improvement proposals at a time (i.e. an SPI package). This is important as it offers clear and well-defined set of goals that are obtainable with regards to resources and time that have to be committed, keeping improvement iterations shorter and delimited.

As such DAIIPS is not dependent on a specific assessment method be used, as long as the output from the assessment consists of delimited improvement suggestions that can be prioritized, dependency mapped and package according to DAIIPS. In the same way DAIIPS is largely independent to what improvement framework be used post-assessment, as long as it can use the DAIIPS SPI packages as input. Given these limitations DAIIPS could be used as a part of any pre-planning activity conducted as a part of an SPI effort.

Paper III - main contributions in comparison to critical factors:

- The involvement of coworkers with different roles in the improvement preparation. Addressing primarily:
  - REF3: Commitment and Perceived benefit.
  - F4: Commitment to SPI and Involvement in the SPI work by coworkers.
- Packaging of improvement proposals into SPI packages. Addressing primarily:
  - F1: Time (long-term work, long-term gain).

- F2: Cost/Resources/Size (the nature of large SPI frameworks implies commitment of much resources over an extended period of time regarding both assessment and the actual improvement).
- F3: Commitment to SPI by management.
- F5: Focus (on the SPI effort with clear and well-defined goals).

Chapter Five (Paper IV) - Improvement Implementation

Requirements Abstraction Model

Following assessment (Paper II) and packaging of improvement proposals into SPI packages (Paper III) an improvement effort was initiated aimed to address the first SPI package. Paper IV reports the result of this process improvement activity through the presentation and validation of the RAM (Requirements Abstraction Model).

The Requirements Abstraction Model was designed to handle requirements coming in from multiple stakeholders on multiple levels of abstraction in a continuous market-driven product development situation. It supports primarily product management with the work-up (abstraction and break-down) of requirements in a structured way, offering abstract requirements comparable to product strategies, as well as refined testable requirements as input to development.

The RAM is based on the notion of requirements initiated development (i.e. projects are initiated by requirements, not vice versa, see Figure 1-4). In addition it offers a multiple abstraction level repository for requirements as opposed to putting all requirements into one repository ignoring abstraction level.

All parties working with development (management in particular) can compare requirements, as they are homogenous regarding abstraction level, and are not forced to decide between an abstract requirement and a detailed one as e.g. planning and prioritization activities are performed.

The RAM can be tailored to satisfy the needs of different organizations and products. In the same way, it can be modified over time to reflect the current situation of a development organization, supporting the collection of requirements over time and product life cycle.

As a part of the model's development it was initially validated in industry with professionals working with requirements as well as management (static validation). Subsequent to this the RAM was piloted in industry (dynamic validation). The static validation was primarily aimed at validating and refining the model in the initial stages of its development. The dynamic validation's primary function was to test the model in a real industry environment.

Paper IV - main contributions from an SPI perspective is (in comparison to critical factors):

- The dynamic validation (pilot) made it possible to evaluate the model prior to its implementation in the organization. This validation yielded primarily qualitative data. Addressing primarily:
  - F6: Measurability of improvement effects.
- The involvement of upper and middle management, product managers and developers in the validation (development) of the RAM ascertained that relevant

feedback regarding both the model (e.g. usability), and the work products (requirements) was collected and taken into consideration. Addressing primarily:

- REF3: Commitment and Perceived benefit.
- F4: Commitment to SPI and Involvement in the SPI work by coworkers.

Paper IV - main contributions from an RE perspective are:

- The introduction of a model that handles and supports requirements on multiple levels of abstraction.
- A structured way to specify, abstract and break-down requirements, producing requirements comparable to product strategies as well as requirements suited as input to a development effort.
- Through using (and offering a repository for) requirements on multiple abstraction levels the RAM gives all parties working with development (management in particular) a way of working that supports comparison of requirements. This is a prerequisite for planning and prioritization activities.

1.5. Research Approach

This section outlines the research approach for this thesis, effectively taking a step back in regards to the papers and their contribution described previously in Section 1.4. The structure of this section is as follows. Section 1.5.1 presents the scope of the research, i.e. exploring the limits of the research presented in Paper I through IV. In Section 1.5.2 the research questions, the basis for the research and contributions, are presented and elaborated upon. Last, Section 1.5.3 presents an overview of the research methods utilized to obtain answers to the research questions.

1.5.1. Research Scope

In this thesis software process assessment and improvement is studied in the context of requirements engineering. The research can be divided into three main parts,

- Process Assessment (Paper I and Paper II)
- Process Improvement pre-planning (Paper III)
- Process Improvement (Paper IV).

Each of these parts can be placed in relation to their individual focus and area of contribution. Figure 1-5 illustrates a generic SPI scheme of sorts (from Figure 1-1, see Section 1.1), but the figure has been augmented with the papers presented above.

Paper I provides an assessment method for fast and easy measurement of the RE process. This can probably not be seen as a thorough assessment, but as a way to get an initial indication of what (if any) areas of the RE process needs to be looked at. This indication can then be a part of the decision support material on which commitment to an in-depth assessment and subsequent improvement effort is based. The limitations of this assessment are that it is not thorough enough to be viewed as an assessment on which improvements can be based, as only practices present in the model are assessed.

However, this is the strong-suite of the model as well since it does not require much resources or time, i.e. there does not have to be any real management commitment to perform the REPM assessment.

The practices in the REPM model as presented in Paper I is primarily targeted at bespoke project evaluation, i.e. the implication being that practices typical for e.g. market-driven development projects may not be fully covered by the practices.

Paper II is an assessment framework that is used to produce improvement proposals. As it demands more time and resources in comparison to the assessment presented in Paper I, management commitment can be considered a prerequisite for the initiation of this activity.

The Paper II assessment produces tangible and relatively well defined improvement proposals (called improvement issues in the paper). The reasoning is for these to be used as input to an SPI planning activity.

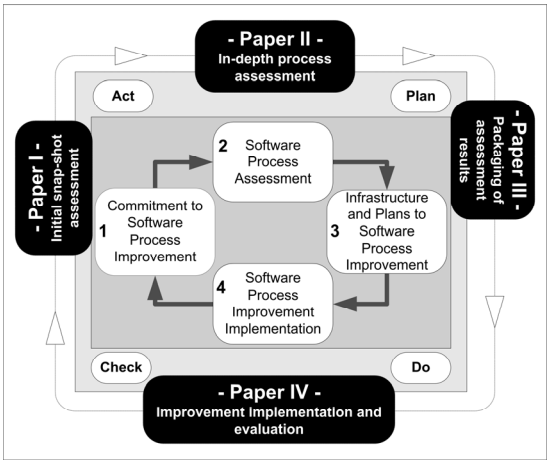


Figure 1-5. Placement of research.

Paper III presents a method that can be used in the initial stages of a planning activity (or even before depending on preference), intended for prioritization and identification of dependencies between improvement proposals. The DAIIPS does not outline a full fledged planning effort, but rather a structured way to choose between proposals and establish what to do first. The DAIIPS produces SPI packages that can be delivered to a detailed planning effort.

Paper IV deals with an actual implementation of an SPI package delivered from a DAIIPS effort. It deals with how an RE model was designed, validated and tested in industry to tackle several improvement proposals deemed of high priority. The improvement implementation presented here gives a hands-on report of how an implementation can be performed, and validated through a pilot project.

Looking at the RAM (Paper IV) the RE specific contribution of the model can be seen as giving structure and supporting product managers in a market-driven continuous product centered development situation. Model v.1.0 (presented in the paper) does however not have support for requirements prioritization and packaging, features that are upcoming in the next release of the model.

1.5.1.1 General Scope

There are several common denominators shared by all papers. Time is one, indicating time to return on investment (ROI), i.e. how long an improvement activity must run before any result is seen. This is reflected in the fact that the assessments presented both are relatively (compared to what they produce) fast to perform.

The time aspect is illustrated further through the DAIIPS effort, as the very nature of SPI packages is to limit the time to ROI, i.e. time and resources that have to be committed, as the improvements are limited to one package at a time (or any amount deemed manageable by the organization subject to SPI). Further, this time awareness also surfaces in Paper IV, as the improvement is based on an SPI package, and the subsequent validation of the improvement is done through a pilot project in a qualitative way, i.e. not demanding extensive and costly measurement practices to be in place before and after the improvement.

Cost is another common denominator indicating a high level of cost awareness during the development of the SPI methods/frameworks presented in papers I through IV.

The general benefit of this is to lower the SPI initiation threshold for organizations by letting them perform assessments, improvements and subsequently evaluate the improvements within cost and time frames that are manageable for small and medium sized enterprises.

In total the research presented does not constitute a complete SPI framework as such as some parts are missing, primarily centered on planning activities for SPI. The focus was put on isolated areas within SPI that were deemed in need of study and where SMEs could benefit from the development of new methods (this can be seen in Section 1.4 where each paper is discussed and compared with SPI critical factors).

A second aspect of the methods/frameworks developed is that they are to some extent modular, i.e. can be used in combination (or as an addition to) already established SPI frameworks like QIP and CMMI. The assessment method presented in Paper I can be used as input to e.g. a QIP SPI effort. DAIIPS can be used post-assessment regardless of assessment method used (as long as the assessment produces the needed input for DAIIPS).

As such, modularity came from the wish to add to the field of SPI pertaining making it easier for SMEs to commit to SPI by giving them "tools" that could (to some extent) be used in combination with already present frameworks.

1.5.2. Research Questions

The research questions posed driving the research can be seen in terms of *general* and *specific*. The general research question, which is more of a goal, sets the direction of the research conducted and can be formulated as the following:

How SPI efforts, targeted at RE, in organizations with limited time and resources can be performed.

This establishes a frame of reference within which more specific research questions can be formulated. The frame of reference relates to the delimitations set up and presented in the previous section, e.g. focus on applicability in SMEs taking factors such as resources and time as important considerations. This initially led to the research question:

How can an organization's RE process be evaluated against a set of best-practices establishing process maturity, with the main focus being assessment performance?

The primary goal of the model-based assessment presented in Paper I was to offer a structured way to establish a status overview of an RE process in a company as fast and

cost effective as possible (through the evaluation of one or several projects). This was accomplished through the creation of the REPM model and based on it an assessment method. The assessments using this model were performed against a set of best practices resulting in that assessment performance was high, i.e. low cost and short time assessments were possible. The results from the assessments could be used as indications as to the need for process improvement activities.

As a step towards refining process assessment to produce specific and tangible improvement proposals (what should be fixed) based on organizational experience and knowledge another research question was posed:

How can specific requirements engineering process improvement proposals be elicited from an organization utilizing experiences and knowledge inherent to the organization as a whole, and at the same time maximizing assessment performance and accuracy?

Assessment performance was still an important consideration, but so was utilizing organizational knowledge (as opposed to following a model's prescriptions) in establishing what needed to be addressed in an improvement effort. The assessment method presented in Paper II looks beyond project scope, i.e. eliciting information from the entire organization, over (several projects) and beyond (looking at project and line organization) project boundaries. Assessment accuracy is improved through data point triangulation. The product of an assessment is a list of improvement proposals that can be used as input to an improvement effort.

As a part of the philosophy of minimizing cost and time the need for prioritization of the improvement suggestions was identified, as a way to help SMEs establish an explicit order of improvement suggestion implementation, i.e. doing the most important thing first. This led to the formulation of a new research question:

How can SPI packages be created in a structured way, based on the organizations experience-base, establishing an implementation order taking priority and dependencies between issues into account?

Paper III presents a framework for dependency adherent improvement suggestions prioritization and packaging. Through the creation of SPI packages an implementation order was established, and furthermore the improvement effort could be focused on a delimited number of improvements at a time.

SPI package one consisted of three improvement suggestions, of which "abstraction level and contents of requirements" was the primary to be addressed. This gave rise to the research question:

How are requirements on varying levels of abstraction specified and refined in a market-driven product development environment?

Paper IV presents the Requirements Abstraction Model (RAM), aimed at supporting primarily product managers in taking care of and working with requirements coming in continuously from multiple stakeholders, and on varying levels of abstraction.

1.5.3. Research Methods

In the pursuit of answers to the research questions posed there was a need to utilize certain research methods. These methods will be listed in order of appearance (Paper I

through IV) below, but initially the general research environment is illustrated to give context to the work performed.

1.5.3.1 Industry Motivated Research - Industry as Laboratory

The issue of technology transfer from research to industry can be and is a real problem. Among the main issues presented by Glass [25] can be described as the following: problems are formulated in academia, the subsequent generation of solutions[2] are put forward in academia, and this solution is subsequently validated in academia - resulting in industrial inapplicability. Stated in this manner this may not be completely surprising. The main point being that there can be no real benefit to industry if the research conducted in software engineering is not based on problems identified in industry, and if the proposed solutions are not tested in an industrial environment prior to "release" [25].

Technology transfer problems have also been identified explicitly in requirements engineering research [24]. Some of the problems highlighted by Glass, i.e. research being isolated from industry, are mentioned, however, issues such as commitment to change, resistance to change, and involvement by engineers in the incorporation of new technology in the form of e.g. models is considered just as crucial (there are clear connections to the factors described in Sections 1.2 and 1.3).

The research presented in this thesis is based on the principle of using industry as a laboratory, the process is illustrated in Figure 1-6. This implies that the main research direction and subsequent research questions formulated (see Section 1.5.2) are based on problems and issues explicitly identified in industry, i.e. the research is industry motivated. Subsequent to the identification of a problem, state-of-the-art (research in academia) is studied in order to see if there already exist one or several solutions (e.g. the REPM model in Paper I was inspired by CMM and REAIMS). The outcome of this activity is, in the worst case scenario, establishing a base of knowledge in the problem area, and in the best case scenario finding a solution (or more commonly inspiration towards a solution) in other research.

A solution is formulated based on the understanding of the problem and the results of the academia study (and through invention to some extent). Subsequent to formulation of the solution there is often an initial validation in industry, usually followed by a modification of the solution based on this validation (e.g. the static validation of the RAM in Paper IV). The final stage is the live dynamic validation of the solution in industry carried out as e.g. a pilot project to validate real applicability (see e.g. the dynamic validation of the RAM in Paper IV). The results of the dynamic validation generally give feedback as to the solutions viability, which can be used for refinement.

[2] The use of the word "solution" in this context is not to be seen in the sense of a definitive "fix" or "a silver bullet" to a identified problem, but rather as a word summarizing "contribution", "research result" and "research product".

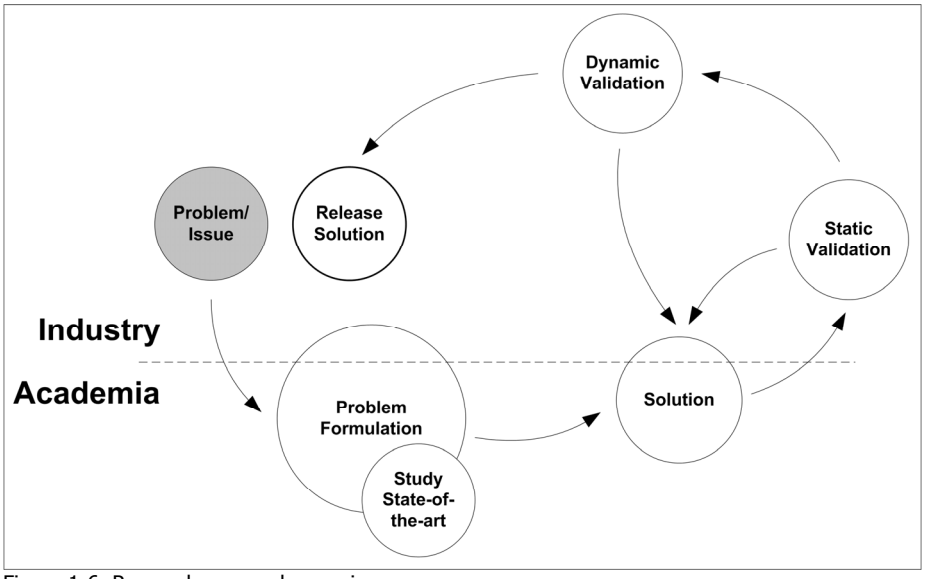


Figure 1-6. Research approach overview.

However, there are issues in using industry as a lab. Working in closer cooperation with industry can threaten the generalizability [65] of the solution, as a scenario of "one study - one case" can give specific research results, i.e. not really usable for organizations in general. By recognizing this threat actions can be taken to avoid issues of this sort. Examples can be formulating research questions and studies in a general way, studying a problem in academia (through e.g. experience reports etc.) can give an indication if the problem is general, and thus taking one step closer to developing a solution that is general, but validated against a specific case.

Another way to avoid problems of this sort is to test (static/dynamic validation) on more than one case, i.e. having a second organization to cooperate with. This is the case pertaining to the research presented in Papers I, II and IV.

Below a list of the papers and a specification of the industry cooperation:

- The REPM assessment presented in Paper I was developed in cooperation with four companies.
- The triangulation assessment method (Paper II) was developed in cooperation with Danaher Motion Särö AB, and later validated at Danaher Motion Särö AB. The evaluation was later also performed at ABB Automation Technology Products AB (replication).
- DAIIPS (Paper III) was developed and validated in cooperation with Danaher Motion Särö AB.
- The RAM (Paper IV) was developed and validated in cooperation with Danaher Motion Särö AB. In addition the RAM will be tested in a pilot project at ABB Automation Technology Products AB, planned for the fall of 2004.

The benefits of working with research in close cooperation with industry generally outweighs the potential drawbacks, as industry professionals add to the value of the solutions in a tangible way through their input and feedback. In addition, as the solution is designed based on tangible problems inspired by industry the chance for future application in industry increases.

1.5.3.2 Utilized Research Methods

The main method used throughout this thesis is based on flexible design research strategy, namely case studies [66]. This implies 'development of a detailed, intensive knowledge about a single case, or of a small number of related "cases"' [67]. For the purposes of this thesis the following distinction is made to clarify the paper specific situations:

- *Macro-case* implies the overall case, i.e. if a study of three projects is performed at *one* company, the company is the macro-case (i.e. one macro-case in this example).
- *Case* implies the individual cases studied within a macro-case, i.e. if three projects are studied in one company (macro-case) the result is *one* macro-case and *three* cases (projects) studied.

The assessments presented in Paper I consisted of the study of four macro-cases and four cases (four companies and one project was studied in each instance). The collection technique utilized was structured interviews [67] via a predefined question form.

Paper II presents an assessment method that is designed to use multiple cases (projects) as input, as well as looking at the single "case" of the line organization within one macro-case, utilizing collection techniques such as semi-structured interviews and document analysis [67]. As this study was replicated the assessment method presented in Paper II was tested on two macro-cases.

Paper III presents the study of one macro-case (i.e. an SPI pre-planning effort at one company) where the main techniques utilized consisted of workshops and questionnaires in combination to obtain the results.

Paper IV presents validation of an RE model in industry through one macro-case. The static validation part of the study utilized unstructured interviews as a primary collection technique, and the dynamic validation consisted of testing the model in a pilot project.

In summary, the research design strategy utilized was case study research, and the main collection methods were interviews (structured, semi-structured and unstructured), the use of questionnaires, workshops, and pilot projects.

1.6. Future Research

There is a real need to actively address the issues of providing models and methods to enable SMEs working with development to improve. There are two parts of this. Process evaluation (fast and relatively accurate models), and improvement planning (answer what to do first and how) tools are one part of this. Assessment and planning are a prerequisite for any SPI effort not using a prescriptive approach. Papers I through III in this thesis address these issues by the introduction and initial validation (test) of

assessments and planning methods. However, there is a need to continue this effort, replicating the studies and through this validation improve on the methods presented. The result of this continuation of the research presented in papers I through III should be to add to the methods that SMEs can choose from in their attempts to improve.

An important aspect that needs further research is measurement of improvement success in general and in the area of RE. High cost long time extensive measurement programs and gathering of metrics have to be adapted to fit the situation of SMEs.

The other part of enabling improvement for SMEs is the development of better RE methods and models suited for SMEs. Looking at the area of market-driven continuous RE (specifically addressed in Paper IV) there is a need to enable SMEs (but maybe even larger enterprises) to handle the stream of "requirements" coming from multiple sources in all shapes and forms (varying levels of abstraction and refinement). This not only calls for a structured way in which the requirements are taken in and refined, covered partly in Paper IV, but also the ability to prioritize and package requirements at an early stage before sending them to (initiating) development projects that are to implement the requirements in question.

Specifically this implies several research activities planned to be the continuation of the research presented in this thesis. The activities can be summarized as following:

- Incorporation of the RAM in increments at Danaher Motion Särö AB (making it part of the RE process), evaluating each increment before continuing with the next. This will not only give valuable results as to the implementation/improvement aspect (i.e. SPI), but also feedback regarding the model itself addressing issues such as scalability, and thus giving a chance to improve the RAM.
- Replication of the study presented in Paper IV at a second company involving tailoring the RAM to a new product/organization and validating the RAM in a new environment through a second pilot project. Based on this work the RAM can be improved upon, and the notion of general applicability of the RAM will be strengthened.
- Evolvement of the RAM to include new parts (addressing e.g. SPI package two and three in Paper III), and validating the new parts in industry at e.g. Danaher Motion Särö AB. Examples of the new parts are: incorporation of prioritization schemes that take customer (market) priorities as well as internal (strategic product management) priorities into account. Furthermore, the packaging of requirements into development packages (used to initiate a project) based on the formerly mentioned priority and with regards to interdependencies [12, 68, 69] between requirements, will be addressed.

The objective is to develop a model that enables organizations to handle requirements in a continuous manner, addressing the market-driven incremental nature of product development.

1.7. Conclusions

There is a need for SPI in RE. This not only implies that the RE processes in organizations are in need of improvement, but also that research has to present models

and tools that makes SPI a realistic and possible endeavor for all organizations, even the ones with limited time and resources. The assessment methods and planning scheme presented in this thesis were developed to address this, and to some extent add to the choices given to SMEs faced with the need for improvement.

Papers I and II present assessment methods that enable organizations to assess and produce tangible improvement proposals that can be used as an input to an SPI planning activity. At the same time assessment performance (low cost, fast and relatively accurate) was considered, as well as other critical factors reported from industry experience with assessments, e.g. the involvement of the organization's constituents in the SPI effort to ensure commitment and that several views were considered.

Paper III presents a method for structured and dependency adherent prioritization and packaging of improvement proposals into SPI packages, enabling an organization with limited resources to explicitly choose what is to be done first, based on their own knowledge and experience of what is critical.

Paper IV presents an SPI implementation initiated to address a select number of improvement proposals (SPI package one, Paper III) identified during assessment (Paper II). The development of an RE model is described, as well as a validation of said model in industry. Thus, the research presented above to some extent completes an iteration of the general SPI steps presented previously (see Figure 1-5), introducing methods to aid SMEs in "evaluation of the current situation", "plan for improvement", "implement the improvements", and "evaluate the effect of the improvements" respectively.

In addition to enabling SPI, there is a need for RE models that are appropriate for SMEs faced with a market-driven product centered development situation. The model presented and validated in Paper IV addresses some of the aspects of this. The RAM was developed to support primarily the product management effort, recognizing that RE from this perspective is not project initiated but rather project initiating. The model assists product managers to take requirements on varying abstraction levels and refining them to the point of being good-enough to offer a base for decision support for management, and at the same time being good enough for project initiation.

The continued refinement and evolvement of the Requirements Abstraction Model will result in the creation of an RE management and development tool aimed at SMEs. The continued validation of the RAM in industry will assess the models applicability and give valuable feedback regarding both benefits and (equally important) limitations.

1.8. References

1. Kotonya G, Sommerville I (1998) Requirements Engineering: Processes and Techniques. John Wiley, New York.
2. Sommerville I (2001) Software Engineering. Addison-Wesley, Essex.
3. Davis AM (1993) Software Requirements: Objects, Functions, and States. Prentice Hall, Englewood Cliffs NJ.
4. Bray IK (2002) An Introduction to Requirements Engineering. Addison-Wesley, Dorset.
5. Glass RL (1998) Software Runaways. Prentice Hall, Upper Saddle River NJ.
6. Connolly TM, Begg CE (1998) Database Systems: A Practical Approach to Design, Implementation and Management. Addison-Wesley, Harlow.
7. Jirotka M, Goguen JA (1994) Requirements Engineering Social and Technical Issues. Academic Press, London.
8. van Buren J, Cook DA (1998) Experiences in the Adoption of Requirements Engineering Technologies. Crosstalk - The Journal of Defense Software Engineering 11(12):3-10.
9. Sommerville I, Sawyer P (1999) Requirements Engineering: A Good Practice Guide. Wiley, Chichester.
10. Ruhe G, Greer D (2003) Quantitative Studies in Software Release Planning under Risk and Resource Constraints. In Proceedings of International Symposium on Empirical Software Engineering (ISESE), IEEE, Los Alamitos CA, pp. 262-271.
11. Butscher SA, Laker M (2000) Market-Driven Product Development. Marketing Management 9(2):48-53.
12. Carlshamre P (2002) Release Planning in Market-Driven Software Product Development: Provoking an Understanding. Requirements Engineering 7(3):139-151.
13. CMMI-PDT (2002) Capability Maturity Model Integration (CMMI), Version 1.1. CMMI for Systems Engineering, Software Engineering, Integrated Product and Process Development, and Supplier Sourcing Version 1.1 (CMMI-SE/SW/IPPD/SS, V1.1).
14. (2004) http://www.sei.cmu.edu/cmm/cmm.html. Last Accessed: 2004-04-11.
15. Zahran S (1998) Software Process Improvement: Practical Guidelines for Business Success. Addison-Wesley, Reading.
16. Calvo-Manzano Villalón JA, Cuevas Agustín G, San Feliu Gilabert T, De Amescua Seco A, García Sánchez L, Pérez Cota M (2002) Experiences in the Application of Software Process Improvement in SMEs. Software Quality Journal 10(3):261-273.
17. Kuilboer JP, Ashrafi N (2000) Software Process and Product Improvement: An Empirical Assessment. Information and Software Technology 42(1):27-34.
18. Reifer DJ (2000) The CMMI: It's Formidable. Journal of Systems and Software 50(2):97-98.

19. El Emam K, Goldenson D, McCurley J, Herbsleb J (2001) Modeling the Likelihood of Software Process Improvement: An Exploratory Study. Empirical Software Engineering 6(3):207-229.
20. Rainer A, Hall T (2003) A Quantitative and Qualitative Analysis of Factors Affecting Software Processes. The Journal of Systems and Software 66(1):7-21.
21. Herbsleb J, Zubrow D, Goldenson D, Hayes W, Paulk M (1997) Software Quality and the Capability Maturity Model. Association for Computing Machinery. Communications of the ACM 40(6):30-40.
22. Herbsleb JD, Goldenson DR (1996) A Systematic Survey of CMM Experience and Results. In Proceedings of the 18th International Conference on Software Engineering, IEEE, Los Alamitos CA, pp. 323-330.
23. Conradi R, Fuggetta A (2002) Improving Software Process Improvement. IEEE Software 19(4):92-100.
24. Kaindl H, Brinkkemper S, Bubenko Jr JA, Farbey B, Greenspan SJ, Heitmeyer CL, Sampaio do Prado Leite JC, Mead NR, Mylopoulos J, Siddiqi J (2002) Requirements Engineering and Technology Transfer: Obstacles, Incentives and Improvement Agenda. Requirements Engineering 7(3):113 - 123.
25. Glass RL (1994) The Software-Research Crisis. IEEE Software 11(6):42-47.
26. Shewhart WA, Deming WE (1986) Statistical Method from the Viewpoint of Quality Control. Dover, New York.
27. Basili VR (1985) Quantitative Evaluation of Software Methodology. University of Maryland, College Park, Maryland.
28. El Emam KE, Madhavji NHE (1999) Elements of Software Process Assessment & Improvement. Wiley-IEEE, Los Alamitos CA.
29. Basili VR (1993) The Experimental Paradigm in Software Engineering. In International Workshop on Experimental Software Engineering Issues: Critical Assessment and Future Directions, Lecture Notes in Computer Software, Springer, Dagstuhl, pp. 3-12.
30. (2004) http://herkules.oulu.fi/isbn9514265084/html/x287.html. Last Accessed: 2004-04-11.
31. Basili VR (1992) Software Modeling and Measurement: The Goal Question Metric Paradigm. Computer Science Technical Report Series, Cs-TR-2956 (UMIACS-TR-92-96). University of Maryland, College Park, Maryland.
32. Fuggetta A, Lavazza L, Morasca S, Cinti S, Oldano G, Orazi E (1998) Applying GQM in an Industrial Software Factory. ACM Transactions on Software Engineering and Methodology 7(4):411-448.
33. Briand L, El Emam K, Melo WL (1995) An Inductive Method for Software Process Improvement: Concrete Steps and Guidelines. In Proceedings of ESI-ISCN'95: Measurement and Training Based Process Improvement, ISCN, Vienna, pp. xx-xxx.
34. Paulk MC, Curtis B, Chrissis MB, Weber CV (1995) The Capability Maturity Model: Guidelines for Improving the Software Process. Addison-Wesley, Reading.
35. Paulk MC (1995) The Capability Maturity Model : Guidelines for Improving the Software Process. Addison-Wesley, Reading.

36. CMMI Product Developement Team (2002) Capability Maturity Model Integration (CMMI), Version 1.1. CMMI for Systems Engineering, Software Engineering, Integrated Product and Process Development, and Supplier Sourcing Version 1.1 (CMMI-SE/SW/IPPD/SS, V1.1).
37. McFeeley B (1991) IDEALSM: A User's Guide for Software Process Improvement. TR-CMU/SEI-92-TR004. Software Engineering Institute,
38. (2004) http://www.sei.cmu.edu/cmm/ipd-cmm.html. Last Accessed: 2004-04-11.
39. (1998) http://www.sei.cmu.edu/iso-15504/. Last Accessed: 2004-01-07.
40. Ahern DM, Clouse A, Turner R (2003) CMMI Distilled: A Practical Introduction to Integrated Process Improvement. Addison-Wesley, Boston.
41. PDT S (2001) Standard CMMI Appraisal Method for Process Improvement (SCAMPI) Version1.1 (CMU/SEI-2001-Hb-001). Carnegie Mellon SEI. Pittsburgh, PA.
42. CMMI-PDT (2001) Appraisal Requirements for CMMISM, Version 1.1 (ARC, V1.1).
43. El Emam K, Drouin J-N, Melo W (1998) SPICE: The Theory and Practice of Software Process Improvement and Capability Determination. IEEE, Los Alamitos CA.
44. (2003) http://www.sqi.gu.edu.au/spice/. Last Accessed: 2003-09-11.
45. Wiegers KE, Sturzenberger DC (2000) A Modular Software Process Mini-Assessment Method. IEEE Software 17(1):62-70.
46. (2003) http://sme.cordis.lu/home/index.cfm. Last Accessed: 2003-05-05.
47. Scott L, Jeffery R, Carvalho L, D'Ambra J, Rutherford P (2001) Practical Software Process Improvement - the IMPACT Project. In Proceedings of the Australian Software Engineering Conference, IEEE, Los Alamitos CA, pp. 182-189.
48. Kautz K, Hansen HW, Thaysen K (2000) Applying and Adjusting a Software Process Improvement Model in Practice: The Use of the IDEAL Model in a Small Software Enterprise. In Proceedings of the 2000 International Conference on Software Engineering, IEEE, Los Alamitos CA, pp. 626-633.
49. Laryd A, Orci T (2000) Dynamic CMM for Small Organizations. In Proceedings of the First Argentine Symposium on Software Engineering (ASSE 2000), Tandil, Argentina, pp. 133-149.
50. Basili VR, McGarry FE, Pajerski R, Zelkowitz MV (2002) Lessons Learned from 25 Years of Process Improvement: The Rise and Fall of the NASA Software Engineering Laboratory. In Proceedings of the 24th International Conference on Software Engineering (ICSE02), ACM, Orlando, pp. 69-79.
51. Jacobs S (1999) Introducing Measurable Quality Requirements: A Case Study. In Proceedings IEEE International Symposium on Requirements Engineering, IEEE, Los Alamitos CA, pp. 172-179.
52. Basili V, Green S (1994) Software Process Evolution at the SEL. IEEE Software 11(4):58-66.
53. Damian D, Chisan J, Vaidyanathsamy L, Pal Y (2003) An Industrial Case Study of the Impact of Requirements Engineering on Downstream Development. In Proceedings of the International Symposium on Empirical Software Engineering (ISESE), IEEE, Los Alamitos CA, pp. 40-49.

54. Johansen J, Jenden KO (2003) Did You Already Know? How Good Is an Organization at Identifying Its Own Strengths and Weaknesses? In Proceedings of European Software Process Improvement Conference (EuroSPI'2003), Verlag der Technischen Universität, Graz, Austria, pp. II.1-II.15.
55. Kitson DH, Masters SM (1993) An Analysis of SEI Software Process Assessment Results: 1987–1991. In Proceedings of the 15th International Conference on Software Engineering, IEEE, Los Alamitos CA, pp. 68 - 77.
56. Haley TJ (1996) Software Process Improvement at Raytheon. IEEE Software 13(6):33-41.
57. Mashiko Y, Basili VR (1997) Using the GQM Paradigm to Investigate Influential Factors for Software Process Improvement. The Journal of Systems and Software 36(1):17-32.
58. Sawyer P, Kotonya G (2001) Software Requirements. Guide to the Software Engineering Body of Knowledge SWEBOK, Trial Version 1.0. IEEE, Los Alamitos CA.
59. Wang Y, Court I, Ross M, Staples G, King G, Dorling A (1997) Quantitative Evaluation of the SPICE, CMM, ISO 9000 and Bootstrap. In Proceedings of the Third IEEE International Software Engineering Standards Symposium and Forum (ISESS 97), IEEE, Los Alamitos, pp. 57-68.
60. Sawyer P, Sommerville I, Viller S (1999) Capturing the Benefits of Requirements Engineering. IEEE Software 16(2):78-85.
61. (2003) http://www.comp.lancs.ac.uk/computing/research/cseg/projects/reaims/index.html. Last Accessed: 2003-05-01.
62. Fritzhanns T, Kudorfer F (2002) Product Management Assessment - a New Approach to Optimize the Early Phases. In Proceedings of IEEE Joint International Conference on Requirements Engineering, IEEE, Los Alamitos CA, pp. 124-134.
63. Wieringa R, Ebert C (2004) Guest Editors' Introduction: RE'03: Practical Requirements Engineering Solutions. IEEE Software 21(2):16-18.
64. Karlsson L, Dahlstedt Å, Natt och Dag J, Regnell B, Persson A (2003) Challenges in Market-Driven Requirements Engineering - an Industrial Interview Study. In Proceedings of the Eighth International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'02), Universität Duisburg-Essen, Essen, Germany, pp. 101-112.
65. Wohlin C, Runeson P, Höst M, Ohlson MC, Regnell B, Wesslén A (2000) Experimentation in Software Engineering: An Introduction. Kluwer Academic, Boston.
66. Stake RE (1995) The Art of Case Study Research. Sage Publications, Thousand Oaks.
67. Robson C (2002) Real World Research: A Resource for Social Scientists and Practitioner-Researchers. Blackwell Publishers, Oxford.
68. Carlshamre P, Sandahl K, Lindvall M, Regnell B, Natt och Dag J (2001) An Industrial Survey of Requirements Interdependencies in Software Product Release Planning. In Proceedings of Fifth IEEE International Symposium on Requirements Engineering, IEEE, Los Alamitos CA, pp. 84-92.

69. Dahlstedt Å, Persson A (2003) Requirements Interdependencies - Molding the State of Research into a Research Agenda. In Proceedings of the Ninth International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'03), Universität Duisburg-Essen, Essen, Germany, pp. 71-80.

Chapter Two

Paper I

Introduction and Application of a Lightweight Requirements Engineering Process Evaluation Method

Tony Gorschek, Mikael Svahnberg and Kaarina Tejle

Proceedings of the Ninth International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'03), Universität Duisburg-Essen, Essen, 2003, pp. 101-112.

Abstract

The lack of an adequate requirements specification is often blamed for the failure of many IT investments. Naturally, the requirements specification is the product of a requirements engineering process.

Methods are required to evaluate the current requirements engineering process and identify where improvements are necessary making it possible to produce requirement specifications of high quality. Existing requirements engineering evaluation methods are often large, costly and time-consuming to use. Therefore we introduce a lightweight evaluation method, which we use to evaluate four industry projects. In this chapter we outline the evaluation method, describe four industrial applications of the method and present an analysis of the findings.

The results suggest that the proposed evaluation method is useful and the studied cases to a large extent have adequate requirements engineering processes although many important aspects are missing from their respective processes.

2. Model-based Process Assessment

2.1. Introduction

According to certain sources the failure rate of IT investments is over 60% [1]. In addition problems introduced through the requirements engineering of a project accounts for something like 50% of the total debugging costs [2]. One of the major causes for this is said to be the lack of a complete and/or adequate requirements specification [2-4] . As the requirements specification is a direct result of the requirements engineering process it stands to reason that an inadequate specification is a result of a requirements engineering process with a low maturity level [5].

Although there exist several methods for assessing software development processes, (e.g. CMM [6] and ISO9000 [7]), few models focus on requirements engineering, and those that do to some extent (e.g. Sommerville & Sawyer [4, 8] , CMMI [9] and SPICE [10]) are large and demand a fair amount of resources in order to be used. This is primarily due to the fact that they are exhaustive and often aimed at large scale evaluations of entire processes, e.g. the whole of a development process. This may not be a problem for large business enterprises, but small and medium size enterprises (SME) often have a limited budget for process evaluation and improvement.

Hence there is no simple and fast way to assess whether or not the requirements engineering processes in a company is inadequate today, or whether other causes are responsible for the aforementioned lack of a complete and adequate requirements specification.

To address this we need a fast and cost effective way to study the status of a requirements engineering process. This initial investigation into the status of requirements engineering in a company need not necessarily be faultless or even exhaustive, instead it should be good enough to give an indication to whether or not a problem exists, and to some extent where the problem areas reside. For this purpose a process evaluation model is introduced, the lightweight model of requirements engineering practices – the *REPM model* [11][1].

In this chapter we describe the results from evaluating the requirements engineering process in four companies, using the REPM model. The contribution of this chapter is thus as a pilot study into requirements engineering practices in industry as well as the introduction and industrial application of the REPM model. This application is described in detail to illustrate (i) *how* the REPM model was used during these evaluations, (ii) *what* results were obtained, and (iii) *how* the results may be interpreted.

The remainder of the chapter is organized as follows. In Section 2.2 we describe the planning for the case studies conducted and the REPM model, and in Section 2.3 we describe the execution of the case studies. The results from the study is presented and discussed in Section 2.4, and the chapter is concluded in Section 2.5.

[1] The REPM model can be obtained from the first author.

2.2. Planning

In this section we describe the context in which we conduct the study and the subjects involved, and present the design of the study as well as address validity issues. Furthermore the section holds a brief introduction to the REPM model itself.

2.2.1. Context and Subjects

The case studies are conducted in industry through in-person interviews by graduate students. The projects evaluated were concluded at the time of the study. This ensures that all of the stages in the RE process are completed at the time of the study.

The case studies involve a total of four companies. We chose to use two medium sized companies, i.e. under 500 employees, and two smaller ones (<150 employees). This to ascertain that the model was tested on both small and medium sized enterprises. The only criterion demanded from the companies selected was that they had projects featuring a customer-developer relationship. Two of the companies are situated in Sweden, and two in Ireland.

These companies were selected because we, or our local contact person on Ireland, had previous relationships with them and because they fit our criteria of large and small companies.

The subjects being interviewed were in senior positions in the projects being evaluated and had knowledge about requirements engineering, and more importantly extensive knowledge about the projects selected for evaluation. The project's main responsible for requirements engineering was designated "project responsible" for each project evaluation session, but we did not put any limitations on the number of people that were present, as the positive effects of having a discussion with more than a single person in a senior position outweigh any risks that may be involved.

Each of the projects being evaluated was selected by the interview subjects. This made it possible for the companies to avoid questions dealing with projects very sensitive to being exposed to outside parties and also made it possible for them to choose a project that was concluded, of the right sort (customer-developer) and of interest to get evaluated. It is important to realize that the companies participating wanted the evaluation to take place in order to get an evaluation of their requirements engineering process. This alleviated the threat that a "good" project was chosen, i.e. it was in the companies own interest to get an accurate evaluation.

However this way of choosing projects introduces several threats to the study. The sampling is very much tainted by the person choosing the project and it may not be representative for the company at hand - it is a case of convenience sampling. In addition the choosing party can be biased, i.e. trying to portray the company in a positive way. We believe these risks to be small as the project responsible have nothing to loose in being honest and portraying the situation correctly – we informed them at a very early stage that the results of the evaluation would be treated as confidential. In addition the data gathered during the evaluations would have been less usable for the companies themselves if the evaluation was corrupted.

2.2.2. Study Design

The study is based around a series of structured interviews (each interview represents one of the four cases). These structured interviews follow a model of requirements engineering practices that has been constructed by the authors, the REPM model [11]. In this section we briefly describe this model and how to interpret the results from it.

2.2.2.1 Requirements Engineering Process Maturity Model

To assess the state of the requirements engineering processes in the companies we have constructed a model of good requirements engineering practices. This model, the REPM model, is further presented in A Method for Assessing Requirements Engineering Process Maturity in Software Projects [11]. The model is inspired mainly by the work done by Somerville et. al. in the REAIMS project [12] but also other existing work, such as Sommerville & Sawyer [4, 8] CMM [6], ISO9000 [7], Jirotka & Goguen [2] and Kotonya & Sommerville [5]. The REPM model was constructed by combining these sources with personal experiences and including additional experts from academia and industry in the construction process. All of these sources were thus used to determine what should be included in the model and at what maturity level.

For reasons of brevity it is impractical to include the entire REPM model in this chapter. Instead, we describe it briefly below and a summary of the actions included in the REPM model is presented in Table 2-1. This is mainly intended to give a brief overview so that an opinion of the usefulness of the REPM model can be formed. For detailed information about the model and the contents please contact the authors.

The REPM model mirrors what should be done to obtain a consistent requirements engineering process. The individual tasks of which the model is comprised are called *actions*. Actions are the smallest constituents of the model and are in turn mapped to one of three main categories (called Main Process Areas or *MPAs* in the model). The MPAs are: *Elicitation*, *Analysis & Negotiation* and *Management*.

Every action resides on a certain *Requirement Engineering Process Maturity level* (REPM level) spanning from 1 to 5, where level 1 represents a rudimentary requirements engineering process and level 5 represents a highly mature process. The actions on each level ensure a consistent and coherent requirements engineering process for the particular maturity level.

The maturity levels enable us to evaluate companies with respect to requirements engineering with a better accuracy than if we simply assume that all actions are equally important. By "base-lining" the actions into maturity levels we can assess that a particular company has potential for a certain maturity in its requirements engineering processes and it enables us to see what actions should be focused on to achieve the particular maturity level.

In Table 2-1 we see the different REPM levels, the goals associated with each maturity level and the actions presented under the relevant level. The actions are divided into groups by the MPAs of Elicitation, Analysis & Negotiation and Management. It is important to notice that achieving REPM level 1 means completing all the actions under REPM level 1, achieving REPM level 2 involves completing all actions under REPM level 1 and all actions under REPM level 2. Thus in order to achieve REPM level 5 one has to complete all actions presented in Table 2-1.

Table 2-1. Action Summary, REPM.

REPM Level 1	
Goals: 1. Basic requirements specification	
	Action Name
Requirements Elicitation	
	Ask Executive Stakeholders
	Technical Domain Consideration
	Executive Stakeholders
	In-house Scenario Creation
Analysis and Negotiation	
	Analysis Through Checklists
Management	
	Document Summary
	Term Definition
	Unambiguous Requirement Description
	Information Interchange Through CARE
	Information handling Through CARE
REPM Level 2	
Goals: 1. Introduction of traceability 2. Introduction of validation of requirements 3. Introduction of a standardized structure for the documentation produced as a result of the requirements engineering process, i.e. the Requirements Document 4. Stakeholder identification	
	Action Name
Requirements Elicitation	
	Research Stakeholders
	In-house Stakeholders
	Scenario Elicitation - Executive Stakeholders
Analysis and Negotiation	
	Requirements Classification
REPM Level 2	
Management	
	Requirements Origin Specification
	Document Usage Description
	Requirements Description Template
	Quantitative Requirements Description
	Prototyping
	User Manual Draft
	Requirements Test Cases
	Requirements Identification
	Backward-from traceability
	Backward-to traceability

Table 2-2. Action Summary, REPM. (Continued)

REPM Level 3	
Goals: 1. Application domain and processes are studied and taken into consideration 2. All stakeholders are consulted 3. Dependencies, interactions and conflicts between requirements are taken into consideration 4. Requirement categorization and prioritization 5. Requirements re-prioritization 6. Peer-reviews 7. Risk assessment	
Requirements Elicitation	
	System Domain Consideration
	Operational Domain Consideration
	General Stakeholders
Analysis and Negotiation	
	Interaction Matrices
	Boundary definition through categorization
	Prioritizing Requirements
	Re-prioritization – New Requirements
	Re-prioritization – New Releases
	Risk Assessment – selected **OG1.03**
Management	
	Global System Requirements Identification
	Record Requirements Rationale
	Business Case
	Descriptive Diagrams usage - **Opt**
	System Models
	Requirements Review
	Forward-from traceability
	Volatile Requirements Identification
	User documentation
	System documentation
Actions REPM Level 4	
Goals: 1. Human domain consideration 2. Business domain consideration 3. Advanced risk assessment 4. Advanced traceability	
Requirements Elicitation	
	Human Domain Consideration
	Business Domain Consideration
	Scenario Elicitation - General Stakeholders
Analysis and Negotiation	
	Ambiguous Requirements refinement **Opt**
	Re-prioritization due to Change
	Risk Assessment – individual - **OG1.01**
	Risk Assessment - sets - **OG1.02**

Table 2-3. Action Summary, REPM. (Continued)

Management	
	Environmental Models
	Requirements Inspection
	Forward-to traceability
	Management documentation
Actions REPM Level 5	
Goals: 1. Requirements reuse 2. Rejected requirements documentation 3. Architectural modeling 4. Advanced validation 5. Advanced requirements re-prioritization	
Requirements Elicitation	
	Requirements Reuse
Analysis and Negotiation	
	Re-prioritization with Regularity
Management	
	Rejected Requirements Documentation
	Architectural Models
	System Model Paraphrasing
	Version traceability

2.2.2.2 Structured Interview Design

Based on the REPM model a checklist is constructed, which we use to guide the structured interviews. This checklist takes each action and formulates it as a question which can be answered with one of the three answers: *completed*, *uncompleted* and *satisfied-explained*.

The purpose of the satisfied-explained category is to take model compatibility into consideration. Companies carrying out projects in special environments unlike the traditional customer-developer environment may deem certain actions unnecessary and have compelling reasons for this opinion.

An example can be a company where the developer and the customer both are specialists in a certain domain and hence "speak the same language". The need for extended clarification and validation of requirements may not be needed, e.g. the construction of prototypes can be omitted. Satisfied-explained thus denotes an action that is not completed but the organization doing the evaluation deems the action not applicable to their project.

The organization doing the evaluation makes the distinction when an action is to be considered satisfied-explained. It is important to notice that an action should not be deemed satisfied-explained for reasons like lack of time, lack of money, lack of know-how or just "did not think of it".

2.2.2.3 Results from Interviews

The results of a project evaluation are presented as four tables, one for each MPA and one summarizing all of the results. An example of such a table is found in Table 2-4. This table is an example of a summary table for all three MPAs for one project

evaluation. We see that the actions for each REPM level are listed separately, and that e.g. REPM level 2 contains a total of 14 actions, of which 9 are completed and 4 are satisfied-explained (14 – (9+4) = 1 is uncompleted).

Table 2-4. Example of project evaluation result.

REPM Level	Total Actions	Completed	Satisfied-Explained
1	10	8	2
2	14	9	4
3	19	11	4
4	11	4	2
5	6	1	4

To assist in the interpretation of the results, we suggest that the results are also presented as graphs, as the example in Figure 2-1. The graphs represent a simple overview of the results and it is recommended that all results be presented in this way as well. However, due to lack of space, we are unfortunately unable to do this in this chapter. The absence of the diagrams should thus not be interpreted as a statement against their worth. A complete list of diagrams can be viewed in [11].

In the graph, the solid gray line represents the total number of actions, the solid black line represents a summary of all actions that are completed and satisfied-explained. The dashed line represents the actions that are actually completed. The area between the dashed line and the solid black line denotes to what extent the REPM model is inapplicable to the project being evaluated (called *model lag*), the area between the solid black line and the gray line represents the area of possible improvement of the RE process.

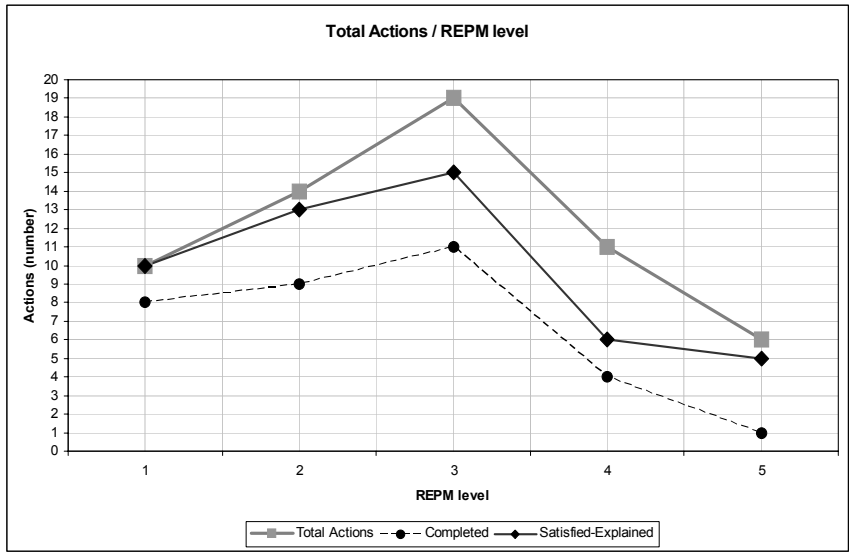


Figure 2-1. Example of result diagram.

The tables and graphs are interpreted as follows. Starting with the first REPM level, if all actions are completed or satisfied-explained, i.e. the solid black line overlays the solid gray line, this level of maturity is achieved. This would mean, if no more REPM levels are achieved, that the company has a consistent and complete requirements engineering process of a low maturity level. This is then repeated for each of the REPM levels. Note that all lower REPM levels must also be completed before a certain REPM level is achieved.

In Table 2-4 and Figure 2-1, for example, we see that REPM level one is achieved and only one action more is required to achieve REPM level two, whereas four more are required to achieve level three. Strictly speaking, this project would be considered to be on REPM level 1, but as only one action is necessary to get it up to level two this is the level that we think this company should aim for in a first step. This would ensure a consistent requirements engineering process that is fairly basic but may be sufficient for this company's needs.

2.2.3. Validity Evaluation

In this section we discuss the threats to this investigation. We base this on the discussion of validity and threats to research projects presented in Wohlin et al. [13].

2.2.3.1 Conclusion Validity

Four cases are inadequate for statistically sound generalization purposes. However, this is not the main intention of this study. The main intention of this study is to serve as a pilot project to see whether it is possible to assess the state of requirements engineering practice using the REPM model, and to possibly provide some early results as to what areas within requirements engineering are subject to most problems.

The checklist used for the case studies (structured interviews) was validated through preliminary testing and proofreading by several independent parties, this to avoid factors like poor question wording and erroneous formulation.

Each case study interview was conducted without any break. Thus the answers were not influenced by internal discussions about the questions during e.g. coffee breaks.

The sampling technique used, i.e. convenience sampling, can pose a threat to the validity of the investigation. The companies selected are not necessarily representative of the general population of software development companies, on the other hand there is no evidence that they are not.

2.2.3.2 Internal Validity

Prior to the interviews the subjects had access to the REPM model so that they could acquaint themselves with the model (see Section 2.2.2.1) In addition the REPM evaluation checklist was explained before it was used. These steps were taken to avoid problems with maturity, i.e. that the first questions would act like a learning process and thus be answered with a different presupposition than the remaining questions. The preparation described was executed in an identical manner for each of the four case studies. We did not encounter any problems with this due to the fact that the subjects being interviewed had similar backgrounds, i.e. experience in the field of requirements engineering and software development.

2.2.3.3 Construct Validity

The REPM model itself can be a threat to the investigation. If the model is unsatisfactory when it comes to content and/or structure it follows that the result of each case study (and thus the investigation) may be threatened to the same extent. Another important point is the usage of the model by the investigators and by the interview subjects. Using the REPM model as a tool in a manner not consistent with the initial intent may diminish the result obtained through the usage.

To reduce these risks the REPM model and the checklist were validated before the case studies were conducted. The validation was two-fold. First the model was scrutinized by the creators and an independent expert from academia. The second round of validation was conducted by an independent expert from industry. During the validation process the models structure and contents was modified when appropriate.

2.3. Operation

Having prepared the model, the checklist and a way to present the results, the next step consists of contacting companies and actually conducting the study. In this section we describe this, starting with the preparations and continuing with experiences from the actual execution.

2.3.1. Preparation

A copy of the REPM Model was sent to the interview subjects in advance. This to give all people involved a chance to study the model and its constituents in order to prepare them for the work during the project evaluations and to note any questions they wished to address.

In addition to this the participating companies were asked to choose a project to be evaluated and instructed that the project should be of a customer-developer type, i.e. not an internal development project but featuring an external customer. This due to the fact that the REPM model at this stage of its development is adapted primarily for these types of projects. In addition we asked the person(s) mainly responsible for the requirements engineering process for the projects selected to be present as the interview subjects at the project evaluation session.

The interviews in Ireland were preceded by personal contact and emails to prepare for the evaluation.

2.3.2. Execution

At the initial stage of the interview the basic layout of the evaluation was explained. Information about how the Project Evaluation Checklist version 1.4 (can be obtained from the authors) is structured, and how it is connected to the REPM model was provided. In addition we asked the people evaluating the project to "think-aloud" when answering the questions. An important thing we clarified was the concept of satisfied-explained actions, i.e. that the model may not suit all projects and that this concept was introduced to compensate for that fact.

All interviews were made on-site and in person. We find in-person interviews beneficial in several ways. It is often possible to extract more information as well as avoiding misunderstandings. However there can also be negative aspects like the interviewer influencing the interviewees [14, 15] . By strictly adhering to the pre-planned structure of the interview and only deviating when further explanations were necessary, we believe that the risk that this has occurred is low.

Each case study interview was estimated to take between 1.5 to 2 hours. This was based on an earlier test interview conducted using a software engineering post graduate student. The time spent on the interviews mirrored our estimation.

During the interviews we were somewhat surprised by the tendency for the subjects to engage in discussions related to the topic of requirements engineering in general and more specific topics as the questions reflected certain actions. We tried to steer the interview towards completing the checklist but did not categorically refuse to discuss matters. More often than not the discussions helped to clarify different points, and often the discussions were more or less one sided, i.e. the interview subjects discussed matters to clarify their own trail of thought to themselves before answering a question. We only got involved in the discussion when necessary, trying not to influence the answers by adding information not already present in the model and/or in the checklist. We took this decision to ensure that each interview subject received the same amount of information, so as not to change the conditions of any one interview in comparison with another.

2.3.3. Data Validation

During the interview the subjects discussed their answers aloud as the questions were answered. This helped us catch misunderstandings, and if necessary we clarified the relevant questions in an effort to elicit a relevant answer to the question at hand.

All of the answers are included in this investigation. Our goal is to ascertain the status of the requirements engineering process in companies today, therefore all answers are relevant. Four case studies, and thus four interviews, were conducted. They are all presented in the study, no case studies were disqualified due to reasons such as lack of conformity.

2.4. Analysis and Interpretation

In this section we present the results from the case studies (each case study is represented by one project). We do this in four tables, one for each project/case study. As mentioned earlier, it is easier to analyze the data if the results are presented as graphs in the same form as the examples in Figure 2-1. Due to lack of space, we are unfortunately not able to do this here, and instead refer you to [11], where these graphs are available.

We refer to the different projects as project alpha, beta, gamma and delta respectively. Projects gamma and delta are from what we consider small companies.

2.4.1. Presentation of Results

In this section we present the results from the case studies (each case study is represented by one project). We do this in four tables, one for each project/case study. We refer to the different projects as project alpha, beta, gamma and delta respectively. Projects gamma and delta are from what we consider small companies.

In Table 2-5 we see the data from project alpha, and in Table 2-6,

Table 2-7 and

Table 2-8 we see the data from project beta, gamma and delta, respectively.

These tables are read as follows. There are four groups of data, each containing three columns. These columns list the total number of actions on each REPM level, the number of actions completed and the last column lists the number of actions satisfied-explained for the REPM level. The first group of three columns is a summary of the following three groups, which are the actions within the MPAs Elicitation, Analysis and Negotiation and Management, respectively.

We see that no project fulfils even REPM level 1. However, if we count those where only one or two actions are missing, we see that project alpha is close to REPM level 1, project beta close to level 2, project gamma close to level 1, and project delta close to level 2. We also see that project beta has potential for being on level 5, and that the others probably would benefit from completing the remaining actions for their respective REPM level and aim for REPM level 3. Level 3 represents a requirement engineering process that is fairly advanced and not just the bare basics, while still being streamlined and suitable for many software organizations.

Table 2-5. Project alpha.

	Action Summary			Elicitation			Analysis & Negotiation			Management		
REPM level	Total Actions	Completed	Satisfied-Explained	Total Actions	Completed	Satisfied-Explained	Total Actions	Completed	Satisfied-Explained	Total Actions	Completed	Satisfied-Explained
1	10	8	0	4	3	0	1	0	0	5	5	0
2	14	8	2	3	1	1	1	1	0	10	6	1
3	19	12	2	3	2	1	6	4	0	10	7	0
4	11	6	1	3	1	1	4	2	0	4	3	0
5	6	2	0	1	0	0	1	0	0	4	2	0

Table 2-6. Project beta.

	Action Summary			Elicitation			Analysis & Negotiation			Management		
REPM level	Total Actions	Completed	Satisfied-Explained	Total Actions	Completed	Satisfied-Explained	Total Actions	Completed	Satisfied-Explained	Total Actions	Completed	Satisfied-Explained
1	10	8	1	4	4	0	1	1	0	5	3	1
2	14	10	2	3	2	1	1	1	0	10	7	1
3	19	15	1	3	3	0	6	5	1	10	7	0
4	11	9	1	3	2	1	4	3	0	4	4	0
5	6	4	0	1	1	0	1	1	0	4	2	0

Table 2-7. Project gamma.

	Action Summary			Elicitation			Analysis & Negotiation			Management		
REPM level	Total Actions	Completed	Satisfied-Explained	Total Actions	Completed	Satisfied-Explained	Total Actions	Completed	Satisfied-Explained	Total Actions	Completed	Satisfied-Explained
1	10	8	0	4	3	0	1	0	0	5	5	0
2	14	7	0	3	2	0	1	1	0	10	4	0
3	19	13	1	3	3	0	6	3	0	10	7	1
4	11	3	2	3	0	1	4	2	0	4	1	1
5	6	1	1	1	0	1	1	0	0	4	1	0

Table 2-8. Project delta.

	Action Summary			Elicitation			Analysis & Negotiation			Management		
REPM level	Total Actions	Completed	Satisfied-Explained	Total Actions	Completed	Satisfied-Explained	Total Actions	Completed	Satisfied-Explained	Total Actions	Completed	Satisfied-Explained
1	10	9	0	4	4	0	1	0	0	5	5	0
2	14	9	4	3	2	1	1	1	0	10	6	3
3	19	11	4	3	3	0	6	1	3	10	7	1
4	11	5	3	3	3	0	4	0	1	4	2	2
5	6	3	1	1	1	0	1	0	1	4	2	0

2.4.2. Analysis of Results

First, we always recommend a homogenous REPM level across the three MPAs. The reason for this being mainly dependencies. If a project's requirements engineering process resides on REPM level five when it comes to the MPA of *Requirements Elicitation* but only on level two in the other two MPAs this would not be a consistent and coherent requirements engineering process. Moreover, in many cases actions under one MPA are dependent on other actions being completed under another MPA.

In this respect, none of the projects succeed. The process area where all of the projects fail is that of *Requirements Management*. There are several reasons for this. Requirements Management is the largest MPA, i.e. housing the largest number of actions, and thus there is more to complete for this area. Moreover, this area contains actions which are notoriously difficult to accomplish in an efficient way, such as traceability and change policies. We are also quite stringent in how we think a requirements document should be structured and the individual requirements written which is reflected in the REPM model under requirements management where the actions for requirements documentation are included.

Assuming that all organizations should strive for at least REPM level 3 there are several actions that two or more projects need to complete. These are:

- In *Elicitation*, the action *In-house Scenario Creation (REPM level 1)*. This action is concerned with creating scenarios for elicitation purposes but to only consult

the developers in the projects. As there are other actions where stakeholders are consulted and none of the projects fail to do this, the companies may think that if stakeholders are consulted it is not necessary to consult the developers.

- In *Elicitation*, the action *Research Stakeholders (REPM level 2)*. Who the stakeholders are is researched during this action. This can help identify groups that are affected by the requirement, and thus the basis for gathering information about a requirement is wider if the action is performed. That this action is not done may stem from a trust that the customer provides information regarding all stakeholder groups as this is in the best interest of the product and hence the customer. The development companies may forget who it is in the customer-developer relationship that has experience with requirements engineering.
- In *Analysis and Negotiation*, the actions *Analysis through Checklists (REPM level 1)* and *Interaction Matrices (REPM level 3)*. Analysis through checklists refers to having standardized checklists that are used to ensure certain standardized qualities of each requirement. Interaction matrices are constructs that can help catch requirements dependencies, conflicts or other interactions between requirements. These are very simple tools that can help catch problems with requirements at an early stage, and we are surprised that this is not done in all companies.
- In *Analysis and Negotiation*, the action *Risk Assessment – Selected (REPM level 3)*. This action is part of a group of actions, of which one, but only one, must be completed. The remaining actions in this group are on REPM level 4, so this is the least possible amount of risk management that should be done with respect to requirements analysis and negotiation. What it means is simply that certain requirements are, based on intuition, identified as being extra prone to risk and then a risk assessment is performed for these requirements.
- In *Management*, the action *Quantitative Requirements Description (REPM level 2)*. With this action we mean that all requirements – especially quality requirements – should be specified such that they can be measured. Quality requirements are a notoriously difficult subject that is still the subject of much research. It is thus not surprising that the companies reflect the amount of uncertainty in the research community regarding quality requirements and their quantifiability.
- In *Management*, the actions *Requirements Review (REPM level 3)* and *User Manual Draft (REPM level 2)*. These two actions are concerned with validating the requirements. The second of these two actions suggests that an initial user manual is written based on the requirements, which serves as an informal walkthrough of the requirements. That this action is not completed in two of the companies may simply be that they have not thought of it.
- In *Management*, the action *Document Usage Description (REPM level 2)*. Different user groups often use the requirements document in different ways. A document usage description is a manual aimed at aiding different users of the document, helping them use and navigate the document. This may often be forgotten as it seems obvious to those writing the document how to read it.

In *Management*, the action *Backward-to Traceability (REPM level 2)*. This action links the design and implementation back to the requirements. That this is not done may be a

result of waterfall-inspired development methods. If previous development steps such as requirements engineering is never re-visited, what is the point in tracing the requirements backwards. Moreover, many companies may not update the requirements specification if the design and implementation deviates from what is specified.

2.4.3. Summary and Interpretation of Results

Only 9 of the 43 actions up to REPM level 3 are not completed by two or more projects. However, many of the uncompleted actions are quite severe, and we are surprised that they are not done in all companies. For example, risk assessment seems to be a neglected area, and interactions between requirements do not appear to be mapped, which of course can cause severe problems if there are in fact conflicting or volatile requirements. Also, we are surprised that companies still spend time and effort on stating requirements without providing a measurement, as there is no way of telling when the requirement is fulfilled if this is not done. As mentioned quality requirements, where this is most often missed and required the most, is a subject with much research focus and methods are still needed for quantifying these quality aspects of software systems.

The size of a company does not seem to have a direct correlation to the maturity of the requirements engineering process to the extent believed initially. If we look at project alpha and beta approximately 68% (for project alpha) and approximately 85% (for project beta) of all the actions are fulfilled. The numbers for gamma and delta (the smaller companies) are 60% and 81%, respectively. At an initial glance the numbers seem very much comparable, even if the larger companies have a small advantage. To see the whole picture however one has to take the amount of actions deemed satisfied-explained under consideration. In project alpha, beta and gamma the number of actions in this category seem to be a fairly constant 8%, whereas in the case of delta the number of satisfied-explained actions is 20%. It is vital to understand whether the latter figure is the result of model inapplicability (See Section 2.2.2.1) or not, in order to find out whether project delta is representative as a small company project or not.

If the number has another reason than model inapplicability this may indicate a distinct line between medium sized and smaller companies when it comes to requirements engineering. The projects in this study were chosen with the same criteria in mind, and thus should be generally compatible when it comes to model applicability. A widened study, involving more companies is necessary to understand whether this distinction between medium sized and smaller companies exist.

The MPA of *Requirements Management* is generally the one needing most improvements, i.e. the MPA with most actions not completed. There are several reasons for this. As mentioned Requirements Management is the largest MPA, i.e. housing the largest number of actions, and thus there is more to be completed. Another reason may be that the actions under this particular MPA are fairly advanced, i.e. on a higher REPM level relative to the total number of actions. An example of this is that there are more actions under the MPA of *Requirements Management* on REPM level 5 than there are under the other MPAs in total.

2.5. Conclusions

The main purpose of the investigation presented above was to (1) to give an idea of the problem scope pertaining to requirements engineering practices in industry, and (2) test a method for quickly ascertaining the status of requirements engineering in companies. This was done through the design of the REPM model, which in turn was used for the investigation.

Firstly, if we look at results gathered from the four cases they may of may not be generalizable to a larger set of companies, i.e. if replications of this study show similar results, there are several ways to react to this, depending on what audience one belongs to.

For researchers, the question seems to be to find effective and attractive methods for risk assessment and for requirements management. For development companies, the reaction should be to first assess whether the very simple actions are done, as we believe this would vastly enhance the quality of the resulting products and, above all, reduce the risks involved.

For companies participating in an evaluation such as the ones presented above, the results should be analyzed to decide whether or not there is an indication of a problem. The second step is then to study the evaluation, examine the improvement suggestions and the conclusions drawn for inaccuracies and/or neglected parts. It is also important to scrutinize the actions deemed to be under the category of Satisfied-Explained to ensure that they are put there for the right reason. After the evaluation review a plan should be constructed based on the improvement suggestions and improvement consequences. This plan should state what measures should be taken. One way proceed is to order a more comprehensive and exhaustive examination of the requirements engineering process using the results from the REPM model evaluation as a first insight to where the problems may reside. Another way to go is to use the REPM results purely as a basis for deciding on a more thorough investigation. In the latter case the results from the two investigations could be validated through a subsequent comparison.

Secondly, if we look at the REPM model itself and the use of it in the four cases there are two main parts, namely the gathering of the data using the Project Evaluation Checklist (see Section 2.2.2.2), and the evaluation and interpretations of the results (see Section 2.4).

The main point in constructing the REPM model was to get a fast, easy and cost effective evaluation of a requirements engineering process. During the industrial application of the model the gathering of the data took no more than eight person-hours per project. The subsequent analysis of the data took approximately 30 to 40 person-hours per project. This gave us at most 48 person-hours in cost for the REPM evaluation of a project, which can be considered to be fast.

In this chapter a fairly detailed description was presented of how the industry cases (projects) were evaluated. During the gathering of data the actions were deemed as belonging to one of three categories, i.e. Completed, Uncompleted or Satisfied-Explained. The diagram example presented in Section 2.2.2.2 gives information of how result diagrams could be constructed to give an overview of the process (as well as parts of the process), and how the data could be evaluated and interpreted is presented in sections 2.4.2 and 2.4.3. One could go so far as to claim that the collection of the data is

fairly easy using the Project Evaluation Checklist. The analysis and interpretation of the results on the other hand does demand some expertise in the area of requirements engineering, although the complete REPM model offers some insights into the area.

Whether usage of the REPM model is cost effective or not is of course dependent on how accurate and exhaustive the gathered results are. As mentioned before emphasis was put on speed and ease, not exhaustiveness. If we look at the results from the case studies quite a few indications point to areas of possible improvements. However this does not give any information about things potentially missed. In our experience the REPM model provides an indication of problem areas that should be scrutinized further. An REPM evaluation could be used to develop a plan for what steps to take in order to improve a requirements engineering process, though it is important to realize that the REPM model is lightweight, and an evaluation taking 48 person-hours is bound to overlook issues.

It is also important to notice that the REPM evaluation is project based. If a company has a "generic" (typical) project to evaluate one instance may suffice. On the other hand most companies may need to evaluate more than one project in order to get an accurate (and more exhaustive) picture of their RE process, this is especially true for companies without a standardized and repeatable process.

As the results from the cases were presented to each of the companies the general view was that some of the results gathered were already more of less known, but for the most part the REPM evaluation results were seen as valuable insights in the respective company's process offering an indication that further study was warranted.

2.5.1. Future Work

This study serves as a pilot study as far as ascertaining (1) the problem scope pertaining to requirements engineering practices in industry, which is the reason why only four companies have participated. In order to gain real understanding of what is done and what is not done in industry today, an extended study involving a larger set of companies needs to be performed. We also need to further understand the relation between what is considered an inadequate requirements specification and the connection between this and the maturity of the requirements engineering process at large.

We encourage others to design and execute similar investigations on their own, as there is a clear and present need for knowledge about the status of requirements engineering in industry today.

(2) If we look at the REPM model there is a need for further evaluation, refinement and validation. There is also a need to validate results gathered with the REPM model by using another approach (other model/method). This would provide further information on of how accurate and cost effective the REPM model is, as well as how the REPM model can be improved upon.

(3) A study of how the results from a REPM evaluation can be used in further and more detailed RE process evaluation is also warranted. I.e. how to use the REPM results obtained to take the next step, either towards further evaluation or maybe even a the creation of an adequate improvement plan.

2.6. References

1. Connolly TM, Begg CE (1998) Database Systems: A Practical Approach to Design, Implementation and Management. Addison-Wesley, Harlow.
2. Jirotka M, Goguen JA (1994) Requirements Engineering Social and Technical Issues. Academic Press, London.
3. van Buren J, Cook DA (1998) Experiences in the Adoption of Requirements Engineering Technologies. Crosstalk - The Journal of Defense Software Engineering 11(12):3-10.
4. Sommerville I, Sawyer P (1999) Requirements Engineering: A Good Practice Guide. Wiley, Chichester.
5. Kotonya G, Sommerville I (1998) Requirements Engineering: Processes and Techniques. John Wiley, New York.
6. Paulk MC (1995) The Capability Maturity Model : Guidelines for Improving the Software Process. Addison-Wesley, Reading.
7. (2002) http://www.tickit.org/. Last Accessed: 2002-10-01.
8. Sawyer P, Sommerville I, Viller S (1999) Capturing the Benefits of Requirements Engineering. IEEE Software 16(2):78-85.
9. CMMI-PDT (2002) Capability Maturity Model Integration (CMMI), Version 1.1. CMMI for Systems Engineering, Software Engineering, Integrated Product and Process Development, and Supplier Sourcing Version 1.1 (CMMI-SE/SW/IPPD/SS, V1.1).
10. (2003) http://www.sqi.gu.edu.au/spice/. Last Accessed: 2003-09-11.
11. Gorschek T, Tejle K (2002) A Method for Assessing Requirements Engineering Process Maturity in Software Projects. Master Thesis in Computer Science. Blekinge Institute of Technology, Ronneby.
12. (2003) http://www.comp.lancs.ac.uk/computing/research/cseg/projects/reaims/index.html. Last Accessed: 2003-05-01.
13. Wohlin C, Runeson P, Höst M, Ohlson MC, Regnell B, Wesslén A (2000) Experimentation in Software Engineering: An Introduction. Kluwer Academic, Boston.
14. Körner S (1985) Statistisk Slutledning. Studentlitteratur, Lund.
15. Robson C (1993) Real World Research : A Resource for Social Scientists and Practitioner-Researchers. Blackwell, Oxford.

Chapter Three

Paper II

Identification of Improvement Issues Using a Lightweight Triangulation Approach

Tony Gorschek and Claes Wohlin

Proceedings of the European Software Process Improvement Conference (EuroSPI'2003), Verlag der Technischen Universität, Graz, Austria, 2003, pp. VI.1-VI.14.

Abstract

One of the challenges in requirements engineering is the ability to improve the process and establish one that is "good-enough". The objective of this chapter is to present a lightweight approach to identify process improvement issues. The approach is developed to capture both the views of different stakeholders and different sources of information. An industrial investigation from a small company is presented. In the investigation both projects and the line organization have been interviewed and documentation from them has been studied to capture key issues for improvement. The issues identified from one source are checked against other sources. The dependencies between the issues have been studied. In total nine issues for improvement of the requirements engineering work at the company were identified. It is concluded that the approach is effective in capturing issues, and that the approach helps different stakeholders to get their view represented in the process improvement work.

3. Inductive Process Assessment

3.1. Introduction

The area of requirements engineering (RE) is often underestimated in value in the area of software engineering, in spite of the fact that requirements problems are the largest cause in project failure [1, 2].

This chapter presents an investigation conducted in industry aimed at introducing a way to identify RE *improvement issues* in small and medium sized enterprises (SME) [3]. The lightweight approach presented uses data point *triangulation* [4] as a means to identify possible improvement issues. The data points consist of two major parts, i.e. a *line study*, and a *project study*. This is depicted in Figure 3-1. The project study is comprised of three exploratory case studies [4], and each case study is in turn comprised of project specific interviews and documentation (see Section 3.3.2).

The line study is comprised of line interviews and documentation (see Section 3.3.3). The main objective of the investigation was to ascertain state-of-practice at Danaher Motion Särö (DHR) (see Section 3.2) and to identify improvement points in their RE process. There are approaches for RE process improvement targeted at e.g. SME. Sommerville and Sawyer [5] for instance present a series of guidelines as well as a guide for process improvement as a product of the REAIMS project [6]. The PERE (Process Evaluation in Requirements Engineering) method used for analysis of the requirements engineering process from two different viewpoints in REAIMS is basically a systematic approach to understanding processes, and a human factors analysis of the process. The lightweight approach presented in this chapter uses data point triangulation with *four* main data sources (see Section 3.3.1), In addition to this the emphasis of the approach presented in this chapter is on fast and fairly low-cost evaluation.

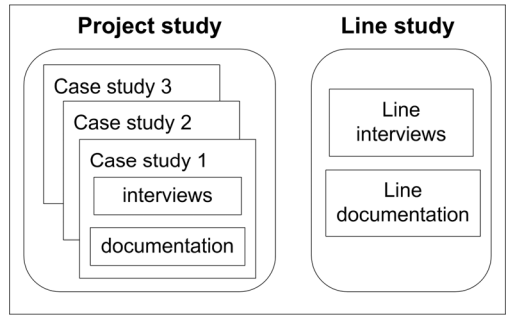


Figure 3-1. Investigation description.

In addition to RE specific improvement strategies there are some more general, targeted at quality assurance in software projects, such as *The TickIT Guide* (ISO 9001:2000) [7]. These process improvement and quality assurance approaches do not offer any lightweight identification of how to establish what needs to be improved in an RE process, i.e. how to identify improvement issues.

The main objectives (contributions) of the chapter is to present (i) the lightweight triangulation approach used for the identification of the improvement issues (presented in Section 3.3), (ii) the improvement points identified and triangulated (presented in

Section 3.4), and (iii) the dependencies between the improvement points (also presented in Section 3.4). In addition to this there is an overview of state-of-the-art in relation to each improvement point (see Section 3.5). Section 3.6 contains the conclusions drawn in the chapter.

3.2. Investigation Context

Danaher Motion Särö AB develops and sells software and hardware equipment for navigation, control, fleet management and service for Automated Guided Vehicle (AGV) systems. More than 50 AGV system suppliers worldwide are using DHR technologies and know-how together with their own products in effective transport and logistic solutions to various markets worldwide. The headquarters and R & D Centre is located in Särö, south of Gothenburg, Sweden. DHR has approximately 100 employees. DHR is certified according to SS-EN ISO 9001:1994, but is uncertified according to CMM and CMMI.

DHR has a wide product portfolio, as the ability to offer partners and customers a wide selection of general variants of hardware and supporting software is regarded as important. Tailoring and especially lighter customer adaptation often follows the procurement and subsequent installation of a system. This in addition to development of new software and hardware makes it a necessity to plan, execute and manage a wide range of projects.

Requirements are one factor that binds all of the projects together. It is not necessarily so that requirements breach project boundaries (although this is known to happen) but rather that most projects involve elicitation, analysis and negotiation, management and documentation of requirements. In addition to this the stakeholders (or groups of stakeholders) are at least as divergent and diversified as the projects themselves. Requirements from general sources like *the market* have to be taken into consideration as well as the ones from *industry-partners*, *end-customers* and *internal sources* such as developers and management. All of these factors contribute to a need for an RE process that is good enough (at present) and has the potential to meet growing demands and complexity (in the future). The ability to prepare for the future, and improve current practices, is the underlying motivator for the work conducted in the partnership between DHR and Blekinge Institute of Technology.

The first step in this partnership was to map the RE process at DHR (establish a baseline), and to determine what the main improvements issues were.

3.3. Investigation Design

In this section the overall design of our *multi method investigation* is described, how the data was elicited for each data source (project study and line study), compiled and analyzed. A validity evaluation is also presented below.

3.3.1. Multi Method

By looking at several data sources instead of relying on a single one, e.g. solely a case study, a higher level of validity can be achieved [8]. Below four major data sources are described, (A) data from the case study interviews, (B) data from project documentation, (C) data from line interviews, and (D) data gathered from line documentation. (A) and (B) together comprise the project study, i.e. interviews and project documentation from three specific projects (see Section 3.3.2). (C) and (D) are interviews and documentation from the line, and together comprise the line study. The idea is not to use all of the data sources solely for the purpose of getting more data, but rather to have a confirmation (validation) of the individual issues identified. This is achieved through triangulation, illustrated in Figure 3-2. One issue is identified and specified, then checked against the other data sources for confirmation. An additional benefit is that this usage of different sources enables several perspectives, decreasing the possibility of missing crucial information.

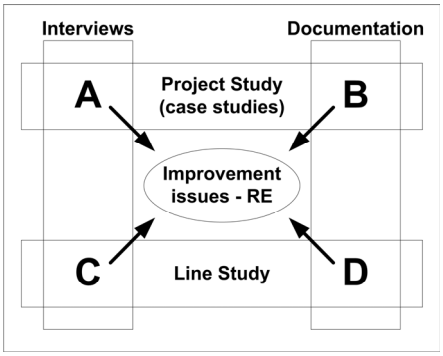


Figure 3-2. Triangulation using several data sources.

In addition to getting project specific data (A) and (B), data is elicited from the line (C) and (D), i.e. the development support and production parts of the organization (see Section 3.3.3).

In addition to the horizontal division of the data collected there is a vertical distinction to be made. Data gathered from the interviews, (A) and (C), are complemented and/or verified (occasionally contradicted) by the data gathered from the documentation, (B) and (D).

When conducting the triangulation two leading data sources were selected, namely the interviews (A) conducted in the case studies and the interviews conducted in the line study (C). The reason for selecting leading data sources was the need for points of reference that could be compared (triangulated). If no leading source was selected every single potential data point had to be considered and triangulated. This would have been possible but very time and resource consuming, and the final improvement issues identified would be the same.

It was however not a foregone conclusion that specifically the interviews were to be chosen as the leading data sources. The main reason for the decision was the fact that the interviews reflected the views of the project and line team members. In addition to this a piece of documentation (whether it was from the project study or the line study) was primarily used for confirmation of the issues identified during the interviews. If the documentation was used for the identification of issues themselves all things not present in the documents could be considered potential issues. This became a problem if the contents of the documents were compared to state-of-the-art, in which case the absence of things in DHR's documentation could yield any number of issues depending on what sources in state-of-the-art were used for comparison. The idea was not to identify all things not performed and turn each these in to an issue (basically getting a maximum

sized RE process), but rather to identify what needed to be changed/added to DHR's RE process based on their own perception and experiences.

Three other reasons existed for making the interviews the leading data source. The case study interviews were the source of pre-verified quantitative data (the pre-verification is described in Section 3.3.2.1), the case studies were the source of the most up-to-date data (three current projects were studied), and last the case studies yielded the lion share of the total data collected.

3.3.2. Project Study (Case Study) Design

In order to get an overview of the RE process no single case could be studied that was representative for most of the projects conducted at DHR, rather a block of projects had to be identified [9]. Several projects had to be considered and three were finally chosen. The main criteria were getting a cross-section of typical projects conducted, and to get a wide representation of the developers and managers involved in the projects, i.e. avoid asking the same people questions about different projects. Company representatives with an overview of DHR's domain, project portfolio and current practices picked three projects. The projects chosen were of three types:

- ***Project Alpha*** Externally initiated software development project aimed at an end customer. Basically an addition of customer specific features to an existing piece of software.
- ***Project Beta*** Externally initiated software development project aimed at an industry partner. Basically development of a new service tool to be used by an industrial partner in their work with end-customers.
- ***Project Gamma*** Internally initiated software and hardware project aimed at updating a certain central product. Basically a generational shift to newer technology for mainly production aspects.

The division of the projects can be seen as two steps, first there is the obvious distinction between the projects from the point of what sort of product they yield, bespoke (alpha and beta) or generic (gamma). As described by Sommerville [10] bespoke products are aimed at specific customers and the projects are initiated by external sources, generic products however are rather aimed at a market and initiated internally by the producer. Second there was a need for a further granulation of the "bespoke project type". This is due to the fact that project alpha was aimed at an end-customer, while beta was aimed at an industrial partner. The preconditions of a partner project differ from that of an end-customer. Industrial partners often have more domain specific and technical knowledge.

Subsequent to project division and selection, interviews were booked with representatives for all *roles* in the projects at hand. Four major roles were identified that were typical for all of the projects in question, *Orderer*, *Project Manager*, *System Engineer* and *Developer*. These official roles can briefly be described in the following manner:

1. The *Orderer* has the task of being the internal owner of a certain project, i.e. has the customer role and if applicable the official contact with an external customer and/or partner. This party is responsible for the official signing-off when it comes to the requirements, i.e. he places an order.

2. The *Project Manager* has the traditional role of managing the project, resources, planning and follow-up. As far as requirements are concerned the Project Manager is responsible for that the requirements engineering is performed, the requirements specification is written and signed off by the System Engineer.
3. The *System Engineer* is the technical responsible for a project. It is also important to recognize that the System Engineer has the official responsibility for the requirements specification in a project.
4. The *Developer* is a representative for the developers (e.g. programmers) in a project, the ones actually implementing the requirements. The developers use the requirements specification.

The researchers in cooperation with representatives for DHR identified these roles. In total 11 people were interviewed during the case studies, four each from projects Alpha and Gamma, but only three from project Beta. The Orderer from project Beta was not accessible at the time of the investigation.

3.3.2.1 Case Study Interview Questions

Semi-structured interviews [4] were used for the case studies. There were 31 questions in total, two warm-up questions where the subjects stated name, title and responsibilities etc. Following warm-up the questions were divided into four main parts, i.e. *requirements elicitation*, *requirements analysis and negotiation*, *requirements management* and a *general part*. The questions posed in the first three parts were aimed at finding out what tasks were performed in the context of RE for the project in question, who performed them, when and how. An example of such a questions could be "was there any classification of the requirements into classes, groups or types during the elicitation process", and the follow-up questions "how and by whom was this done", and "what were the criteria for the classification…". The subjects were asked to elaborate on their answers, especially if the answer was a simple "no". The intention is to get an idea of why a certain task was not performed. The structure of the questionnaire and the tasks that the questions were based upon were inspired primarily by the work of Sommerville & Sawyer [5] and Gorschek et al. [11]. The questions posed in the first three parts were of a qualitative nature.

The general part (fourth) contained three questions (from a total of four) of a quantitative nature. Each of these questions was comprised of several sub-questions, and two of them were *open*, i.e. up to the subject to come up with the contents and not merely answer if a certain task was performed. An example of such a question was "list three things that work best with respect to RE at you company" and the follow-up "…three things that have the best improvement potential with respect to RE…". These open questions were beneficial from at least three aspects, i.e. the data gathered during preceding questions could substantiate the answers given here, items missed by earlier questions could be observed, and last the subjects were asked for their opinion without being led in a certain direction.

When the interviews were completed the results were compiled and the quantitative results were used as a primary source of data, and the qualitative results were used as a secondary source as well as to validate the quantitative data. Figure 3-3 shows the procedure, the reasoning behind this was to be able to see patterns in the quantitative data that could be substantiated (validation) and augmented by the data gathered from

the qualitative part. The confirmed data was then used as one of the sources (A) in the triangulation (see Figure 3-2).

The subjects were given the questions on location. Each interview took about one to one and a half hours. All of the interviews were conducted over a period of five consecutive days. In addition to general information given to the subjects they were assured that all answers would be anonymous. The questionnaire used can be obtained through the authors. A digital recorder was used during the interviews in addition to the notes taken, this to enable further clarification and analysis after the fact.

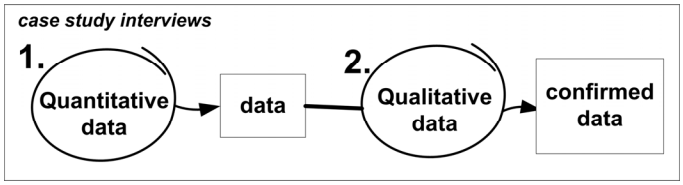


Figure 3-3. Pre-validation of case study data.

3.3.2.2 Project Documentation

The study of documentation was a substantial part of the overall investigation. The documentation in question was of two types, first there were documents pertaining to the projects investigated in the case studies (B), and second there was line documentation (D) (see Figure 3-2).

Project documentation included pre-study documents, project specifications, project plans, requirements specification, subsequent follow-up documentation, budget and time reports and meeting minutes.

3.3.3. Line Study

In addition to the 11 case study interviews conducted three line interviews were performed. This was done to elicit data from three additional roles not present in any of the block projects. The roles interviewed consisted of representatives for *System Test*, *Production* and *Application Developer*. The descriptions of the roles below are according to the official line documentation.

- The *System Test* role can be described as the traditional role of application and feature test. This role is officially present during initial project meetings and is a part of the verification of the requirements specification.
- The *Production* role also has a presence in projects. This representation consists of verifying that the production aspects are met, i.e. that production is taken into consideration at an early stage.
- The *Application Developer* role represents installation and adaptation aimed at industry partners and/or end customers. Adaptation here translates to tailoring, development of some light customer features and some support.

The interviews conducted here were unstructured [4], i.e. some directional questions were used, but for the most part informal conversation dominated. The conversational interview's goal was to elicit data about the lines role in the RE process, the

representatives opinions and experience. The structure observed during the case study interviews (see Section 3.3.2.1) was present to a certain degree, i.e. the interviewer wanted answers to some of the same questions, but no questionnaire was formally used. The reason for using a more unstructured approach during the line interviews was due to that they were somewhat more general in nature, i.e. did not refer to a specific project. The same general information and assurance of anonymity as before was issued.

3.3.3.1 Line Documentation

Line documentation (D) (see Figure 3-2) included general process descriptions, development process descriptions, process improvement strategies, documentation pertaining to roles and responsibilities, and the requirements engineering process description. Generally one could say that the line documentation was a collection of documents in which the official processes and responsibilities were specified.

3.3.4. Validity Evaluation

In this section we discuss the threats to this investigation. We base this on the discussion of validity and threats to research projects presented in Wohlin et al. [12]. One type of threats mentioned in [12] is not relevant, since the investigation is conducted in an industrial environment. The threat not considered is construct validity, which mainly is concerned with the mapping from the real world to the laboratory. The investigation presented here is however conducted in the real world. The validity threats considered are: conclusion, internal and external validity threats respectively.

3.3.4.1 Conclusion validity

The questionnaire used for the structured interviews was validated through preliminary testing and proofreading by several independent parties, this to avoid factors like poor question wording and erroneous formulation.

Each case study interview was conducted without any break. Thus the answers were not influenced by internal discussions about the questions during e.g. coffee breaks.

The sampling technique used for the case studies, i.e. projects selected and interview subjects for every project, can pose a threat to the validity of the investigation. The projects may not be totally representative for the projects conducted at DHR, in a similar manner the interview subjects may also not be representative for the role they represent. Three things alleviate these potential threats. First the fact that three cases was studied, and this also gives us three representatives interviewed for each role identified. The exception to this is the Orderer in project B. Third is the triangulation effect, i.e. data from different data sources is used for validation.

3.3.4.2 Internal Validity

The use of a digital recorder during the interviews could be considered as a problem due to the fact that certain people may feel constrained to answer differently if an interview is recorded. This potential problem was alleviated by the guarantee of anonymity as to all information divulged during the interview, and that the recordings are only to be used by the researchers.

3.3.4.3 External Validity

The external validity is concerned with the ability to generalize the results. This is not a main threat in the investigation, since the objective is not to generalize the improvement issues to another environment. The important generalization here is whether the applied approach for identifying improvement issues is possible to apply in other environments. There is nothing in the approach that makes it tailored to the specific setting ad hence the approach should be useful at other small and medium sized enterprises that would like to identify improvement issues in the requirements engineering process.

3.4. Results

3.4.1. Project Study

3.4.1.1 Case Study Interviews (A)

Data from the case study interviews resulted in nine general improvements issues summarized in Table 3-1. In the first column there is a unique id for each improvement issue, the second column holds the name of each issue, and the last column houses the support number of each issue. The support number is the total number of times an issue was brought up during the interviews, i.e. issue 1 has a support number of 5, meaning that five out of eleven people interviewed brought up this issue during the open questions (see section 3.3.2.1).

Issue-1: A central issue for improvement is how the requirements are specified, contents of each requirement and to establish a detailed enough level of description. This relates to usability and understandability of the requirements (how well a specified requirement depicts what it is intended to), and to comparability between requirements.

Table 3-1. Improvement issue case study.

Issue Id	Improvement Issue	Support number
1	Abstraction level & Contents of requirements	5
2	Requirements overview in projects & over project boundaries	4
3	Requirements prioritization	4
4	Requirements upkeep during & post project	4
5	Requirements responsible/owner with overview of the requirements	3
6	Roles and responsibilities RE process	2
7	System test performed against requirements	2
8	Customer relations during RE	2
9	RE process/methods	3

Issue-2: Requirements overview in projects and over project boundaries is an issue viewed as important. In order to facilitate this overview three main other issues have to be satisfied, i.e. Issue-1, Issue-4 and Issue-5. Figure 3-4 illustrates this, where the relations between the nine issues can be viewed. Issue-2 is *dependent on* Issue-1, Issue-4 and Issue-5 (denoted by the arrows). The relations in Figure 3-4 can be several levels deep, e.g. Issue-2 is

dependent on Issue-4 which in turn is dependent on Issue-5 and so on. Below only the direct relations are addressed during the description of each issue.

Issue-3: In order to meet the demands posed on projects by resource and time limitations, requirements should be prioritized. The level of prioritization should reflect the need on a project basis. Prerequisites for this issue are that the requirements are specified in an adequate way and at a comparable level of abstraction (Issue-1), and that there is an overview of the requirements in question (Issue-2).

Issue-4: The requirements (requirements specification) should be kept up-to-date during and post project. The point of keeping the requirements 'alive' during the project, and not discard them after a certain point in the project, was made.

Issue-5: There should be a person(s) responsible for the RE as a whole that has an overview of the requirements. This role should further be the owner of the requirements, i.e. be up-to-date on the requirements and have executive powers and responsibilities pertaining to change to and addition of new requirements. This issue has an obvious relation to Issue-6.

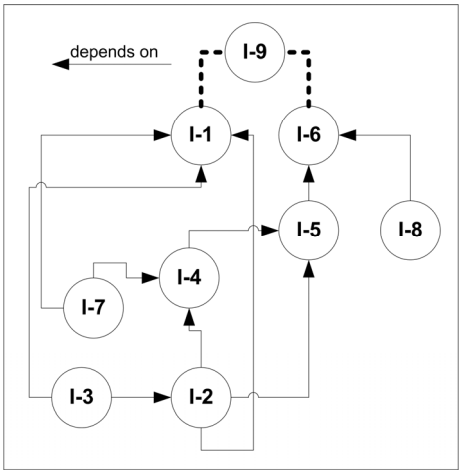


Figure 3-4. Dependency diagram.

Issue-6: The roles and responsibilities of project members and other stakeholders, e.g. customer, should be clearly and unambiguously defined pertaining to RE.

Issue-7: System tests should be performed against the requirements. Prerequisites for this are that the requirements specification is kept up to date during the project (and post project in some cases) (Issue-4), and that the requirements are specified adequately and at a specified level of abstraction.

Issue-8: The relations and particularly the flow of information between the customer (a group name used for all executive stakeholders, i.e. stakeholders representing the customer in a formal and executive capacity [11]) and the project team vary with project. There is a general view that the information from the customer is passed through too many filters before it reaches the project team. In addition to this the customer is often not accessible (at least not directly) for elicitation and clarification purposes during a project. A prerequisite identified here for this issue is that roles and responsibilities during RE are defined (Issue-6) (see Figure 3-4), i.e. all persons involved in the RE process (including the customers) are stakeholders, thus they are a part of the responsibility paradigm as well as the members of a project team.

Issue-9: This issue refers to the view that there is a lack of a formalized and/or structured RE process, or methods supporting certain tasks performed during RE at DHR. Issue-9 is to some extent outside of the dependency diagram (see Figure 3-4) as far as the level of abstraction. All issues except Issue-9 can be described as more or less specific tasks to be performed or rules to be upheld, while Issue-9 rather is an 'umbrella'

issue under which the rest can be sorted. Many of the other issues contributes to Issue-9, this is illustrated in Figure 3-4.

3.4.1.2 Case Study Project Documentation (B)

Looking at the project documentation in the case study projects several of the improvement issues described in Section 3.4.1.1 are substantiated. Table 3-1 shows each issue (sorted by issue id), the cells in the B-column (B for triangulation point B, see Figure 3-2) with a ticked box denote the substantiation, and the note-column offers a short account for it. Four issues could not be directly substantiated by the project documentation, i.e. Issue-2, Issue-5, Issue-6 and Issue-8.

3.4.2. Line Study

3.4.2.1 Line Study Interviews (C)

Looking at the line interviews several of the issues described in 4.1.1. are substantiated (denoted by the ticked boxes in column C of Table 3-3, C standing for triangulation point C, see Figure 3-2). Four issues could not be directly substantiated by the line interviews, i.e. Issue-2, Issue-3, Issue-5 and Issue-8. It is important to notice that three new improvement issues are identified during the line interviews, i.e. Issue-10 to Issue-12.

Table 3-2. Improvement issues documentation.

B	Issue Id	Note
☑	1	The abstraction level and contents of the requirements tend to vary. This is seen within projects, and over project boundaries, i.e. requirements are not comparable.
	2	
☑	3	No requirements prioritization is documented, and no priority grouping can be observed.
☑	4	The update of the requirements specifications (RS) varies, however no RS was ever updated after the first 50% of the project calendar-time had elapsed.
	5	
	6	
☑	7	The problems with Issue-1 and Issue-4 make system test based on requirements difficult.
	8	
☑	9	The RE process/methods seems not to be good enough. This is based primarily on issues substantiated above, but the fact that the level of RE and the methods used varies with project, also indicates this.

Issue-10: There are no mechanisms or practices implemented for requirements reuse. There is reuse on the design/code/function/ solution levels however.

Issue-11: No traceability policies are implemented for requirements. In some cases it is possible to trace requirements to the initiating party (backward-from traceability) [5, 11], and to partially trace requirements through design documents to implemented components (forward-from traceability) [5, 11]. These traceability possibilities are

however dependent on individual knowledge by project team members and they are not documented.

Issue-12: There is no version handling of individual requirements. The requirements document itself is however updated and released with different versions.

Table 3-3. Improvement issues line interviews.

C	Issue Id	Note
☑	1	Abstraction level & Contents of requirements varies and may be inadequate for tasks such as system test.
	2	
	3	
☑	4	Outdated req. specifications make it difficult to use the req. document.
	5	
☑	6	The roles of the interviewed as to their roles and responsibilities in RE are not clear.
☑	7	System test are seldom performed against requirements due to Issue-1 and Issue-4.
	8	
☑	9	See Issue-1, Issue-2, Issue-4 to Issue-8
Improvement Issues from line interviews		
NEW	10	Requirements reuse
NEW	11	Requirements traceability
NEW	12	Requirements version handling

3.4.2.2 Line Study Documentation (D)

Several issues identified thus far (see sections 3.4.1.1 and 3.4.1.2) were not substantiated by the line documentation, i.e. Issue-2 and Issue-4 to Issue-8. Table 3-4 offers an overview of this.

Table 3-4. Improvement issues line documentation.

D	Issue Id	note
☑	1	The template used for specifying requirements is fairly abstract, i.e. on a high level of abstraction. There are no detailed instructions and/or no detailed examples.
	2	
☑	3	No instructions/help/regulations/method description exists to aid in the prioritization of requirements.
	4	
	5	
	6	
	7	
	8	
☑	9	see Issue-1, Issue-3
Improvement Issues from line interviews		
☑	10	No policies for requirements reuse exist.
☑	11	No policies for requirements traceability exist.
☑	12	No policies for requirements version handling exist.

3.4.3. Triangulation of Results

Table 3-5 offers an overview of all issues presented, and information about substantiation from the different data sources, i.e. A to D. Nine out of a total of twelve issues were substantiated by two or more data sources (see Section 3.3.1), and are considered to be triangulated.

Table 3-5. Triangulation of improvement issues.

Issue Id	A	B	C	D	TOTAL
1	☑	☑	☑	☑	***4***
2	☑				***1***
3	☑	☑		☑	***3***
4	☑	☑	☑		***3***
5	☑				***1***
6	☑		☑		***2***
7	☑	☑	☑		***3***
8	☑				***1***
9	☑	☑	☑	☑	***4***
10			☑	☑	***2***
11			☑	☑	***2***
12			☑	☑	***2***

Issues 2, 5 and 8 each have a total triangulation value of one, i.e. the proof for the issues could not be distinctively identified in the project documentation (B), were not mentioned during the line interviews (C), and could not be found in the line documentation (D). Thus, these are not viewed as being triangulated.

3.5. Relation to State-of-the-art

In this section each triangulated issue (see Section 3.4.3) is posted with a description of how some sources in state-of-the-art can contribute in resolving them.

Issue-1: *Abstraction level & Contents of requirements.* Requirements should be stated in a clear and unambiguous way. Although there are many textbooks describing how this is done many specifications are inadequate [1, 5, 13, 14]. Requirements are in addition totally dependent on the customer, i.e. the stakeholders making the demands. Here it is assumed that the elicitation process and stakeholder consulting [5, 11, 15] is not a factor, i.e. all information is available and understood by the party writing the specification. This is however seldom the case, i.e. lack of domain knowledge and miscommunication between developers and customers is often a factor influencing the quality of a requirement [16].

Natural language (NL) specifications are commonly used and there are several ways in which to approach this, spanning from the psychology behind getting the right requirements on paper [17], to how important linguistics [18] and ethnography [19] are in the context.

Formal specifications [20] are an alternative (or a complement) to NL specifications. Either way several other techniques can be used in combination with both, e.g. prototypes, scenarios, state diagrams, use cases, data-flow diagrams and so on, to validate and/or clarify a certain requirement [5, 10, 13, 14].

Issue-3: *Requirements prioritization.* To have a shortage of requirements is seldom a real issue, quite the opposite. This makes prioritization of the requirements a valuable tool in the initial stages of a project. There are several methods for prioritization [21, 22], one of

the most well known is the Analytic Hierarchy Process or AHP for short [23]. The main issue with this method is scale-up issues, i.e. prioritizing of a lot of requirements is rather time consuming [21]. Getting the customers (or other departments) to give input, i.e. do some prioritization of their own, and show what is important to them, is an approach that could be beneficial, and combined with ones own prioritization efforts [24]. The exact method used depends on any number of factors, e.g. like amount of requirements, time devoted to prioritization and so on. The main issue here is not to recommend a prioritization method, but to recommend that a method be used.

Issue-4: *Requirements upkeep during & post project.* Requirements change. Making the requirements reflect these changes is crucial for several reasons. It is a prerequisite for being able to conduct tests based on the requirements (Issue-7). Furthermore in order to achieve any reuse (Issue-10) or re-prioritization of requirements this issue has to be resolved. Figure 3-5 illustrates an updated version (only triangulated issues) of dependencies between issues. Issue-4 is here dependent on Issue-6, i.e. that there is a clear and unambiguous definition of the roles and responsibilities pertaining to RE, this includes who should be responsible for making sure that the requirements are kept up to date.

Issue-6: *Roles and responsibilities RE process.* The importance of making roles and responsibilities clear is not an issue reserved for RE, but pertinent in any organization involved in software development [25]. One issue for RE is to identify the tasks to be conducted [5, 11], and then assign responsibility for them. Typically one would assume that there is one person responsible for the RE pertaining to a certain project, this does not however necessarily imply that this individual has to perform all tasks in question. The tasks could be delegated to any number of project team members. It is important that there be no question about who should perform what tasks. The delegation and division of tasks should be documented and rooted in the project organization in the same manner as the general project model.

Issue-7: *System tests performed against requirements.* Requirements are typically described as *what* is to be delivered to a customer [5]. By performing system tests on requirements it is possible to validate if the implemented features are really based on the requirements (functional requirements) [14], and if the system complies with other requirements (non-functional requirements) [14]. Making test scenarios/cases during initial requirements engineering can also be a good way of validating the requirements themselves at an early stage [5].

Issue-9: *RE process/methods.* All issues are included here. There are several RE processes suggested in literature [5, 11, 14], all more or less based on similar concepts and ideas. The main question for an organization to decide is what tasks are necessary to achieve their goals.

Issue-10: *Requirements reuse.* Reuse is a fairly common term in software engineering and often refers to the reuse of everything from code to architectures. The reuse of requirements however is not as commonly adopted [26]. This does not mean that the potential benefits of such a reuse are less, quite the opposite actually. By reusing requirements (traceability is a prerequisite, Issue-11, see Figure 3-5 [27]) everything from the requirement itself to analysis, design, implemented components, test cases, scenarios, and use cases etc. can be reused [5].

Issue-11: *Requirements traceability.* Several types of traceability could be identified in RE [11]. Only two are mentioned here, i.e. *Backward-from traceability* denoting a link from

requirement to their source in other documents or people, and *Forward-from traceability* denoting a link from requirements to the design and indirectly to the implemented components [5, 11]. The first is important for verifying what sources there are for a certain requirement [28], and the second for making reuse of more that just the requirements themselves possible. There are however several views on requirements traceability [5, 11, 29] which should be studied before a distinction about what model for traceability should be implemented in an organization.

Issue-12: *Requirements version handling.* This issue could effectively be sorted under the umbrella of traceability [5, 11], but is separated due to the fact that this issue was identified separately from Issue-11 during the line interviews (see Section 3.4.2.1). Version handling of code in classes, components etc. is widely used in software engineering. The benefits are many, the most obvious one being the ability to go back and look at/use/compare to previous versions of an item. The same is true for a requirement, i.e. version handling enables an evolutionary view. In order to obtain this some kind of version handling system must be used, e.g. as a part of a requirements handling application.

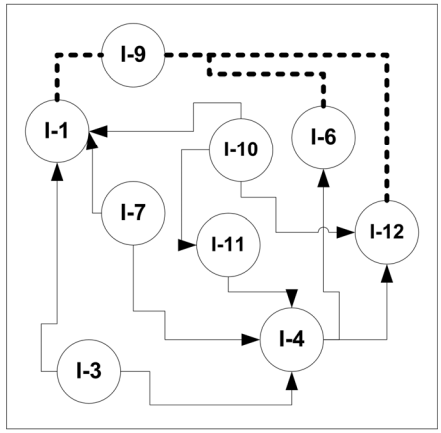


Figure 3-5. Dependency diagram (updated).

3.6. Conclusions

By conducting two separate studies, i.e. the project study and the line study, we were able to identify several improvement issues in DHR's RE process. Using the lightweight triangulation approach nine (out of a total of twelve) improvement issues were triangulated. The result was nine tangible issues that positively identified improvements to be made in the RE process at DHR.

A good enough RE process is crucial in order for development projects to be successful, it is however also crucial to have the ability to plan for the evolvement of the process. Process improvement is however often a costly enterprise tying up valuable resources during the improvement, and subsequent indoctrination of the improved process often means that substantial educational measures have to be taken. Ad hoc improvement measures, i.e. trying to improve the RE process as a whole, can mean that issues not crucial are improved/implemented, thus the process improvement costs are higher. Cost is a central concern for all organizations, and especially in the case of SME:s.

By identifying tangible and crucial improvement issues through eliciting information from personnel, management and documentation, and subsequently triangulating the issues identified, validate that the right problems are prioritized. A prioritization of issues gives the organization the raw material for the construction of a process improvement

plan that addresses primarily crucial issues. Secondary issues, e.g. the ones identified during the studies but not triangulated, can be addressed at a later stage or the improvement approach could be used again.

It is also important to realize that the nature of the lightweight approach described in this chapter bases its results on the knowledge and views of the people in the organization whose RE process is to be improved. This is beneficial in terms of rooting the coming changes in the organization at all levels.

Furthermore the usage of a lightweight triangulation approach to find improvement issues has a moderate price tag attached to it, as well as being an alternative to a more extensive evaluation process. This makes the use of a lightweight approach beneficial for SME:s. For a large enterprise the cost of using the lightweight approach would probably be higher. This is based on the assumption that the studies would be larger and the analysis more resource consuming, unless it is applied to a part of the enterprise. The investigation presented here took about 150 person-hours to complete.

The main risk in using the lightweight approach described in this chapter is that issues may escape identification. This in turn can cause critical issues to remain unidentified and unattended. However, two things alleviate this risk. One, the identification of the dependencies between the issues was mapped, establishing an understanding of the interrelations of the issues in question. These dependencies help establish that there is not any crucial step missing as the issues are identified, as well as being a useful tool for the planning of the process improvement itself.

Second, following a process improvement plan, and the execution of such a plan, the studies and triangulation can be repeated. This would hopefully yield confirmation that the addressed issues were turned into non-issues, as well as identifying new improvement issues. The new issues could be issues not prioritized before, but issues missed the first time around could also be caught.

Future work consists of taking the next step, i.e. ascertaining what improvement issues are to be addressed first. This can be done using several different approaches. One is basically to base the priority of the issues on the relations in the dependency diagrams. Another is taking advantage of the expertise present in the organization that has been evaluated, i.e. making the decision up to the same people that were involved in the studies, and letting them prioritize the issues using some prioritization scheme, e.g. AHP.

Process improvement is an ongoing process with several steps, but the first one should be to identify what to improve.

3.7. References

1. Bray IK (2002) An Introduction to Requirements Engineering. Addison-Wesley, Dorset.
2. Glass RL (1998) Software Runaways. Prentice Hall, Upper Saddle River NJ.
3. (2003) http://sme.cordis.lu/home/index.cfm. Last Accessed: 2003-05-05.
4. Robson C (2002) Real World Research: A Resource for Social Scientists and Practitioner-Researchers. Blackwell Publishers, Oxford.
5. Sommerville I, Sawyer P (1999) Requirements Engineering: A Good Practice Guide. Wiley, Chichester.
6. (2003) http://www.comp.lancs.ac.uk/computing/research/cseg/projects/reaims/index.html. Last Accessed: 2003-05-01.
7. (2002) http://www.tickit.org/. Last Accessed: 2002-10-01.
8. Bratthall L, Joergensen M (2002) Can You Trust a Single Data Source Exploratory Software Engineering Case Study? Empirical Software Engineering 7(1):9-26.
9. Kitchenham B, Pfleeger SL, Pickard L (1995) Case Studies for Method and Tool Evaluation. IEEE Software 12(4):52-63.
10. Sommerville I (2001) Software Engineering. Addison-Wesley, Essex.
11. Gorschek T, Tejle K, Svahnberg M (2002) A Study of the State of Requirements Engineering in Four Industry Cases. In Proceedings of Software Engineering Research and Practice in Sweden (SERPS'02), Blekinge Institute of Technology, Karlskrona, pp. 111-119.
12. Wohlin C, Runeson P, Höst M, Ohlson MC, Regnell B, Wesslén A (2000) Experimentation in Software Engineering: An Introduction. Kluwer Academic, Boston.
13. Kotonya G, Sommerville I (1998) Requirements Engineering: Processes and Techniques. John Wiley, New York.
14. Robertson S, Robertson J (1999) Mastering the Requirements Process. Addison-Wesley, Harlow.
15. Sharp H, Finkelstein A, Galal G (1999) Stakeholder Identification in the Requirements Engineering Process. In Proceedings of the Tenth International Workshop on Database and Expert Systems Applications, IEEE, Los Alamitos CA, pp. 387-391.
16. Hanks KS, Knight JC, Strunk EA (2002) Erroneous Requirements: A Linguistic Basis for Their Occurrence and an Approach to Their Reduction. In Proceedings of 26th Annual NASA Goddard Software Engineering Workshop, IEEE, Los Alamitos CA, pp. 115-119.
17. Rupp C (2002) Requirements and Psychology. IEEE Software 19(3):16-19.
18. Potts C (2001) Metaphors of Intent. In Proceedings of Fifth IEEE International Symposium on Requirements Engineering, IEEE, Los Alamitos CA, pp. 31-38.

19. Sommerville I, Rodden T, Sawyer P, Bentley R, Twidale M (1992) Integrating Ethnography into the Requirements Engineering Process. In Proceedings of IEEE International Symposium on Requirements Engineering, IEEE, Los Alamitos CA, pp. 165-173.
20. Potter B, Sinclair J, Till D (1996) An Introduction to Formal Specification and Z. Prentice Hall, London.
21. Karlsson J, Wohlin C, Regnell B (1998) An Evaluation of Methods for Prioritizing Software Requirements. Information and Software Technology 39(14-15):939-947.
22. Karlsson J (1996) Software Requirements Prioritizing. In Proceedings of the Second International Conference on Requirements Engineering, IEEE, Los Alamitos CA, pp. 110-116.
23. Saaty TL, Vargas LG (2001) Models, Methods, Concepts & Applications of the Analytic Hierarchy Process. Kluwer Academic Publishers, Boston.
24. Regnell B, Host M, Natt och Dag J, Hjelm T (2001) An Industrial Case Study on Distributed Prioritisation in Market-Driven Requirements Engineering for Packaged Software. Requirements Engineering 6(1):51-62.
25. Kivisto K (1999) Roles of Developers as Part of a Software Process Model. In Proceedings of the 32nd Annual Hawaii International Conference on Systems Sciences, IEEE, Los Alamitos CA, pp. 1-19.
26. Lam W (1998) A Case-Study of Requirements Reuse through Product Families. Annals of Software Engineering 5(X):253-277.
27. Roudies O, Fredj M (2001) A Reuse Based Approach for Requirements Engineering. In Proceedings of ACS/IEEE International Conference on Computer Systems and Applications, IEEE, Los Alamitos CA, pp. 448-450.
28. Gotel O, Finkelstein A (1997) Extended Requirements Traceability: Results of an Industrial Case Study. In Proceedings of the Third IEEE International Symposium on Requirements Engineering, IEEE, Los Alamitos CA, pp. 169-178.
29. Ramesh B, Jarke M (2001) Toward Reference Models for Requirements Traceability. IEEE Transactions on Software Engineering 27(1):58-94.

Chapter Four

Paper III

Packaging Software Process Improvement Issues – A Method and a Case Study

Tony Gorschek and Claes Wohlin

In (to be published) Software: Practice & Experience, Wiley.

Abstract

Software process improvement is a challenge in general and in particular for small and medium sized companies. Assessment is one important step in improvement. However, given that a list of improvement issues has been derived, it is often very important to be able to prioritize the improvement proposals and also look at the potential dependencies between them. This chapter comes from an industrial need to enable prioritization of improvement proposals and to identify their dependencies. The need was identified in a small and medium sized software development company. Based on the need, a method for prioritization and identification of dependencies of improvement proposals was developed. The prioritization part of the method is based on a multi-decision criteria method and the dependencies are identified using a dependency graph. The developed method has been successfully applied in the company, where people with different roles applied the method. The chapter presents both the method as such and the successful application of it. It is concluded that the method worked as a means for prioritization and identification of dependencies. Moreover, the method also allowed the employees to discuss and reason about the improvement actions to be taken in a structured and systematic way.

4. Process Improvement (Pre-) Planning

4.1. Introduction

The production of high quality software with a limited amount of resources, and within a certain period, is the goal of most software producing organizations. They vary from large corporations with thousands of engineers, to small ones with only a handful. All of them, regardless of location, size or even success-rate, are largely dependent on their software *processes* [1] to reach their goals.

It stands to reason that continuous evaluation and improvement of an existing process (Software Process Improvement - SPI [1]) is crucial to ensure that the organization is successful in its pursuits of quality, and in order to be effective enough to stay competitive in the world of business.

The work presented in this chapter introduces a structured way in which software development practitioners (whose organization is subject to SPI efforts), and SPI practitioners alike, can make dependency adherent prioritizations of identified improvement issues (findings from process assessments). The goal is to give small and medium sized enterprises (SMEs) a tool to focus their SPI efforts, and not to present "yet another method" as such. The work presented here should complement already existing SPI frameworks with a modular addition intended to minimize some of the main issues with SPI efforts identified in literature (see Section 4.2.1).

The chapter is structured as follows. Section 4.2 gives some background information pertaining to SPI concerns and the motivation behind the work presented in this chapter. Section 4.3 introduces the "Dependency Adherent Improvement Issue Prioritization Scheme" or "DAIIPS" for short, and the techniques on which it stands. Section 4.4 presents a study using DAIIPS in an industry SPI effort. The results from the industry study are subsequently validated through an additional study performed in academia, and the results from the two studies are compared for validation purposes. Section 4.5 has the discussion and the conclusions.

4.2. SPI – Related Work and Motivation

An SPI scheme is usually based in four fairly straightforward steps, "evaluation of the current situation", "plan for improvement", "implement the improvements", "evaluate the effect of the improvements", and then the work takes another cycle (see Figure 4-1, inspired by [2]).

[1] A sequence of steps performed for the purpose of producing the software in question (IEEE-STD-610).

Most well known process improvement (quality improvement/assurance) frameworks are based on this general principle. From the Shewart–Deming "Plan-Do-Check-Act" paradigm [3] and Basili's "Quality Improvement Paradigm" [4], to standards like CMM [5], CMMI,[6] ISO/IEC 15504 (a.k.a. SPICE) [7] and the Software Engineering Institutes IDEAL SPI Guide [8].

Looking at the frameworks mentioned above (primarily CMMI, CMM, SPICE and IDEAL) they have been used for SPI over a number of years (except for CMMI) and there are quite a lot experiences reported from industry.

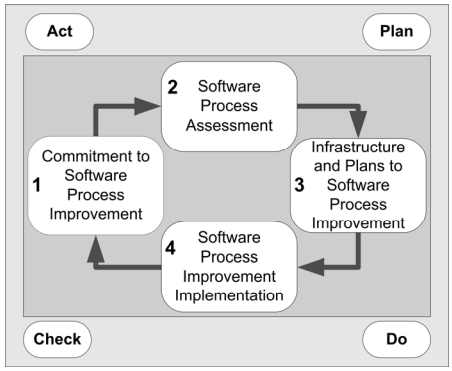


Figure 4-1. Generic process improvement scheme.

The main issues seem to be *cost* and *time*. An assessment-improvement cycle is often rather expensive and time consuming [9]. A typical SPI cycle using e.g. CMM can take anything from 18 to 24 months to complete [10] and demand much resources and long-time commitments in order to be successful. In addition to being time consuming many view extensive SPI frameworks as too large and bulky to get an overview of and to implement [2, 11, 12].

However, this is not the same as saying that frameworks like e.g. CMMI are inapplicable in general. Organizations with time and resources available report high return on their investment over an extended time period (results indicate both lower costs and higher customer satisfaction) [2, 11]. The main issue here is rather whether or not a small and medium sized enterprise (SME) has the ability to commit to a large-scale SPI project spanning over a long time period as far as the work and pay-off is concerned.

There are examples of attempts to adapt larger SPI frameworks to be more suitable for SMEs. The IMPACT project [13] reports an initiative to use elements from proven technologies like CMMI and SPICE to implement a lightweight SPI framework. Another example is presented in [14] where IDEAL is adjusted for use in SMEs. The work presented in this chapter is meant to add to this effort of making SPI available for SMEs.

4.2.1. SPI Concerns and DAIIPS Goals

In addition to the factors mentioned above, which may have a negative impact on SPI efforts (pertaining mainly to SMEs) i.e. (I) Time (long-term work, long-term gain) and (II) Cost/Resources (the nature of large SPI frameworks imply commitment of much resources over an extended period of time), there are a number of more general critical aspects (not directly linked to SMEs) presented in literature [15-19].

Some of the central are[2] (III) Commitment (to the SPI effort by management, middle management and the staff e.g. engineers), (IV) Focus (on the SPI effort with clear and well-defined goals) and (V) Involvement (in the SPI work by staff).

If we look at DAIIPS the first and foremost motivation for the development of the framework was to give organizations with limited resources for SPI a chance to choose *what to do first* based on their needs.

The need for this was first recognized in relation to cooperative SPI work conducted at DanaherMotion Särö AB (see Section 4.4), a medium sized company about to undertake process assessment and process improvement work of their requirements engineering process. With limited resources available it was crucial that the improvements be manageable in size and time, thus not posing a threat against their core activities (the business of producing income generating products).

The idea was to prioritize and package improvement issues in smaller and more manageable groups based on how important they were to the organization, and the dependencies between the improvement issues. This division of a potentially large amount of improvement issues (depending on what was found during the software process assessment phase) was to help organizations take small steps towards quality improvement at a manageable rate and with a realistic scope. This addresses the factors of (I) time, smaller packages of improvement issues can be implemented and evaluated faster (the results are felt in the organization sooner rather than later), and (II) cost/resources in terms of smaller steps each costs less and large resources do not have to be committed over long periods of time. By dividing a potentially large amount of improvement issues into smaller packages for implementation one could argue that it is easier to define clear and manageable goals for the SPI activities, e.g. results lie closer in time and there are a limited number of issues that are addressed (speaking of aspect IV). Many SPI efforts fail before they start, i.e. an assessment is made, then the work stops and no real improvements are made. A lack of follow-thorough is usual, and this may be contributed to several of the reasons mentioned above, of which commitment of time and resources are not the least [9]. This is the motivation behind DAIIPS, to offer SMEs a framework to choose what to do, i.e. limit the commitment to a manageable level.

DAIIPS is directly dependent on the involvement, contribution and expertise of the personnel involved in the SPI activity, namely a selection of representatives from the organization (as are most SPI paradigms). This is the nature of DAIIPS, i.e. all the decisions (regarding priority and dependencies) are made by the people working with the current process in the organization. The direct nature of the involvement in the work and decision-making speaks to securing the SPI work in the minds of the very people that the SPI activities influence the most, i.e. relating to aspects III and IV.

DAIIPS is not an absolute method, which should be followed to the letter, rather a framework for gathering data in a certain way, formatting it and presenting the results in way that should ultimately act as decision support for SPI activities.

[2] In addition to these there are other factors that influence the success of SPI activities. However these are not elaborated upon here due to the fact that they are out of the scope of this chapter.

4.2.2. DAIIPS and Modularity

As mentioned earlier DAIIPS is not meant to compete with any SPI model, in fact DAIIPS is just a structured way to prioritize and check dependencies amongst the improvement issues already identified using a software process assessment (SPA) tool, e.g. CMM, CMMI, SPICE or IDEAL. Thus, the prerequisite for using DAIIPS is that an assessment has been made and that the improvement issues are documented and explained in such a way that the constituents of the SPI targeted organization can understand them. Given this restriction almost any assessment framework/standard/method can be used, including lightweight methods like the one presented in Chapter 3.

4.3. DAIIPS – An Overview

DAIIPS consists of three basic steps, (A) prioritization, (B) dependency mapping, and (C) packaging (illustrated in Figure 4-2). Steps A and B are performed in sequence at a workshop, while step C is performed at a later stage.

As mentioned earlier improvement issues identified during the SPA stage of the SPI effort are used as input. Each of the steps is described in further detail below.

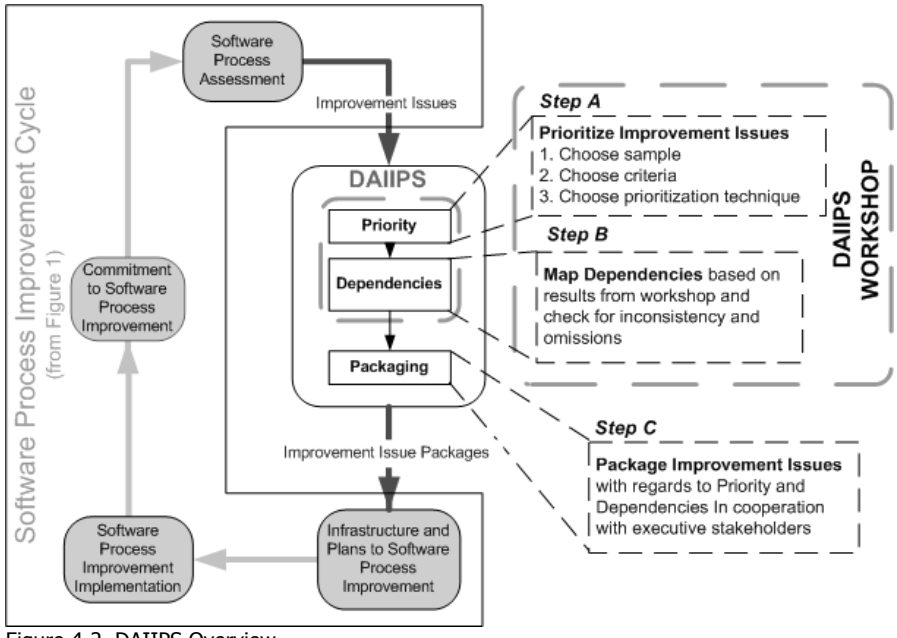


Figure 4-2. DAIIPS Overview.

4.3.1. Prioritization (Step A)

The idea behind prioritization of the improvement issues in DAIIPS is to introduce an explicit choice amongst the issues identified during assessment, i.e. all improvement issues are channeled into the prioritization unordered and with no "importance" attributes attached after the assessment. A number of sub-steps should be performed in prioritization (see Figure 4-2), each described below.

For an example of the prioritization (Step A) and results from an industry case, see Section 4.4.3.2.

4.3.1.1 Sample

There should be a conscious decision behind the choice of what stakeholders should be invited to participate in the prioritization step. The sample could be based on a sampling technique, what technique depends on applicability in the case at hand. One alternative could be to invite all of the participants of the SPA performed earlier (if this is a manageable number of people). The participants of the SPA should have been selected in line with some plan (although this cannot be assumed). However, if the SPA sample is deemed good-enough it can be reused, saving time both in regards to selecting the sample and introducing the sampled subjects to the SPI activity. There are quite a lot of sampling techniques available, see e.g. [20, 21] for examples.

The sample size is also important and once again it should be possible to look at the sample size used during the SPA. A main issue is to get a sample large enough making it possible to generalize, i.e. a large variability in the population (typical for a software development organization) demands a larger sample.

Quota sampling and Stratified random sampling [20] are both based on having elements (strata/groups) from which a representation is wanted, e.g. programmers, middle management, testers and so on.

Through sampling certain views (agendas) could be premiered over others, i.e. by having an overrepresentation of a certain group (role) in the sample. An example of this could be having an overrepresentation of developers – thus adding weight to their influence on the prioritization.

Despite of how the sampling is conducted it should be a conscious action with regard to the consequences it may have on the prioritization.

For an example of sampling performed in an industry case, see Section 4.4.2.1.1.

4.3.1.2 Criteria

There are characteristics of the process that we want either to minimize or to maximize. Quality, time, cost and risk are three examples of criteria. When prioritizing improvement issues it is done in relation to a criterion. If quality is the criterion (improve/maximize quality) the prioritization is skewed in one direction, if time is the criterion other aspects may be more important. Which criterion is chosen must be up the SPA team (including management), and should be based on strategic goals of the organization subject to the SPI as well as the findings of the SPA.

The choice of criteria should be conveyed to the participants beforehand (and consensus reached about what a criterion means) in order for the prioritization to be done with the same focus in mind.

For an example of criteria formulation in an industry case, see Section 4.4.2.1.1.

4.3.1.3 Prioritization Techniques

Several techniques can be used for the prioritization of improvement issues. The main idea is to have a structured way in which prioritization can be performed in the same manner (technique) and with the same intent (criterion) by all participants. Furthermore the results from the prioritization should be comparable to each other, this to enable a compilation of the results.

In this chapter we briefly present one prioritization method used in our study (see Section 4.4), and mention three more. Note that AHP is just one of several available prioritization techniques that may be used.

The Analytical Hierarchy Process (AHP) [22] is a method using scaled pair-wise comparisons between variables, as illustrated in Figure 4-3.

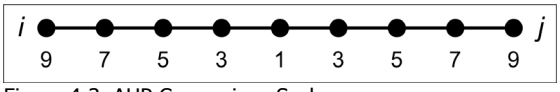


Figure 4-3. AHP Comparison Scale

Here the variables are ***i*** and ***j*** and the scale between them denotes relative importance. The importance ratings can be seen in Table 4-1 below.

Table 4-1. AHP Comparison scale.

Relative intensity	Definition	Explanation
1	Of equal importance	The two variables (i and j) are of equal importance.
3	Slightly more important	One variable is slightly more important than the other.
5	Highly more important	One variable is highly more important than the other.
7	Very highly more important	One variable is very highly more important than the other.
9	Extremely more important	One variable is extremely more important than the other.
2, 4, 6, 8	Intermediate values	Used when compromising between the other numbers.
Reciprocal	If variable i has one of the above numbers assigned to it when compared with variable j, then j has the value 1/number assigned to it when compared with i. More formally if n i j = x then n j i = 1/x.	

As the variables have been compared the comparisons are transferred into an ***n x n*** matrix with their reciprocal values (***n*** is the number of variables). Subsequently the eigenvector of the matrix is computed. The method used for this is called *averaging over normalized column* and the product is the *priority vector,* which is the main output of using AHP for pair-wise comparisons.

AHP uses more comparisons than necessary, i.e. ***n x (n – 1) / 2*** comparisons, and this is used for calculating the consistency of the comparisons. By looking at the *consistency ratio (CR)* an indication of the amount of inconsistent and contradictory comparisons can be obtained. In general a CR of ≤ 0.10 is considered to be acceptable according to Saaty [22], but a CR of > 0.10 is often obtained. There has been some debate as to the applicability of results that have a CR of > 0.10, see [23] and [24], and this is an ongoing debate. A rule of thumb is that a CR of ≤ 0.10 is optimal, although higher results are often obtained in the real world. Further details about AHP can be found in [22] and [25].

AHP Example (Using AHP to prioritize mobile phone features)

answer	feature																		feature
1	SMS	9	+	7	+	5	+	**3**	+	(1)	+	3	+	5	+	7	+	9	Color display
2	SMS	9	+	7	+	5	+	3	+	(1)	+	3	+	**5**	+	7	+	9	WAP
3	SMS	9	+	7	+	5	+	3	+	(1)	**+**	3	+	5	+	7	+	9	Vibrating Call Alert
4	WAP	9	+	7	+	5	+	3	+	(1)	+	**3**	+	5	+	7	+	9	Color display
5	WAP	9	+	7	+	5	+	3	+	(1)	+	3	+	**5**	+	7	+	9	Vibrating Call Alert
6	Vibrating Call Alert	9	+	7	+	**5**	+	3	+	(1)	+	3	+	5	+	7	+	9	Color display

Having 4 features *[n=4]* to prioritize against each other would demand a table with 4 x (4 - 1) / 2 = 6 rows *[n x (n – 1)/ 2]*, i.e. require 6 answers in order to complete the AHP prioritization. Note that all features are compared to each other once. (The answers in our example are denoted by the black circles)
The CR (consistency ratio) comes from how consistent the answers are in comparison to each other, e.g. in the example above (answer 1) "SMS" is considered slightly more important than "Color display", (answer 2) "WAP" is considered highly more important than "SMS". The inconsistency comes in (answer 4) when "Color display" is considered slightly more important than "WAP". A consistent answer would be that "WAP" was more important than "Color display".
The more inconsistent answers, like in the example above, the higher the CR is for the prioritization.

The CR from the prioritization above is 0.19, which is well above the recommended limit proposed by Saaty. This example shows that a high inconsistency (i.e. CR > 0.10) is not especially difficult to obtain with only a few variables to prioritize, not to mention if you have e.g. 10 variables (10 variables means 45 prioritizations). The positive thing is that the CR actually indicates consistency, which can be seen as a quality indicator of a person's answers.

The Planning Game is a more informal way in which the improvement issues can be sorted and ordered according to priority. This method is used extensively in extreme programming [26] for feature and release planning, but can easily be used in prioritizing improvement issues.

Ranking involves putting the improvement issues in a list; where the higher in the list an issue is the more important it is deemed and vice versa. The result of this ranking is a list of ordered issues on an ordinal scale (it is possible to observe that e.g. issue ***I*** is more important than ***J*** but not how much more important it is).

The "100-points method" can be used to distribute tokens of some sort (e.g. "money") amongst the issues. The issues are prioritized with regards to how much "money" they receive. An example of this could be to give each person doing the prioritization $100 (or $1000 etc) to distribute amongst the improvement issues. The result from this method is a weighted prioritization on a ratio scale (the issues are ordered and a distance between them can be observed, e.g. issue ***I*** is $10 more important than ***J***). For further information see [27].

The primary goal with prioritization is to ascertain what improvement issues are considered important, and which can be put off for future SPI cycles. The choice of technique depends on e.g. how many issues you have to prioritize. AHP demands ***n x (n – 1) / 2*** comparisons, e.g. 45 comparisons for 10 issues. This amount is manageable, but when considering 20 issues (190 comparisons) it may start to get unmanageable. To get e.g. 10 professionals to sit down and do 190 comparisons (and be consistent) is not easy. Using the 100-point method and giving them the task of distributing $100 over the 20 issues may be more feasible in this case.

It is important however to realize that different prioritization methods have different pros and cons, e.g. AHP gives information about consistency and relative priority, whereas e.g. the planning game does not but is much faster in performing when dealing with many improvement issues. A comparison of prioritization techniques is provided in [28].

4.3.1.4 Weighing the Priorities after the Fact

As each contributor's results are gathered there is the possibility to weight the results, and hence enabling to decide on the importance of a certain viewpoint. An example of this could be a desire to premiere developers and their views by multiplying their results by 1.0, while using the multiplier of 0.6 for e.g. managers, effectively premiering one view over another. This is the same as discussed in Section 4.3.1.1, where the sample selection was as a weighting method, but this could also be done after the prioritization, independent of sample.

The danger of adding weights is evident, i.e. it could initiate conflicts, as well as belittle valuable views. The possibility of weighting results exists independent of prioritization technique used, but should only be performed after careful consideration, and with clear reasons.

4.3.2. Mapping Dependencies (Step B)

This step is aimed at mapping the dependencies between improvement issues. For an example of results obtained during a dependency mapping performed industry, see Section 4.4.3.3.

This step should involve the same participants as the prioritization (given that the prioritization participants were sampled from the organization in an adequate manner, see Section 4.3.1.1). Some variations may be necessary, i.e. being able to prioritize issues is not the same as being able to identify dependencies between them. Whether or not the prioritization sample can be reused depends largely on the constituents of the sample itself, i.e. what people are involved, where they work (department) and what their competences are. Identified dependencies are drawn (by each participant) between the

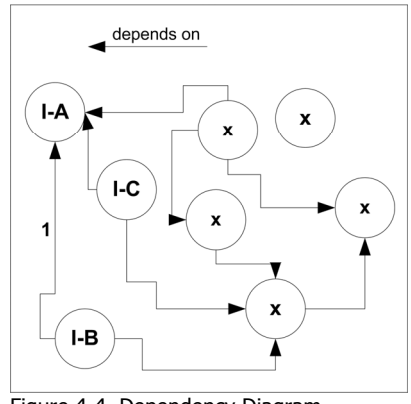


Figure 4-4. Dependency Diagram Example

issues with two major attributes registered, i.e. *direction* and *motivation*. The arrow itself denotes a dependency between two issues, e.g. ***I-B*** and ***I-A*** as illustrated in Figure 4-4, and the direction of the dependency can be seen through the direction of the arrow. In Figure 4-4 issue ***I-B*** depends on ***I-A*** (as does ***I-C***).

Dependency Example

A real life example of a potential dependency could be if there were two new practices to be introduced into an organization, e.g. *Programming in Java* and *Object Oriented Design*. Java is an object oriented programming language, therefore the practice of *Object Oriented Design* should be implemented first (to make use of Java as an object-oriented programming language), i.e. *Programming in Java* is dependent on having an *Object Oriented Design*. If this example is transferred to Figure 4-4 *Object Oriented Design* could be denoted by ***I-A*** and *Programming in Java* could be denoted by ***I-B***. The dependency is denoted by the arrow (1).

The motivation could in this case be "there is no point in introducing an OO programming language if we cannot design in an OO fashion…"

In addition to drawing the arrows a *motivation* for each arrow is registered. This is done to avoid arbitrary and vague dependencies and dependencies can be sorted and compared during the compilation of the results, i.e. it is possible to see if two participants have the same type of dependency between two issues, or if the same arrow denotes two different views. This is also a way in which different types (views) of dependencies can be elicited.

The result of this step should be a list of dependencies between the issues as well as their relative weight, i.e. how many times they are specified by the participants. Table 4-2 illustrates the results of an example dependency mapping. The dependencies present (e.g. I-B on I-A) is decided by what dependencies are identified. P 1, P 2, …, and P *n* denote the participants, and the numbers in each participant's row denotes if he/she stated the dependency or not (1 for YES). By summarizing the number of identifications of a certain dependency weights are ascertained. In this case I-B on I-A has a weight of 4, I-B on I-C has 3, and so on.

Table 4-2. Dependency Table

Dependency	P 1	P 2	P 3	P 4	P n
I-B on I-A	1	1	1	1	
I-B on I-C	1	1		1	
I-C on I-A		1		1	
I-C on I-E		1			
I-D on I-B	1	1	1		
I-*n* on I-*n*					

The weights are compiled and drawn into a dependency diagram, see Figure 4-5, which is like the one presented earlier in Figure 4-4, but with the addition of weights, and "white" issues (I-D and I-F) denoting issues that have no dependencies on them. Note that issue ***I-F*** has no dependencies on it, and furthermore is not dependent on any other issue either.

If an improvement issue dependency has a low weight, e.g. 1 (***I-C*** on ***I-E*** in Figure 4-5 has a weight of 1), *one* person has only identified the dependency. Such single occurrences may be a result of misunderstandings and/or other anomalies (given that e.g. 10 persons participate in the dependency mapping) and can be omitted to avoid having a great number of weak dependencies to take into consideration. However, all "anomalies" should be scrutinized by the SPI group before they are dismissed in order to assure that the dependency is not relevant. The SPI group should not dismiss dependencies unless a consensus can be reached about the issue.

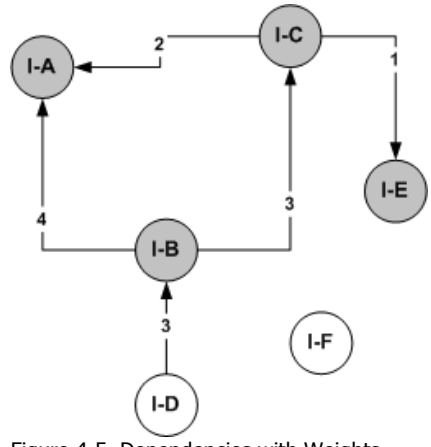


Figure 4-5. Dependencies with Weights

A good rule of thumb is that there should be a "threshold" set by the SPI group beforehand. The idea with mapping dependencies is to catch the important relations between improvement issues. Too many dependencies amongst a large number of issues can result in a dependency diagram that is unmanageable, and thereby useless. On what level this threshold should reside should be governed by the circumstances of the dependency mapping occurrence.

4.3.3. Package Improvement Issues (Step C)

Thus far, we have the priority of the improvement issues, and a mapping of the dependencies amongst them. The last stage of DAIIPS is to compile the information in order to generate packages of improvement issues that can be used as a base for planning and implementing an SPI cycle.

For an example of improvement issue packaging in an industry case, see Section 4.4.3.4.

Figure 4-6 is a continuation of our example from previous sections with all information gathered in one figure. The improvement issues ***I-A*** through ***I-F*** are augmented with relative priority (denoted by the number within parenthesis) and relations with weights. The larger numeral (upper left hand in each circle) denotes the rank of each improvement issue, i.e. ***I-C*** has the highest priority and

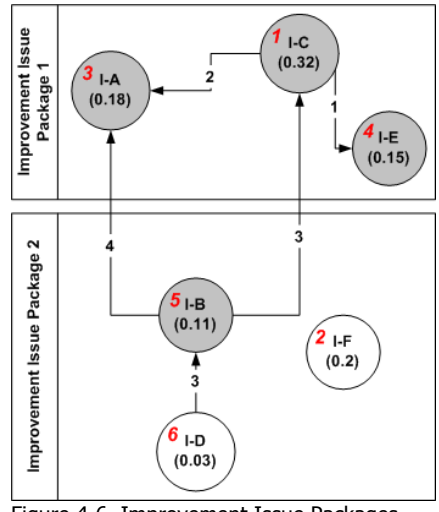


Figure 4-6. Improvement Issue Packages

I-D the lowest (note that all priority methods do not produce relative values of priorities, AHP is used in this example, see Section 4.3.1.3).

The packaging of the issues largely depends on what is entailed in each improvement issue. In our simple exemplification all issues demand equal time and resources to implement, thus the division of them into two packages is fairly simple. ***I-C*** is of highest priority, i.e. it governs that ***I-A*** and ***I-E*** be packaged with it due to dependencies on these two issues. Observe that issue ***I-F*** that is the second issue in relative priority is not included in package 1, i.e. the relations have precedence over priority in this example.

Priority, *dependencies* and *cost* (estimated resources demanded for implementation) are the primary variables that have to be adhered to during packaging. The packaging should be done to reflect the current and near-future needs of the organization (*priority*) as well as the available resources for process improvement (attention to *cost*), and last but not least attention should be paid to the order (*dependencies*) of the implementation of improvement issues. It seems that packaging one issue that cannot be realized until another is implemented, e.g. packaging issue ***I-A*** in package 2 in our example would be less than optimal. This is not to say that dependencies always should take precedence over the other variables. There are more ways in which the packaging can be performed, which is chosen is up to the SPI group. The main concern in this step is to package the issues so that a compromise between priority, dependencies and cost can be reached as the issues are packaged into units that are appropriate for an SPI cycle. The SPI group decides this with the individual improvement issues as a base, i.e. an initial estimation and planning has to be done to ascertain what each improvement issue entails as far as time and resources are concerned. In the example above the division is simple, which may not be the case in reality. Diagrams (like the one displayed in Figure 6) should act as a decision support tool for the SPI group when undertaking the task of establishing what is to be done first, second and so on.

As the three steps are completed the SPI group needs to perform a validity review. This should be an official action performed to ascertain that (1) the results from the prioritization and dependency mapping are good enough to proceed, and (2) a formal review of the documentation produced to ascertain that no omissions/mistakes crept in during the processing of the data gathered from the DAIIPS work. Reviews of dismissed dependencies (e.g. was the threshold set at a proper level), and to what extent is high inconsistency (CR) a threat against the ascertained priority of the improvement issues are examples of important issues. This validity review should help ensure that the quality of the material produced through the usage of DAIIPS is high.

4.4. Industry and Academia Study

This section presents the design and results of the industry and academia (validation) study performed in order to test the DAIIPS framework in an industry setting. The section is structured as follows. Sub-section 4.4.2 the designs of the studies are described. The results from the industry study are presented in Sub-sections 4.4.3.2 (Step A), 4.4.3.3 (Step B), and 4.4.3.4 (Step C), corresponding to DAIIPS three major steps (see Figure 4-2). Section 4.4.2.2 presents the academia (validation) study results, and in Sub-section 4.4.5 the industry and academia study are compared in a validation attempt.

The industry study described in this chapter is from a SPI project performed at DanaherMotion Särö AB (DHR), where the use of DAIIPS was a part of the SPI work.

DHR develops and sells software and hardware equipment for navigation, control, fleet management and service for Automated Guided Vehicle (AGV) systems. More than 50 AGV system suppliers worldwide are using DHR technologies and know-how together with their own products in effective transport and logistic solutions to various markets worldwide. The headquarters and R & D Centre is located in Särö, south of Gothenburg, Sweden. DHR has 85 employees. DHR is certified according to SS-EN ISO 9001:1994 (currently working on certification according to ISO 9001:2000), but there have not been any attempts towards CMM or CMMI certification.

DHR has a wide product portfolio, as the ability to offer partners and customers a wide selection of general variants of hardware and supporting software is regarded as important. Tailoring and especially lighter customer adaptation often follows the procurement and subsequent installation of a system. This in addition to development of new software and hardware makes it a necessity to plan, execute and manage a wide range of projects.

4.4.1. Related work

The need for continuous process improvement is well known at DHR. They identified the area of requirements engineering as a good candidate for improvement work. This chapter (the DAIIPS framework and the industry study presented below) is a product of research conducted at DHR based on their need for improvements in their requirements engineering process. Although DAIIPS was formulated in conjunction with requirements engineering process improvement work, it is not tailored towards a single sub-process area (e.g. requirements engineering), but can be used in any process and/or sub-process improvement effort.

A proceeding process assessment concentrated on the area of requirements engineering was conducted at DHR using the lightweight triangulation approach presented in Chapter 3.

The nine improvement issues (see Section 4.4.3) used as input to DAIIPS (as viewed in Table 4-5) came from this assessment of the requirements engineering process at DHR.

4.4.2. Study Design

This section covers the design of the industry study performed. The design is based on the DAIIPS steps described in Section 4.3, and can be seen as the preparation for the use of DAIIPS.

In addition to performing a study in industry a second study was performed in academia. The academia study's purpose was to validate some of the results obtained in the industry study, e.g. an effort to secure the external validity of the study (see Section 4.4.2.3.3), and to help in increasing the confidence that no important dependencies were missed during the industry study, i.e. the use of DAIIPS in an industry SPI effort. An overview of the studies is illustrated in Figure 4-7. Improvement issues (obtained during process assessment) are used as input for DAIIPS used in the two studies. Observe that

only the dependency mapping was performed during the academia study, as the prioritization results were not used.

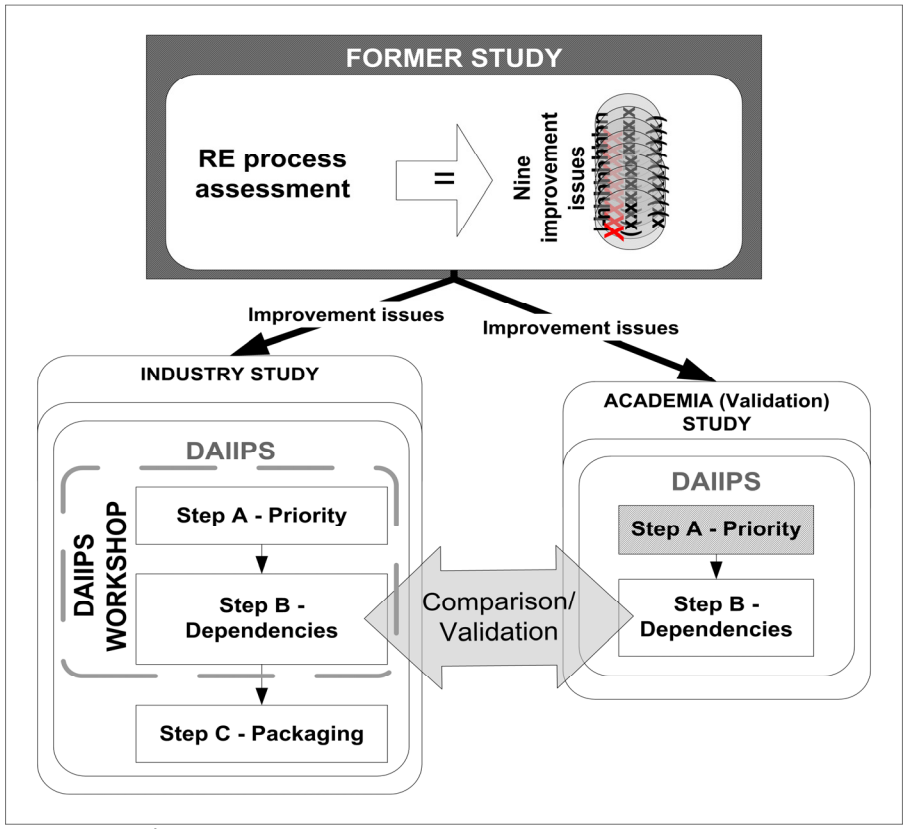


Figure 4-7. Study Overview

4.4.2.1 Industry Study Design (Preparation Step A and B)

4.4.2.1.1 Prioritization Preparation (Step A)

Sampling (see Section 4.3.1.1) of the subjects for the prioritization (and later the dependency mapping) was based on the sample selected for the process assessment (PA) conducted (as mentioned in Section 4.4.1). It was based on quota sampling, in an effort to get subjects from different parts of the population [20]. The quotas were selected based on ten different roles identified in development at DHR:

1. The *Orderer* has the task of being the internal owner of a certain project, i.e. has the customer role and if applicable the official contact with an external customer and/or partner. This party is responsible for the official signing-off when it comes to the requirements, i.e. he/she places an order.
2. The *Project Manager* has the traditional role of managing the project, resources, planning and follow-up. As far as requirements are concerned the Project

Manager is responsible for that the requirements engineering is performed, the requirements specification is written and signed off by the System Engineer.

3. The *System Engineer* is the technical responsible for a project. It is also important to recognize that the System Engineer has the official responsibility for the requirements specification in a project.
4. The *Developer* is a representative for the developers (e.g. programmers) in a project, the ones actually implementing the requirements. The developers use the requirements specification.
5. *The System Test* role can be described as the traditional role of application and feature test. This role is officially present during initial project meetings and is a part of the verification of the requirements specification.
6. *The Production* role also has a presence in projects. This representation consists of verifying that the production aspects are met, i.e. that production is taken into consideration at an early stage.
7. *The Application Developer* role represents installation and adaptation aimed at industry partners and/or end customers. Adaptation here translates to tailoring, development of some light customer features and some support.

In addition to having representatives for the roles presented above three more general roles were represented:

8. *Management*
9. *Sales/Marketing*
10. *Sales Support*

The last three roles were more general in nature in comparison to the development specific roles presented before. In total 14 people participated distributed over the roles as can be viewed in Table 4-3.

Table 4-3. Participant Distribution over Roles - Prioritization

Role	Number of participants		Role	Number of participants
1	1		6	1
2	2		7	1
3	2		8	2
4	2		9	1
5	1		10	1

The reason for having ten roles and 14 participants (i.e. an overrepresentation of some roles) was twofold. First the overrepresented roles housed most senior personnel and it was deemed positive to maximize the amount of senior staff participating. The second reason was executive, i.e. the roles of Project managers and System engineers to a degree represented the (in this study) undefined role of middle management.

Criteria (see Section 4.3.1.2) were based on maximizing the quality of the produced products in terms of customer satisfaction. The focus was to increase the quality of the requirements engineering process, thus meeting the objective of increasing customer satisfaction.

Prioritization technique (see Section 4.3.1.3) was chosen because there were only nine improvement issues to prioritize. With this in mind AHP was suitable, and could provide priority and information about e.g. consistency (see Section 4.3.1.3 – AHP for further information), and [28] for a evaluation of AHP in comparison to other prioritization techniques.

4.4.2.1.2 Dependency Mapping Preparation (Step B)

Dependencies were mapped by the same group of professionals as described in the sample above (participants in the same DAIIPS workshop) but with a modified representation over the different roles. In total 10 participants as can be seen in Table 4-4.

Table 4-4. Participation Distribution over Roles - Dependency Mapping

Role	Number of participants
1: Orderer	1
2: Project Manager	2
3: System Engineer	1
4: Developer	2
5: System Test	1
6: Production	0
7: Application Developer	1
8: Management	2
9: Sales/Marketing	0
10: Sales Support	0
Total	**10**

Some roles, i.e. Production, Sales/Marketing and Sales Support, were not present during the dependency mapping. While other roles more directly linked to system development (active participants and "owners" of the process) were represented.

The main reason for not inviting all roles was that the dependencies between the improvement issues were not obvious to people not working directly with the development, and thus the input from these roles was not premiered. The ten persons chosen for the task of dependency mapping were all active participants within the research and development department, many of which were senior members of the staff with experience from multiple roles over a number of years.

The roles not elicited during the dependency mapping were however present during the prioritization. The rationale behind the two samples was that all roles (presented in Section 4.4.2.1) could have relevant and important input to what parts of a process that needed to be improved. While knowledge of how the improvement issues were dependent on each other was deemed better explored by the ones that worked with the process every day.

4.4.2.2 Academia (Validation) Study Design

Sampling (see Section 4.3.1.1) of the subjects for the prioritization (and later the dependency mapping) was based on convenience sampling [21]. The sample consisted of six PhD students from the Department of Software Engineering & Computer Science at Blekinge Institute of Technology. The students in question were two senior students (had been PhD students for more than 2 years), and 4 junior students (< 2 years).

Criteria were set based on the one used in the industry study, i.e. maximizing the quality of the RE process.

Prioritization technique and **dependency mapping** were done in the same manner as in the industry study described in Section 4.4.2.1.

4.4.2.3 Validity Evaluation

In this section we discuss the threats to this investigation. We base this on the discussion of validity and threats to research projects presented in Wohlin et al. [20]. One type of threats mentioned in [20] is not relevant, since the investigation is conducted in an industrial environment. The threat not considered is construct validity, which mainly is concerned with the mapping from the real world to the laboratory. The investigation

presented here is however conducted in the real world. The validity threats considered are: conclusion, internal and external validity threats respectively.

4.4.2.3.1 Conclusion validity

The questionnaire used for the prioritization and dependency mapping was validated through preliminary testing and proofreading by several independent parties, to avoid factors like poor question wording and erroneous formulation.

Each prioritization and dependency mapping was done in one uninterrupted work session. Thus the answers were not influenced by internal discussions about the questions during e.g. coffee breaks.

The sampling techniques used for the industry study can pose a threat to the validity of the investigation. The subjects selected may not be totally representative for the role they should represent at DHR.

The main assurance that this misrepresentation is minimal is the fact that the subjects were selected in cooperation with three senior managers with extensive knowledge and experience with regards to the development processes and the personnel at DHR.

4.4.2.3.2 Internal Validity

As the prioritization and dependency mapping was done on paper (i.e. there was a record of people's opinions and views) this could have constrained people in their answers. This potential problem was alleviated by the guarantee of anonymity as to all information divulged during the study, and that recorded answers was only to be used by the researchers.

4.4.2.3.3 External Validity

The external validity is concerned with the ability to generalize the results. As far as the results pertaining to priority is concerned this is not a main threat to the study, since the objective is not generalizing DHR's priorities to other environments (i.e. things important to DHR may be less critical for another company). The important generalization here is whether the applied approach for prioritizing, dependency mapping and packaging improvement issues is possible to apply in other environments. There is nothing in the approach that makes it tailored to the specific setting hence the approach should be useful at other small and medium sized enterprises that would like to choose what improvement issues to undertake in their SPI enterprise.

The academia study was performed to validate that the identified dependencies between the improvement issues were representative for state-of-the art, e.g. that no important and unforeseen dependencies were missed in the industry study. As far as the priority of improvement issues obtained from the academia study is concerned no real validation and/or comparison is relevant. This is because the PhD students are not participants in the DHR organization, thus have no appreciation or insight into the strengths or weaknesses perceived by DHR employees.

4.4.3. Industry Study Execution and Results

The first part of the DAIIPS workshop (see Figure 4-7 and Section 4.3) was to introduce the participants to the nine improvement issues and the official formulation of

them, i.e. what each issue meant and what they involved. This was a straightforward procedure since all people had participated in the PA activities earlier. The improvement issues can be viewed in Table 4-5. The descriptions of the issues (in italic under each issue) are given in abbreviated format.

The next part was to initiate the workshop. A general description covering the form, how answers should be specified, and information about anonymity issues, proceeded the handing out of the forms. Examples were given as to the meaning of the priority scale and questions were answered. The organizing body of this workshop was present during the entire duration and the participants could ask questions as needed throughout the prioritization.

Table 4-5. Improvement Issues at DHR

Improvement Issues (in no specific order)
Issue-1: Abstraction level & Contents of requirements Each requirement should be specified on a predefined level of abstraction with certain characteristics (attributes attached to it), enabling requirements to be comparable and specified to a certain predefined degree of detail.
Issue-2: Requirements prioritization This issue suggests a systematic prioritization of all requirements in a standardized way.
Issue-3: Requirements upkeep during & post project In order to keep the requirements up to date during and post project the requirements have to be updated as they change.
Issue-4: Roles and responsibilities - RE process To avoid misunderstandings as well as avoiding certain tasks not being completed the roles and responsibilities of all project members should be clearly defined before project start.
Issue-5: System tests performed against requirements All system tests should be performed against requirements (e.g. using requirements as a base to construct test-cases).
Issue-6: RE process/methods This issue is basically the creation and incorporation of a complete and comprehensive Requirements Engineering process at DHR that is well defined and documented.
Issue-7: Requirements reuse By reusing requirements everything from the requirement itself to analysis, design, implemented components, test cases, scenarios, and use cases etc. can be reused.
Issue-8: Requirements traceability Policies and support for traceability to and from the requirements are to be established.
Issue-9: Requirements version handling Policies and support for version handling of each requirement (not only on requirement's document level) should be established.

The handout included several examples of how priority and dependencies were to be specified. The workshop lasted over two hours time.

4.4.3.1 Sample and Consistency Ratio

During the compilation of the prioritization results the consistency ratio (CR) was observed for each participant. It varied from a CR of 0.05 (well below the recommended limit) to 0.59 (regarded highly inconsistent). The average CR was ≈ 0.22 (median ≈ 0.16). A summary of the participants CR is given in Table 4-6.

The decision was made to include only the results from participants having a CR of 0.20 or less. Going by the recommendation by Saaty of 0.10 would exclude all results except for one participant, i.e. not a practical solution.

This of course had a negative impact on the sample, i.e. the distribution of participants over the roles was altered due to the invalidation of some results (see Table 4-6, the grey cells).

Table 4-6. CR for DHR Participants

Subject	CR		Subject	CR
DHR6	0.05		DHR1	0.17
DHR9	0.11		DHR5	0.19
DHR7	0.12		*DHR12*	*0.28*
DHR4	0.12		*DHR13*	*0.30*
DHR14	0.13		*DHR8*	*0.35*
DHR2	0.14		*DHR10*	*0.38*
DHR3	0.14		*DHR11*	*0.59*

In Table 4-7 the impact of the participants with too high CR (>0.20) can be observed. In the left column the roles are listed, the middle column displays the number of people that participated in the prioritization for each role. The last column denotes the total amount that was incorporated in the prioritization results, i.e. those with a CR of less than 0.20 (within the parenthesis the number of people that were excluded from the prioritization can be viewed). The bottom row shows the total, i.e. five people were excluded due to too high CR, leaving nine people.

Some roles were diminished and others vanished altogether, i.e. the Production and Application Developer roles were not to influence the prioritization of the improvement issues at all.

This "total non-representation" by the two roles was however deemed acceptable, as was the diminished representation of three other roles. The reason for this was that the improvement issue's priority differed relatively little after excluding the persons with high CR (see Section 4.4.3.2 and Figure 4-1).

Table 4-7. Result of Invalidation Impact on Sample

Role	Number of participants	Result (Difference)
1: Orderer	1	1 (0)
2: Project Manager	2	2 (0)
3: System Engineer	2	1 (-1)
4: Developer	2	1 (-1)
5: System Test	1	1 (0)
6: Production	1	0 (-1)
7: Application Developer	1	0 (-1)
8: Management	2	1 (-1)
9: Sales/Marketing	1	1 (0)
10: Sales Support	1	1 (0)
TOTAL	**14**	**9 (-5)**

4.4.3.2 Priority of Improvement Issues (Results - Step A)

The results of the prioritization can be viewed in Table 4-8 where all nine issues are present. The views of the participants are fairly scattered, although some tendencies of consensus can be observed.

From the *rank* row in Table 4-8 it is observable that issue *5: System Test* has the highest priority by a marginal of almost 43% ((0.213-0.149)/0.149) in comparison to issue *2: Upkeep* ranked as no. 2.

The issues ranked 2:nd to 4:th only have a moderate difference in priority in comparison (i.e. the difference between issues 2 and 4 is in comparison a moderate 20% ((0.149-0.124)/0.124)).

Looking at Figure 4-8 the issues are grouped together on a "shelf structure" which illustrates the jumps in priority. The top shelf is occupied with issue 5, then there is a large drop in priority to the next shelf where issues *3: Requirements upkeep during & post project*, *1: Abstraction level & Contents of requirements* and *2: Requirements prioritization* are grouped, and so on. It is noticeable that there are several jumps in priority, and a substantial difference between the issues ranked in the top four and the issues ranked five and lower.

There is a quite large difference in opinion regarding issue *5: System Test* (ranked number one), i.e. a scattering from a priority of only 0.05 to an extreme 0.44 (i.e. more than eight times as high). The scattering is less in issues 2-4 which lie closer in priority.

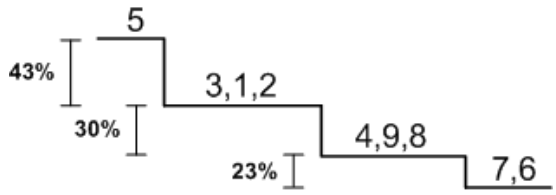
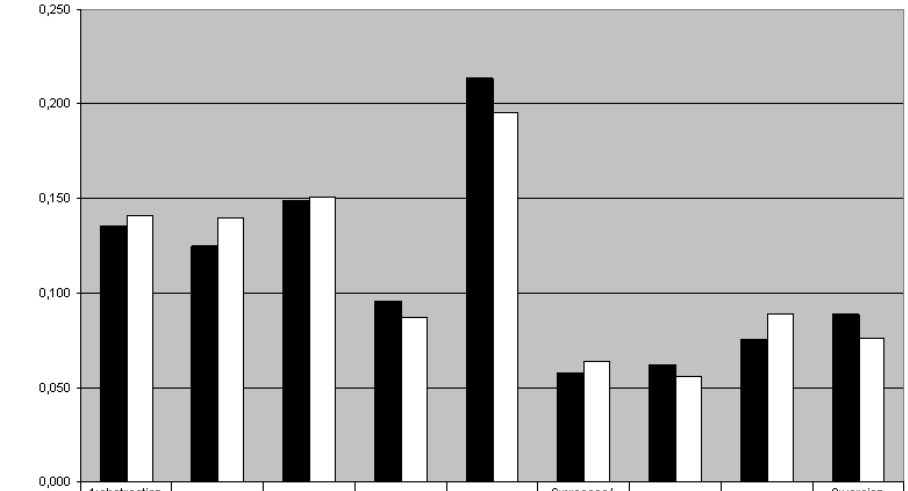


Figure 4-8. Priority Shelves

Figure 4-9 also illustrates the priority of the improvement issues. There are two series of data presented, the priorities of the issues by only the participants with CR<0.20 (black bars), and the priorities not excluding the ones with CR>0.20, i.e. all 14 participants included (white bars). It is observable that the priority of the improvement issues is not substantially changed when all 14 participants' results are taken into consideration, at least not for the fist two shelves, i.e. issue 5 is still on step 1, and issues 3, 1, 2 are still on step two. And the changes amongst the other issues are also rather moderate.

Table 4-8. Priority Results (CR<0.20)

ID	1:abstr/ contents	2:prio	3:upkeep	4:roles	5:sys. test	6:process / methods	7:reuse	8:traceab	9:version handling	CR
DHR1	0.22	0.07	0.19	0.03	0.26	0.03	0.05	0.07	0.09	0.17
DHR2	0.2	0.14	0.19	0.05	0.22	0.07	0.06	0.06	0.02	0.14
DHR3	0.17	0.07	0.18	0.28	0.1	0.02	0.04	0.05	0.09	0.14
DHR4	0.08	0.24	0.08	0.09	0.25	0.12	0.06	0.03	0.04	0.12
DHR5	0.12	0.14	0.14	0.03	0.17	0.05	0.06	0.1	0.2	0.19
DHR6	0.04	0.16	0.17	0.17	0.05	0.12	0.09	0.13	0.07	0.05
DHR7	0.13	0.15	0.19	0.05	0.18	0.03	0.1	0.08	0.1	0.12
DHR9	0.14	0.09	0.14	0.06	0.25	0.03	0.04	0.11	0.14	0.11
DHR14	0.12	0.06	0.06	0.1	0.44	0.05	0.06	0.05	0.05	0.13
average (CI<0.20)	0.14	0.12	0.15	0.1	0.21	0.06	0.06	0.08	0.09	0.13
rank	3	4	2	5	1	9	8	7	6	



	1:abstraction /contents	2:prioritization	3:upkeep	4:roles	5:system test	6:process/ methods	7:reuse	8:traceability	9:version handling
■ CR < 0.20	0,136	0,124	0,149	0,096	0,213	0,058	0,062	0,076	0,089
□ CR > 0.20	0,141	0,14	0,151	0,087	0,195	0,064	0,056	0,089	0,076

Figure 4-9. Priority Comparison

4.4.3.3 Dependencies between Improvement Issues (Results - Step B)

As the results from the dependency mapping were compiled the weight of each dependency was recalculated into percent (what percentage of the subjects identified the dependency), and a threshold was set to 20%, i.e. more than 20% of the subjects doing the dependency mapping had to identify the dependency (see Section 4.3.2).

The results from the dependency mapping can be seen in Table 4-9, where the grey cells denote dependencies deemed under the threshold.

The dependencies under the threshold were scrutinized and dismissed, all except one, i.e. dependency *8 on 1* (8: traceability was deemed dependent on 1: abstraction/contents). The reasoning behind this was discussed and consensus reached thus making an exception to the general threshold limit.

Table 4-9. Dependencies between Improvement Issues Identified at DHR

Dependency (Issue *i* on issue *j*)	Weight in %		Dependency (Issue *i* on issue *j*)	Weight in %
2 on 1	60		*5 on 8*	*20*
3 on 1	*20*		7 on 1	30
3 on 4	40		7 on 3	60
3 on 6	*20*		*7 on 6*	*20*
3 on 9	*20*		7 on 8	40
4 on 6	*20*		7 on 9	30
5 on 1	60		*8 on 1*	*20*
5 on 2	30		*8 on 3*	*10*
5 on 3	70		*9 on 3*	*10*

A dependency diagram was drawn (see Figure 4-10). Here the relations, their weight (e.g. 0.6 is the same as 60% in Table 4-9), and the relations' direction can be observed.

Relative priority (the value inside the parenthesis) and rank (top bold numeral) are also visible.

4.4.3.4 Package Improvement Issues (Results – Step C)

Issue *5: system test* has the highest priority by far, and depends on issues 1, 2 and 3. Issue 3 in turn depends on issue 4. This "unit", i.e. issues 5, 3, 2, 1 and 4 were possible to break out, and put into an SPI package, see Figure 4-11. This was a seemingly optimal choice with regards to priority and the mapped dependencies. However this packaging (as illustrated in Figure 4-11) was rejected due to the strain on resources such a large SPI package would demand (especially in terms of time to return on investment). Instead a second proposal for packaging was drawn up taking resources and time into account.

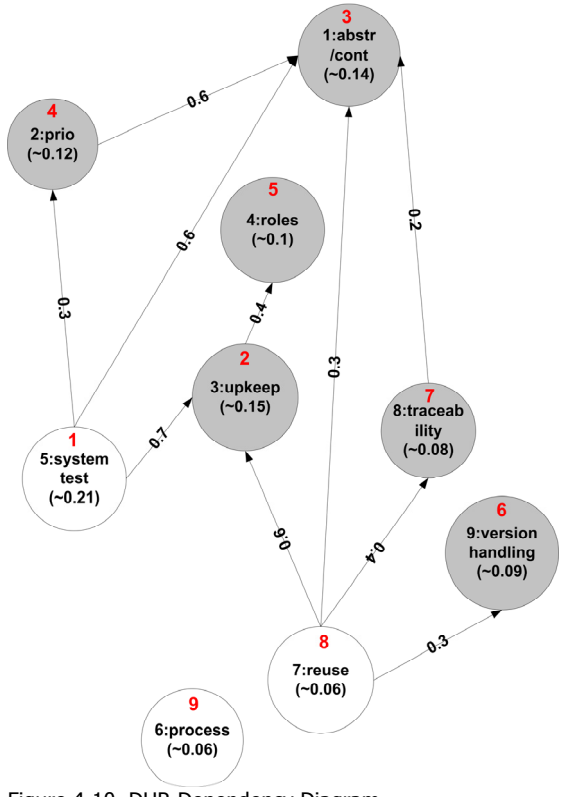


Figure 4-10. DHR Dependency Diagram

The previously suggested SPI Package *A* (see Figure 4-11) was divided into two packages, i.e. *2: prioritization* and *5: system test* was broken out to its own package (see Figure 4-12). This meant that issue 5 (which had the highest priority) was postponed. The reasoning was that SPI Package 1 (see Figure 4-12) was a prerequisite for SPI Package 2, and that it would probably be implemented in this order anyway, i.e. even if the packaging had been accepted from the first suggestion as seen in Figure 4-11. The major difference in breaking up package *A* further was for the implementation and evaluation of the implementation to be faster. If package *A* had been kept as the original suggestion more resources and time had to be spent on the SPI before feedback on the implementation of the activities was available. By the further division an evaluation of the work could be done earlier, and a subsequent decision to continue (i.e. with package 2 in Figure 4-12) or not could be made earlier and at a lesser cost in regards to time and resources.

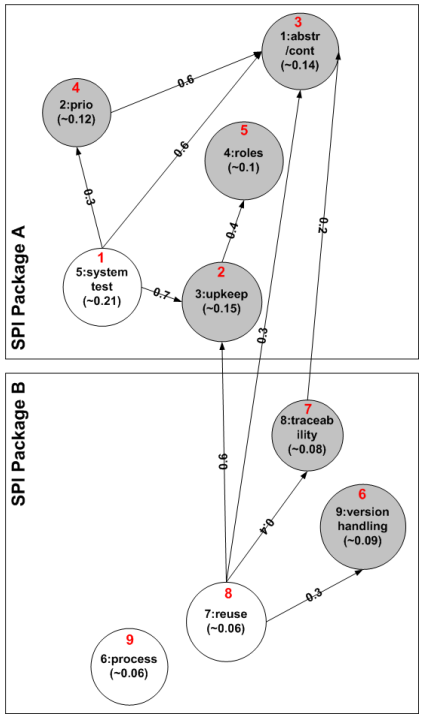


Figure 4-11. SPI Package Suggestion at DHR

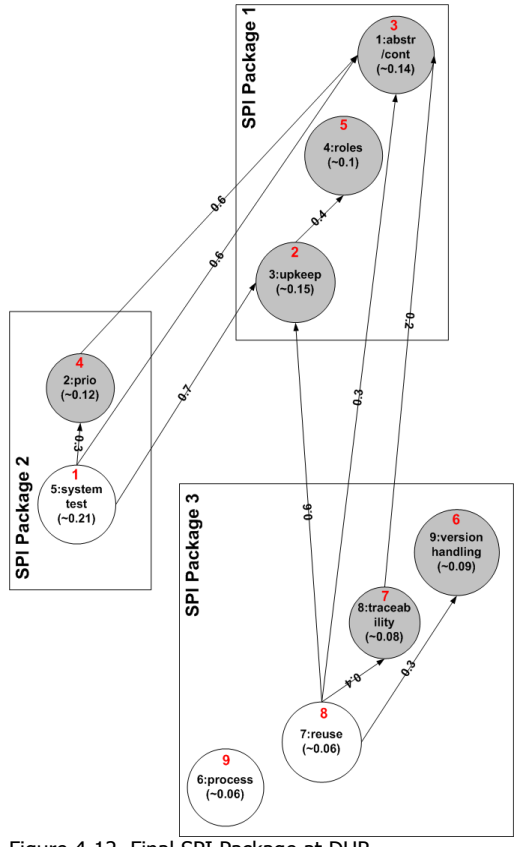


Figure 4-12. Final SPI Package at DHR

4.4.4. Academia (Validation) Study Execution and Results

The study in academia preceded much in the same manner as the industry study. The subjects selected were given the same information and the same form to fill out regarding prioritization and dependency mapping of improvement issues.

The workshop was budgeted to two hours, but took only about one hour in total.

4.4.4.1 Sample and Consistency Ratio

During the compilation of the prioritization results the consistency ratio (CR) was observed for each participant. It varied from a CR of 0.23 (well above the recommended limit set by Saaty and the limit deemed acceptable during the industry study) to 0.62 (regarded highly inconsistent). The average CR was ≈ 0.40 (median ≈ 0.35). A summary of the participants CR can be viewed in Table 4-10.

In Section 4.4.2.3.3 it was stated that there would be no comparison between academia and industry as to the priority of improvement issues as it was not considered relevant. Furthermore it could be said that the CR was very high over all (well over the limit of CR<0.2 used in the industry study), in general disqualifying all participants in the academia study.

However this is as said before of subordinate importance as the academia study was not done in order to confirm/validate the priorities obtained from industry, but rather to compare the dependencies being mapped, in order to get an idea if the dependencies found in the industry study were also found in academia, and vice versa.

Table 4-10. CR for Academia Participants

Subject	CR
BTH1	0.42
BTH2	0.62
BTH3	0.27
BTH4	0.23
BTH5	0.26
BTH6	0.58

4.4.4.2 Dependencies between Improvement Issues

As the results from the dependency mapping were compiled the weight of each dependency was recalculated into percent (what percentage of the subjects identified the dependency), and a threshold was set to 20%, i.e. more than 20% of the subjects doing the dependency mapping had to identify the dependency (see Section 4.3.2).

The results from the dependency mapping can be seen in Table 4-11.

It is noticeable that no dependencies identified were specified by less than 33% (i.e. 2 participants in this case).

Table 4-11. Dependencies between Improvement Issues Identified in Academia

Dependency (Issue i on issue j)	Weight in %		Dependency (Issue i on issue j)	Weight in %
2 on 1	83		6 on 4	33
3 on 1	33		7 on 1	67
3 on 4	33		7 on 3	33
3 on 6	50		7 on 6	33
3 on 9	33		7 on 8	33
4 on 6	33		7 on 9	50
5 on 1	83		8 on 1	33
5 on 3	83		8 on 3	50
5 on 8	50		9 on 3	33

4.4.5. Comparison – Industry vs. Academia

4.4.5.1 Introduction

In this section some comparisons are made between the dependencies identified in the industry study at DHR and the study performed in academia at Blekinge Institute of Technology.

The motivation for the comparison is to validate the dependencies identified in industry (see Section 4.4.2.3.3). In addition this section provides a chance to observe some differences between the outlook on dependencies between industry and academia.

Table 4-12 presents a summation of the dependencies presented previously (i.e. in Table 4-9 and Table 4-11). The weights for both industry (column two) and academia

(column three) are presented in percent. The dependencies are also presented in Figure 4-13 where dependency weights are presented on the lines (industry | academia).

Table 4-12. Dependency Comparison between DHR and Academia

Dependency (Issue i on issue j)	Weight (%) DHR	Weight (%) Acad.	Note (Group)
2 on 1	60	83	1
3 on 1	20	33	4
3 on 4	40	33	2
3 on 6	20	50	3
3 on 9	20	33	4
4 on 6	20	33	4
5 on 1	60	83	1
5 on 2	30	*0*	6
5 on 3	70	83	1
5 on 8	20	50	3
6 on 4	*0*	33	5
7 on 1	30	67	2
7 on 3	60	33	2
7 on 6	20	33	4
7 on 8	40	33	2
7 on 9	30	50	2
8 on 1	20	33	2
8 on 3	10	50	3
9 on 3	10	33	5

4.4.5.2 Comparison Analysis

Below some analysis and augmentation is provided on the dependencies listed in Table 4-12. The note column holds numbers that have corresponding numbers in the text below, i.e. some analysis is offered for each number. In some instances the dependencies are grouped together with regards to common properties or tendencies.

Note 1: There is strong support for the dependencies in this group, both in industry and in academia. All of the dependencies in this category were taken into consideration during the packaging activity performed (see Section 4.4.3.3). The dependencies here are not analyzed or commented upon any further.

Note 2: This group also holds dependencies that all were taken into consideration during the dependency mapping, i.e. all had a weight above the threshold in both industry and in academia. However in comparison to group 1 the weights are lower, i.e. most weights are ≤ 50%. In addition to this there are some discrepancies between industry and academia. This is especially clear in the case of dependency *7 on 1* (having a much stronger weight in academia than in industry), and *7 on 3* (showing the opposite, i.e. stronger weight in industry). In academia issue *7: Requirements reuse* is considered to be linked primarily to how the requirements are specified (issue 1) while the dependency on issue *3: Requirements upkeep during & post project* is considered of less weight. In industry the tables are turned, i.e. issue 3 is premiered. One explanation for this could be that the participants from industry see out-of-date documents, e.g. requirement specifications, as an undesirable but relatively common occurrence in projects. While people based in academia in this case may have premiered other aspects due to that they are not faced with this particular problem in their daily work.

Note 3: In this group there is a clear discrepancy between industry and academia, i.e. dependencies identified as rather strong (50%) in academia are considered weak in industry (≤ 20%). This group is has the strongest candidates for adding to the dependencies identified in industry, i.e. catching dependencies missed in the industry mapping. The three cases in this group are analyzed one by one below.

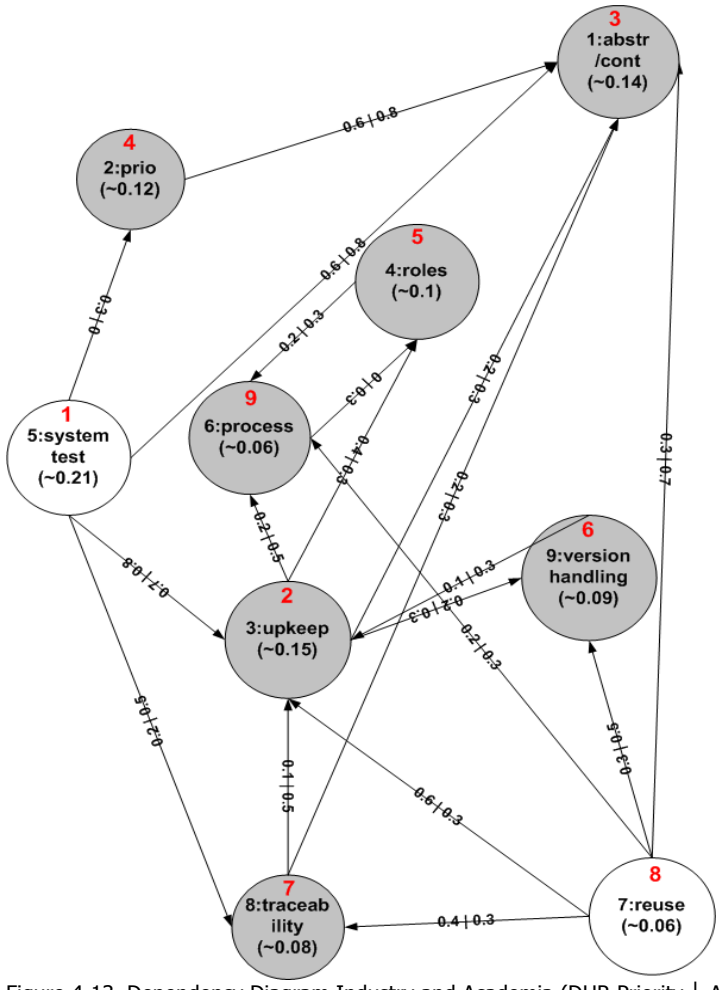


Figure 4-13. Dependency Diagram Industry and Academia (DHR Priority | Academia Priority)

3 on 6: In the case of dependency *3 on 6* (*6: RE process/methods*) the motivation behind this in academia was generally that a documented process helped in the upkeep of requirements. In industry however the general perception was that the presence of a documented process did not ensure anything, and that focus should be put on establishing practices for issues, and that a complete and comprehensive documented process was not a prerequisite for any of the issues identified.

Looking at issue *6: RE process/methods* there are a number of instances where dependencies are identified both to and from this issue (in groups 4 and 5). In industry the identified dependencies were under the threshold (see Section 4.4.3.3) in all cases and thus had no impact during the dependency mapping. In academia the dependencies were over the threshold. One possible explanation for industry not premiering this issue (not in the case of either priority or during the dependency mapping) could be that there is some disillusionment towards official documented processes, i.e. in many cases representing a largely unused set of steps, rules, practices etc. The mindset was that

documenting something and making it official did not necessarily promote it or make it used (or even useful) by the participants in the organization.

5 on 8: In the case of dependency *5 on 8* the general reasoning in academia was that there had to be traceability policies in place in order for system tests to be performed against requirements. This would enable e.g. faster tracking of faults from test cases (based on requirements) and the modules/classes/code.

A dependency between test and traceability was identified in industry also but from a different perspective, i.e. the reason given was that test could be performed on requirements only if it would be possible to trace which of the requirements that were implemented. This rationale (and thus the dependency) was however not seen as critical due to that all requirements allocated to a certain project should be implemented. If the requirements in question were not to be implemented for any reason they would be removed from the project.

In the case of this dependency there is a good chance that the academia view could benefit the industry case.

8 on 3: The reasoning behind this dependency in academia was that requirements not updated were prone to make traceability difficult, if not impossible. However this dependency was not considered critical in industry.

In the case of this dependency there is a good chance that the academia view could benefit the industry case.

Note 4: In this group the dependencies were identified in both cases, but the weight in industry was just below (or actually on) the threshold, i.e. 20%. In the case of academia all the dependencies had a weight of 33%, just above the threshold, i.e. identified by two persons (2 out of 6 = 33%). There are some implications that could be attributed to this, i.e. that one person less in the case of academia would make this category a non-issue in this analysis.

Given that the dependencies here are rather weak looking at the combination of both cases the dependencies in this group are not considered as strong candidates for adding to the dependencies identified in industry. This is further supported by the fact that two out of the four dependencies in this groups are on issue *6: RE process/methods* (the rationale behind dependencies and this issue was discussed above under note 3).

The two other dependencies under this group are discussed below.

3 on 1: The reasoning behind this dependency in academia was that the abstraction level and contents of a requirement made it easy or hard to keep a requirement updated. This dependency was also identified in industry but the reasoning behind the dependency was rather that someone had to keep the requirements updated (thus the dependency on issue 4), i.e. identifying that *how* the requirement was specified could impact on keeping it updated, was not premiered.

3 on 9: This dependency was identified in both studies, and the rationale behind it was that version handling helped in the upkeep of requirements. This was however not considered critical in industry because version handling of the requirements document was considered adequate for the time being and that issue 9 would be implemented later, possibly in combination with CASE tool support.

Note 5: This group shows a large discrepancy between industry (dependency weights well below the threshold) and academia (dependency weights just below the threshold). The weights are fairly weak even in the academia case, thus these dependencies are not considered as strong candidates for adding to the dependencies identified in industry.

This is further supported by the fact that one out of the two dependencies in this group is from issue *6: RE process/methods* (the rationale behind dependencies and this issue was discussed above under note 3).

In the case of dependency *9 on 3* the reasoning in academia was that upkeep of requirements was the basis for creating versions, i.e. the point of creating versions.

In industry the dependency was seen as versions had to be created manually (i.e. upkept). This reasoning was however not seen as critical since the idea was for this to be handled by some sort of tool later on (e.g. a CASE tool as mentioned in relation to dependency *3 on 9*).

Note 6: The dependency *5: System tests performed against requirements* on *2: Requirements prioritization* was identified in industry but not in academia. The motivation behind this dependency in industry was that in order to know what requirements to premiere in the testing it was important to ascertain which were prioritized, i.e. implying that there were not always resources to pay equal attention to all parts and test all parts as extensively as would be preferable.

4.4.5.3 Comparison Summary and Discussion

It should be restated that the comparison between dependency mappings in industry and academia was performed to validate the dependencies found in industry, i.e. strengthening the external validity of this study by alleviating the risk of crucial dependencies being missed.

Two prime candidates for adding to the dependencies found in industry were identified through the academia study, namely 5 on 8 and 8 on 3 (see note 3).

The impact of these dependency additions to the industry case in this instance has to be taken under advisement.

In the case of dependency 8 on 3 the effects on the SPI package is minor, almost non-existent. Looking at Figure 4-12, which displays the final SPI packaging at DHR, no action has to be taken. I.e. the new dependency does not force any reallocation of improvement issues between the packages 1 through 3.

In the case of dependency 5 on 8 the impact is greater though. In order for the dependency to be satisfied issue 8 has to be moved to either package 2 (and supersede implementation of issue 5), or be moved to package 1. In either case there is a reallocation of an improvement issue and a subsequent increase in the size of the affected package.

Looking at the other dependency comparisons the discrepancies between the studies were of less impact. Most dependencies identified in one study were also seen in the other, barring a few exceptions.

Some of the most prominent were dependencies to and from issue *6: RE process/methods*. The industry view was more pragmatic than the one in academia, i.e. recognizing that an official and documented process is seldom a guarantee for things actually being done at all, not to mention done in a certain way which is set down on paper.

This difference in outlook between industry and academia is also seen in the case of testing being dependent on having priority on the requirements, i.e. *5 on 2* (note 6). The assumption in academia can be that all implemented components are tested. This is true

in the industry case as well (as probably all parts are tested), but limited resources and tight deadlines may result in some prioritization as to the degree of testing.

4.5. Discussion and Conclusions

In the introduction to this chapter several reasons were given motivating the construction of the DAIIPS scheme. The primary motivation was to give SMEs a decision support scheme that considered several aspects identified as critical for the success of SPI endeavors. The need for this was initially identified in industry, i.e. in the case of the SPI activity performed at Danaher Motion Särö AB.

In the first part (Section 3) we present DAIIPS, both how improvement issues are prioritized, and how dependencies between issues are identified and visualized. The objective is that this information act as input to an informed decision regarding what actions to take in terms of process improvement.

The developed method is subsequently applied at a company (Section 4), and it is shown how it was used in an industry setting to allocate improvement issues into SPI packages taking priority, dependencies and (to a certain extent) cost into consideration.

Modularity was also premiered in the development of DAIIPS. The scheme is independent, speaking to that any method (e.g. CMMI, SPICE etc) can be used for the proceeding SPA activity, as long as the assessment data be available. The same is true for the implementation of the issues after the DAIIPS packaging, enabling organizations to lift in DAIIPS as a decision support scheme regardless of environment. This is of course dependent on factors such as that there is input from the SPA in some comparable form, i.e. considering abstraction level.

Looking at the industry case presented in this chapter the focus was not on general SPI but rather on improving a sub-process, namely requirements engineering. In the DHR case this was the area that needed to be improved, using DAIIPS enabled the creation of decision support material for the continuation of the improvement work after the assessment stage.

SPI efforts are often an expensive undertaking, thus effectively raising the threshold for companies and especially SMEs. The general idea behind DAIIPS was to offer a way in which the threshold could be lowered, by offering a structured way to increase the control of aspects such as *time* to return on investment and *cost* of SPI.

Furthermore, other points argued as critical for the success of SPI undertakings (see Section 4.2.1) were also taken into consideration during the development and use of DAIIPS:

- *Commitment* by management and middle management is easier to secure if management feels in control of the issues of *cost* and *time*.
- *Commitment* by staff, e.g. engineers, (often seen as the most critical issue in regards to SPI success) can be positively influenced by the fact that the prioritization and dependency mapping is largely done by representatives from their "ranks".
- *Focus* on a delimited number of improvement issues at a time (i.e. a SPI package) offers clear and well-defined goals that are obtainable.
- *Involvement* (in the SPI work by staff) is easier to secure if the staff has a say in what is to be done from the start.

Enabling the professionals in the SPI targeted organization to "take matters into their own hands", prioritizing, mapping dependencies and packaging issues according to their needs, should be premiered. It is important to remember that no method, framework or scheme (or even DAIIPS for that matter) is useful if the professionals, whose organization is targeted for SPI, are not committed.

In summary, the chapter presents a method for prioritization and identification of dependencies between software process improvement proposals. The method is applied successfully in an industrial case study involving an organization being classified as a small and medium sized company.

4.6. Further Work

Refinement, Expansion and Replication are three key words that are central for the future of the DAIIPS scheme.

DAIIPS as a scheme needs to be refined through the tweaking of the steps of which it is comprised. Lessons learned from the studies thus far need to be evaluated, weaknesses need to be identified and dealt with by further simplification of the scheme were possible.

DAIIPS could be augmented by the incorporation of cost as an official and modeled part or the scheme, thus offering further dimensions to the decision support offered.

Replication of the study presented above is almost a prerequisite in order for the DAIIPS scheme to be tested, and in doing so producing results that allows DAIIPS to evolve. This includes testing the use of alternative prioritization techniques.

4.7. References

1. Sommerville I (2001) Software Engineering. Addison-Wesley, Essex.
2. Calvo-Manzano Villalón JA, Cuevas Agustín G, San Feliu Gilabert T, De Amescua Seco A, García Sánchez L, Pérez Cota M (2002) Experiences in the Application of Software Process Improvement in SMEs. Software Quality Journal 10(3):261-273.
3. Deming WE (1986) Out of the Crisis. Massachusetts Institute of Technology Center for Advanced Engineering Study, Cambridge.
4. Basili VR (1985) Quantitative Evaluation of Software Methodology. University of Maryland, College Park, Maryland.
5. Paulk MC, Curtis B, Chrissis MB, Weber CV (1995) The Capability Maturity Model: Guidelines for Improving the Software Process. Addison-Wesley, Reading.
6. CMMI Product Developement Team (2002) Capability Maturity Model Integration (CMMI), Version 1.1. CMMI for Systems Engineering, Software Engineering, Integrated Product and Process Development, and Supplier Sourcing Version 1.1 (CMMI-SE/SW/IPPD/SS, V1.1).
7. (1998) http://www.sei.cmu.edu/iso-15504/. Last Accessed: 2004-01-07.
8. McFeeley B (1996) IDEALSM: A User's Guide for Software Process Improvement. TR-CMU/SEI-92-TR004. SEI, Pittsburgh.
9. Wiegers KE, Sturzenberger DC (2000) A Modular Software Process Mini-Assessment Method. IEEE Software 17(1):62-70.
10. Zahran S (1998) Software Process Improvement: Practical Guidelines for Business Success. Addison-Wesley, Reading.
11. Kuilboer JP, Ashrafi N (2000) Software Process and Product Improvement: An Empirical Assessment. Information and Software Technology 42(1):27-34.
12. Reifer DJ (2000) The CMMI: It's Formidable. Journal of Systems and Software 50(2):97-98.
13. Scott L, Jeffery R, Carvalho L, D'Ambra J, Rutherford P (2001) Practical Software Process Improvement - the IMPACT Project. In Proceedings of the Australian Software Engineering Conference, IEEE, Los Alamitos CA, pp. 182-189.
14. Kautz K, Hansen HW, Thaysen K (2000) Applying and Adjusting a Software Process Improvement Model in Practice: The Use of the IDEAL Model in a Small Software Enterprise. In Proceedings of the 2000 International Conference on Software Engineering, IEEE, Los Alamitos CA, pp. 626-633.
15. El Emam K, Goldenson D, McCurley J, Herbsleb J (2001) Modeling the Likelihood of Software Process Improvement: An Exploratory Study. Empirical Software Engineering 6(3):207-229.

16. Rainer A, Hall T (2003) A Quantitative and Qualitative Analysis of Factors Affecting Software Processes. The Journal of Systems and Software 66(1):7-21.
17. Herbsleb J, Zubrow D, Goldenson D, Hayes W, Paulk M (1997) Software Quality and the Capability Maturity Model. Association for Computing Machinery. Communications of the ACM 40(6):30-40.
18. Herbsleb JD, Goldenson DR (1996) A Systematic Survey of CMM Experience and Results. In Proceedings of the 18th International Conference on Software Engineering, IEEE, Los Alamitos CA, pp. 323-330.
19. Conradi R, Fuggetta A (2002) Improving Software Process Improvement. IEEE Software 19(4):92-100.
20. Wohlin C, Runeson P, Höst M, Ohlson MC, Regnell B, Wesslén A (2000) Experimentation in Software Engineering: An Introduction. Kluwer Academic, Boston.
21. Robson C (2002) Real World Research: A Resource for Social Scientists and Practitioner-Researchers. Blackwell Publishers, Oxford.
22. Saaty TL (1980) The Analytic Hierarchy Process: Planning, Priority Setting, Resource Allocation. McGraw-Hill, London.
23. Chu P, Liu JK-H (2002) Note on Consistency Ratio. Mathematical and Computer Modeling 35(9-10):1077-1080.
24. Apostolou B, Hassell JM (2002) Note on Consistency Ratio: A Reply. Mathematical and Computer Modeling 35(9-10):1081-1083.
25. Saaty TL, Vargas LG (2001) Models, Methods, Concepts & Applications of the Analytic Hierarchy Process. Kluwer Academic Publishers, Boston.
26. Beck K, Fowler M (2001) Planning Extreme Programming. Addison-Wesley, Boston.
27. Leffingwell D, Widrig D (2000) Managing Software Requirements: A Unified Approach. Addison-Wesley, Reading.
28. Karlsson J, Wohlin C, Regnell B (1998) An Evaluation of Methods for Prioritizing Software Requirements. Information and Software Technology 39(14-15):939-947.

Chapter Five

Paper IV

Requirements Abstraction Model

Tony Gorschek and Claes Wohlin

Submitted to Requirements Engineering journal, Springer-Verlag, 2004.

Abstract

Software requirements arrive in different shapes and forms to development organizations. This is particularly the case in market-driven requirements engineering, where the requirements are on products rather than directed towards projects. This result in challenges related to making different requirements comparable. In particular, this situation was identified in a collaborative effort between academia and industry. A model, with four abstraction levels, was developed as a response to the industrial need. The model allows for placement of requirements on different levels, and it supports abstraction or break down of requirements to make them comparable to each other. The model was successfully validated in several steps at a company. The results from the industrial validation point to the usefulness of the model. The model will allow companies to ensure comparability between requirements, and hence it generates important input to activities such as prioritization and packaging of requirements before launching a development project.

5. Improvement Implementation

5.1. Introduction

Market driven incremental product development and delivery (release) is becoming increasingly commonplace in software industry [1, 2]. Incremental product development is planned and executed with the goal of delivering an optimal subset of requirements in a certain release (version of a product that is distributed to customers) [3]. The idea is to select *what* a release should contain (requirements), *when* it should be released (time), and at what *cost* (pertaining to the resources needed designing and implementing a requirement) this should be achieved. The decision about which customers get what features and quality at what point in time has to be taken, i.e. making these activities a major determinant of the success of a product [4].

All activities described above, i.e. establishing *what* should be included in a release, *when* it should be released and at what *cost*, are vitally dependent on the product requirements and that they are elicited/caught, analyzed, and specified before any planning and development activity can commence.

The situation is far more complex than the one presented by the classical bespoke [3] development situation where elicitation could be targeted and concentrated mainly to the customer organization and stakeholders identified there. The development activity was initiated by e.g. an order, which generally resulted in a project (maybe preceded by a pre-study) and then requirements engineering was initiated through this project. As illustrated in Figure 5-1, the situation in a market driven development situation is different since the requirements flow is not limited to a development instance (e.g. a project), but rather continuous in nature, and the requirements themselves should act as the catalyst for initiating development.

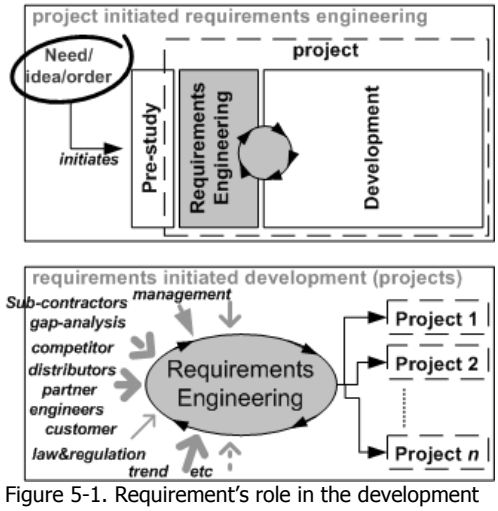


Figure 5-1. Requirement's role in the development process.

In market driven development requirements are generally generated by multiple sources, both internal (e.g. engineers to management) and external (e.g. customers and partners). It can be everything from direct requests for added functionality from existing and/or potential customers to updates proposed by engineers working on the product.

In addition, *indirect requirements* need to be caught. This can be everything from idea-like requirements caught by the marketing department during a competitor analysis or a market survey, to information gathered during product support and conveyed internally.

As the sources of the requirements vary and the requirements themselves are both direct and indirect in nature it is not surprising that they come in different shapes and forms, at multiple levels of abstraction, and described on varying levels of refinement.

In Figure 5-1 this is depicted as *requirements initiated development*, where all these types of requirements are input to a requirements engineering process which has the capability of performing the needed refinement, analysis and negotiation, producing good-enough requirements to initiate and drive development activities (e.g. projects) in an continuous manner, i.e. development is initiated when the requirements warrant it.

The market driven product development situation described above was identified during a cooperative software process improvement (SPI) venture in industry, performed at Danaher Motion Särö AB (DHR). There was a need for adapting requirements engineering at DHR to a continuous process, moving from traditional *project initiated requirements engineering* to *requirements initiated development*. This involved not only creating a new way of working (e.g. how to specify requirements, what roles and responsibilities should be present etc), but also a new way of thinking (e.g. that requirements are the basis for product development).

The Product Managers (the ones charged with implementing the new way of working) were faced with the challenge of how to take care of the continuous incoming stream of requirements ranging from abstract to technically detailed. Based on this problem the *Requirements Abstraction Model* was developed. This chapter presents the *Requirements Abstraction Model (RAM)*, how it was created in close cooperation with industry, and validated through feedback from professionals as well as how it was tested in a live project.

Although, the model was developed based on needs identified at DHR, the objective was for the model to be generally usable, i.e. the aim of the RAM was to give professionals working with product planning/development (e.g. Product Managers) a requirements engineering model that help them in their work. To this end the RAM is modeled towards a product perspective, supporting a continuous requirement engineering effort, aimed at taking requirements of multiple types (abstraction level) as input, and offer a structure for the *work-up* of these requirements, i.e. breaking down abstract requirements into detailed ones, and vice-versa (see Section 5.4.3).

The benefits of using the RAM as a support in product centered continuous requirements engineering can be summarized in four bullets.

(I) All requirements are compared to the product strategies, offering an assurance that requirements do not violate the overall goals set by management. This offers the possibility to dismiss requirements early in the process, freeing up resources to work on/refine relevant requirements that are in line with the product strategies.

(II) All requirements are broken down to an abstraction level where they are good-enough for initiating a development effort (project). This assures that the projects (whose aim it is to realize the requirements) get good-enough requirements to base their development efforts on (e.g. testable and unambiguous).

(III) Work-up of requirements means that they are formulated on the *same* level of abstraction, and hence they can be compared and set against one another. The ability to compare requirements is a prerequisite to effective release planning and prioritization.

(IV) All requirements can be followed through several levels of abstraction giving a richer understanding of each requirement, and thus better decision support can be obtained for all professionals, from management to developers.

The chapter is outlined as follows. Section 5.2 offers an overview of related work. In Section 5.3 the background and motivation is presented, along with how it relates to both DHR and the general industry case. Section 5.4 offers an introduction and exemplification of the Requirements Abstraction Model (RAM), and Section 5.5 presents how the RAM was evaluated through static and dynamic validation. Section 5.6 presents the Conclusions.

5.2. Related Work

The idea of market driven incremental development is not new. An example of this is the IBM REQUEST technique presented in 1992 by Yeh in [5]. Here arguments are put forward for the necessity of market driven (i.e. listening to the customer) development and having mechanisms (prioritization) in place to select what requirements to develop. Recent research focuses on release planning from a perspective of what requirements to put forward taking issues of e.g. interdependencies between the requirements into account [4, 6, 7] when planning a release, as well as priority from the customer perspective [8]. Dependencies, allocation to a certain release, risk, effort and priority are all factors addressed by the *Evolve method* presented by Greer and Ruhe in [1, 9] .

In the same manner the realization that requirements are often on different levels of abstraction and in varying stages of refinement has been recognized in both industry and research [10, 11].

This is often used as a way of working in industry by having different requirements documents for different reasons, e.g. one aimed for market/management were requirements are abstract and are closer to visions than actual requirements (Market Requirements Specification - MRS). The MRS is later refined to product requirements (Product Requirements Specification - PRS) by engineers and/or managers that interpret the MRS to form actual requirements (that should be testable and unambiguous, although it may not always the case). The next step is to refine and add to the requirements by adding technical details and producing Technical Requirements Specifications (TRS) based on the PRS. This offers a refinement of requirements from vision to technical requirements, almost design, through a document oriented perspective where different people work on different documents interpreting requirement statements along the way. The Product Manager's role may span from actively participating in the work with high level (MRS), to the next stage lower level (PRS) requirements, and planning for what requirements should be allocated to what projects. Exactly what requirements are used as input to projects varies, as does the abstraction level within most documents (MRS, PRS, and TRS).

Requirements on different levels of abstraction and at varying levels of refinement are considered by some as crucial input to the development in order to get a better understanding of what should be developed and why [10]. The basic notion is that both the abstract (long-term overview) and the detailed (giving context and the short term-view) is important [12, 13], and the two used in combinations offers a better understanding.

The field of goal based requirements engineering (see e.g. [14-16]) focuses on the elicitation of goals that are to become requirements at some point, working from the top down.

Wiegers [17] and Lauesen [18] describe that requirements can be of different types pertaining to what they should be used for, not totally unlike the industry view of dividing requirements of different types into documents, from goal (abstract natural language formulations [19]) to technical design like specifications. The focus is on that requirements be specified on different abstraction levels depending on usage, e.g. project type.

The contribution of the Requirements Abstraction Model is set around taking advantage of the fact that continuous requirements engineering means that requirements are caught/elicited on different abstraction levels. Abstraction levels subsequently are used as a "motor" to facilitate work-up (not flattening, i.e. forcing all requirements to one level of abstraction) of requirements. Work-up is accomplished by breaking down abstract requirements into detailed ones, and vice-versa (see Section 5.4.3).

This work-up facilitates initial analysis and refinement of requirements to the degree of producing good-enough requirements for project initiation, as well as explicitly linking all requirements to product strategies as a means to offer decision support for e.g. management.

The RAM does not assume a starting point (e.g. starting with goals), but rather takes what requirement is available as input and uses it as a base. Furthermore there is no choice of *one* abstraction level, i.e. flattening, all requirements depending on project type since the continuous requirements engineering is not project initiated rather product oriented in nature. In a product development situation there is a need for decision support on multiple levels before any release planning and development activity can be undertaken (e.g. in project form). Through work with the RAM requirements on a high level of abstraction (comparable to product strategy), and requirements on a low level of abstraction (good-enough as input to a project) are available. In addition, as several levels of abstraction are offered, a richer understanding can be obtained as to the purpose of a requirement, its origin, and so on, by looking at requirements over the abstraction level boundaries.

The fact that requirements are not flattened (forced to the same level of abstraction), or put into one repository (regardless of abstraction level), means that requirements produced when using the RAM offer the possibility for comparison of requirements against each other on one or several abstraction levels. This is a prerequisite for later planning and prioritization activities.

5.3. Research Context – Background and Motivation

The development of the RAM was prompted by the increased pressure on product development organizations to handle an escalating load of requirements coming in to product management on varying levels of abstraction and detail. Moving away from project centered development (project initiated requirements engineering) towards product centered development (requirements initiated development) demands support

for handling the incoming requirements. Giving product management a model for how to handle (and use) the requirements and their varying abstraction levels was the central motivation behind the development of the RAM.

The need for a supporting model for handling and working with requirements in this context was also explicitly identified during a software process assessment activity performed at DanaherMotion Särö AB (DHR) (see Chapter 3). The DHR case is used as an illustration of the problem and the needs described throughout this chapter.

DHR develops and sells software and hardware equipment for navigation, control, fleet management and service for Automated Guided Vehicle (AGV) systems. More than 50 AGV system suppliers worldwide are using DHR technologies and expertise together with their own products in effective transport and logistic solutions to various markets worldwide. The headquarters and R & D Centre is located in Särö, south of Gothenburg, Sweden. DHR has 85 employees. DHR is certified according to SS-EN ISO 9001:1994 (currently working on certification according to ISO 9001:2000), but there have not been any attempts towards CMM or CMMI certification.

DHR has a wide product portfolio, as the ability to offer partners and customers a wide selection of general variants of hardware and supporting software is regarded as important. Product Managers oversee development and new releases of products.

The initiative for an extensive process improvement program was initiated by DHR in recognition of the importance of optimizing their product development, and especially the area of requirements engineering was targeted. The first step of the improvement program was to perform an assessment activity to establish a baseline and identify possible improvement issues.

5.3.1. Process Assessment Results

During the process assessment conducted at DHR in total nine major improvement issues were identified and formulated (see Chapter 3 for detailed information). These nine issues were subsequently prioritized and packaged (according to dependencies and priority) into three separate improvement packages to be addressed in turn (see Chapter 4 for detailed information). The Requirements Abstraction Model was primarily built to address the first of these three improvement packages, but also to prepare for the subsequent two (see Section 5.7). Improvement issue package 1 consisted of three issues as can be seen in Table 5-1[1]. Below the issues are expanded upon and described to illustrate the state they were in at initiation of the process improvement activity that initiated the creation of the RAM.

[1] The improvement issues "issue-id" has been altered from their original state for reasons of simplification. Title and description in Table 5-1 are unaltered.

5.3.1.1 Abstraction Level & Contents of Requirements

Issue-1 speaks to the need of looking over and establishing how the requirements are specified regarding abstraction level and level of detail. During the process assessment, it was ascertained that requirements (obtained from multiple sources) were often specified on different levels of abstraction, and that some were detailed while other were not. This depended on several factors, e.g. who stated the requirement (source), who specified the requirement (experience and expertise), and to what extent the requirement was analyzed and refined subsequent to the initial draft. See Example 5-5 for an example.

Table 5-1. Improvement issue package 1.

Improvement Issue Package 1
Issue-1: Abstraction level & Contents of requirements Each requirement should be specified on a predefined level of abstraction with certain characteristics (attributes attached to it), enabling requirements to be comparable and specified to a certain predefined degree of detail.
Issue-2: Roles and responsibilities - RE process To avoid misunderstandings as well as avoiding certain tasks not being completed the roles and responsibilities of all project members should be clearly defined before project start.
Issue-3: Requirements upkeep during & post project In order to keep the requirements up to date during and post project the requirements have to be updated as they change.

Issue-1 and State-of-the-art: Looking at literature and previous studies conducted regarding the state of requirements engineering in industry the points described in Issue-1 are not exclusive to the DHR case in any way. An example is the findings in context of the REAIMS Esprit project [20, 21]. This is further supported by the findings in [10, 22-24] and findings presented in Chapter 2, where certain issues were identified as general deficiencies in the requirements engineering process, i.e.

Example 5-1. Abstraction level and level of detail.

This is an example of two requirements specified on different levels of abstraction and at different levels of detail (i.e. more information is given in the case of Req. 2).
Requirement 1:
TITLE: "Support standardized formats"
DESC: "The system should support standardized formats"
Requirement 2:
ID: "X-11B"
TITLE: "Save output to XML"
DESC: "A user should be able to save output to a file in xml format in order for the data to be exported to the ERP system. Requirement O-7C needs to be implemented before this requirement."
SOURCE: "Kevin Incognito"

- Analysis and Negotiation (leading to good-enough specification),
- Interpretation and Structuring of requirements,
- Testability and Measurability (of the specified requirements),
- Reviews (of the requirements), and

Varying abstraction level of specified requirements leading to problems in e.g. comparing requirements.

5.3.1.2 Roles and Responsibilities – RE Process

Issue-2 is related to there being an unambiguous and clear description of the responsibilities needed to support the requirements engineering process, as well as an explicitly defined structure for what these responsibilities entailed.

The market driven product development at DHR made it necessary to elicit/catch requirements continuously, i.e. there was a need for a continuous requirement engineering process and roles to support the activities this involved, which partly lie outside of the traditional project initiated requirements engineering process.

Issue-2 and State-of-the-art: The importance of making roles and subsequent responsibilities clear is not an issue reserved for requirements engineering, but pertinent in any organization involved in product development [25, 26]. Roles and responsibilities needed to handle requirements and in essence manage products often lay outside the focus of traditional quality assurance frameworks like CMM [27] since it is not a direct constituent of the development, rather an initiator of it.

5.3.1.3 Requirements Upkeep During & Post Project

Issue-3 is about keeping requirements "alive" as long as they are relevant. Keeping requirements updated is largely connected to Issue-2 (i.e. who has the responsibility to update requirements). The reasoning was that the requirements should be used throughout the development cycle, i.e. initially for establishing what is to be done and why, later as a basis for validation (e.g. system test), and for purposes of possible reuse. Keeping requirements updated was deemed necessary in order for requirements to be usable over (and beyond) the entire development cycle.

Issue-3 and State-of-the-art: There are any number of reasons for keeping the requirements up-to-date during and post development. If changes are made to what is done and the requirements are not updated the requirements do not reflect the end-result of the development activity. This can be a problem both technically and legally since the requirements cannot be used as a basis for e.g. system test [28] or as a binding agreement (contract) between customer and developers [18].

In the case of DHR's continuous product development the requirements sources are many (see Figure 5-1) and the customer is not generally identifiable as one external customer, but rather the role of customer is taken by e.g. the Product Managers. Thus, the requirements are the contract between management (which Product Managers are a part of) and the projects designing and implementing new products/product versions.

5.3.2. Motivation Summary

Two main factors motivated the creation and evolvement of the Requirements Abstraction Model, (i) a direct need identified in industry, (ii) and that a suitable model was not be found in literature, i.e. a model for continuous requirements engineering catching and handling requirements on multiple levels of abstraction.

Regarding the industry need (i) there was an expressed interest to make the way of working with requirements market driven and product centered [29]. Utilizing product management and the multitude of requirements gathered (from multiple sources, but more importantly on multiple levels of abstraction) by the organization to improve the product alignment towards the needs of the market. The actual need for this type of

model has also become even more apparent after the development of the RAM. A different organization has shown an interest in applying the model to their organization, due to that they experienced similar challenges as DHR.

The way of working with requirements was to reflect good practices identified in state-of-the-art, i.e. adopting appropriate good practices (see e.g. [20, 21, 30]) in the RAM. Moreover, aspects like repeatability and structure were seen as important, but only to the extent of offering a clear support-framework without being cumbersome and overbearing. The plan for achieving a process improvement was to develop a way of working with requirements based on the needs and experiences of the management, Product Managers, and engineers.

Looking at state-of-the-art there is research being conducted in the area of continuous requirements engineering in a market driven development situation, although many problems remain (as described above in combination with the improvement issues), and there is a lack of an appropriate model (ii). Offering support for continuous product centered requirements engineering that not only considers abstraction levels, but also use them, prompted the explicit effort to develop the RAM in a way suitable for organizations faced with certain issues, rather than tailoring the model towards one organization.

5.3.3. Evolvement of the Requirements Abstraction Model

The RAM was developed in several stages as can be viewed in Figure 5-2.

Subsequent to the initial formulation the RAM went through two major validations (Validation One and Two are seen as *static validation*) before it was deemed good-enough to be tested in a live industry setting (*dynamic validation*). The *static validation* steps involved brainstorming/interview sessions with product and Project Managers as well as discussions with representatives for development, system test, and upper management.

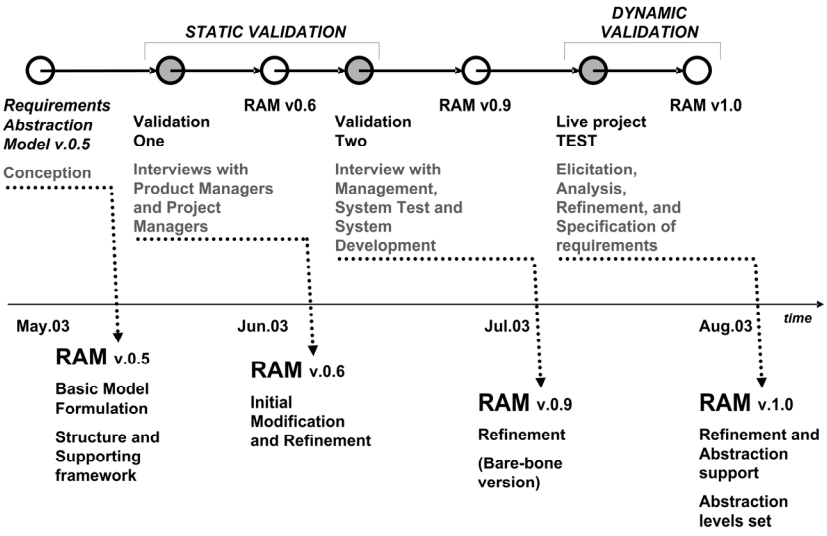


Figure 5-2. Evolvement time-line of the RAM.

The *dynamic validation* consisted of using the RAM for a real live requirements engineering effort. Both the static and dynamic validations had a fundamental impact on the model, and they are described in Section 5.5.

The RAM version presented in this chapter is version 1.0 (see Section 5.4), i.e. the model that evolved as a result of the validations. RAM v.1.0 was mainly aimed towards establishing support for the initial stages (specification, placement and work-up) of the continuous requirements engineering performed by Product Managers. This is illustrated in Figure 5-3 where two types of requirements engineering can be seen, continuous and dedicated. The continuous requirements engineering is considered a prerequisite for being able to deliver a sub-set of the total requirements to one or more development projects.

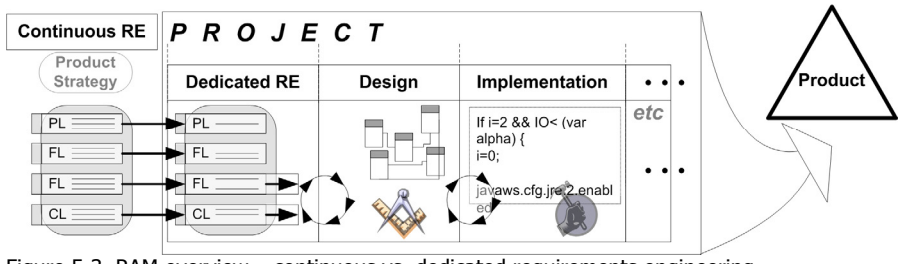


Figure 5-3. RAM overview – continuous vs. dedicated requirements engineering.

5.4. RAM Structure & Supporting Process – An Overview

This section describes the Requirements Abstraction Model's structure and the supporting process developed as a part of it. The gray example boxes in this section offer exemplification continuously throughout the text. Examples are important when working with the RAM since the model is example-driven, i.e. relevant examples (regarding the requirements engineering of the product in question) are important as a means to support training and use of the RAM for continuous requirements engineering. Developing relevant examples is a part of the Model Tailoring (see Section 5.4.5), as is ascertaining the number of abstraction levels appropriate, and attributes needed for each requirement. The version of the RAM presented in this chapter (Sections 5.4.1 through 5.4.3) is based on the one developed at DHR, and it is an example of what the RAM can look like. The examples are not specific to any domain or organization, rather general in nature. This offers a general model exemplification and overview. The version of the model presented here is

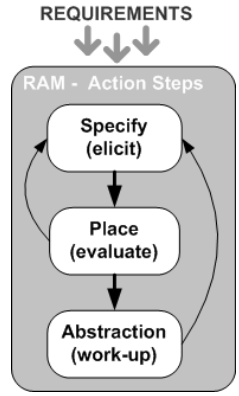


Figure 5-4. RAM action steps.

intended as a starting point for anyone wanting to tailor it for their specific organization and products.

Requirements engineering using the model involves the following three basic steps, as can be seen in Figure 5-4. The first step (Specify) involves specifying the initial (raw) requirement and eliciting enough information about it to specify a number of attributes. The second step (Place) is centered around what abstraction level the now specified requirements resides on, and last (Abstraction) each requirement goes through a work-up. Each of these steps is described in further detail below.

5.4.1. Action Step One – Specify (elicit)

In order to get a uniform way of specifying requirements, four main attributes are to be specified manually in this initial step. The goal of this step is to get an overview of the raw requirement to the extent of it being understood by the Product Manager performing the continuous requirements engineering.

The four attributes are:

1. **Description** The requirement description should not be more than about 5 sentences, and should in broad strokes describe the central essence of the requirement. Example 5-2 offers examples of this attribute.

Example 5-2. Description.

Requirement A	Requirement B	Requirement C
The system shall be able to use standardized formats when communicating with the surrounding environment.	The user shall be able to print information from the system. This print function shall be offered every time information of any kind is presented to the user.	The system shall have support for multiple languages.

2. **Reason/Benefit/Rationale** This attribute consists of two parts; **WHY** the requirement is specified and **BENEFIT** of the specified requirement. "Benefit" should be seen in the context of the subject of the requirement being stated. If the subject is an *user* it should illustrate the benefit to him/her (see Example 5-3, Requirement B), if the subject is the product itself, e.g. in a non-functional requirement, the benefit should reflect the benefit from the requirements perspective (see Example 5-3, Requirement A and C).

Example 5-3. Reason/Benefit /Rationale.

Requirement A	Requirement B	Requirement C
WHY: Use output from system in other systems, e.g. ERP system, logistics systems etc. **BENEFIT:** Be usable in a environment with other systems.	**WHY:** The user wants information on paper. **BENEFIT:** The user can choose how to view and/or distribute information.	**WHY:** Many users don't understand English and prefer the system be in their native language. **BENEFIT:** Make usage of the system more attractive (easier) in an international setting.

3. **Restrictions/Risks** This attribute describes the restrictions and/or risks with the requirement that is not obvious in the description. It is possible to say that this attribute constitutes the negotiation space in the requirement. Example 5-4 offers examples of this attribute.

Example 5-4. Restrictions/Risks

Requirement A	Requirement B	Requirement C
Using third party standards means a dependency on external organizations. By opening up our system to communication that we don't control we lose control over what other systems may be used with ours. In the future this may result in us being unable to control things such as which parts are bought from us and which parts are bought from other suppliers.	Communication with printers is an issue. The format for this communication has to be defined in more detail.	Should there be a restriction to languages that use the Latin alphabet? If e.g. Chinese is included it may result in problems with user interface logic etc.

4. **Title** The title should reflect the contents of the requirement and should not be longer than five words. Example 5-5 offers examples of this attribute.

Example 5-5. Title.

Requirement A	Requirement B	Requirement C
Standardized formats for communication	Print system information	Support for multiple languages

As these four attributes have been specified, the next step is to ascertain the abstraction level of the requirement in question. Additional attributes to be specified are described in Section 5.4.4.

5.4.2. Action Step Two - Place

The RAM consists of a number of abstraction levels (the driving motor of the model). This step involves analyzing what level a requirement is on, and placing it on this level.

Looking at Figure 5-5, four abstraction levels can be seen, i.e. Product Level, Feature Level, Function Level, and Component Level. (See Example 5-6 for exemplification of the placement of requirements on abstraction level described here)

The Product Level is the most abstract level. Requirements on this level are goal-like in nature, i.e. not fitting the normal definition of a requirement (e.g. testable and unambiguous, [30]), thus the use of the term "requirement" can be

Figure 5-5. RAM abstraction levels.

viewed as somewhat questionable in this case. Nevertheless, "requirement" is used for statements on all the four levels of abstraction. This is motivated by the fact that requirements in the model do not exist in isolation, but are always broken down to a level that is in line with the traditional meaning of the word "requirement" (see Section 5.4.3), as a part of the RAM work-up.

Example 5-6. Placing example requirements.

Looking at Requirements A, B and C (see examples 2 through 5) the following is an exemplification of the placement procedure following the how-to guide displayed in Figure 5-9 in Appendix A.

As illustrated, the guide poses a series of questions to steer the initial placement of the requirement on a certain level. By following the guide step-by-step (grey dashed line with the arrows) an attempt is made to place the example-requirements ***A: Standardized formats for communication***, ***B: Print system information*** and ***C: Support for multiple languages***.

The first question is if the requirement is functional or not, or if the requirement in question described what (testable) characteristics a system should provide. This applies to ***B: Print system information***, but not to the other two requirements since none of them fall into the description of providing testable characteristics to the system.
The next question is if the requirement consists of specific suggestions of HOW things are solved. This does not apply to any of the example-requirements. However if ***B: Print system information*** had information in it that spoke to details for solutions, e.g. "support the post-script standard for printing" it would have been a candidate for the Component Level.
The next question is if the requirement is comparable to the product strategies. Both requirements ***A: Standardized formats for communication*** and ***C: Support for multiple languages*** are fairly abstract in nature. Requirement ***A*** basically means that the product should open up to communicating in a standardized way with the surrounding environment. This goes against a strategy saying, "to box in the customer to a certain standard (maybe specific to the product developing organization) and a certain range of products". Requirement ***A*** is however in-line with a strategy saying "to offer a customer the choice of other products by enabling them to communicate with ours". We place requirement **A** on the Product Level as it is directly comparable to product strategies. If requirement ***C*** is compared to the product strategies it may be possible to deduct whether or not it complies, but probably only indirectly. It is probably not within the product's strategy "to support multiple languages in their products", however "to offer the product to an international market" may be the case. Thus, requirement ***C*** is not directly comparable to the product strategies, but close.
The next question posed is if the requirement describes "what the system should include/support". This fits requirement ***C*** since it speaks to that the system should have support for multiple languages, i.e. requirement ***C*** is placed on Feature Level.

Product Level requirements are considered abstract enough to be comparable directly to the product strategies, and indirectly to the organizational strategies. In the context of the RAM, product strategies are e.g. rules, long and short-term goals, and visions pertaining to a product specified by management. Product strategies are in other words what govern what is to be done (direction of the product), and what is not, depending on e.g. targeted market segments, competitor situation, and so on.

The Feature Level is the next level in the model. The requirements on this level are features that the product supports. Feature Level requirements should not offer details as to what functions are needed in order for the product to support a feature; rather the requirements should be an abstract description of the feature itself.

The Function Level is as the name suggests a repository for functional requirements, i.e. what a user should be able to do (actions that are possible to perform), but also for non-functional requirements. The main criterion is that the requirement should be descriptive of what a user (or the system in the case of non-functional requirements) should be able to perform/do. In general, Function Level requirements are detailed and complete enough to be handed over to a system designer

for further evolution and finally be a basis for the design. Functional Level requirements should strive to be testable and unambiguous.

The Component Level is the last level of abstraction in the RAM. Component Level requirements are of a detailed nature depicting information that is closer to (or even examples of) how something should be solved, i.e. on the boundary of design information. The main reason for the existence of this level is twofold. Many requirements that come from internal sources (e.g. engineers) are on this level of abstraction. The Component Level can also act as a possibility to break down Function Level requirements in more detail and/or set limits to a Function Level requirement. This last point is elaborated upon in Section 5.4.3.

In order for this RAM action step to be completed (i.e. the initial placement of the requirement on appropriate abstraction level) a how-to guide was developed, as can be seen in Figure 5-9 in Appendix A. This guide operates on the principle of asking a series of questions and thus guiding the initial placement of the requirement. It should be noted that the how-to guide is only part of the support material offered to the professionals working with the requirements. Other support materials consist of e.g. a list of multiple examples (of requirements on all abstraction levels) relevant to the requirements engineering of the product in question. Example 5-6 offers exemplification of the placement of a requirement on a certain abstraction level.

It is important to realize that the initial placement of requirements as described in this section (and exemplified in Example 5-6) is not an absolute, but a balance, where the requirements have to be weighed against the examples and instructions in e.g. the how-to guide. It is imperative for the requirements engineers (Requirements Manager) working with the RAM to be consistent, i.e. placing requirements of comparable abstraction levels on the same level, and enforcing consistency over time.

5.4.3. Action Step Three – Abstraction (Work-up)

Subsequent to the initial placement of the requirement on an appropriate abstraction level the work-up of the requirement can commence. This third step of the RAM involves abstracting and/or breakdown of a requirement, depending on the initial placement of the *original* requirement. The work-up process involves creating new requirements (called *work-up requirements* hereafter) on adjacent abstraction levels or linking to already existing ones, depending on the situation.

This process takes place for several reasons. First, as mentioned before, one of the model's goals is for every requirement to be comparable with the product strategies. Thus, every requirement (on Feature Level or lower) has to be abstracted up to the Product Level (see Figure 5-5) in order for this to be possible. This creates the first work-up rule (R1):

R1: No requirement may exist without having a connection to the Product Level.

R1 can be met in one of two ways, one or more new *work-up requirements* are created, or the requirement in question is linked to already existing requirements on an adjacent upper level. In either case, the original requirement is abstracted upward and can be compared (indirectly) to the product strategies.

In addition to abstraction, there may also be reason for requirements to be broken down enough to act as a basis for design (good-enough for project initiation). For a

requirement to be detailed enough and on the right level of abstraction for this to be possible every requirement (on Feature Level or higher) has to be broken down to Function Level (testable and unambiguous). This creates the second work-up rule (R2):

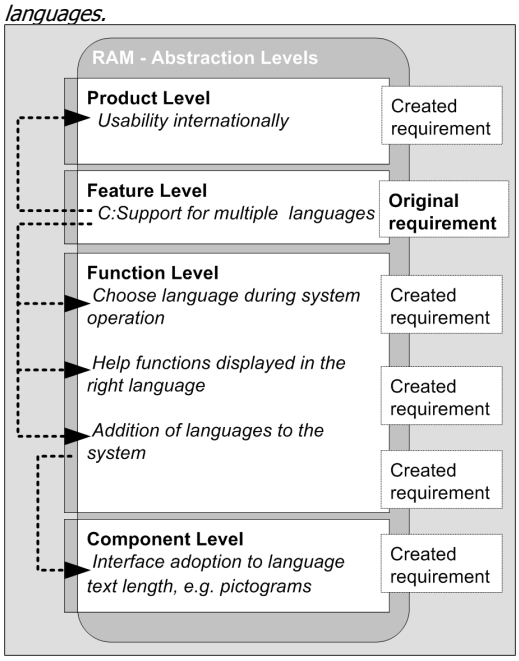
R2: All requirements have to be broken down to Function Level.

Like in the case of R1 this can mean creating one or more work-up requirements on adjacent levels, or linking the requirement in question to already existing requirements on lower adjacent levels. Either is acceptable as long as the requirement(s) on lower levels satisfy the upper level one.

Satisfy in this case pertains to the issue of breaking down a high-level requirements (from e.g. Feature Level) to Function Level, where the requirements on lower level together satisfy the original one to the extent of giving a foundation good-enough for the initiation of realization (design) of the requirement. The main reasoning behind this is that requirements are meaningless if they cannot be delivered to a development effort (i.e. left on a too abstract level). Typically, R2 involves the creation of several work-up requirements being created.

An example of R1 and R2 can be viewed in Example 5-7. Here the original requirement "*C:Support for multiple languages"* (placed on Feature Level) is abstracted to Product Level (R1) through the creation of a work-up requirement "*Usability internationally"*, and broken down to Functional Level (R2) where three work-up requirements are created as a part of the breakdown.

Example 5-7. Abstraction and breakdown of example-requirement *C: Support for multiple languages.*



The breakdown to Component Level offers further details to relatively low-level requirements and is often associated with suggestions of *how* to implement a requirement. This level is not mandatory in the process of breaking down requirements. However, it should be observed that it is not unusual that several of the incoming requirements are on the Component Level of abstraction, thus the level cannot be discarded. The need for the Component Level is illustrated in the validation part in Section 5. In addition, the Component Level can also act as clarification (refinement and/or limitation) to one or more Function Level requirements. An example of this can be viewed in Example 5-7, where a technical limitation pertaining to the user interface adds to (and limits the possible solutions by adding a design element) the requirement on Function Level.

As was the case in the previous step (initial placement of the requirement) a how-to guide was developed to aid in the work-up of requirements. This guide (see Figure 5-10 in Appendix A) is example-driven adhering to the two rules (R1 and R2) described above. It shows mandatory as well as optional work-up of the requirements.

It should be noted that none of the RAM action steps is final, i.e. in some instances it is necessary to iterate (go back) and rethink the initial placement of a requirement (as the work-up offers an analysis that may change the perception of the original requirement). This can also involve eliciting additional information from the requirement's source if this is possible.

During the work-up, it is important to stay true to the original requirement, or rather the intention of it. The creation of new requirements as a part of the work-up should not stray too far from the initial intention of the original requirement, and thus give rise to totally new requirements that are related to but outside the scope of initial intention. It is inevitable to create new requirements (especially if the original requirement is on a high abstraction level) as the work-up is designed to create new relevant work-up requirements to the extent that they satisfy the original requirement. However, this is not the same as including new requirements based on "this might also be a good idea" philosophy, as this could give rise to a mass of new requirements. As the model (and the usage of it) is aimed at offering support to professionals, it is also very dependent on the same professionals pertaining to how well it works. A recommendation when performing work-up of original requirements is always to ask the question "is this new (work-up created) requirement really necessary in order to satisfy the original requirement"? If the answer is "yes" then there is no problem, but if there is uncertainty, the work-up should be stopped.

This does not mean that good ideas pertaining to new requirements should be discarded along the way in any case, but they should not be a part of the work-up, rather be specified as new original requirements on an appropriate level (and in turn get a work-up themselves).

5.4.4. Additional Structure and Process – Roles, Attributes and Rules

As requirements are specified, placed and worked-up additional attributes are specified (in addition to those introduced in Section 5.4.1) for each requirement as applicable. summarizes the additional attributes with a short description linked to each. Under the comment column there is indication whether or not the attribute has to be specified (mandatory) or not (optional), as well as if it has to be specified manually or if it is auto generated (assuming tool use). The subsequent subsections (5.4.4.1 and 5.4.4.2) elaborate regarding the attributes and their rationale.

Table 5-2. RAM requirement's attributes. (For attribute 1 to 4, see Section 5.4.1).

Attribute Title	Description	Comment
5. Requirement Source	This is a link to the source of the requirement. This can be a physical person, document, group, or meeting. The exactness depends on the information available.	Mandatory (Manual)
6. Requirement Owner	A link to the person who "owns" the requirement and is responsible for the follow-up of the requirement. This person acts as the advocate of the requirement. This role is always held by a internal person in the product development organization.	Mandatory (Manual)
7. Requirements manager	An identification of the Product Manager responsible for the specification, placement, and work-up of the requirement.	Mandatory (Auto generated)
8. Relation/Dependency	One or several links to other requirements on the same level of abstraction. This attribute's aim is to record important relations/interdependencies of different types between requirements. Every link can be augmented by a explanation in free-text.	Optional (Manual)
9. State	A requirement in the RAM can have different states giving information of the status of the requirement, see Figure 5-6 for details.	Mandatory (Manual)
10. Reject Reason	If a requirements state is set to "Rejected Requirement", i.e. it is deemed out of scope in relation to the product strategies, then this attribute records the rationale of the decision.	Mandatory if requirement is rejected. (Manual)
11. Due Date	This attribute's purpose is to ascertain that requirements are not forgotten, e.g. put as draft indefinitely. The manual usage of the attribute can be in case of e.g. customer deadline etc.	Mandatory, auto set to 30 days if nothing else is specified.
12. Version	Records version on requirements level rather than document level. Enables requirements' history to be viewed.	Mandatory (Auto generated)
13. Date of Creation	Records the creation date of the requirement.	Mandatory (Auto generated)
14. Last Changed	Indicates when the last change was performed.	Mandatory (Auto generated)

5.4.4.1 Traceability and Role Attributes

Attributes 5, 6 and 7 are linked to traceability issues, enabling (roles) people to be linked to a certain requirement, ensuring that responsibilities are clear not open to interpretation, on a requirement level rather than a document level, to avoid issues like certain requirements being neglected or even overlooked entirely.

As requirements are caught/elicited from multiple sources everything from a person, to a market survey or a competitor analysis, can be the official "source" of the requirement (in the latter two the sources are the reports/documents produced). This makes the role of Requirement Owner important as this person is charged with the responsibility of seeing that the requirement is followed through on. A Requirement Owner is the representative of the Requirement Source when this role is silent, e.g. when the source is a survey or an external party like a group of customers. In some instances

the Requirement Source and the Requirement Owner can be the same person (e.g. in the case of an internal engineer formulating the original requirement).

The Requirements Manager role is typically represented by the Product Manager charged with the responsibility of actually working with the requirement throughout its lifecycle. During the RAM action steps (see Figure 5-4) the Requirements Manager can utilize resources needed, e.g. asking system experts and/or domain specialists for input during the work-up of the requirement. The cooperation between the Requirements Manager, Owner and Source (when applicable) is especially important as they respectively possess different perspectives and knowledge pertaining the requirement in question.

If a requirement is created as a part of the work-up (a work-up requirement not an original one), the attributes of Requirement Source/Owner/Manager are set to the Requirements Manager responsible for the work-up.

5.4.4.2 Process (Attributes)

State reflects how the requirement is handled in the product development organization and how it is set to different states reflecting the status. Figure 5-6 offers an overview of the different states. Looking at Figure 5-6 states *A*, *B* and *C* are a part of the RAM action steps (see Figure 5-4) and the continuous (project independent) requirements engineering and is basically about drafting the requirement. The work done associated with the three states is specified in further detail in Table 5-3 where each separate step and sub-step is described. Observe that states *D (dependent on prioritization)* and *F (dependent on packaging into release packages)* are addressed in future work (see Section 5.7).

In RAM v.1.0 the possible states a requirement can exist in are *A: Draft requirement*, *B: Rejected requirement*, *C: Incompletely specified,* and *G: Refined requirement*. A requirement can reach states *A*, *B*, and *C* during the continuous requirements engineering, i.e. the work done as a part of the RAM action steps. State *G* is however reached as the requirement in question is subjected to further refinement and validation during the *dedicated* requirements engineering.

Dedicated requirements engineering is performed on a chosen subset of requirements after project initiation, and involves refinement by the development project and system test departments to assure testability and unambiguity as well as completeness of the requirements.

It should be noted that state *B* is dependent on the requirement being out of scope, i.e. that it is not in line with the product strategies. Generally the out of scope requirement is rejected off hand (and attribute *10: Reject reason* is specified), but in some instances an alternate decision can be taken. If a requirement is considered out of scope but is important for any reason (e.g. an important customer is the source) an exception can be made. However, this exception is an explicit action and has to be approved by both the Requirements Manager and the Requirement Owner, as well as checked against upper management. The general rule is that all requirements not rejected should be in line with the strategies formulated for a product. Exceptions to this rule should be kept at a minimum in order to use the full potential of the RAM.

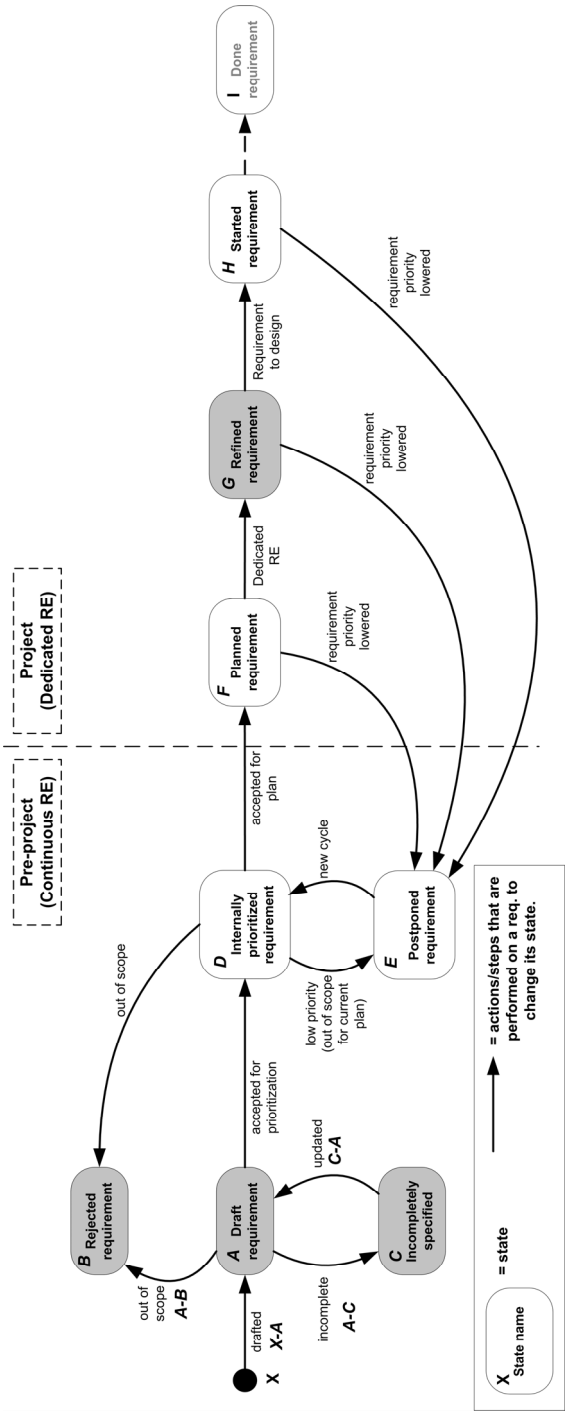


Figure 5-6. Requirement's states in RAM.

Table 5-3. RAM state steps (see Figure 5-6).

Step ID	Step Description	Sub-Steps
X-A	The requirement (req.) is caught/elicited and drafted by the Requirements Manager . The work is done by the Requirements Manager , that utilizes relevant resources if the need arises (e.g. Requirement Owner, experts and so on are important parties in this process).	**X-A-1:** Specify attribute 1 to 4 (Specify). **X-A-2:** Determine on which level the original requirement is on (Place). **X-A-3:** Specify all attributes on relevant level. **X-A-4:** Abstract and/or Breakdown the original requirement according to work-up rule **R1** and **R2** (work-up). **X-A-5:** Validate requirement against Requirement Owner and Requirement Source.
A-B	During (or rather, as the requirement is abstracted) work-up the requirement is checked against product strategy. Requirements deemed as not in line with strategies are deemed out of scope and thus rejected. As a requirement is rejected, directly related requirements (work-up requirements) on adjacent abstraction levels either are removed as well or re-linked (enforcing **R1** and **R2**).	**A-B-1:** Compare requirement (directly or indirectly) to product strategies. If the req. is not in line with strategy it is rejected. **A-B-2:** If a requirement is rejected for any reason created work-up req. have to be evaluated if they should also be removed. If all are removed, nothing further is needed. However if a work-up req. is left on any level it has to be re-linked to requirements above and/or beyond for the work-up rules **R1** and **R2** to be enforced.
A-C	During validation against the Requirement Source/Owner (**X-A-5**) the req. can be deemed incomplete. This can be a result of e.g. incorrect initial placement, unsatisfactory work-up and so on.	

5.4.5. Model Tailoring

The basic elements of the Requirements Abstraction Model revolve around using abstraction levels, instead of flattening them (or dividing them into separate documents), to work with requirements. The RAM action steps described in Sections 5.4.1, 5.4.2, and 5.4.3 follow this ambition, and the use of attributes (Section 5.4.4) enables a requirements oriented (not document oriented) way of working.

This does not mean that the RAM was developed to be prescriptive in nature. One model fitting all types of products is not a realistic aim, not to mention differences between organizations. To this end, the model is intended to be a framework on which continuous requirements engineering can be based. Several things need to be addressed prior to the model being set into operation in an organization. This can be seen as a tailoring of the RAM to fit a specific product (organization), as well as giving the adopting organization's representatives (e.g. Product Managers) a chance to acquaint themselves with the RAM.

Below an overview of the tailoring aspects is offered.

- **Abstraction Levels** (and usage of them) are a fundamental part of the RAM. However, the *number of abstraction levels* and the *abstraction degree* of each level is not to be considered absolute.
  The number of abstraction levels needed depends on the product and organization. To facilitate abstraction (to a level comparable to product strategy) and breakdown (to a level good-enough for project initiation) different organizations may have different needs. Three levels may suffice in some cases where requirements are generally only caught/elicited on three levels, other organizations may have the need for five abstraction levels, and so on.
  The same reasoning applies to the abstraction degree of each level. It is dependent on the number of abstraction levels used, i.e. the jumps between levels are greater if few levels are used and vice versa.
  As a rule, two things determine the number of abstraction levels and the abstraction degree of these levels. First, the requirements caught/elicited can be used as an indicator (if requirements are typically caught on four levels this amount may be appropriate). Second, the need for work-up of the requirements, i.e. does e.g. four levels (instead of three) help in making the work-up of the requirements easier? This mainly pertains to the distance between the abstraction levels.
- **Example-driven** is a term used in the description of the RAM. As different organizations (products) are dependent on different vocabularies, domains, technical terminology, traditions and needs, in terms of e.g. the number of abstraction levels and abstraction degree, relevant examples have to be developed. The examples are to illustrate most "common" situations encountered by the Product Manager charged with the job of performing the continuous requirements engineering.
- **Attributes** need to be reviewed and adapted to the needs of the organization. It is important to realize that perfect attributes are meaningless if they are not specified. A balance between benefit and cost has to be reached, and the benefit of specifying a certain attribute should be well anchored in the organization.
- **Roles** present differ from organization to organization. In some cases new roles may be needed, e.g. if no Product Manager or equivalent role exists charged with the continuous requirements engineering, in other cases already present roles can be redefined. This is not primarily to fit the model, but rather to fit the nature of requirements initiated development (market driven and continuous).
- **Naming** of abstraction levels, attributes, roles and so on should be adapted to fit the vocabulary of the organization. The main concern is that all parties have the same understanding of what is meant and a naming standard is used.
- **Product Focus** means that an organization with homogenous products (pertaining to the development) can work with the same version of the RAM (tailored to their needs). If an organization has heterogeneous products (e.g. large organization with different types of products/product lines) several tailored versions of the RAM may be beneficial, e.g. one for each "group/type" of product.

- **Support Material**, e.g. how-to guides is a result of the other tailoring points described above. The number of abstraction levels, abstraction degree, relevant examples, and so on all govern how the guides are developed and what contents they have.

Details pertaining to the tailoring of the RAM are considered outside the scope of this chapter. However, it should be noted that a tailoring effort of the RAM is underway at present, which is a prelude to the test and implementation of the RAM in a second organization faced with a situation similar to the one identified at DHR.

5.4.6. Requirement Abstraction Model – Summary of the Action Steps

Product Managers are offered a model to handle requirements on multiple levels of abstraction in a continuous requirement engineering environment, and a structured framework for how to work-up these requirements. The main steps of the RAM are summarized below:

1. **Specify** initial attributes (attributes 1-4) to form a basic understanding of the requirement and ascertain its abstraction level. This is generally done by the Requirements Manager (Product Manager) in cooperation with the Requirement Owner (and the Requirement Source when applicable) (see Section 5.4.1 and 5.4.4).
2. **Place** the requirement (see Section 5.4.2) on an appropriate abstraction level (using product relevant examples, see Section 5.4.5, and the how-to guide in Figure 5-9 in Appendix A).
3. **Work-up** (see Section 5.4.3) the requirement by specifying additional requirements (called work-up requirements) on adjacent abstraction levels until work-up rule R1 and R2 are satisfied (using product relevant examples, see Section 5.4.5, and the how-to guide in Figure 5-10 in Appendix A).

During work-up attributes 1-4 are specified for the new work-up requirements. As the work-up is being completed additional attributes need to be specified (see Section 5.4.4).

During work-up it may be necessary to rethink the initial placement of the original requirement and reinitiate work-up.

5.5. RAM - Validation

This section offers a presentation of the evolutionary development of the RAM as it was validated in industry against an organization faced with the improvement issues described in Section 5.3.

The validation is divided into two main parts, static and dynamic validation (see Figure 5-2). The static validation consists of reviews and walkthroughs of the RAM, and the dynamic validation consists of a live industry requirements engineering effort where the model was put through its phases.

5.5.1. Static Validation

As the RAM was developed it was considered important to get input from all stakeholders involved with the continuous requirements engineering, as well as the ones using requirements as input to their work, to participate in the development of the RAM.

The first step was to identify relevant general roles at DHR, and their relation to the requirements engineering process. The roles identified were project centered, as the organization largely was centered around development projects (i.e. project initiated requirements engineering) at the time of the model's development. This also meant that the roles were not directly compatible with the ones described by the RAM (see Section 5.4.4).

Below the identified roles are listed. The roles were to some extent influenced by the roles identified during the process assessment activity performed earlier at DHR (see Chapter 3) (see Section 5.3.1).

A. The *Product Manager* role was new to the DHR organization (<1 year). The role's responsibilities were centered on product coordination and product (release) planning, as well as development project coordination. The Product Manager was considered responsible for a product and answered to upper management.
B. The *Orderer* had the task of being the internal owner of a certain project, i.e. having the customer role and if applicable the official contact with an external customer and/or partner. This role is responsible for the official signing-off when it comes to the requirements, i.e. he/she places an order.
C. The *Project Manager* has the traditional role of managing the project, resources, planning, and follow-up. As far as requirements are concerned, the Project Manager is responsible for that the requirements engineering is performed, the requirements specification is written and signed off by the System Engineer.
D. The *System Engineer* is the technical responsible for a project. It is also important to recognize that the System Engineer has the official responsibility for the requirements specification in a project.
E. The *Developer* is a representative for the developers (e.g. programmers) in a project, the ones actually implementing the requirements. The developers use the requirements specification.
F. The System Test role can be described as the traditional role of system test. This role is officially present during initial project meetings and is a part of the verification of the requirements specification.
G. *Upper Management* was identified as a crucial stakeholder since executive power to a large extent resided here. This role also represented the process owners, as well as the ones actively participating in the creation of product strategies.

Static Validation One involved eliciting feedback/input regarding the RAM from the following roles:

- A. Product Manager (2 persons interviewed)
- B. Orderer (1 person interviewed)
- C. Project Manager (2 persons interviewed)
- D. System Engineer (1 person interviewed)

Static Validation Two involved eliciting feedback/input regarding the RAM from the following roles:

- E. Developer (1 person interviewed)
- F. System Test (1 person interviewed)
- G. Upper Management (1 person interviewed)

The division of roles between the two static validations was deliberate. Static Validation One was centered on the initial formulation of the model and cooperation with roles working with requirements (elicitation, analysis, management, and so on) were considered as crucial. Static Validation Two was centered on the input to and output from the model (what the model needed, e.g. product strategies, and what the model produced, e.g. requirements good-enough for project initiation). Developers and System Test were in the position to give input as to e.g. when a requirement was good-enough for project imitation, testability, and so on. Upper Management was directly involved with product strategies, as well as the process as a whole.

Both validations were carried out in the form of unstructured interviews [31] in the offices of the interview subjects and lasted between one and two hours each.

In addition to the two official static validations described above it should be noted that the development of the RAM was conducted on-site and that unofficial discussions and feedback were commonplace from multiple sources not limited to, but sometimes including, the roles and interview subjects that participated in the official validations.

5.5.1.1 Static Validation Impact and Lessons Learned

Static Validation One mainly influenced the RAM pertaining to *size* and *complexity*.

The initial designs of the RAM were considered too ambitious in terms of how requirements were specified (e.g. more attributes pertaining to relations and dependencies). It led to a larger model demanding further detail in the specification. There was a view amongst the Product Managers that too rigorous demands would result in either that the model would not be used (at least not as intended), or that too much time would go into specifying details that were never used (even if potentially beneficial). It was realized that by simplifying the model pertaining to what was specified in relation to a requirement also dulled the requirement in question. This trade-off was considered and the general view was that it was better to have a model (requirements) which was good-enough and used, than a larger more complete model that was considered too cumbersome.

In addition, the usability of the model was debated and the consensus was that the example-driven nature of the model was important as far as usability was concerned. Product centered relevant examples of specification, placement and work-up were considered necessary, and in combination with abbreviated how-to guides (see Appendix A) enhanced usability of the model substantially, especially in comparison to just offering e.g. a list of rules describing the RAM.

The structure and rules of the RAM were considered important to get a repeatable and comparable continuous requirements engineering process that produced requirements on a level of detail that was predetermined, and not ad-hoc. The ability to compare (directly or indirectly through work-up requirements) requirements to product

strategies was considered very beneficial as it theoretically would be possible to dismiss some requirements off-hand if they were considered to be out of scope. The main risk identified in this scenario was that product strategies had to be present and explicit in order for this to work.

Static Validation Two mainly influenced the RAM pertaining to *process* related issues.

This validation can be divided into two main parts: *RAM produced requirements* and *RAM general process*. RAM produced requirements was the part discussed at length with the Development and System Test roles, and involved how to ascertain that the RAM produced requirements (on Function/Component Level) passed to development (input to initial design) were refined enough, unambiguous and testable. These aspects were used as input to the validation with upper management regarding the *RAM general process* and gave rise to a division of the RAM requirements engineering into two main parts: a *continuous* (pre-project) part and *dedicated* (during project) part. Figure 5-3 illustrates this.

To ensure testability and to get a pre-design review of all requirements a *testability review* was proposed. It meant creating test cases based on the requirements and thus get an indirect review of both testability and (to large extent) completeness and refinement. The creation of test cases based on the requirements was not adding any work effort to the development as this was to be done in any case, but usually at a much later stage (i.e. as implementation is close to completion).

The testability review was not deemed possible to perform during the continuous part, but was placed in the dedicated part of the process. The main reason for this was not to waste effort on requirements not yet planned for development, i.e. only the Function/Component Level requirements allocated to a project would be subjected to the review.

This meant that testability could not be assured for all Function/Component Level requirements in the RAM requirements "repository", but rather only for the ones prioritized enough to be allocated to a project. As a general rule was set that, a requirement on Function/Component Level should strive to be testable, but no formal review would be conducted until project allocation (i.e. performed during the dedicated part).

In addition to the division of the requirements engineering into two parts, upper management identified product strategies as an important factor in the context of performing requirements engineering with the RAM. The general view was that the creation of explicitly defined product strategies would allow better control over product development, as well as tie development efforts (projects) closer to the goals for the product (and indirectly the organization).

5.5.2. Dynamic Validation

As a candidate model (RAM v.0.9, see Figure 5-2) was completed, it was set to be tested in a live development situation (referred to as *dynamic validation* hereafter). The idea was to use the model for actual requirements engineering in order validate its components (e.g. attributes) and if the model was scalable to a real development situation. This involved validating several aspects of the RAM:

1. Abstraction levels
    a. On what abstraction levels did requirements actually come in, and what sources produced what requirements (pertaining to abstraction level)?

   b. Did the work-up create any problems with e.g. too many new requirements (i.e. work-up requirements) created as a result of work-up? This aspect was largely linked to the scalability of the RAM.
2. Was it possible to use requirements directly from the model for the intended purposes, i.e.
   a. Were Product Level requirements comparable to product strategies?
   b. Did the Function Level (together with corresponding Component Level) requirements offer enough material for a development project to perform dedicated requirements engineering?
3. Usability, i.e. was the RAM easy and intuitive enough to use practically?

The main factor limiting the effectiveness of the dynamic validation were the difficulties of actual performing continuous requirements engineering over a long time-period, as the dynamic validation was limited in time and scope. This meant that the "continuous" part was not really tested, i.e. only one development project was to be initiated as a result of the requirements engineering performed during the dynamic validation. Following a continuous product centered requirements engineering view (as intended when using the RAM and described in e.g. Figure 5-1) the dynamic validation would have to be performed over a much longer period eliciting/catching requirements continuously, and initiation several development projects as needed (based on the requirements).

However, the consensus at DHR was that a limited, but nevertheless *real,* test of the RAM would produce a clear indication as to the applicability of the model to solve the problems described in Section 120, as well as answer the questions posed above (see questions 1-3).

Looking at the process of the dynamic validation the Product Manager role (at DHR) corresponded to the Requirement Manager role in RAM (see Section 5.4.4). Two persons in cooperation conducted the actual requirements engineering work performed with the RAM, i.e. the normal Requirements Manager role of RAM was divided during the validation between a DHR Product Manager (driving the work) and the researcher (collection/support). In reality, the researcher's role was not only collection, but also cooperation and collaboration to some extent. The main idea was for a professional in industry to use the RAM in a requirements engineering situation, and by doing so involve other parts of the organization described by the RAM, e.g. Requirement Owner and Requirement Source, but also e.g. system experts and developers as needed to complete the tasks. The guarantee for scalability of the RAM resided in this fact, i.e. that seasoned professionals used the model for a real requirements engineering effort, and could estimate scalability based on their experience. The researcher's role was to support and help in the work.

The dynamic validation (requirements engineering effort) was centered on the development of a new small product needed at DHR. It was set up to be as close to a real market driven continuous requirements engineering effort as possible (but focused in length). Figure 5-7 illustrates the set-up. Requirements were caught /elicited from multiple sources both internal and external. Each requirement was specified, placed, and worked-up in turn. The stop condition for the dynamic validation was when there were adequate requirements in the RAM repository for one development project to be initiated. In a continuous requirements situation the requirements engineering with the

RAM would never stop, but continue initiating project after project over time as the requirements came in.

During the dynamic validation the dedicated part of the requirements engineering (performed in the project) was largely omitted as the aim was to validate RAM v.0.9 (supporting the Product Manager during the continuous part), however a number of testability reviews were performed. The motivation for this was to check Function/Component Level requirements (as they were delivered to the development project) for testability, completeness and ambiguity since the formal review ensuring these aspects was moved to the dedicated requirements engineering (see Section 5.5.1.1, Static Validation Two).

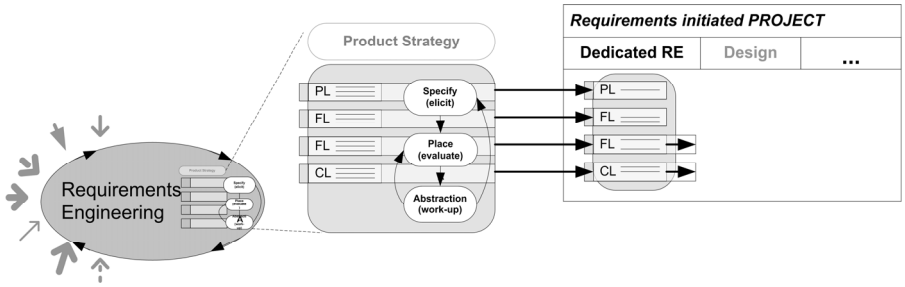


Figure 5-7. Requirements engineering during dynamic validation.

The role of Requirement Source was represented by a diverse group of different internal stakeholders with varying agendas. Some of these internal stakeholders (primarily marketing and sales personnel) represented external parties during the dynamic validation (as they often are pertaining to the "normal" case in everyday work).

5.5.2.1 Dynamic Validation - Lessons Learned

Below each of the questions posed in Section 5.5.2 (i.e. question 1 to 3 with sub-questions) is discussed based on the experiences from the dynamic validation of the RAM.

Looking at the requirements engineering performed, 78 original requirements were elicited/caught before Product Management considered a development effort possible.

1.a. As the original requirements came in, the distribution over abstraction levels was diverse, as can be seen in Figure 5-8. In the center of Figure 5-8 the incoming requirements' distribution over abstraction levels can be viewed as percentages of the total amount of requirements. To the left in the figure the sources (grouped) of the requirements can be seen along with a percentage indicating their relative contribution of the total amount of requirements on each abstraction level. E.g. 6% of the requirements came in (were placed) on Product Level, and the two sources (Customer and Management) had an equal (50%-50%) share of these requirements.

As the dynamic validation was a limited test-run of the RAM, and the organization had limited infrastructure to register requirements sources, some grouping was necessary. The grouping is based on general source, e.g. the group "Developers" is populated by all requirements elicited/caught from the development department, regardless of e.g. who the developer was. In the same way, the "Management" group consists of requirements

from all management sources (whether it was upper or middle management). The Partner group was elicited indirectly through the marketing department, as direct access was limited, as was the case in most Customer group cases.

The distribution over the abstraction levels was relatively even except for the Product Level. This was not surprising as the Product Level only holds relatively abstract requirements, and not statements such as "the product should support" which was the most common type (as indicated by 37% at Feature Level). The abstract nature of requirements on the Product Level means that in reality the requirements will result in many more requirements on lower levels when the requirements are broken down. On the one hand, this means that the percentage figures are not comparable. On the other hand, the inability to compare between different levels of abstraction is one of the main motivations for the development of the model.

Most of the requirements came in on Feature Level, not testable or broken down enough for development, but feature focused, as were the Customer and Management groups that dominated this abstraction level. There was a limited introduction of requirements from both the Partner and Developer groups on Feature Level as well.

The Developer group stated most of the requirements on Function Level, and had a fair share on Component Level as well. This is not surprising, as developers generally are

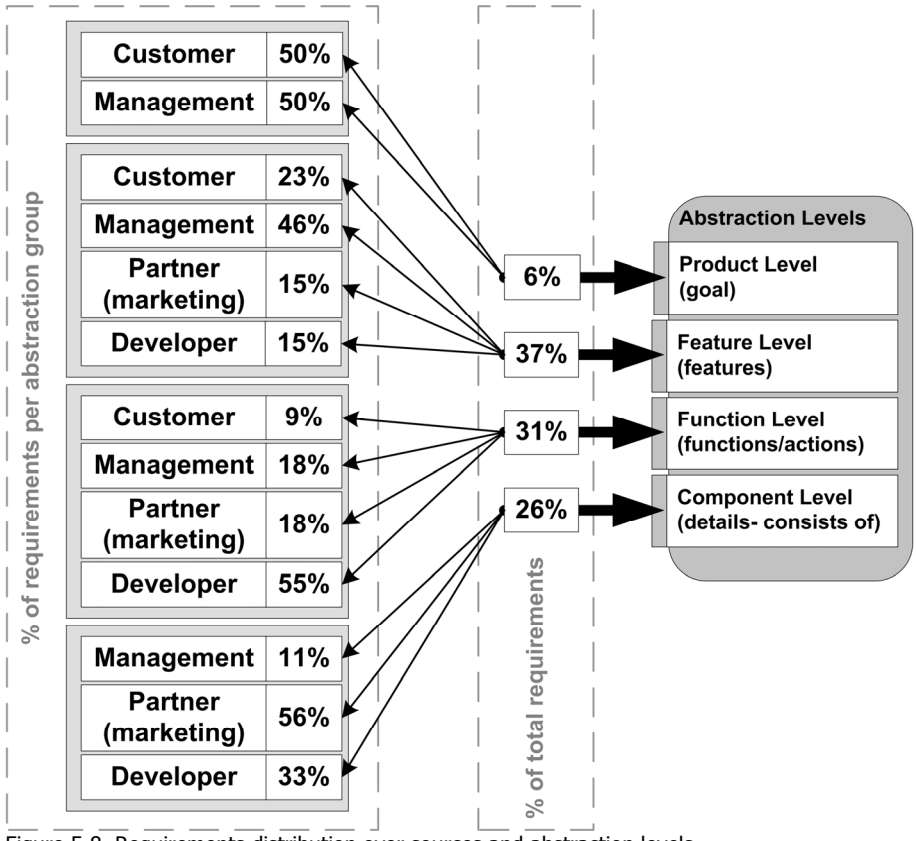


Figure 5-8. Requirements distribution over sources and abstraction levels.

perceived as more hands-on and not prone to abstract non-verifiable statements. The Partner group consisted of requirements were the source could be derived to industry partners which used DHR's products as components in their own products. The partners' high level of technical expertise and insight into DHR product's technical detail probably led to their high representation on Component Level.

It should be noted is that the Management group has relatively high representation on the low abstraction levels. This might be a result of that many management representatives are experienced engineers, well versed in technical aspects, and have been with the organization for a large number of years.

In regards to the distribution, it was seen as beneficial that the RAM supported varying levels of abstraction for initial placement of requirements since incoming requirements varied a great deal regarding abstraction level. The alternative was to put all requirements into one single repository, regardless of abstraction level, as was largely done prior to the RAM.

1.b. The total number of requirements after work-up was about twice as many as the original requirements, i.e. for every original requirement in average one work-up requirement was created. The original requirements that came in on Product Level (6%) were relatively few in comparison, but they resulted in the creation of several work-up requirements, more so than original requirements coming in on lower levels of abstraction. I.e. the product impact of the requirements on high abstraction levels (Product Level) was substantial even if the amount of original requirements coming in on this level was relatively low.

The main issue with creating work-up requirements (new requirements as a result of work-up) was knowing when to stop. Satisfying the work-up rules (R1 and R2) were not the real problem. The main challenge was to avoid inventing new and not completely relevant requirements. Relevance in this case was staying true to the original requirement and only perform the work-up that was necessary to satisfy it. The Product Manager felt that it could be easy to lose focus and create additional requirements that were semi-relevant, e.g. adding functionality that could be "good to have", but was not sought after when looking at the original requirement and the Requirement Source.

The main experiences from the work-up was that staying true to the original requirements was very important, and probably a matter of experience in using the RAM. The doubling of the total amount of requirements as a result of work-up was not considered a problem since the benefits outweighed the costs. The benefits were not only comparison to strategies and producing good-enough requirements for development, but also that the work-up process itself gave a better overview of the requirements, and allowed for a better holistic view. In addition, the increase in effort, despite the creation of work-up requirements, was considered moderate in comparison to the requirements engineering performed before the use of the RAM. In conclusion, the work-up (i.e. abstraction and breakdown) of requirements was considered beneficial in terms of that it implied analysis, refinement, and completion of requirements in a structured way.

2.a. As the requirements were abstracted upwards, it was possible to compare most of the requirements to the product strategies. It should be noted that some of the product strategies were not in place prior to the requirements engineering (as the product was new to the organization this was rather expected). However, the benefit was that explicit decisions had to be made during the requirements engineering as to what requirements

were to be dismissed and what were not. This activity can be seen as a "reverse engineering" of product strategies, but it can also be seen as the creation (or rather explicit specification) of product strategies that are lacking, using requirements as a road-map to what is needed. Either way, as the strategic decisions were made they offered the possibility to dismiss requirements that were out of scope at an early stage, and keeping resources spent on irrelevant requirements at a minimum.

The main concern was for a need for management to establish routines for creating and re-creating (e.g. adding to/reformulating) product strategies continuously, as needed, to support the continuous requirements engineering. A lack of these routines was not a direct problem during the dynamic validation as the scope of the requirements engineering effort was limited, and did not influence critical core products.

2.b. As the requirements engineering come close to its stop condition (adequate requirements for project initiation) a requirements review was performed in order to assess if the requirements on Feature Level and Component Level were good-enough for use in a project and dedicated requirements engineering. In total ten requirements were reviewed on Feature Level (chosen by random). Three of these were considered as testable and unambiguous right away. The other seven were in need of some additional work before testability could be assured.

During the review, all ten requirements were considered good-enough as input to the dedicated requirements engineering. There was adequate material present (looking at the entire abstraction chain) to continue the work of refinement and completion before the design stage.

The input to the project planning effort was also improved as all requirements were broken down to a level were greater accuracy regarding resource cost could be achieved, especially in comparison to having a mix of abstract and detailed technical requirements as a basis for project planning and initiation.

It should be noted that some communication with the Product Manager and minor refinement (reformulation) of requirements would have been necessary if the requirements should have been good-enough in terms of being unambiguous and testable. This was deemed as acceptable and considered a substantial improvement over previous input to projects.

3. The overall impression of using the RAM was positive. The action steps of specification, placement, and work-up were performed without much trouble, and if supported by relevant examples rather intuitive. As the dynamic validation was the first attempt to actually use the RAM there were of course some problems, mainly pertaining to coming up with relevant examples and figuring out strategy related issues. However, the model helped in raising number of issues early instead of leaving more uncertainties to development. Thus, the model is also indirectly a risk mitigation tool.

The material produced by using the RAM was considered more than adequate, as well as offering it on an even level, i.e. as the requirements were specified and worked-up in the same way, most requirements were equally refined and comparable (on the same level of abstraction).

The potential problems with the model were not directly associated with the RAM, but rather the infrastructure needed to support the RAM. This included issues mentioned before, e.g. product strategies, but also issues such as adequate tool support and organizational infrastructure (traceability to original requirement source), and so on.

Looking at the dynamic validation as a first live run of the RAM, the overall usability of the model was considered more than adequate by the Product Manager using the RAM, as were the results produced by the RAM.

5.5.3. Validity Evaluation

In this section, we discuss the threats to the validations performed. We base this on the discussion of validity and threats to research projects presented in Wohlin et al. [32]. The validity threats considered are conclusion, internal, external and construct validity threats respectively.

5.5.3.1 Conclusion validity

Each static validation session was done in one uninterrupted work session. Thus, the answers were not influenced by internal discussions about the questions during e.g. coffee breaks.

The sampling techniques used for the static validation can pose a threat to the validity of the investigation. The subjects selected may not be totally representative for the role they should represent at DHR. The main assurance that this misrepresentation is minimal is the fact that the subjects were selected in cooperation with three senior managers with extensive knowledge and experience concerning the development processes and the personnel at DHR.

5.5.3.2 Internal Validity

As the discussions and static validation of the RAM was performed with the different interview subjects, they were called upon to voice their opinions and views regarding e.g. requirements engineering in general and the RAM in particular. As their answers (the discussion summarized) were registered by the researcher this could have constrained people in their answers. This potential problem was alleviated by the guarantee of anonymity as to all information divulged during the validation, and that recorded answers was only to be used by the researcher, i.e. not showed or used by any other party. In addition, the researcher has worked with DHR personnel over a period of just under two years, and has earned a degree of trust, as well as established a professional relationship that in all likelihood made it easier for the personnel to voice their views.

5.5.3.3 External Validity

The external validity is concerned with the ability to generalize the results, i.e. in this case the applicability of the RAM in industry based on the literature survey and interest from another company than DHR. As the problems identified in this chapter must be considered as general issues in a market driven development situation in general, the use of the RAM as a step towards alleviating the issues speaks for the model's applicability in general. The use of DHR as a validation case for the RAM was beneficial as direct industry input regarding all aspects was considered crucial. The tailoring of the RAM towards DHR is not directly a threat since the model is largely example-driven, and can be tailored and adapted to fit other organizations by the development of e.g. product relevant examples.

5.5.3.4 Construct Validity

The construct validity is mainly concerned with the mapping from the real world to the laboratory. In this case the real world is the ongoing requirements engineering effort at DHR, and the "laboratory" is the dynamic validation performed. It was performed in industry with industry professionals, but since it was limited in time and scope it can be seen as an artificial environment.

The main threat is scalability, i.e. if the RAM will work as well during a continuous requirements engineering. Scalability is to some extent assured by the statements and views of the professionals involved in the dynamic validation.

5.6. Conclusions

The Requirements Abstraction Model was developed in response to direct needs identified in industry, and the lack of an appropriate model to address these needs. The goal was to offer Product Managers a model supporting the ability to handle and work with requirements on multiple levels of abstraction in a continuous product centered requirements engineering effort. This meant going from e.g. one repository where all requirements were kept, regardless of abstraction level, to a structure mirroring the reality of the requirements coming in.

The RAM enforces work-up rules that abstracts and breaks down requirements as needed, offering requirements on several levels of abstraction reflecting the needs of a development organization. Product Level requirements can be compared to product strategies, in an attempt to dismiss out of scope requirements at an early stage. Function and Component Level requirements (needed for project initiation) effectively assure that developers are not burdened with requirements too abstract for development. In addition, working with the model gave a homogenous refinement and analysis of requirements, and the effect that several issues, normally left to projects, were caught early on. This mitigating the risk of some problems creeping into development, and thus caught only after the development effort (e.g. project) was planned and initiated.

As requirements engineering is an integrated part of development the effective use of the RAM is dependent on certain infrastructure being in place, this pertains mainly to explicitly formulated product strategies, which existence cannot be taken for granted. As parts of the benefit is dependent on this fact it could be considered a drawback, however it should also be noted that the abstraction of requirements can trigger explicit questions, challenging management to refine product goals and strategies to reflect the needs of the development organization.

All parties working with development (management in particular) can compare requirements, as they are homogenous regarding abstraction level, and are not forced to decide between an abstract requirement and a detailed one as e.g. planning and prioritization activities are performed.

As a part of its development, the RAM was validated in industry through both static validations, giving feedback from professionals working with requirements engineering and product development, and through dynamic validation, giving a real live test-run of the model. The validations were performed to assure that the model complied with the needs of industry.

During the validations it was ascertained that requirements did come in on different levels of abstraction, and that specification, placement, and work-up were feasible in a real live requirements engineering situation. The usability of the model was premiered in its development, and was partly assured during the static validation, and tested during the dynamic validation.

As the RAM is not prescriptive in nature, but rather adaptable and example-driven, tailoring towards a product (or organization) may be required prior to use in order to develop support materials like guides and relevant examples. The main stakeholder pertaining to the model is the Requirements Manager (e.g. a Product Manager), and as the model should be tailored to support the work performed, the work performed must adhere to work-up rules, but also the consistency regarding abstraction levels is critical and how requirements are handled in relation to these rules. The main point is not having a certain requirement of a certain abstraction on a particular level, but rather having all requirements of a certain abstraction on the *same* level. This (i.e. adequate usage of the model) is obtained through supporting users with guides, relevant examples, and explicitly formulated product strategies, but also through training in model usage prior and during model implementation in an organization. Issues such as the number of abstraction levels needed is up to the organization in question and their particular case, and is a part of the tailoring.

Requirements specified using the RAM were based on attributes validated against industry professionals, and were considered adequate in amount and detail pertaining to fulfilling the functions needed. The usability was premiered in the models development, but not at the expense of substantially lowering the quality of the requirements produced. This was assured through the validity reviews performed.

As stated earlier, the RAM presented in this chapter was designed with Product Managers in mind, supporting them in their requirements engineering effort producing requirements good-enough for planning activities, giving detailed abstraction as well as a big picture view. However, it was also developed with engineers (developers) in mind, controlling the abstraction level and refinement of requirements handed over to projects. The motivation for the RAM was to address needs identified at DHR, these same needs that were later also confirmed by a second (independent) development organization, and their explicit interest in tailoring the RAM to their products.

The non-prescriptive, adaptable nature of the model is based on the assumption that a perfect model and way of working cannot be specified, not even for one organization with homogenous products, since the products and needs of the organization may change over time. The RAM can be tailored to satisfy the needs of different organizations and products. In the same way, it can be modified over time to reflect the current situation of a development organization, supporting the collection of requirements over time and product life cycle.

5.7. Future Work – Evolvement of the RAM

RAM v.1.0 was aimed at assuring requirements be specified on multiple abstraction levels in a repeatable and homogenous way. This was considered a prerequisite for subsequent parts, i.e. requirements prioritization, packaging of requirements concerning coupling between requirements and so on. RAM v.2.0 will support this (giving additional

support to Product Managers), as well as a detailed process for the dedicated requirements engineering effort performed within projects (giving development personnel additional support). These improvements will be a part of complementing the RAM to support multiple stakeholders in the developing organization, thus effectively addressing issues identified in industry (see e.g. Improvement Package Two and Three in Chapter 4).

A study of the RAM's performance over an extended period will be performed to investigate scalability of the model in regards to continuous requirements engineering.

As a part of further validating the RAM's applicability in industry, the model will be tailored and tested in a second development organization experiencing some of the same issues identified in the DHR case.

5.8. References

1. Ruhe G, Greer D (2003) Quantitative Studies in Software Release Planning under Risk and Resource Constraints. In Proceedings of International Symposium on Empirical Software Engineering (ISESE), IEEE, Los Alamitos CA, pp. 262-271.
2. Butscher SA, Laker M (2000) Market-Driven Product Development. Marketing Management 9(2):48-53.
3. Sommerville I (2001) Software Engineering. Addison-Wesley, Essex.
4. Carlshamre P (2002) Release Planning in Market-Driven Software Product Development: Provoking an Understanding. Requirements Engineering 7(3):139-151.
5. Yeh AC (1992) Requirements Engineering Support Technique (REQUEST): A Market Driven Requirements Management Process. In Proceedings of the Second Symposium on Assessment of Quality Software Development Tools, IEEE, Los Alamitos CA, pp. 211-223.
6. Carlshamre P, Sandahl K, Lindvall M, Regnell B, Natt och Dag J (2001) An Industrial Survey of Requirements Interdependencies in Software Product Release Planning. In Proceedings of Fifth IEEE International Symposium on Requirements Engineering, IEEE, Los Alamitos CA, pp. 84-92.
7. Dahlstedt Å, Persson A (2003) Requirements Interdependencies - Molding the State of Research into a Research Agenda. In Proceedings of the Ninth International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'03), Universität Duisburg-Essen, Essen, Germany, pp. 71-80.
8. Regnell B, Host M, Natt och Dag J, Hjelm T (2001) An Industrial Case Study on Distributed Prioritisation in Market-Driven Requirements Engineering for Packaged Software. Requirements Engineering 6(1):51-62.
9. Greer D, Ruhe G (2004) Software Release Planning: An Evolutionary and Iterative Approach. Information and Software Technology 46(4):243-253.
10. Weber M, Weisbrod J (2003) Requirements Engineering in Automotive Development: Experiences and Challenges. IEEE Software 20(1):16-24.
11. Higgins SA, de Laat M, Gieles PMC (2003) Managing Requirements for Medical It Products. IEEE Software 20(1):26-34.
12. Potts C (1997) Requirements Models in Context. In Proceedings of the Third IEEE International Symposium on Requirements Engineering, IEEE, Los Alamitos CA, pp. 102-104.
13. Potts C, Hsi I (1997) Abstraction and Context in Requirements Engineering: Toward a Synthesis. Annals of Software Engineering 3(X):23-61.
14. Anton AI (1996) Goal-Based Requirements Analysis. In Proceedings of the Second International Conference on Requirements Engineering, IEEE, Los Alamitos CA, pp. 136-144.

15. Kavakli E, Loucopoulos P, Filippidou D (1996) Using Scenarios to Systematically Support Goal-Directed Elaboration for Information System Requirements. In Proceedings of IEEE Symposium and Workshop on Engineering of Computer-Based Systems, IEEE, Los Alamitos CA, pp. 308-314.
16. Castro J, Kolp M, Mylopoulos J (2002) Towards Requirements-Driven Information Systems Engineering: The Tropos Project. Information Systems 27(6):365-389.
17. Wiegers KE (1999) Software Requirements. Microsoft Press, Redmond WA.
18. Lauesen S (2000) Software Requirements: Styles and Techniques. Samfundslitteratur, Fredriksberg.
19. Robertson S, Robertson J (1999) Mastering the Requirements Process. Addison-Wesley, Harlow.
20. Sawyer P, Sommerville I, Viller S (1999) Capturing the Benefits of Requirements Engineering. IEEE Software 16(2):78-85.
21. Sommerville I, Sawyer P (1999) Requirements Engineering: A Good Practice Guide. Wiley, Chichester.
22. Pfleeger SL (1998) Software Engineering: Theory and Practice. Prentice-Hall, Upper Saddle River NJ.
23. Martin S, Aurum A, Jeffery R, Barbara P (2002) Requirements Engineering Process Models in Practice. In Proceedings of the Seventh Australian Workshop on Requirements Engineering (AWRE'02), Melbourne, Australia, pp. 141-155.
24. Juristo N, Moreno AM, Silva A (2002) Is the European Industry Moving toward Solving Requirements Engineering Problems? IEEE Software 19(6):70-78.
25. Kivisto K (1999) Roles of Developers as Part of a Software Process Model. In Proceedings of the 32nd Annual Hawaii International Conference on Systems Sciences, IEEE, Los Alamitos CA, pp. 1-19.
26. Humphrey WS (2000) Introduction to the Team Software Process. Addison-Wesley, Reading.
27. Fritzhanns T, Kudorfer F (2002) Product Management Assessment - a New Approach to Optimize the Early Phases. In Proceedings of IEEE Joint International Conference on Requirements Engineering, IEEE, Los Alamitos CA, pp. 124-134.
28. Rakitin SR (2001) Software Verification and Validation for Practitioners and Managers. Artech House, Boston.
29. Lehmann DR, Winer RS (2002) Product Management. McGraw-Hill, Boston.
30. Kotonya G, Sommerville I (1998) Requirements Engineering: Processes and Techniques. John Wiley, New York.
31. Robson C (2002) Real World Research: A Resource for Social Scientists and Practitioner-Researchers. Blackwell Publishers, Oxford.
32. Wohlin C, Runeson P, Höst M, Ohlson MC, Regnell B, Wesslén A (2000) Experimentation in Software Engineering: An Introduction. Kluwer Academic, Boston.

Appendix A

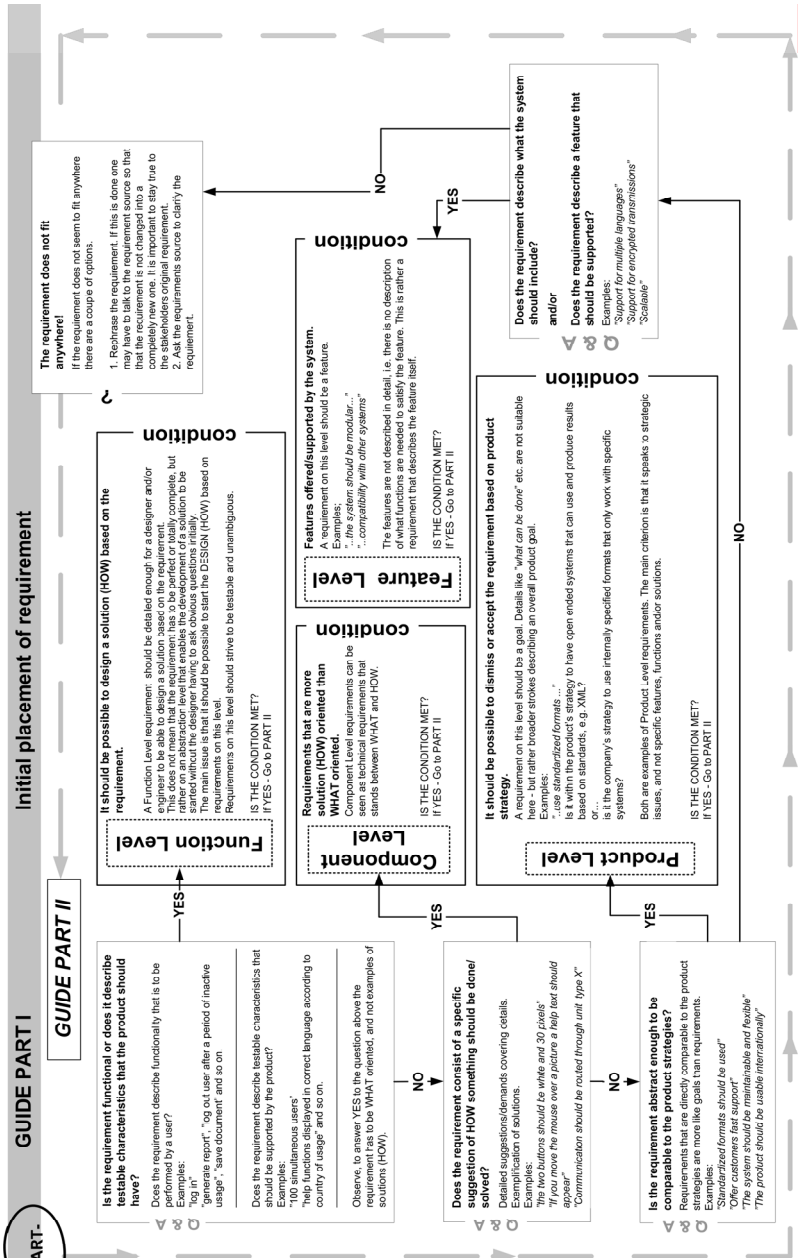


Figure 5-9. RAM how-to guide part I - initial placement of requirements.

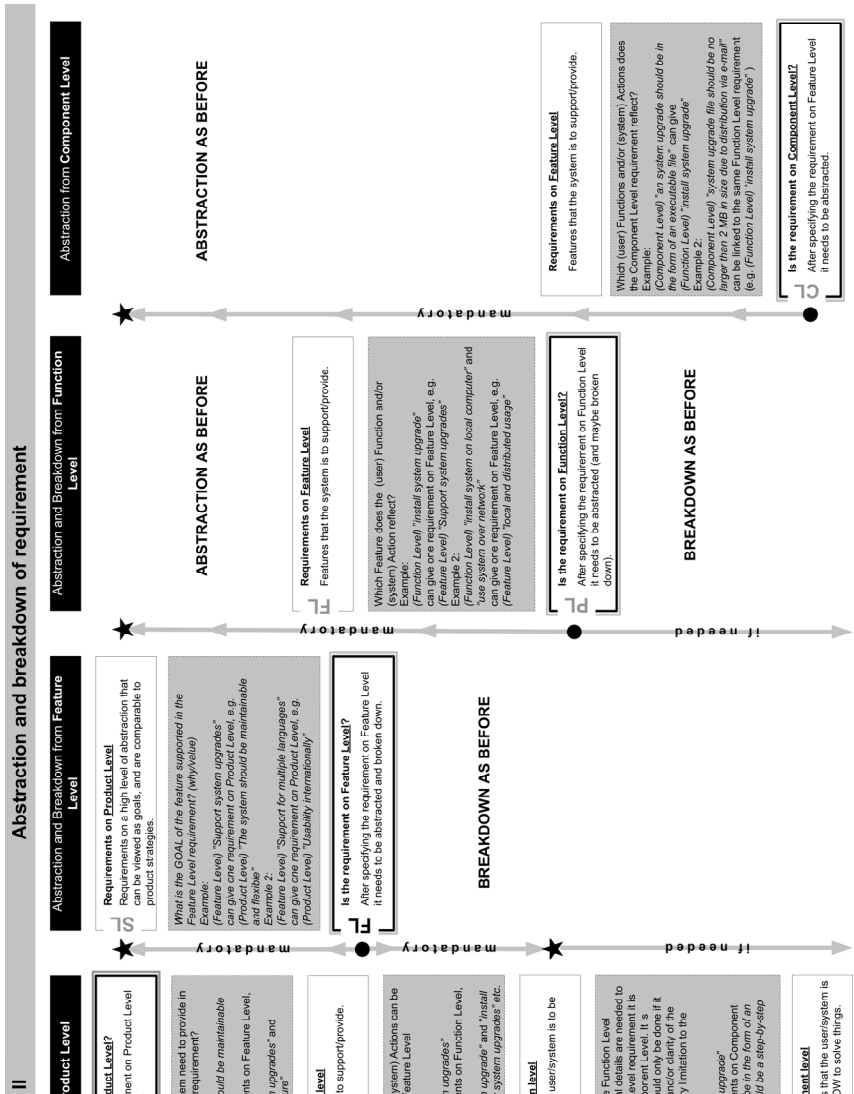


Figure 5-10. RAM how-to guide part II – Abstraction and breakdown of requirement.