

# **Requirements Engineering Supporting Technical Product Management**

Tony Gorschek



Blekinge Institute of Technology Doctoral Dissertation Series

No 2006:01

ISSN 1653-2090

ISBN 91-7295-081-1

# Requirements Engineering Supporting Technical Product Management

**Tony Gorschek**



Department of Systems and Software Engineering

School of Engineering

Blekinge Institute of Technology

SWEDEN

© 2006 Tony Gorschek  
Department of Systems and Software Engineering  
School of Engineering  
Publisher: Blekinge Institute of Technology  
Printed by Kaserstryckeriet, Karlskrona, Sweden 2006  
ISBN 91-7295-081-1

*to my father, the most decent man I know,  
to my mother, always thinking about everyone else,  
to my sister, for all the support and love,  
to my Cecilia, for inspiring me to be a better man*



# ABSTRACT

Market-Driven Requirements Engineering (MDRE) handles the continuous flow of requirements in an engineering effort, and is not limited to a development instance but part of technical product management as a whole. The market-driven environment generates large amounts of requirements from multiple sources, internal and external, threatening to overload the technical management of products. MDRE involves requirements analysis, resource estimation, prioritization, and ultimately release planning. These activities allow for effectively deciding which customers get what features and quality at what point in time, making the accuracy and efficiency of MDRE a major determinant of the success of a product.

This thesis presents research aimed at supporting technical product management in MDRE, based on needs identified in industry. One contribution of the thesis is the development and validation of a process assessment and improvement planning framework, making the identification of process improvement issues possible. The major characteristics of the framework can be described as resource efficiency and relative accuracy by utilizing multiple perspectives and data point triangulation. As a complement to the assessment, the improvement planning enables industry professionals to focus their efforts on one improvement package at a time, supporting step-by-step evolution with less time to return on investment.

Challenges identified during process assessment using the framework resulted in the development of the Requirements Abstraction Model (RAM), which is the central contribution of the thesis. RAM employs multiple levels of abstraction for requirements elicitation, analysis, refinement and management. The model offers the possibility for early requirements triage using product strategies/roadmaps, as well as supporting a structured and repeatable way to perform continuous requirements engineering. This enables product managers to specify requirements that can be traced across abstraction levels, from abstract requirements that can be used by managers, to refined requirements specific enough to be used for estimation and input to development efforts.

RAM was engineered based on industry needs, validated and refined through several empirical evaluations, utilizing both academia and industry as laboratory. This involved close collaboration with industrial partners, Danaher Motion Särö AB and ABB Robotics, where the model was introduced. Monitoring the process improvement (introduction of RAM) indicated substantial improvements, both in accuracy of the work performed using RAM, and in the quality of the requirements produced, with only moderate increase in effort.

Another contribution in the thesis is a compilation of lessons learned from practical hands-on experience of technology and knowledge transfer from academia to industry, with focus of producing industry relevant usable and useful results.

The main contribution of the thesis is improving the practice of product management by providing and evaluating frameworks for software process improvement and market-driven requirements engineering. The results of the research presented in the thesis are aimed at supporting technical product management taking the unique challenges of market-driven development into account.



# ACKNOWLEDGEMENTS

First and foremost I would like to extend my sincere gratitude to my supervisor and collaborator Professor Claes Wohlin. Without his patient guidance the goal would not have been attained as swiftly, nor would the road have been as pleasant.

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 Per Garre, Arne Hedström and Stefan Börlin.

I would also like to thank all participants of the process improvement work conducted at ABB Robotics, in particular Stefan Forssander for his support. In addition, Stig Larsson and Samuel Fricker at ABB Corporate Research have contributed greatly with their commitment and encouragement.

The industry collaboration 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 BESQ participants and the SERL group for being central parts of creating this environment. I would especially like to thank Mikael and Miroslaw for always challenging me to reach higher and Michael for being in my corner.

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

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>).



# TABLE OF CONTENTS

<b>CHAPTER 1 - INTRODUCTION .....</b>	<b>1</b>
1. CONCEPTS AND RELATED WORK .....	6
1.1. <i>Software Process Assessment &amp; Improvement – General Concepts</i> .....	6
1.2. <i>Market-driven Requirements Engineering – General Concepts</i> .....	17
2. CONTRIBUTION AND OUTLINE OF THE THESIS .....	35
2.1. <i>PART I – Process Assessment and Improvement</i> .....	36
2.2. <i>PART II – MDRE Model Presentation and Validation</i> .....	38
2.3. <i>PART III – Knowledge and Technology Transfer</i> .....	46
3. RESEARCH APPROACH .....	47
3.1. <i>Research Questions</i> .....	47
3.2. <i>Research Methods</i> .....	53
4. LIST OF PUBLICATIONS .....	59
4.1. <i>Publications included in the thesis</i> .....	59
4.2. <i>Publications Not Included in the Thesis</i> .....	60
5. CONCLUSIONS .....	61
5.1. <i>Future Research</i> .....	64

## **PART I – PROCESS ASSESSMENT AND IMPROVEMENT**

<b>CHAPTER 2 - IDENTIFICATION OF IMPROVEMENT ISSUES USING A LIGHTWEIGHT TRIANGULATION APPROACH .....</b>	<b>69</b>
1. INDUCTIVE PROCESS ASSESSMENT .....	70
1.1. <i>Introduction</i> .....	70
1.2. <i>Investigation Context</i> .....	71
1.3. <i>Investigation Design</i> .....	72
1.4. <i>Results</i> .....	79
1.5. <i>Relation to State-of-the-art</i> .....	85
1.6. <i>Conclusions</i> .....	88
<b>CHAPTER 3 - PACKAGING SOFTWARE PROCESS IMPROVEMENT ISSUES – A METHOD AND A CASE STUDY .....</b>	<b>91</b>
1. PROCESS IMPROVEMENT (PRE-) PLANNING .....	92
1.1. <i>Introduction</i> .....	92
1.2. <i>SPI – Related Work and Motivation</i> .....	93
1.3. <i>DAIIPS – An Overview</i> .....	96
1.4. <i>Industry and Academia Study</i> .....	106
1.5. <i>Discussion and Conclusions</i> .....	128
1.6. <i>Further Work</i> .....	129

## PART II – MDRE MODEL PRESENTATION AND VALIDATION

<b>CHAPTER 4 - REQUIREMENTS ABSTRACTION MODEL .....</b>	<b>133</b>
1. THE REQUIREMENTS ABSTRACTION MODEL .....	134
2. RELATED WORK .....	137
3. RESEARCH CONTEXT – BACKGROUND AND MOTIVATION .....	139
3.1. <i>Process Assessment Results</i> .....	140
3.2. <i>Motivation Summary</i> .....	142
3.3. <i>Evolution of the Requirements Abstraction Model</i> .....	143
4. RAM STRUCTURE & SUPPORTING PROCESS – AN OVERVIEW .....	145
4.1. <i>Action Step One – Specify (elicit)</i> .....	146
4.2. <i>Action Step Two - Place</i> .....	148
4.3. <i>Action Step Three – Abstraction (Work-up)</i> .....	150
4.4. <i>Additional Structure and Process – Roles, Attributes and Rules</i> .....	155
4.5. <i>Model Tailoring</i> .....	159
4.6. <i>Requirement Abstraction Model – Summary of the Action Steps</i> .....	161
5. RAM - VALIDATION .....	162
5.1. <i>Static Validation</i> .....	162
5.2. <i>Dynamic Validation</i> .....	166
5.3. <i>Validity Evaluation</i> .....	172
6. CONCLUSIONS .....	174
7. FUTURE WORK – RAM VALIDATION AND EVOLUTION .....	176
8. ACKNOWLEDGEMENTS .....	177
9. APPENDIX – CHAPTER 4 .....	178
<b>CHAPTER 5 - A CONTROLLED EMPIRICAL EVALUATION OF A REQUIREMENTS ABSTRACTION MODEL .....</b>	<b>181</b>
1. INTRODUCTION .....	182
2. BACKGROUND AND RELATED WORK .....	183
2.1. <i>The Requirements Abstraction Model</i> .....	183
2.2. <i>Problem Statement</i> .....	186
2.3. <i>Related Work</i> .....	187
3. PLANNING OF THE EVALUATION .....	189
3.1. <i>Context</i> .....	189
3.2. <i>Subjects</i> .....	189
3.3. <i>Design</i> .....	190
3.4. <i>Instrumentation</i> .....	190
3.5. <i>Validity Evaluation</i> .....	193
4. RESEARCH QUESTIONS .....	195
4.1. <i>Part I</i> .....	195

4.2.	<i>Part II</i> .....	196
4.3.	<i>Part III</i> .....	197
5.	OPERATION.....	198
5.1.	<i>Preparation</i> .....	198
5.2.	<i>Execution</i> .....	199
6.	RESULTS AND ANALYSIS .....	200
6.1.	<i>Part I</i> .....	200
6.2.	<i>Part II</i> .....	204
7.	RESEARCH QUESTIONS REVISITED .....	211
7.1.	<i>Part I</i> .....	211
7.2.	<i>Part II</i> .....	211
7.3.	<i>Part III</i> .....	212
8.	CONCLUSIONS.....	213
8.1.	<i>Future Work</i> .....	214
<b>CHAPTER 6 - A REPLICATED CONTROLLED EMPIRICAL EVALUATION OF A REQUIREMENTS ABSTRACTION MODEL.....</b>		<b>217</b>
1.	INTRODUCTION .....	218
2.	BACKGROUND AND RELATED WORK .....	219
2.1.	<i>The Requirements Abstraction Model</i> .....	219
2.2.	<i>Problem Statement</i> .....	222
2.3.	<i>Related Work</i> .....	224
3.	PLANNING OF THE EVALUATION.....	225
3.1.	<i>Context</i> .....	225
3.2.	<i>Subjects</i> .....	225
3.3.	<i>Design</i> .....	226
3.4.	<i>Instrumentation</i> .....	226
3.5.	<i>Validity Evaluation</i> .....	229
4.	RESEARCH QUESTIONS.....	231
4.1.	<i>Part I</i> .....	231
4.2.	<i>Part II</i> .....	232
4.3.	<i>Part III</i> .....	234
5.	OPERATION.....	235
5.1.	<i>Preparation</i> .....	235
5.2.	<i>Execution</i> .....	235
6.	RESULTS AND ANALYSIS .....	236
6.1.	<i>Part I</i> .....	236
6.2.	<i>Part II</i> .....	239
7.	RESEARCH QUESTIONS REVISITED .....	246
7.1.	<i>Part I</i> .....	246
7.2.	<i>Part II</i> .....	247

7.3. <i>Part III</i> .....	248
8.    CONCLUSIONS.....	249
8.1. <i>Future Work</i> .....	250
<b>CHAPTER 7 - TEST-CASE DRIVEN INSPECTION OF PRE-PROJECT REQUIREMENTS - PROCESS PROPOSAL AND INDUSTRY EXPERIENCE REPORT.....</b>	<b>251</b>
1.    INTRODUCTION .....	252
2.    INSPECTION - GENERAL OVERVIEW.....	254
2.1. <i>Inspection Process</i> .....	254
2.2. <i>Reading Techniques</i> .....	256
3.    BENEFITS AND ISSUES – INSPECTION EXPERIENCES FROM INDUSTRY AND ACADEMIA.....	257
3.1. <i>Benefits of Inspections</i> .....	257
3.2. <i>Issues with Inspections</i> .....	258
3.3. <i>Motivation to Find Alternatives</i> .....	259
4.    TEST-CASE DRIVEN INSPECTION .....	259
4.1. <i>TCD Inspection</i> .....	261
5.    DISCUSSION AND STUDY RESULTS.....	264
5.1. <i>TCD vs. Traditional Approaches</i> .....	264
5.2. <i>Study results</i> .....	267
6.    CONCLUSIONS.....	268
<b>CHAPTER 8 - INDUSTRY EVALUATION OF THE REQUIREMENTS ABSTRACTION MODEL.....</b>	<b>269</b>
1.    INTRODUCTION .....	270
2.    BACKGROUND AND RELATED WORK .....	271
2.1. <i>Introduction to the Requirements Abstraction Model</i> .....	272
3.    THE COMPANIES .....	275
3.1. <i>DanaherMotion Särö AB (DHR)</i> .....	275
3.2. <i>ABB (ABB)</i> .....	276
4.    MODEL TAILORING .....	276
4.1. <i>Implementation</i> .....	278
5.    EVALUATION DESIGN .....	280
5.1. <i>PART I</i> .....	281
5.2. <i>PART II</i> .....	284
5.3. <i>Analysis PART I and PART II</i> .....	284
6.    EVALUATION RESULTS .....	286
6.1. <i>DHR</i> .....	286
6.2. <i>ABB</i> .....	296
6.3. <i>Comparison of RAM Evaluation Results at DHR and ABB</i> .....	306

7. CONCLUSIONS .....	307
8. FUTURE WORK .....	308

**PART III – KNOWLEDGE AND TECHNOLOGY TRANSFER**

<b>CHAPTER 9 - TECHNOLOGY AND KNOWLEDGE TRANSFER IN PRACTICE – REQUIREMENTS ENGINEERING PROCESS IMPROVEMENT IMPLEMENTATION.....</b>	<b>313</b>
---	------------

1. INTRODUCTION .....	314
1.1. <i>Step 1 - Basing research agenda on industry needs</i> .....	315
1.2. <i>Step 2 – Problem formulation</i> .....	316
1.3. <i>Step 3 – Formulate a candidate solution</i> .....	318
2. EVOLUTION AND TRANSFER PREPARATION THROUGH VALIDATION.....	319
2.1. <i>Step 4 – Lab validation</i> .....	319
2.2. <i>Step 5 – Static validation</i> .....	320
2.3. <i>Step 6 – Dynamic validation (piloting)</i> .....	321
2.4. <i>Step 7 – Release solution</i> .....	322
3. CONCLUSIONS IN PERFECT HINDSIGHT.....	326

<b>REFERENCES .....</b>	<b>328</b>
-------------------------	------------



# Chapter 1

---

## Introduction

The development of software intensive products is changing focus, moving from a traditional bespoke (producer-customer) to a market-driven (producer-marketplace) perspective [1-3]. The overall implication being that the product development cost as well as producer revenues are not linked to one specific customer but a market, comprised of any number of potential buyers [4].

Requirements engineering (RE) is also very much affected by the change in perspective. Traditionally RE is described as “what services a product should provide and under what constraints it should operate” [3], and in the bespoke perspective RE consists of the systematic process of eliciting, understanding, analyzing, documenting and managing requirements throughout a product’s life cycle [5]. The focus here is on the development instance (project) itself, i.e. the RE effort is project initiated and part of e.g. a pre-study or focused to the initial stages of development.

In contrast, market-driven requirements engineering (MDRE) has a continuous flow of requirements and the requirements engineering effort is not limited to a development instance but a part of product management as a whole [6, 7]. In this environment requirements come from several sources both internal (e.g. developers, marketing, sales, support personnel, bug reports etc) and external (e.g. users, customers and competitors, often gathered via surveys, interviews, focus groups, competitor analysis etc) [8-11]. This can give rise to very large amounts of requirements, and all of them need to be caught, specified, analyzed and managed continuously as the product evolves over time through releases. In MDRE requirements selection (release planning) activities are central, i.e. the decision about which customers get what features and quality at what point in time. This makes the accuracy of release planning a major determinant of the success of a product [12]. In MDRE the focus is on the product and requirements, and development efforts (projects) are initiated by requirements.

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 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 [13-18]. Many of the challenges are relevant for both bespoke RE and MDRE. For example, problems (inadequate quality) in requirements filter down to design and implementation [3]. 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 [19],

and several other sources indicate that inadequate requirements are the leading source for project failure [13-18]. This is further compounded in the case of MDRE as the initial selection of requirements is crucial, in particular since large volumes of requirements from multiple sources risk overloading companies. It is vital that incoming requirements can be handled in a structured way, dismissing irrelevant ones at an early stage, thus expending as little effort as possible in order to save resources for refining requirements that will actually be selected and allocated to development instances [20]. In addition to the volume, the requirements themselves are of varying quality, state of refinement, and level of abstraction. In traditional bespoke development a requirements engineer can actively elicit requirements and thus hope to control or at least substantially influence these aspects. In a market-driven situation this is seldom the case. Most requirements are already stated in one way or another when they reach the requirements engineer (e.g. a product manager). The knowledge and experience of the developing organization, of which the requirements engineer in this case is instrumental, is central to making sense of the requirements as they are processed [21].

There exists many RE best-practice guides and software process improvement frameworks (see e.g. [18, 22-26]) that are targeted at finding challenges/issues that can be used as a basis for improving RE practices. However, most of them are adapted to suit a bespoke environment with traditional, project focused, customer-developer relationships. Another problem is that many software process improvement (SPI) frameworks are often too large and bulky to get an overview of and to implement [27-29], and if implemented return on investment can take anything from 18 to 24 months [30]. This makes it difficult for organizations, in particular small and medium sized enterprises (SMEs), to initiate and perform assessment and improvement activities as cost and time are crucial considerations [27-29].

There is a need for the development and evaluation of best-practice guides, techniques and models that support professionals working in market-driven environments – taking the unique challenges/issues of MDRE into account. A prerequisite for identifying these challenges in industry is the development of appropriate process assessment and improvement frameworks.

The work presented in this thesis can be seen as three parts, all very closely linked. (**Part I**) First, the challenges facing market-driven product development organizations need to be identified, analyzed and prioritized; in other words what are the challenges/issues facing industry, and of these which are the most critical? For this purpose, current processes

and practices need to be assessed using a process assessment technique suitable for the task. Suitability speaks to several aspects of which assessment complexity, effort and accuracy are central, as well as enabling assessments and improvements for SMEs or other organizations with limited resources.

Following the assessment the issues need to be analyzed and prioritized according to criticality and dependencies, enabling the organization being assessed to decide what the most critical aspects are. Performing process improvements in smaller increments shortens improvement lead time.

**(Part II)** The second part of the thesis uses the results from the first, and is focused on the incremental development, validation and refinement of MDRE practices that address the issues identified in Part I. This is performed in close cooperation with industry in several iterative steps performed both using academia and industry as laboratory. The concrete result was the conception of the Requirements Abstraction Model (RAM), designed towards a product development perspective. It supports a continuous requirement engineering effort, and can handle large quantities of requirements of varying degrees of detail and abstraction. RAM aids practitioners in handling requirements in a structured and repeatable way, during requirements elicitation, analysis, refinement and management. The nature of the model is to *use* the fact that requirements come on different levels of abstraction instead of trying to flatten all or mixing different types in a document. Using RAM makes abstraction (comparing to strategies), and breakdown (e.g. until testable) of all requirements a part of the analysis and refinement work. As all requirements are abstracted and broken down it is also possible to compare them, making prioritization and estimation realistic undertakings.

The model is intended to incorporate possible solutions for many of the challenges/issues identified during the assessments in industry, and primarily offers product planning and product management support for organizations working in an MDRE environment.

**(Part III)** The third part can be seen as an overall knowledge and technology transfer process. This part includes process assessment and the development, validation, and ultimately the transfer of results to industry practice. In addition it summarizes and describes the research approach used in all of the work presented in this thesis. Part III described close industry collaboration, and using industry as laboratory to assure that state-of-practice is taken into consideration, and thus avoiding problems with industrial applicability and scalability of research results. Industrial applicability and scalability have been identified as a major problem with tech-

nology and knowledge transfer of research results to industry practice [31, 32].

The prerequisite of Part III and thus all work presented in the thesis is close industry collaboration. This was made possible with the aid and collaboration of three research partners: Danaher Motion Särö AB, ABB Robotics, and ABB Corporate Research.

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

**Section 1** is aimed at introducing the underlying concepts and background of the research presented in this thesis.

**Section 1.1** introduces the area of Software Process Assessment and Improvement. This information serves as a background primarily for the critical SPI success factors presented in Sections 1.1.3 and 1.1.4. These SPI success factors are challenges facing SPI and indirectly Knowledge and Technology Transfer. They need to be taken into consideration when performing process assessment in general, for organizations with limited resources, but also for the assessment of RE processes in particular. The research under Part I of this thesis addresses these success factors making it possible to identify RE and MDRE issues, and thus identifying the basis for the research in Part II.

**Section 1.2** introduces the area of Market-driven Requirements Engineering. This serves as a background to several challenges facing MDRE practitioners (called MDRE issues hereafter). The issues are identified and presented in Section 1.2.3. The research under Part II of this thesis addresses some of these issues.

**Section 2** gives the outline and contribution of the research presented in this thesis by introducing eight chapters. The contribution of each chapter is clarified in the context of the SPI success factors (for Part I) and the MDRE issues (for Part II). This is done to clarify the contribution as well as place the research in a relevant context.

**Section 3** gives an overview of the research approach used, and the overall technology and knowledge transfer process (Part III) is presented and linked to the SPI success factors. This section also presents the scope, research questions and methods/tools used to answer the research questions.

**Section 4** lists the relevant publications for the thesis. **Section 5** has the conclusions, and presents some future work planned.

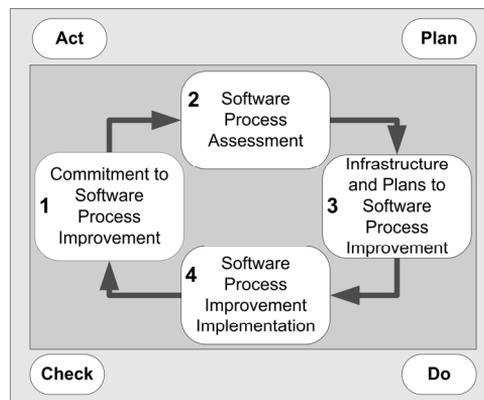
# 1. CONCEPTS AND RELATED WORK

This section gives some background to software process assessment and improvement, and requirements engineering focusing on market-driven requirements engineering. Each section is summarized in a number of success factors for SPI (see Section 1.1.3 and 1.1.4) and MDRE challenges/issues (see Section 1.2.3).

## 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, inspired by [27]).

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 embraces the necessity for a cyclic and continuous process improvement as early as 1939 [33].



**Figure 1.** *Generic process improvement scheme.*

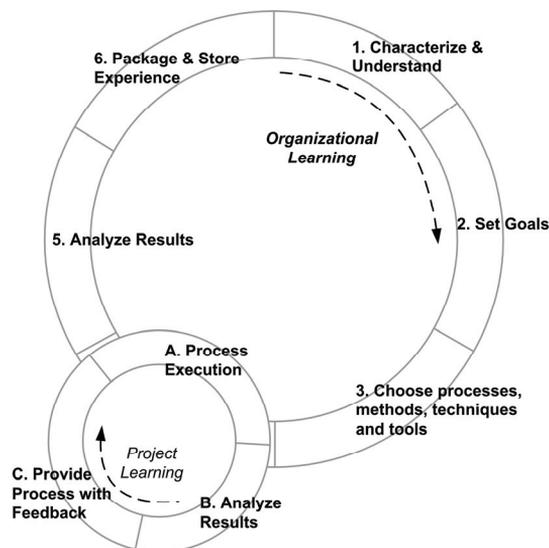
However, there is a clear and distinctive difference between frameworks and how they approach process improvement. A rudimentary division of SPI frameworks can be done based on whether or not they are bottom-up (inductive) or top-down model-based (prescriptive) in nature. Below in-

ductive and prescriptive SPI frameworks are characterized through exemplification of well-known examples of each.

### 1.1.1. INDUCTIVE FRAMEWORKS

Basili's QIP (Quality Improvement Paradigm) [34] 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) [35]. 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 an official change is made to the process [36].

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 2, inspired by [37], quantifiable goals are set, based on this improvements are chosen (which can be in the form of e.g. new processes, methods, techniques and/or tools).



**Figure 2.** *Quality Improvement Paradigm (QIP).*

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 [36].

## 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 organization's 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 [30, 38]. 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) [39-41] 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, 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) [30].

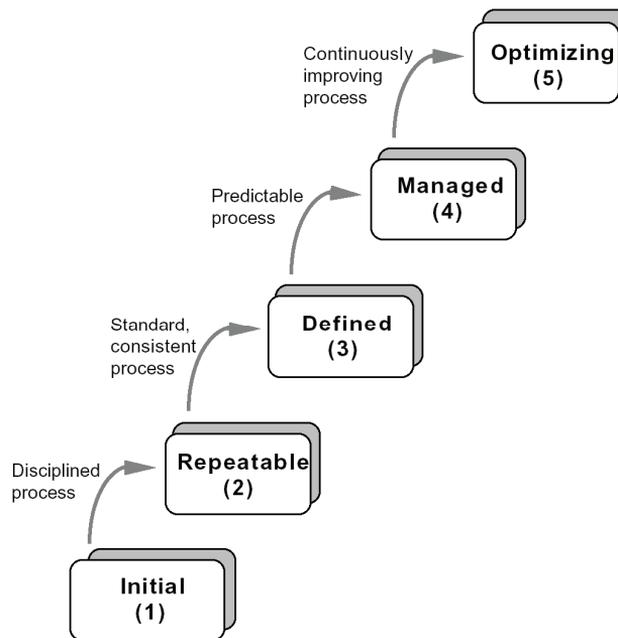
Subsequent to the assessment there is planning and preparation for introducing relevant practices (or changing present ones) to conform to CMM. This concerns both what practices are used, and in what order they are implemented.

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 [42].

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 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 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) [43] and IPD-CMM (Integrated Product Development CMM) [44]. 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) [41], ISO/IEC 15504 (a.k.a. SPICE – Software Process Improvement & Capability dEtermination) [45], and BOOTSTRAP [30]. The development of the new standards was also attempts to address some of the issues identified with CMM, as can be seen below.

### 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 [46].

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 [46].

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) [46]. In general CMMI supports three classes of appraisals [47, 48]:

- **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 C 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 [46, 49]. 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<sup>1</sup> 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 [49, 50].

### 1.1.3. GENERAL SPI - SUCCESS 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 (success factors) have been identified during the practical application of these frameworks as determinants of success [27-30, 51-62]. Some of these factors

---

<sup>1</sup> 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).

(relevant to the research presented in this thesis) are summarized under a number of headings below<sup>2</sup>, and to some extent exemplified with the use of the frameworks described above.

### 1.1.3.1 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 [51]. 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 [30].

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 [27-29]. This is in particular the case for small and medium sized enterprises (SMEs) (e.g. less than 250 employees) [63] where time and resources always are an issue both regarding assessment and improvement [27-29].

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 [64] where a QIP inspired framework is presented. Adaptations and versions of prescriptive frameworks like CMM has also been presented, see e.g. IDEAL [26] and Dynamic CMM [65].

### 1.1.3.2 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 [55-60]. Management commitment is of course to some extent connected to cost and resource issues presented above, as manage-

---

<sup>2</sup> It should be observed that the naming of the success factors was done to summarize industry experiences with SPI, thus the naming of the factors are not directly mapped to any one source.

ment 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 [55-59, 61, 62, 66]. 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 [67].

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 [68].

### 1.1.3.3 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) [55-59, 69]. As improvements should be performed in a continuous manner, taking small evolutionary steps, prioritization of what to improve and in which order is implied [39, 60, 62]. 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 [60, 69-71]. 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 pre-

defined practices primarily, although measurement methods can be used as a part of this.

#### 1.1.3.4 Summary of General SPI - Success 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 staff/coworkers (e.g. technical managers and developers).
- **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.

The formulation of F1 through F6 is 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. The factors are used to discuss the thesis contribution in Section 2.

#### 1.1.4. REQUIREMENTS ENGINEERING SPI - SUCCESS 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.

##### 1.1.4.1 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 [25]. This is of course not surprising as many prescriptive model-based frameworks like the ones described earlier in Section 1.1.1 and 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 [41, 46, 72]. However, compared to RE specific best practice models like e.g. the ones presented by Sommerville & Sawyer in their practice guide [18, 73], the area is still marginalized [25].

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 [18, 74]. But there are also efforts to define RE process improvement models based on a CMM like structure. Gorschek *et al.* developed the Requirements Engineering Process Maturity model (REPM), which drew inspiration from results from the REAIMS project and CMM, to develop a fast and low-cost project assessment model [75]. Later Beecham *et al.* performed a similar effort in their definition of R-CMM [76], although the focus of R-CMM was CMM compliance (but from a RE perspective) and not fast and low effort assessments like in the case of the REPM model.

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.

#### **1.1.4.2 Requirements Engineering and Project Focus**

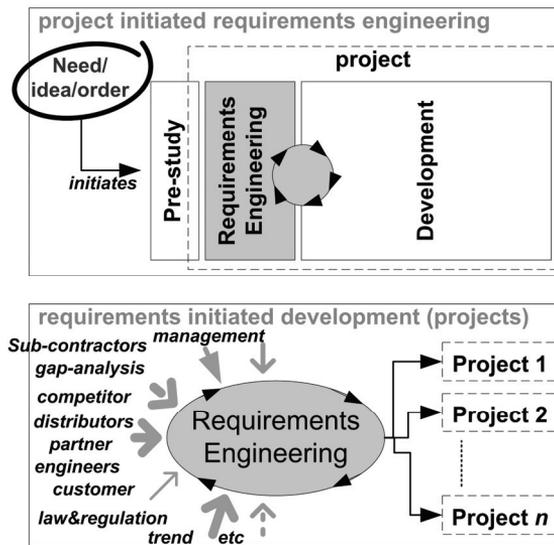
An underlying issue complicating SPI efforts targeted towards RE is the fact that RE more and more transcends projects in an organization as market-driven product development is becoming increasingly commonplace in software industry [1-3].

This basically implies that the situation is far more complex than the one presented by the classical bespoke [3] 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 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) [77].

Requirements engineering, and especially MDRE is not limited to projects but also a part of e.g. product management activities [6, 7].



**Figure 4.** *Requirements' role in the development process.*

Market-driven requirements engineering (MDRE) is covered in more detail in Section 1.2.2, but from a perspective of challenges/issues unique for MDRE.

### 1.1.4.3 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 [61, 66]. This also impacts the willingness to commit and participate in an SPI activity [18].

This is connected to the general SPI factor described in Section 1.1.3.2 (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 see them as beneficial.

#### 1.1.4.4 Quantification and Measurement

As stated in Section 1.1.3.3 the ability to measure SPI success is considered important. Looking at requirements engineering quantification of process improvement results is difficult [18, 27, 61]. This in particular applies in the case of SMEs as having good long-term measurement programs in place is costly [60]. 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 1.1.3.1.

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).

#### 1.1.4.5 Summary of RE SPI – Success Factors

A summary of the requirements engineering specific factors (REF) gives the following list:

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

The factors REF1 through REF4 are used to discuss the thesis contribution in Section 2.

## 1.2. MARKET-DRIVEN REQUIREMENTS ENGINEERING – GENERAL CONCEPTS

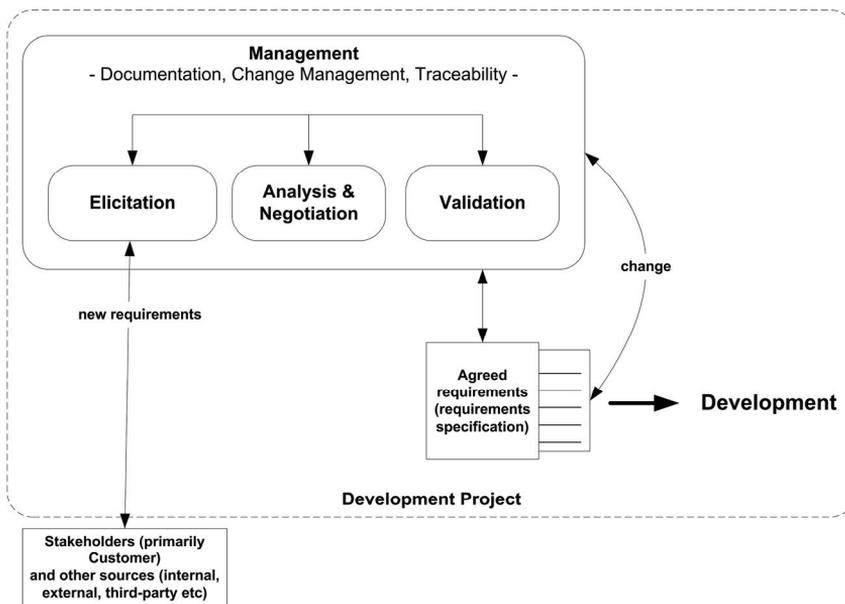
Market-driven requirements engineering (MDRE) is in many ways very similar to traditional bespoke requirements engineering (RE). Many of the practices are the same, as are many of the skills necessary. However, there are several crucial differences which need to be recognized in order to fully comprehend the challenges facing professionals working in a market-driven environment.

Many organizations are faced with a mix of both market-driven and bespoke product development. An example could be a company offering products to a marketplace, but at the same time performing adaptations for customers wanting to pay for tailoring of the products. This situation makes it necessary to adopt both bespoke RE for e.g. adaptation projects, while having an MDRE process handling the continuous requirements engineering governing product development for the market. The mixed situation is not explicitly addressed here as it is out of scope of the work presented in this thesis.

This section offers a brief introduction to the area of traditional bespoke RE, and then moves on to introduce MDRE highlighting some of the major differences between bespoke RE and MDRE. Following this the unique challenges/issues facing MDRE are presented. Some of these MDRE issues act as a basis for what is addressed in the research reported in Part II of this thesis.

### 1.2.1. **REQUIREMENTS ENGINEERING – AN INTRODUCTION**

Requirements engineering can be described as a discipline that covers all of the activities involved in discovering (sometimes called catching or eliciting), documenting (specifying) and maintaining a set of requirements for a product. These main process areas are commonly described in literature as *Elicitation, Analysis and Negotiation, Validation* and *Management* [5, 13, 16, 18, 75, 78]. A simplified overview can be seen in Figure 5. The purpose of the figure is not give the impression that each process area is separated from the other (as they generally are intertwined), but rather present an overview. In this description Management is seen as an overall process of documenting the requirements being formed through elicitation, but also managing the inevitable changes to requirements as they are analyzed and validated. Each of the activities is described in more detail below based on descriptions and techniques found in literature [5, 13, 16, 18, 25, 75, 78].



**Figure 5.** Overview of RE process activities.

**Requirements Elicitation** is about finding and revealing the requirements from different sources. The main sources of information are usually the stakeholders of a product. The stakeholder group is comprised of all people that have a direct or indirect influence on the system requirements, e.g. customers, end-users, people involved in processes that are influenced by the product, engineers producing and managing the product and third party bodies such as regulators. In addition there are other important sources which can yield requirements, e.g. documentation, laws and regulations, standards, operating manuals of products to be replaced, products that interact with the one being developed, domain specific documentation such as company (customer) strategy documents and mission/policy statements etc. Within the elicitation area there are many elicitation techniques, some examples are listed below:

- *Conventional techniques* include the use of interviews, surveys and questionnaires to gather data from stakeholders. In addition analysis of existing documentation e.g. process models, organizational charts, standards and manuals for existing systems can be conducted. Scenarios and use cases can be a very effective way to concretize abstract descriptions into real-life examples helping in the data gathering from stakeholders.
- *Observation (Contextual techniques)* can be a good way of eliciting *how* things are done in contrast to asking stakeholders to describe *what* is

done and how. It can be difficult for stakeholders to articulate fairly simple routines and tasks. Furthermore the context and environment in which the stakeholder works can be of importance.

- *Reuse* is basically the process of using existing knowledge when developing a new system. Requirements already specified and validated can be reused if they fit (or maybe parts of them can be)
- *Group Elicitation techniques* such as brainstorming, Joint Application Design (JAD) workshops, group interviews and brainstorming sessions can also be valuable for elicitation.

**Requirements Analysis and Negotiation.** The work of analyzing the requirements is done with the intention to find possible conflicts, overlaps, dependencies, omissions or inconsistencies as the requirements are elicited and specified. In addition, feasibility and risks can be assessed for the requirements (individually or in groups) in order to build an understanding of potential problems, and estimations are made in order to approximate what resources are needed and the cost is associated with the implementation of a requirement(s).

The requirements are examined at an early stage and unrealistic requirements can be sorted out. The work of sorting out the requirements can also be mentioned as the negotiation part – basically different stakeholders have different views of what is important. Different stakeholders also have different power over the decisions being made [5, 79]. Another activity here is usually to prioritize the requirements and to ultimately decide which requirements should be included in the requirements specification, which effectively acts as a contract between the developing organization and the customer. The main goal of Analysis and Negotiation is to answer the question ‘have we got the right requirements’ [5].

- *Analysis techniques/tools* such as Interaction Matrices and Requirements classification/grouping and sorting can be used to determine dependencies and interactions between requirements.

The development of e.g. Prototypes, Mock-ups, early draft of User Manuals, and Test Cases can be used to analyze requirements, and make sure that requirements are understood correctly and in the same manner by all stakeholders (e.g. customers as well as the engineers). In addition these techniques can be very powerful elicitation tools.

- *Prioritization techniques* such as the Pair-wise comparison technique - AHP, The Planning Game, and Token Assignment can all be used to prioritize requirements, acting as decision support for what requirements should be satisfied by the delivered product. An overview of prioritization techniques can be found in [80, 81].

**Requirements Validation** is generally performed a bit later in the process as requirements stabilize, and is not primarily concerned with dealing with the “raw” requirements coming in during elicitation, but rather validating the final draft of the requirements document together with customer. This is not to say that the requirements completely stable, but rather not as volatile as in the initial stages during elicitation. The main goal of Validation is to answer the question ‘have we got the requirements right’ [5].

- Reviews and Inspections can be used to find defects in requirements and improve quality, assure testability, conformance to standards and so on. More information on reviews and inspections can be found in [82-87].
- Checklists can be used to assure that certain criteria are met regarding information elicited and specified etc.

**Requirements Management** is an ongoing process lasting the entire development life-cycle. It is concerned with the documentation and continuous management of the elicited requirements. The documentation of requirements can take several forms, the most commonly used is specification in Natural Language (NL), but Use-cases, requirements modeling [88], and formal specifications [89] are other possibilities. Combinatory solutions are also possible where some requirements are complemented with e.g. formal parts or a use-case is added to a requirement.

During a development effort new requirements emerge and old ones change, these changes need to be analyzed, controlled and decisions made whether or not a change should be accepted or not. Some changes may have substantial impact on a development project, while others are minor and more of an evolution of requirements as domain and requirements understanding improves during development. Formal change management may involve the use of change control boards that review, analyze, and accept or dismiss change requests, informal change management may be more ad-hoc, in any case, decisions should be made explicit and communicated to all stakeholders, and requirements updated as needed.

A part of the management process is making decisions regarding how requirements should be documented. This implies deciding formatting and information to be stored, and decisions regarding what traceability policies that are needed. One type of traceability is to be able to trace requirements backwards to their source in terms of people, documents etc. Another is forward to e.g. test-cases and design. Version handling is also handled here (for further information see [18, 75]).

## 1.2.2. AN OVERVIEW OF MARKET-DRIVEN REQUIREMENTS ENGINEERING

The main distinguishing feature that separates MDRE from bespoke RE is the fact that there is no customer, but rather a market(s) consisting of any number of customers. This fact influences all other aspects of MDRE including elicitation practices, analysis, and management. Figure 6 gives an overview of a “generic” MDRE process based on several sources in literature [7, 81, 90-94]. Each part of the process is described below.

### 1.2.2.1 Requirement Sources (Elicitation)

The *initiation* of an MDRE effort is not governed by an external request or customer order, but it is continuous in nature, where development efforts (projects) are initiated as needed per a release plan (following a product roadmap). Saying that a *market(s)* is the customer is true, if somewhat simplified. There are of course certain entities within the market(s) which can be identified. Key-customers are a part of the market but due to their importance they usually have the possibility to put forward requirements directly to the developing organization in addition to going through indirect channels like for example marketing/sales/support.

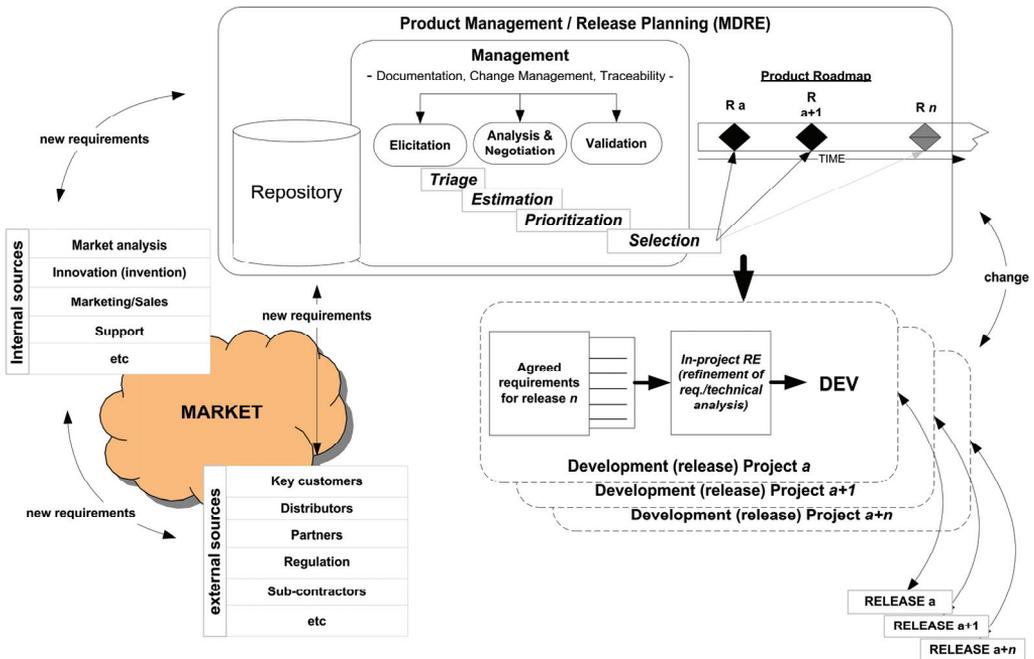


Figure 6. Overview of a generic MDRE process.

Figure 6 shows additional potential external sources of requirements, e.g. distributors and partners that many times have a partnership relation to the developing organization. This is especially the case for developing organizations offering their products to a market indirectly through a vendor or an overall larger product offering. An example could be development organization offering software and hardware components for integration on motherboards. In this case the market consists of all motherboard manufacturers, repair shops, and other companies using the development organization's products as a part of their own product offering. In addition end-users can be the direct customers for e.g. software upgrades (firmware and drivers).

The use of external sources (e.g. key-customers) for direct elicitation activities, as opposed to getting their input through e.g. surveys and other marketing efforts, has been proposed and tested by companies in the form of Customer Working Groups (CWG) [4]. Although this may generate new requirements and give good insight into individual customer priorities there is a risk that multiple preferences (between different customers, and between customers and the development organization) conflict [4].

Many requirements come from internal sources as requirements in MDRE are often *innovations* (invented by the development organization) attempting to add new and unique features and/or characteristics to a product. Development organizations often possess extensive *domain knowledge* making innovation not only possible but often profitable, allowing for differentiation of a product in comparison to competitors by offering unique or better features and/or quality [10].

### 1.2.2.2 Requirements Analysis

All requirements are *documented* in a central repository independent of source and subsequently analyzed. This first involves performing early dismissal/acceptance of the incoming requirements (often called triage [95]) in order to avoid requirements overload, a major issue in MDRE [7, 20, 96].

Following early triage *analysis* is performed where implementation costs and resources are *estimated* for requirements going in to comparison and prioritization. Having accurate estimates is directly related to the ability to perform release planning activities and selecting the requirements of high priority to be allocated to a certain release project [94, 97, 98]. This is mainly due to the fact that time-to-market often is central. Fixed delivery/release dates are generally rigidly enforced, postponing requirements to a later release instead of going over time [91, 97]. Regarding estimation techniques, attempts to estimate costs and schedules using requirements

employ function points or feature points [99]. However, the use of non-structured informal techniques for requirements estimations seems to be pre-dominant, where professionals rely on expert judgment [100].

An important factor to be investigated during analysis is requirements interdependencies which can influence requirements selection. For example, a biometric sensor (e.g. fingerprint identification) on a laptop may decrease the value of having a card reader. Another case could be that a requirement stating “128-bit RSA encryption” would make network performance requirements more expensive to realize. In the market-driven context value based dependencies, directly related to customer value and development cost seem to be predominant. Carlshamre *et al.* [101] report that only about 20% of all requirements are relatively singular, making dependencies a highly relevant factor in requirement analysis.

### 1.2.2.3 Requirements Prioritization

Requirements *prioritization* has to be conducted taking several aspects into account. The central notion is that all requirements should be compared and prioritized independent of source. The *objective* of the prioritization is to get input for requirements selection for subsequent release planning. The overall goal is to deliver the right product to the market at the right time and thus selling and generating revenues. Success is defined by out-performing competitors in the market and delivering a high perceived benefit to customers. From this perspective customer satisfaction is central and optimally customers (and potential customers) should perform prioritizations. Regnell *et al.* [98] report on attempts with distributed prioritization in a market-driven context involving distributed marketing departments in the prioritization of requirements. However, scalability of having large amounts of requirements prioritized by several stakeholders can be an issue. Potential scalability issues also apply to the prioritization technique chosen [81].

Internal considerations regarding technical aspects (e.g. architecture and maintainability), business aspects (e.g. strategic decisions regarding focusing on new market-segments), and implementation aspects (e.g. dependencies) are also input to prioritization and selection. Several methods for attaining requirement priority exist, including AHP [102], the 100-point method [103], attainment [104], and the planning-game [105].

Common for any prioritization activity is that the requirements being prioritized need to be comparable with regards to abstraction level otherwise there is a risk that requirements of higher abstraction level gets higher priority than requirements on a low level of abstraction For exam-

ple, a light in the glove compartment of a car is probably prioritized lower than the car having a boot [81, 106].

#### 1.2.2.4 Requirements Selection (Release Planning)

##### **Product Strategy and Roadmaps**

After initial triage requirements are analyzed (estimated, prioritized, and crucial dependencies mapped) the actual selection and release allocation takes place. A product roadmap (see Figure 6) can be used as an approach to document and communicate plans for future releases [107]. There are many types of roadmaps described in literature, in fact sometimes any forward looking document is called a roadmap [108]. The one used for the purposes of MDRE release planning is described as Product-Technology Roadmaps [108], and has the purpose of “mapping” and aligning efforts and resources towards common goals. A roadmap should convey several aspects, for example:

- themes of a certain product release (e.g. a theme could be offering a certain type of functionality, concentrating on improving quality, security and so on)
- restrictions (e.g. what are the restrictions in terms of risk, time, resources available, internal technical considerations and so on)
- goals (what are the overall product goals, and what are the goals for every release)
- milestones (for releases and goals)

A roadmap can be seen as an explicit concretization of product strategies depicting the long-term plans of a product. Product strategies are in turn a concretization of company and business strategies pertaining to an individual product [109].

Product strategies should reflect not only current market knowledge and customer priorities (attained through e.g. market analysis, from internal experts etc.) but also the long term goals set for a certain product. The requirements selection should be done within the boundaries set by product strategies. For example, if a certain release has the theme of “security”, requirements pertaining to security should be prioritized over other requirements for that release. Ignoring product strategies (and only looking at current priorities) may mean that a product is successful short-term, but at the expense of the long-term goals [11, 110]. For example, security requirements may be deemed less important at present, but the long-term plans for the product is to eventually break into new market segments where security is deemed crucial. One of the fundamental aims of a product strategy is to explicitly plot the goals and the limits of a product – fo-

cluding all efforts and aligning them in one deliberate direction [11, 111]. On the other hand, constraining development too much to a business strategy can cause a developing organization to miss new, out-of-the-box, business opportunities that can leverage business technology strengths [112]. Thus another aspect which must be included in the formulation of product strategies (and thus specified in roadmaps) is the balance between technology-push and market-pull when selecting requirements for a release. There is always a risk that one perspective dominates, predominantly market-pull (business perspective) as reported by Wohlin and Aurum [113].

It should be noticed that requirements selection within the boundaries of product strategy following a roadmap does not guarantee success, as the strategies followed can be flawed. However, having up-to-date product strategies, which reflect all vital knowledge – from both technology and marketing perspectives - will most certainly increase the chance of developing a successful product [11, 110, 114].

For more information about roadmaps see e.g. Kostoff and Schaller [107] and Kappel [108], for information about how to construct roadmaps for SMEs see e.g. [115].

### **Performing Release Planning**

The actual “mechanics” of allocating requirements to a certain release has been described by for example Carlshamre and Regnell [116] in the use of the REPEAT process (Requirements Engineering Process At Telelogic). REPEAT is centered on having fixed release dates and intervals, allowing for requirements to be allocated to Select-lists with a “must” part and a “wish” part. Together the must and wish requirements are estimated to take 1.3 times the available resources (must part taking 0.7, and the wish part 0.6). Allowing for 30% requirement estimation error while still realizing the must part. For a description of the entire REPEAT process see Regnell *et al.* [91].

Greer and Ruhe [117] describe the EVOLVE approach based on a genetic algorithm where customers prioritize and reprioritize requirements for incremental development and delivery. Although the exemplification of EVOLVE assumes continuous customer involvement and is only exemplified using a limited amount of candidate requirements (20) it can be used for decision support generating a limited amount of candidate solutions.

Both processes/methods used as examples here are dependent on relative accurate effort estimations, priorities, guiding strategies (roadmaps), and clear release dates.

### **Requirements Selection Quality**

Being able to measure the success of the requirement selection process is crucial as it allows for continuous improvement. Regnell *et al.* present a model that allows for principal reasoning about both the quality and the current capacity of the requirement selection process [118]. Karlsson *et al.* present the PARSEQ method which focuses on post-release analysis of requirements selection quality, examining the requirements actually selected for realization [119]. This can be described as a post-mortem of sorts for requirements selection quality.

The perceived customer value pertaining to a product can also be used to gauge selection quality (as an indirect indication). GAP analysis can be used to measure positive and negative “gaps” between what the product offers and what the customer perceives. Features and characteristics of the product are identified and their fulfillment of customer needs is mapped. A positive gap represents when a product delivers more than is expected, a negative gap the opposite. One of the earliest descriptions of the need to measure this “gap” was described in [120-122]. Customer Value Analysis (CVA) is similar to GAP analysis but also includes the perspective of using competitor products in the analysis and the evaluated product is graded with regards to the value of a need in comparison to alternatives [123]. Both GAP and CVA can be used to measure selection quality post-release, but the results can also be used actively as input to the next round of requirements selection for the product in question. The relationship between requirements change, development time, and customer satisfaction is presented in [124].

### **1.2.2.5 Requirements Validation**

Requirements validation is traditionally performed in close cooperation with the customer (bespoke RE), but in MDRE this is complicated for obvious reasons. There is the possibility to cooperate with key-customers but their input may be hard to obtain, and in addition they may not be representative of all requirements in a release [97, 125]. This is further complicated by the fact that some requirements are invented to suit an imaginary customer [90]. Validation can be performed internally using the knowledge of the development organization; this is particularly relevant in the case of invented requirements. Traceability to requirements source is important in this case [94]

An additional method for requirements’ validation is using beta-releases of upcoming releases, and having real customers (or potential ones) test the product.

### 1.2.2.6 Requirements Management (Specification)

The requirements repository (e.g. a RE tool or database with a front-end) can act as a central storage for all requirements. For reasons of scalability, traceability, distributed work (e.g. prioritization), and overall management document based (e.g. word-editors) is not appropriate.

The use of a tool also enables the use of attributes. Attributes can be a good way of structuring requirements in a repository. Examples of attributes can be seen in Table 1. Some can be mandatory (have to be stated) others optional (stated if specifier deems it necessary), and some can be auto-generated by the tool used.

Attribute	Description
ID	Unique identifier e.g. auto-number
Title	Title for the requirement
Description	Free description of the requirement
Rationale	A description of the rationale/benefit of the requirement from the req. source's perspective
State	What state the req. is in at present, e.g. new, dismissed, specified, planned for release, released etc.
Version	Version of the requirement (maybe with the possibility to view different versions and differences btw. versions)
Source	The source of the requirement
Estimation	Cost/time for implementation
Dependency	Dependency and type of dependency
Priority	The priority of the req. e.g. on a scale of 1-5 where 5 is more
Test	Links to e.g. test-cases
<i>etc</i>	<i>etc</i>

**Table 1.** *Example of Requirement Attributes (Inspired by [91] and Chapter 4).*

Clear benefits of using attributes are:

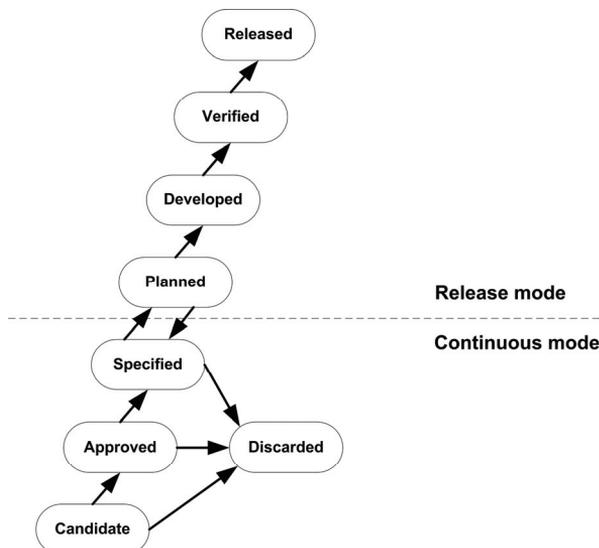
- separation of information regarding a requirement enabling e.g. filtering and different views, sorting according to a specific attribute etc.
- allowing for requirements to be specified in a similar manner by several people as attributes steer what is to be specified
- assuring that a minimum amount of information (mandatory attributes) is specified independent of who is specifying the requirement

- connecting attributes to the overall process and requirement states enables control and traceability (for example to be eligible for prioritization a requirement has to have the “Estimation” attribute specified)

It should be noted that different development organizations may need different attributes (see Chapter 4 and 8). More information and examples on the use of attributes can be found in Regnell *et al.* [91] and Chapter 4.

Figure 7 gives an example of a requirement state model called the “salmon ladder” showing the up-ward road a requirement has to take in order to be implemented (described in Regnell and Brinkkemper [4]).

The use of state models in MDRE enables not only control but also the ability to sort requirements enabling different views. For example, a manager performing allocation of requirements to a release can choose to view only the “Specified” requirements (avoiding the ones not yet specified/analyzed). Requirements in the states “Discarded” and “Released” may be filtered out entirely when viewing requirements for the purpose of planning, and so on. This can also be seen as one way of battling requirements management overload. For examples of other state models see e.g. Regnell *et al.* [91], Carlshamre and Regnell [116], and Chapter 4.



**Figure 7.** Example of Requirement State Model [4].

Requirements change in MDRE just as in the case of bespoke RE. In case of bespoke RE negotiations can be performed with the customer allowing for e.g. deadlines to be renegotiated. In MDRE changes to requirements prior to release allocation can be due to the emergence of new high-priority requirements, market changes can lower the priority of an already prioritized requirement and so on. Changes can be handled through general change management processes (see [126] for examples). However, the issue is complicated if changes occur after a requirement has been allocated to a release. Changes may give the need for re-prioritization, re-selection and re-allocation of all requirements to a certain release in the worst case, due to e.g. dependencies.

The work described can be performed by e.g. a steering committee [97] or through the use of a traditional Change Control Board (CCB) [126]. The main parties involved in change management in the case of MDRE are internal to the development organization as external stakeholders (e.g. key-customers) seldom possess the big-picture needed for requirements selection. An important factor to remember is that the time-to-market (release dates) are generally set, thus the selection of requirements for a release has to adapt to this fact [97, 127]. In MDRE time aspects are even crucial enough to be prioritized over e.g. quality aspects [128].

Due to the potentially large amount of requirements (which is continuously growing) traceability issues are crucial in this context [94]. For example, traceability to source enables sources to be elicited for input regarding e.g. re-prioritization, or if the source is external they can be informed about delays if a requirement is reallocated to a later release. In addition, traceability to the actual specifier of a requirement can also be crucial, especially in a large organization where any number of people can specify the requirements (see Chapter 4).

### 1.2.2.7 Distinguishing features of MDRE - Summary

To get an overview Table 2 shows the main distinguishing features of MDRE in comparison to bespoke RE.

	<b>Bespoke RE</b>	<b>MDRE</b>
<b>initiation</b>	RE process is initiated and terminated in accordance to a development project	RE process is continuous and serves the evolution of a product(s), projects are initiated as needed
<b>objective</b>	Contractual fulfillment, adherence to requirements specification	Right product, right time, take market shares
<b>success criteria</b>	User acceptance, customer satisfaction	Sales, market share, product reviews, customer satisfaction (many customers)
<b>life cycle</b>	Development → maintenance	Evolution through continuous releases
<b>elicitation</b>	One organization is customer T: Interviews, observation etc.	Market is customer T: Innovation (invention of req.), market analysis, focus groups, competitor analysis etc.
<b>domain knowledge</b>	The developing organization and the customer can cooperate to ensure that the domain is understood	The developing organization has to be experts in the domain, or at least have internal experts
<b>analysis &amp; negotiation</b>	Analysis and Negotiation with customer regarding what to implement, conflict resolution etc.	Early triage, estimation, prioritization, selection, release planning for fixed time
<b>validation</b>	Continuously with customer/stakeholders	Late product tests with key-customers, beta releases, focus groups, surveys etc. Internal validations against e.g. Marketing and other req. sources
<b>specification (management)</b>	Finite amount of requirements, specification technique depending on need and amount of requirements T: NL, formal, modeling etc.	Large amount of requirements continuously growing T: NL (scalability (cost) of e.g. modeling and formal specifications is an issue)
<b>change (management)</b>	Requirements change handled in communication with customer, new requirements alternatives allow for e.g. more resources, and/or extended time delivery time i.e. renegotiation	Requirements change handled by the developing organization in communication with internal parties (development, management, marketing etc.). Generally rigid demands on time-to-market (fixed release dates), new requirements can be allocated to next release

**Table 2.** *Overview of Bespoke RE vs. MDRE.*

### 1.2.3. MDRE – CHALLENGES

There are several challenges/issues that can be identified in relation to MDRE. Some evident from the description of the generic MDRE process described previously in Section 1.2.2, others based on industry experiences described in literature [7, 81, 90, 91, 96, 97, 101, 116, 127, 128]. Some of

these issues can also be applicable for bespoke RE, although for the purpose of the work presented here a market-driven perspective is adopted.

Each of the issues are discussed below, and summarized in Table 3, uniquely identified (i1, i2, and so on). Note that several of these issues are addressed through the research presented in Part II of this thesis (see Section 2.2).

- **i1: Requirements Overload**

Large amounts of requirements are not only a potential threat (management overload) but also an opportunity as they can be seen as input to the MDRE process containing information about the needs and wants of different stakeholders. The implication being that curtailing the inflow of requirements is not an option, but rather the MDRE process needs to be able to handle large amounts of data continuously. The heavier the load an MDRE process can handle the greater the chance of catching more relevant requirements (more input gives more information). However, more input does not equal better product offering if the requirements overload the organization.

- **i2: Abstraction Level & Contents of Requirements**

Large amounts of requirements from multiple and diverse sources generate requirements that can be described as everything from abstract “one-liners” and goal-like statements from e.g. marketing channels, to detailed technical solution proposals from technically adapt customers. In traditional requirements engineering these raw “requirements” would not even be considered as proper requirements until they were “filtered” through a specification and refinement process, involving elicitation activities performed by a requirements engineer getting all information needed to formulate the information as “proper” requirements. In market-driven development requirements come in the raw form, and any MDRE process needs to take this into consideration as it influences all aspects of later processing, whether it be early triage, analysis and refinement, estimation, prioritization and ultimately selection. An MDRE process needs to be flexible enough to handle multiple types of requirements.

- **i3: Requirements Dependencies**

Requirements dependencies (a.k.a. interdependencies) influence primarily requirements selection and release planning. An MDRE process needs to enable professionals to take dependencies into account when handling large amounts of requirements. The predominant type of dependencies identified as crucial for the market-driven develop-

ment involves value-based dependencies, directly related to customer value and development cost.

- **i4: Selection / Release Planning**
  - **i4a: Fixed Releases (time-to-market)**
  - **i4b: Estimation**
  - **i4c: Prioritization**

In order for estimation and prioritization activities to be possible (pre-requisites for Selection/Release Planning taking time-to-market into account) requirements stated need to be good-enough for estimation and prioritization. Good-enough implies that the consequence of a specific requirement needs to be known. For example, a too abstract requirement may need additional analysis and refinement (break-up and/or more information) prior to estimation efforts. Accurate estimates are especially important in the market-driven context, i.e. crucial for planning activities.

The same can be said for prioritization purposes, although comparability is paramount here. Requirements that are to be compared to each other and prioritized need to reside on a similar abstraction level in addition to giving a good-enough view of what they imply.

- **i5: Gap between Marketing/Management and Technical Management/Development**

Requirements can be seen as the-least-common-denominator on which decisions regarding *what* to include in the product offering is decided. This makes communication between marketing/management and technical management/development crucial. All parties should be able to participate in the decision process from early triage to final requirement selection. In addition all parties are substantial requirement sources in themselves. For these reasons an MDRE process needs to offer relevant decision support material to multiple roles in order to encourage and support cooperation and joint work.

- **i6: Market Pull vs. Technology Push**

Requirements can be categorized into two main types: requirements stemming from aspirations of creating (technical) innovations (technology-push), and requirements based on requests/whishes/needs identified in the market environment (market-pull). Premiering either over the other can have severe consequences. For example, prioritizing innovation may leave current customers unsatisfied, and prioritizing market needs may result in missing the development of innovations that could generate great value in the future.

The two aspects need to be balanced by the way of enabling communication of requirements and their implications between market and technical propagators (see i5). In addition the use of product strategies (roadmaps) in requirement selection activities can help the balancing of these two aspects.

- **i7: Elaborate vs. Elementary Process**

- **i7a: Simple Tools for Basic Needs**

An MDRE process/model/technique has to be suitable to the organization using it, i.e. the process needs to adapt to the needs and the resources of the organization using it. A too elaborate process in for example an SME will probably not be used, a too elementary process in a larger organization can be insufficient. In any case there is a need for the development of simple tools/techniques/models/processes that address the unique challenges facing development organizations operating in a market-driven environment.

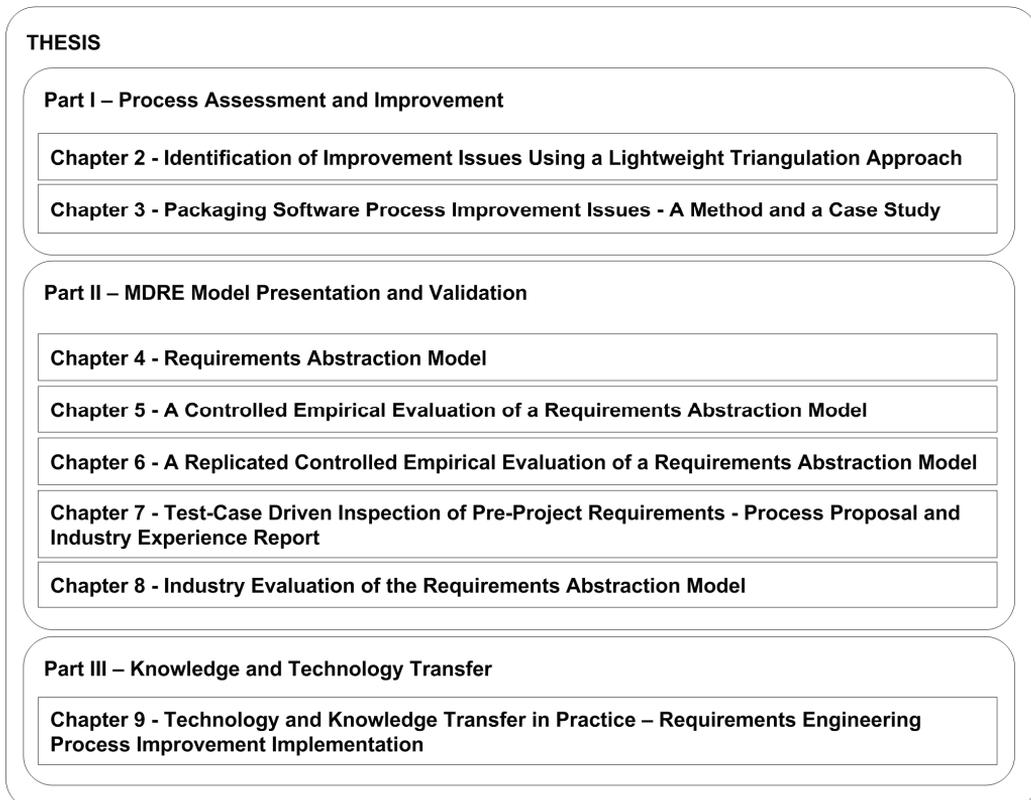
ID	Issue
i1	Requirements Overload
i2	Abstraction Level & Contents of Requirements
i3	Requirements Dependencies
i4	Selection / Release Planning
	i4a Fixed Releases (time-to-market)
	i4b Estimation
	i4c Prioritization
i5	Gap between Marketing/Management and Technical Management/Development
i6	Market Pull vs. Technology Push
i7	Elaborate vs. Elementary Process
	i7a Simple Tools for Basic Needs

**Table 3.** Overview of MDRE Issues.

## 2. CONTRIBUTION AND OUTLINE OF THE THESIS

The research in this thesis is presented in three parts (Parts I, II and III), each containing a number of chapters (see Figure 8). In this section each part with corresponding chapters is presented briefly and related to the previous sections, in an attempt to clarify the contribution of each chapter. (Information pertaining to authors and publication can be viewed in Section 4.1).

It is important to realize that the research presented through the chapters below is not intended to replace or even compete with any existing framework, process or model. The motivation is to complement existing research in the areas of process assessment and improvement (Part I), MDRE (Part II), and Technology Transfer (Part III).



**Figure 8.** *Thesis outline overview.*

## 2.1. PART I – PROCESS ASSESSMENT AND IMPROVEMENT

### Chapter 2 - Identification of Improvement Issues Using a Lightweight Triangulation Approach

---

Chapter 2 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).

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), covering not only bespoke RE but also issues pertaining to overall product development.

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 process is to be assessed and 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 chapter. 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, since the assessment framework was created with primarily SMEs in mind.

Chapter 2 - main contributions in comparison to the success factors presented in Section 1.1.3 and Section 1.1.4:

- 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 3 - 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 Chapter 2) and also look at the potential dependencies between them. Chapter 3 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 chapter 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 has to be committed, keeping improvement it-

erations shorter and delimited. In addition, as a process improvement based on a package is completed evaluations are performed. This makes it possible to assess the effects of an improvement early, and if needed change the focus of the next improvement package to reflect changes in the environment.

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 is used in 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.

Chapter 3 - main contributions in comparison to the success factors presented in Section 1.1.3 and Section 1.1.4:

- 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 many 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).*

## **2.2. PART II – MDRE MODEL PRESENTATION AND VALIDATION**

### **Chapter 4 – Requirements Abstraction Model**

---

Following assessment (Chapter 2) and packaging of improvement proposals into SPI packages (Chapter 3) an improvement effort was initiated aimed to address the first SPI package. Chapter 4 reports the result of this process improvement activity through the presentation and initial validation of RAM (Requirements Abstraction Model).

RAM is an MDRE model with supporting processes designed to handle requirements coming in from multiple stakeholders on multiple levels of abstraction. It was designed to support primarily product managers, and takes the form of a multilevel requirements model offering different views of the requirements base to different roles involved in the MDRE process. A requirement coming in is initially specified (using attributes), and placed on an appropriate abstraction level. Subsequent to this the requirement undergoes *work-up*, i.e. abstraction to a level comparable to product strategies (roadmaps), and break-down until it is refined enough to be a base for estimation efforts and input to development. The final product of work-up allows for product managers to offer different views of a requirement traceable across abstraction levels. These different views make it possible for different roles to choose different views of the requirements base, offering an abstract view to for example marketing, and a detailed view for technical management.

The abstraction part of the work-up can allow for early acceptance/dismissal of a candidate requirement (triage) by comparison of a requirement to product strategies.

One of the main concepts with RAM is that the requirements on a particular abstraction level be fairly homogenous in nature with regards to abstraction level and contents. This enables prioritization efforts. In addition, as all requirements are broken down the implications of a particular requirement becomes evident, supporting estimation efforts. From these two perspectives RAM prepares the requirements for selection/release planning. As of yet RAM does not incorporate a release planning functionality, although any release planning method can be attached to RAM as the model delivers all needed input. For example, the “must/wish list” approach used by REPEAT can be used in combination with RAM (See Section 1.2.2.4).

From a process improvement perspective RAM can be tailored to satisfy the needs of different organizations and products. For example, the number of abstraction levels, attributes, state model, role descriptions, and the choice of tool support can be adapted to suit the organization adopting the model.

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 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.

Chapter 4 - main contributions in comparison to the success factors presented in Section 1.1.3 and Section 1.1.4:

- 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 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.*

Chapter 4 - main contributions in comparison to the MDRE issues presented in Section 1.2.3 are:

- Support of multiple abstraction levels, and offering a structured view of requirements from abstract to detailed level. Addressing primarily:
  - *i2: Abstraction Level & Contents of Requirements.*
  - *I5: Gap between Marketing/Management and Technical Management/Development.*
  - *i6: Market Pull vs. Technology Push.* (Supporting discussions between different roles based on the same requirements but on different abstraction levels).
- Abstraction and comparison to product strategies (roadmap). Addressing primarily:
  - *i1: Requirements Overload.* (Through enabling early dismissal/acceptance of a requirement).
  - *i6: Market-Pull vs. Technology-Push.*
- Break-down of requirements. Addressing primarily:
  - *i4a: Estimation.*
  - *i4b: Prioritization.*
- The tailorability of RAM. Addressing primarily:
  - *i7: Elaborate vs. Elementary Process.*

The following chapters relate to RAM in the following manner. Chapters 5 and 6 describe additional validations of RAM in a laboratory setting. Chapter 7 describes a requirements review process designed to fit with RAM. Chapter 8 describes an assessment of RAM post industry trials.

Chapter 9 describes the overall technology transfer process employed during the development, validation and transfer of RAM to industry practice through a process improvement effort.

### **Chapter 5 – A Controlled Empirical Evaluation of a Requirements Abstraction Model**

---

Chapter 5 describes the controlled evaluation of RAM in a laboratory setting. The study was conducted in an academic setting, with the help of graduate students (normally their fourth year) and some undergraduate students at Blekinge Institute of Technology. A total of 21 students participated.

The main purpose of the evaluation was to assess the usefulness and usability of RAM in a controlled environment prior to widespread industry piloting. Initial evaluation of the model intended to investigate whether or not industry piloting was mandated, or if RAM needed further refinement or even redesign before directly involving industry in validation activities. It was important to get early indications whether or not the fundamental features inherent in the model were useful and usable, and if the assumptions implied by the model were sound.

This was achieved by evaluating the model in two parts:

**(Part I)** Top level (Product Level) requirements should be used to ascertain if a new incoming requirement is in line with product strategies or not (decision support for acceptance or early dismissal of a new requirement). Product Level requirements should therefore convey an overview (“big-picture”) of the product (i.e. Product Level requirements should summarize and represent a considerable amount of requirements on lower levels).

In Part I we evaluate if abstract Product Level requirements are general enough to summarize several requirements on lower levels, but still concrete enough so that it is possible to determine if they represent the product or not. The implication being that Product Level requirements can be used to convey the aforementioned big-picture product overview.

This big-picture overview is also directly related to the possibility of e.g. managers and marketing to participate in MDRE of a product utilizing relatively few top-level requirements to get a product overview to base discussions on.

**(Part II)** This part evaluates if it is possible in practice, with relatively few abstract top-level requirements, to understand what features and attributes a product should and should not consist of. As new requirements come in it should be possible to:

- Dismiss or accept a particular requirement using product strategies (explicitly expressed as Product Level requirements in the evaluation), and
- place the requirement on an appropriate level of abstraction using already present requirements on each level as indication of where the new requirement should be placed.

The usability of the model depends on these two points and thus the utilization of requirements abstraction. Requirements abstraction is the ability to use relatively few abstract Product Level requirements to derive decisions across the hierarchy of abstraction with regards to including or excluding new requirements from a product. In addition, the ability to place new requirements on an appropriate level (using already present requirements as decision support) keeps the levels homogenous, i.e. requirements on each level are comparable with other requirements on the same abstraction level. The ability to place requirements in the model also preserves model consistency over time.

The results indicate that the concept of abstraction and the usage of abstraction levels seems to be fairly intuitive. Overall, the usability and usefulness of RAM (issues evaluated in this chapter) seems to be very promising, supporting further validation activities of RAM.

Chapter 5 - main contributions in comparison to the success factors presented in Section 1.1.3 and Section 1.1.4 are:

- The assessment of RAM prior to industry trials, and the ability to show these results establishes trust and decision support prior to commitment to improvement activities (transferring RAM to industry). Addressing primarily:
  - *F3: Commitment to SPI by management.*
  - *REF3: Commitment and Perceived benefit.*

Chapter 5 - main contributions in comparison to the MDRE issues presented in Section 1.2.3 are:

- By assessing RAM and the underlying concepts behind the model the validity of the claims made in relation to the model in Chapter 4 are validated.

## **Chapter 6 – A Replicated Controlled Empirical Evaluation of a Requirements Abstraction Model**

---

Chapter 6 describes a replication of the controlled evaluation of RAM in a laboratory setting that was presented in Chapter 5. This study was also conducted in an academic setting, but extended from 21 subjects to utilizing 179 subjects from Blekinge Institute of Technology, Linköping University and Umeå University. The result was a substantial increase in the number of subjects, and also utilizing subjects from different universities in Sweden.

The results from this controlled evaluation to a large degree confirm the results observed in the evaluation presented in Chapter 5, only minor differences can be observed. Thus, the conclusions drawn regarding usability and the usefulness of RAM based on the first evaluation are confirmed in this second evaluation.

Chapter 6 - main contributions:

- Strengthening the validity of the conclusions drawn in Chapter 5 regarding the RAM model.
- From a research methodology perspective, the use of subjects from multiple (three) universities to perform a controlled evaluation may report valuable lessons learned (although this is not reported in this thesis).

## **Chapter 7 – Test-Case Driven Inspection of Pre-Project Requirements - Process Proposal and Industry Experience Report**

---

Chapter 7 presents an inspection process designed for early requirement inspections, called TCD inspections. The overall purpose of TCD inspections is to offer support to the MDRE process (RAM) by helping professionals incorporate early inspections (pre-release) assuring that requirements are good-enough for estimation activities and as a basis for initiating development efforts. In summation the TCD Inspection can be characterized by:

- Involving experienced testers in the inspection process, thus reducing the educational and training costs of inspectors
- Using testers' competence and effort for double purposes, i.e. testing and inspections. In software development projects it is common that testers review requirements specification in order to plan and execute testing activities. Thus using testers in pre-project inspections can be seen as an effective usage of company resources.

- Producing reusable inspection artifacts: test-cases produced during inspections are to be used in subsequent stages of the project such as during implementation (as an appendix to requirements offering another view to e.g. developers) and testing.
- As managers (writing the requirements) and testers are directly involved in TCD Inspections a learning effect (becoming better at writing and formulating requirements) is achieved.
- By performing inspections pre-project (prior to final selection of requirements for realization) managers get better requirements as input for requirements cost estimations, prioritization and selection.

The chapter presents both the TCD Inspection process and an experience report from industry application.

Chapter 7 - main contributions in comparison to the MDRE issues presented in Section 1.2.3, and in relation to RAM presented in Chapter 4:

- According to RAM, all requirements should be broken down and refined to a level where they are good-enough to be used as estimation base, to ensure that the implications of a (abstract) requirement is known, and input to a development effort. Thus using TCD inspections makes it possible to “test” requirements with regards to these aspects. Thus indirectly addressing:
  - *i2: Abstraction Level & Contents of Requirements.*
  - *i4a: Estimation.*
  - *i4b: Prioritization.*
- In addition, TCD inspections can be used in any RE scenario to assure requirements quality prior to estimation and development efforts commencing (even in bespoke development).
- TCD inspection is centered on the notion of maximizing the output of an inspection activity, e.g. by the reuse of inspection artifacts, thus spreading the cost of the inspection over several stages of development.

### **Chapter 8 – Industry Evaluation of the Requirements Abstraction Model**

This chapter presents two cases of RAM tailoring, implementation and most important evaluation, conducted at Danaher Motion Särö AB (DHR) and ABB Robotics (ABB). The main purpose is to give a brief overview of how RAM was tailored to fit two different organizations, and how the

model performed in terms of usability and usefulness based on evaluations performed with professionals using it in their day-to-day work, thus establishing the relative value of using RAM.

The relatively large scale industry trials presented in this chapter brings the focus back to industry, in comparison to the evaluations presented in Chapters 5 and 6, allowing us to validate the models usability and usefulness in a non-simulated environment.

The overall results of the evaluations indicate that the implementation of RAM at DHR and ABB has yielded substantial increases in both accuracy of the practices performed during MDRE, and in requirements quality. It is only natural that these improvements have a price, which can be observed in some effort increase over the board (also presented in the chapter), although there are also examples of the opposite. A learning curve effect can be used to explain some of the increase, but for the most part increased accuracy and quality will have some cost as more work is performed. Although in the case of DHR and ABB, these costs are very moderate in total.

Chapter 8 - main contributions in comparison to the success factors presented in Section 1.1.3 and Section 1.1.4 are:

- The assessment of RAM in an industry environment, eliciting information from professionals using RAM. Addressing primarily:
  - *F3: Commitment to SPI by management.*
  - *F4: Commitment to SPI and Involvement in the SPI work by coworkers.*
  - *F6: Measurability of improvement effects.*
  - *REF3: Commitment and Perceived benefit.*
  - *REF4: Quantification and Measurement.*

Chapter 8 - main contributions in comparison to the MDRE issues presented in Section 1.2.3 are:

- By assessing RAM and the underlying concepts behind the model the validity of the claims made in relation to the model in Chapter 4 are validated in an industry setting.

## 2.3. PART III – KNOWLEDGE AND TECHNOLOGY TRANSFER

### **Chapter 9 – Technology and Knowledge Transfer in Practice – Requirements Engineering Process Improvement Implementation**

---

Chapter 9 presents a knowledge and technology transfer process and experiences in using it. The process described is the one used during the planning and execution of the research presented in this thesis. Thus this chapter gives an overview of both the overall research approach used (see also Section 3.2.1), and how results from research were transferred to industry practice. The knowledge and technology transfer can be seen from two perspectives. From a research perspective it revolves around basing research agenda on industry needs, developing solutions through industry collaboration, and validating results in industry practice.

The second perspective is the industry view. Through participation industry (company) representatives can make sure that their perspectives are taken into consideration. For example, usability and usefulness of proposed improvements can be premiered, and the research results can be continuously validated thus minimizing risk and gradually collecting data on relative value of a the new potential improvements prior to commitment.

From our experiences technology transfer (and industry relevant research) is not about producing research results and handing them over in the form of publications and technical reports. The process we used involved several steps, one building on the other, carried out during a long-term joint commitment. These steps were devised in close collaboration with industry, and created in an evolutionary manner, adding steps as needed. This evolution also dictated what each step of the process contained, e.g. how validation is performed depends on the needs of the company (what they trust), as well as the needs of the researchers to validate new technology for academic purposes.

The contribution of this chapter is to present a technology transfer model that was devised on-demand in collaboration with industry and equally important, to report experiences and lessons-learned. The contribution is relevant for all success factors presented in Section 1.1.3 and Section 1.1.4, as it addresses the overall SPI process.

### 3. RESEARCH APPROACH

This section outlines the research approach used in this thesis, effectively taking a step back in regards to the chapters and their contribution described previously in Section 2. The structure of this section is as follows. Section 3.1 the research questions, the basis for the research and contributions, are presented and elaborated upon. Section 3.2 presents an overview of the research methods utilized to obtain answers to these research questions.

#### 3.1. RESEARCH QUESTIONS

The main research question posed in this thesis in a way summarizes all others. It can be seen more like an overall mission statement or goal, driving all following efforts rather than a traditional research question. It is formulated in the following manner:

*How can requirements management be improved in close cooperation and collaboration with industry?*

This overall original question limited the field, but opened up for an evolution of follow-up questions that needed to be answered one by one in order for progress to be made. The first subsequent question that needed to be addressed was devising a way to find out what the industry needs were. Close collaboration and cooperation with industry implied that industry relevant research was the focus, thus basing the research agenda on issues identified in cooperation with industry was the first step.

This in turn led to the realization that process assessment and improvement activities needed to be adapted to suit an industry environment where resources for SPI were very limited, but at the same time offer relative accuracy, and address the target area of RE. This led to the following general research question:

*General RQ: How can SPI efforts, targeted at RE, in organizations with limited time and resources be performed?*

This establishes a frame of reference and the formulation of more specific research questions. The main frame of reference is related to the delimitations set up, e.g. focus on applicability in organizations with limited time and resources to allocate to improvement activities.

In order to produce specific and tangible improvement proposals (what are the needs/what should be addressed) within the frame of reference the following research question was posed:

*RQ1: 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 maximize assessment performance and accuracy?*

Assessment performance was an important consideration (scope), 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 Chapter 2 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, and 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 perspective of minimizing cost and time the need for prioritization of the improvement suggestions was identified, as a way to help organizations establish an explicit order of improvement suggestion implementation, i.e. doing the most important things first. This led to the formulation of a new research question:

*RQ2: 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?*

Chapter 3 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.

In this case the implementation order also helped to define the continued research agenda. SPI package one consisted of three improvement suggestions, of which "abstraction level and contents of requirements" was the primary to be addressed (see Chapter 3). This gave rise to the research question:

*RQ3: How are requirements on varying levels of abstraction specified and refined in a market-driven product development environment?*

Chapter 4 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.

Enabling a development organization in a market-driven environment to elicit and specify requirements on multiple levels of abstraction, effectively enabling more requirements to be elicited and specified, made certain threats evident. The threat of requirements management overload was very clear, especially since long-term applicability (scalability) of any research result was paramount if it was to be usable and useful in an industry environment. This led to the next research question:

*RQ4: How can we limit the risk of requirement management process overload?*

The answer to this research question was not felt to be in utilizing any *one* tool/process/model but rather in a series of activities. Involving everything from making sure that a suggested MDRE process supported state models and attribute based specifications, to a clear division of roles and responsibilities in the supporting process. A combination of which would offer e.g. the possibility to divide responsibilities, offer different views the requirement base and so on.

Although there was a need to introduce more protection against overload, leading to the question:

*RQ5: How can we make early dismissal/acceptance of requirements possible in a structured and repeatable way, using the development organizations goals, while minimizing the chance that relevant requirements are dismissed?*

RQ5 can be seen as directly related to RQ4 as it is one step in avoiding overload. However, the main point of RQ5 is not only to enable triage, but also to in doing so minimizing the chance that relevant requirements are dismissed. In addition, repeatability implies that triage cannot be left up to a total dependency on a few key-personnel taking the correct intuitive decisions.

Using requirements abstraction and product strategies/roadmaps for the task of early dismissal/acceptance (triage) of requirements is one of the central parts of RAM (see Chapter 4). Enabling the overall goals of an organization, and for a certain product, to be used for triage purposes, offering support but not in any way replacing the key-personnel involved in the MDRE process.

One central issue in the MDRE process following early triage is the “production” of requirements good-enough to act as a base for all parts involving release planning. This leads to the question:

*RQ6: How can the requirements process support the central activities of estimation and prioritization to enable input to release planning and development activities?*

The need to break-down all requirements to an abstraction level suitable for development activities was one aspect, thus enabling input to development, but making the implications of abstract requirements explicit for estimation purposes was also crucial. The concept of having multiple levels of abstraction where every level is homogenous enough that requirements are comparable, enabling prioritization is also a part of RQ6.

Following RQ6 the need for quality assurance to minimize the risk of non-compliance with the goals set was evident and identified during initial validation. This led to RQ7:

*RQ7: How can the requirements quality, contents and abstraction level be systematically tested to assure that the requirements engineering process produces good-enough requirements for input to release planning and development activities?*

The development of new inspection technology for early requirements inspections was driven by the need for assurance that the MDRE process does “produce” good-enough requirements as input to release planning and development.

A central part of any MDRE process is the ability for multiple groups of professionals representing different views and competencies in a development organization to interact, collaborate and take decisions based on the same requirements base. An important step is to enable communication, leading to RQ8:

*RQ8: How can the requirements process support communication and cooperative work between technical management and marketing/sales management?*

By offering different views of the requirements base, in the form of multiple abstraction levels, but also the ability to premiere certain information with regards to attributes and states, gives different professionals relevant information without information overload. For example, by looking at a limited amount of fairly abstract requirements a marketing representative can get a relevant overview of the future potential product offering. Should more detail be needed the abstraction level is traversable down to more concrete requirements showing the implications of an abstract view. In the same way a developer can start at a concrete level and traverse upward to get the overall view of a certain product offering, coming closer to the goals and overall motivation for a certain requirement (or groups of requirements).

A central part of the validation of research results is the ability to assure applicability in more than one specialized case (e.g. one development organization). Therefore it was seen as crucial to assure that the require-

ments model and process devised was tailorable, making it usable in any organization faced with the challenges of market-driven product development. In other words, the ability to test the concepts related to RQ3 to RQ8 in more than one organization, leading to the question:

*RQ9: How can we assure that research results are applicable in more organizations than the one primarily involved in the conception of said research results?*

Ensuring tailorability of RAM at an early stage prepared for validation of the research results (concepts of RAM) in multiple environments.

As a part of research validation it was important to enable the test of central concepts on a large scale prior to industry trials. The primary motivation was to limit the risks for industry by doing pre-pilot tests in a less critical environment. This led to the question:

*RQ10: How can research results be evaluated prior to industry piloting?*

Tests in a controlled environment (a laboratory setting) were seen as an option as the resources needed were available to the researchers in the form of the academic environment (University and students). This allowed for a controlled evaluation to be devised and executed without using valuable industry resources (see Chapters 5 and 6). It should be noted that using the academic environment as laboratory, utilizing students does present some validity issues, but also ethical considerations need to be taken into consideration (see Chapters 5 and 6).

As a part of testing the usability and usefulness of RAM in a controlled environment the central concept of requirements abstraction needed to be evaluated, leading to RQ11:

*RQ11: How can the underlying concepts of abstraction inherent to RAM be tested on a large scale in a controlled environment prior to industry piloting?*

The controlled evaluations described in Chapters 5 and 6 were devised to test some of the underlying concepts behind RAM utilizing students as subjects.

RQ12 is closely related to RQ9, but from an empirical viewpoint. RQ9 was addressed by incorporating tailorability of the research results (many of them collected in RAM), in effect the goal was to make RAM tailorable. RQ12 challenges the intentions explicitly stated through RQ9 by asking if the goal of RAM tailorability was achieved in terms of industry applicability:

*RQ12: Does RAM support tailorability to suit different industrial environments?*

RAM was tailored to fit two organizations in industry (see Chapter 8) prior to industry piloting.

It should be noted that the SPI research presented in Chapters 2 and 3 were also tested in more than one environment, in effect it was used to elicit improvement issues from the same two organizations were RAM was subsequently piloted.

Most research questions posed thus far have been at least partly focused in industry relevance and applicability. A central issue following RQ12 which related to tailorability is stated in RQ13:

*RQ13: Can RAM be transferred to industry practice?*

The transfer to industry practice is of course dependent on the research answering most research questions posed earlier, but RQ13 also has a focus of its own. Even if all research questions up to RQ12 are explored, and answered through successful research with positive results there is no guarantee that the overall package (e.g. RAM) can be transferred to industry practice. The actual transfer to industry involves many potential problems, many of them dependent on commitment, perceived benefit and convincing practitioners of the relative benefit of a new solution over an existing one. These concepts are explored as a part of the overall technology transfer process described in Chapter 9. The actual transfer of RAM to industry is described in Chapter 8.

The successful initial transfer of RAM to industry practice does not only depend on the ability to implement the model and the accompanying new process in industry, but also on the success of the overall MDRE process using RAM. This leads to RQ14:

*RQ14: Does the use of RAM as a central part of the MDRE process result in improvements with regards to accuracy of activities performed, and offer an increase in overall requirements quality?*

Obtaining a positive answer to RQ14 is a prerequisite for verifying that the concepts explored and formulated in this thesis (some expressed through RAM) are indeed usable and useful in industry. In one regard all depends on the answer to RQ14, although in reality individual research results can have merit in themselves independent of RQ14. However, the main concepts presented in this thesis are summarized in RAM, and RQ14 points to the core of usability and usefulness of the model with accompanying processes. The original research question posed in the beginning of this section speaks to how RE can be improved in collaboration with industry. RQ14 addresses the heart of this and the answer to it is explored in Chapter 8 where the evaluation of the industry usage of RAM is summarized.

Initially in this section an overall goal research question was posed with the intention of exploring how collaboration with industry in a long-term research effort could be performed to improve the area of RE. An appropriate follow-up question can be formulated based on this:

*RQ15: How can research be conducted in collaboration with industry partners to assure usability and usefulness of conceived research results?*

RQ15 is explored explicitly in Chapter 9, although it depends on all parts presented in this thesis. The overall contribution is more than the sum of its parts which can be nicely illustrated though the contribution of an overall technology and knowledge transfer process.

## 3.2. 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 below in order of appearance (Chapter 2 through 8), but initially the general research environment is illustrated to give a context to the work performed. This overall research environment is closely related to the technology and knowledge transfer process presented in Chapter 9.

### 3.2.1. 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 [32] can be described as the following: problems are formulated in academia, the subsequent generation of solutions<sup>3</sup> are put forward in academia, and these solutions are 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" [32].

Technology transfer problems have also been identified explicitly in requirements engineering research [31]. Some of the problems highlighted

---

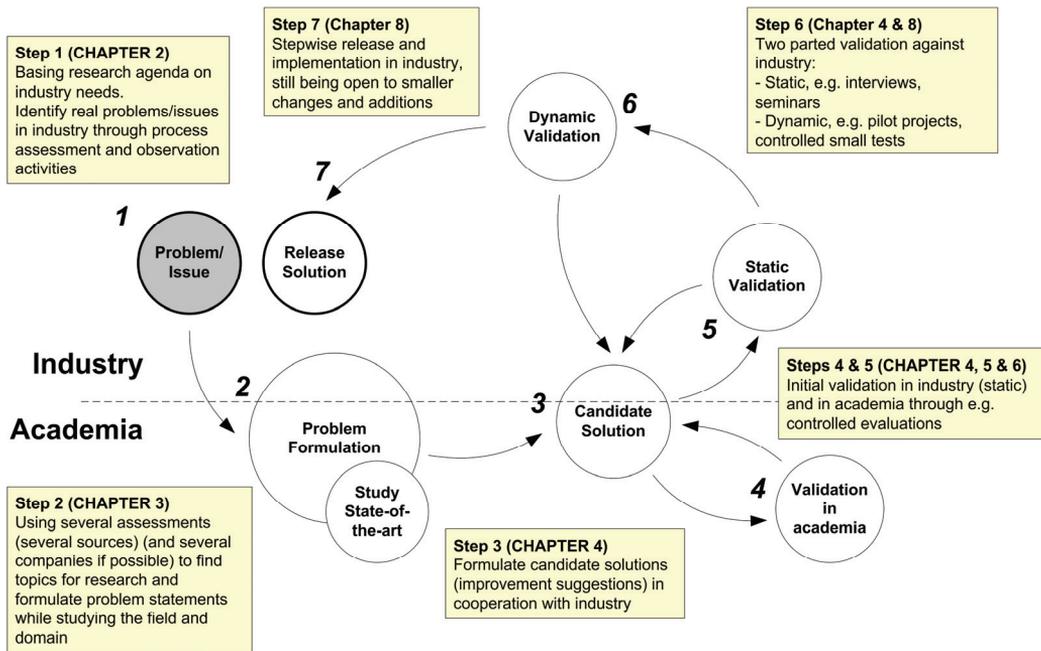
<sup>3</sup> The use of the word "solution" in this context is not to be seen in the sense of a definitive "fix" or "a silver bullet" to a identified problem, but rather as a word summarizing "contribution", "research result" and "research product".

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 1.1.3 and 1.1.4).

### **General Overview of Research Process**

The research presented in this thesis is based on the principle of using industry as laboratory in attempts to counter the overall problem with industry inapplicability. The overall research process (and technology transfer process) is illustrated in Figure 9 as seven main steps:

- *Step 1* implies that the main research direction and subsequent research questions formulated (see Section 3.1) are based on problems and issues explicitly identified in industry, i.e. the research is industry motivated.
- *Step 2*. Subsequent to the identification of possibilities for improvement, state-of-the-art (research in academia) was studied in order to see if solutions to the problems already existed (e.g. the assessment method in Chapter 2 was inspired by other inductive approaches). 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.
- *Step 3*. A solution is formulated based on the understanding of the problem, the results of the academia study and through innovation.
- *Steps 3 and 5*. Subsequent to formulation of the solution there was an initial validation in industry, followed by a modification of the solution based on this validation (e.g. the static validation of the RAM in Chapter 4).
- *Step 4*. Validation in academia can be seen as controlled evaluations in a laboratory environment as the factors surrounding the evaluation can be controlled to a much larger extent than during e.g. a pilot in industry.
- *Step 5*. A second larger round of static validations was performed prior to the pilot study.



**Figure 9.** *Research approach overview.*

- *Step 6.* The dynamic validation of the solution in industry was carried out as a pilot project to validate real applicability (see e.g. the dynamic validation of the RAM in Chapter 4, and description of the second pilot in Chapter 8). The results of the dynamic validation gave feedback as to the solutions viability, which was used for refinement.
- *Step 7.* As the solution matures with the help of previous validations it is released and implemented into industry practice (see Chapter 8 describing the industry implementation and subsequent evaluation of industry usage).

The approach described in Figure 9 is by no means totally sequential. Some overlaps and parallel activities are to be expected, and this was the case during the research presented in this thesis as well. For example, Steps 4 and 5 were to some extent carried out in parallel. In addition some of the validations were performed more than once, e.g. the static and dynamic validations were carried out twice, once initially during the presentation of RAM, then once more after validation in academia.

### Potential Validity Issues

It is important to realize that there can be issues in using industry as a lab. Working in closer cooperation with industry can threaten the generaliza-

bility [129] of the solution devised. The main reason for this is that a relatively few cases are studied, in this case two organizations. This can give specific research results, not really usable for organizations in general.

However, by recognizing this threat actions can be taken minimize these risks. For example, the study of a problem in academia (through e.g. experience reports etc.) can give indications if the problems identified in one or a few cases are general or not (see e.g. the critical/success factors identified in sections 1.1.3 for SPI, 1.1.4 for RE SPI, and 1.2.3 for MDRE). This enables a wider view than the one obtained from the study of few cases. Enabling a research agenda to be based on both the general scenario, and on the specific cases, and at the same time allowing for the results to be validated and tested against specific cases.

The benefits of working 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 identified in industry the chance for commitment to initial validation, and future application in industry increases.

### 3.2.2. RESEARCH PARTNERS

This section offers a short description of the industry partners referred to throughout the thesis.

#### **Danaher Motion Särö AB**

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.

Development projects range from six to nine months in calendar time, with a budget of 2000-5000 person-hours.

**ABB Robotics**

ABB is a leader in power and automation technologies that enable utility and industry customers to improve performance while lowering environmental impact. The ABB Group of companies operates in around 100 countries and employs about 102,000 people. The transfer of new methods for requirement engineering was performed with one of the ABB development centers in Sweden. The product development part of this organization has 200 employees, including development and product management. The organization is primarily working with product development, production and service, supporting ABB sales organizations as well as partners developing solutions for industrial use.

The introduction of RAM was made on the organization developing the controller part of the product offering. The controller includes electronics and software, and project typically involves development of functions in both hardware and software. Projects are divided into release projects with a typical duration of 9 months comprising 20-40 person years, and functional development projects with duration from 3 to 24 months with a large diversity in effort.

Product Management has the responsibility for the functionality of the controller, and orders development from the development organization. Over the last five years, the Product Development organization, including Product Management, has been re-organized several times. This indicates that there is a willingness and need to find improved working methods, also for the requirements engineering.

Process improvement is initiated and managed by process owners that are part of the development organization. Plans and progress are regularly monitored by senior management, and the interest in improving requirement engineering is steadily increasing.

**ABB Corporate Research**

The ABB Group R&D activities are performed in the Corporate Research organization. Corporate Research has two different main activities: to execute strategic R&D activities which are corporate funded, and to provide expertise to the different business units as consultants. The activities are concentrated on the main focus areas for ABB, Power and Automation technologies.

The corporate funded activities are organized in several research programs. The Software Architecture and Processes program includes one part which is related to process improvement. The objective of this work is

to improve the software development practices within the product development organizations in ABB

### 3.2.3. UTILIZED RESEARCH METHODS

The research presented in this thesis employs an empirical research approach as described by Glass in [32] and by Wohlin *et al.* in [129]. The observation of human behavior in the real world (industry) and the use of methods/models/practices (process) enable the possibility of identifying research issues on which a research agenda can be based.

The main empirical strategy used throughout this thesis is based on a flexible design research strategy, namely case studies [130]. This implies ‘development of a detailed, intensive knowledge about a single case, or of a small number of related “cases”’ [131]. For the purposes of this thesis the following distinction is made to clarify the specific situations often encountered:

- *Macro-case* implies the overall case, i.e. if a study of three projects is performed at *one* company, then the company is the macro-case (i.e. two macro-cases in the thesis, Danaher Motion Särö AB (DHR) and ABB Robotics (ABB)).
- *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 (e.g. projects) studied.

The following parts of the thesis employ the case study strategy:

- **Chapter 2** 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 [131]. As this study was replicated the assessment method presented in Chapter 2 was tested on two macro-cases (DHR and ABB).
- **Chapter 3** 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.
- **Chapter 4** presents RAM and the initial validation of the 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, using interviews, group interviews and document analysis for the subsequent evaluation.

- **Chapter 7** presents an inspection technology that was tested in industry in a limited fashion. The test utilized unstructured interviews as well as document analysis.
- **Chapter 8** reports on industry application and a subsequent evaluation of said application. Two macro-cases were studied (DHR and ABB). The main tools used were structured interviews, questionnaires, and document analysis [131].

In summary, the main 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 document analysis.

The exceptions to using case study research can be found in:

- **Chapters 5 and 6** present evaluations of RAM using controlled empirical evaluations. These evaluations were performed in a controlled environment, gathering quantitative data, but cannot be characterized as formal experiments. Only one treatment and one group was utilized [129].

## 4. LIST OF PUBLICATIONS

Chapters 2 through 9 are comprised of papers that are either published or submitted and under review for publication. In addition several papers were not included in the thesis, but may be of interest in relation to the ones presented in the thesis.

### 4.1. PUBLICATIONS INCLUDED IN THE THESIS

#### PART I

- Chapter 2:** Gorschek T. and Wohlin C., "Identification of Improvement Issues Using a Lightweight Triangulation Approach", in Proceedings of the European Software Process Improvement Conference (EuroSPI'2003), Verlag der Technischen Universität, Graz, Austria, 2003, pp. VI.1-VI.14.
- Chapter 3:** Gorschek T. and Wohlin C., "Packaging Software Process Improvement Issues - a Method and a Case Study", *Software: Practice & Experience*, vol. 34, 2004, pp. 1311-1344.

## PART II

- Chapter 4:** Gorschek T. and Wohlin C., "Requirements Abstraction Model", *Requirements Engineering journal*, vol. 11, 2006, pp. 79-101.
- Chapter 5:** Gorschek T. and Svahnberg M., "A Controlled Empirical Evaluation of a Requirements Abstraction Model", *Submitted to Information and Software Technology*, 2005.
- Chapter 6:** Gorschek T., Svahnberg M., Borg A., Börstler J., Eriksson M., Lonconsole A., and Sandahl K., "A Replicated Controlled Empirical Evaluation of a Requirements Abstraction Model", *Submitted to IEEE Transactions on Software Engineering*, 2005.<sup>4</sup>
- Chapter 7:** Gorschek T. and Dzamashvili-Fogelström N., "Test-Case Driven Inspection of Pre-Project Requirements - Process Proposal and Industry Experience Report", in Proceedings of the Requirements Engineering Decision Support Workshop held in conjunction with the 13th IEEE International Conference on Requirements Engineering, Paris, 2005.
- Chapter 8:** Gorschek T., Garre P., Larsson S., and Wohlin C., "Industry Evaluation of the Requirements Abstraction Model", *Submitted to Requirements Engineering journal*, 2005.

## PART III

- Chapter 9:** Gorschek T., Garre P., Larsson L., and Wohlin C., "Technology and Knowledge Transfer in Practice – Requirements Engineering Process Improvement Implementation", *in print IEEE Software*, 2005.

## 4.2. PUBLICATIONS NOT INCLUDED IN THE THESIS

- Paper 1:** Gorschek T., Tejle K., and Svahnberg M., "A Study of the State of Requirements Engineering in Four Industry Cases", in Proceedings of Software Engineering Research and Practice in Sweden (SERPS'02), Karlskrona, 2002, pp. 111-119.
- Paper 2:** Gorschek T., Svahnberg M., and Tejle K., "Introduction and Application of a Lightweight Requirements Engineering Process Evaluation Method", In the Proceedings of the

---

<sup>4</sup> The contribution of Professor Sandahl was limited to the execution of an evaluation subpart and data collection/coding.

- Ninth International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'03), Essen, Germany, 2003, pp. 101-112.<sup>5</sup>
- Paper 3:** Gorschek T. and Davis A., "Assessing the Quality of Requirements Process Changes", In the Proceedings of the Eleventh International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'05), Porto, Portugal, 2005, pp. 101-112.
- Paper 4:** Gorschek T. and Davis A., "Requirements Engineering: In Search of the Dependent Variables", *Submitted to Information and Software Technology*, 2005.
- Paper 5:** Svahnberg M. and Gorschek T., "Multi-Perspective Requirements Engineering Education with Focus on Industry Relevance", in the Proceedings of the 1st International Workshop on Requirements Engineering Education and Training, held in conjunction with the 13th IEEE International Conference on Requirements Engineering, 2005.
- Paper 6:** Gorschek T., Svahnberg M., Borg A., Börstler J., Eriksson M., Lonconsole A., and Sandahl K., "Understanding Perspectives on Requirements Understandability", *Submitted to the International Journal of Software Engineering and Knowledge Engineering*, 2005.
- Book Chapter:** Gorschek T. and Svahnberg M., "Requirements Experience in Practice: Studies of Six Companies", in *Engineering and Managing Software Requirements*, C. Wohlin and A. Aurum, Eds., 1st ed. New York, NY: Springer, 2005, pp. 405-426.

## 5. CONCLUSIONS

Research should *enable* product development organizations to improve, both short terms and long term. This implies that research should be pragmatic and based on industry needs, solving real problems for the real world, and testing the results and assumptions *in* the real world. Software engineering in general, and requirements engineering in particular, can be seen as applied fields of research. This is if nothing else inherent in the "engineering" part which by its very nature is closely related to the appli-

---

<sup>5</sup> This paper is a refinement of Paper 1.

cability of a proposed solution within the constraints posed by an environment.

The research presented in this thesis stemmed from concrete needs identified in industry. The process assessment and improvement (Part I) was needed to figure out *what* to do, the MDRE process improvement (Part II) described *how* it was done, and the validation and transfer to industry was a way of refining results, testing them and contributing to industry practice (Part III).

The individual contribution of each part of the thesis, and of each chapter, has been explored in previous sections. The overall contribution of the thesis is offering a relatively extensive set of practices on how research can be performed in close cooperation with industry, improving practice and adding to a field of research. The knowledge and technology transfer process presented in Chapter 9 is a snap-shot of this reporting lessons learned, but all parts of the thesis are equally relevant as they all contribute to the overall result.

An important contribution of the thesis is that all results, whether they are in the form of an assessment framework or an RE model, are designed to operate in an industry environment rather than a lab. For this reason tailorability and modularity was premiered. For example, RAM can (and is intended to) be tailored to suit different organizations. In addition, it can be used in combination with other methods and techniques to form an overall MDRE process. Recognition that one-size-does-not-fit-all, and that one solution does not work in isolation but has to be adaptable to the overall environment, is crucial. Solutions showing great promise in a laboratory environment may be inapplicable in industry practice as the complex environment demands things unidentifiable prior to industry piloting.

It is important to realize that the research results presented in this thesis although successful from certain perspectives, are not meant to be perfect in any way, rather good-enough to solve problems identified in industry. Good-enough is relative not only to solving a problem, but also to the issues of cost and time in doing so. This goes to the heart of being successful in terms of transfer of research results to industry.

When performing research based on needs identified in industry it is easy to get overwhelmed by the challenges inherent to producing results that should be validated in practice. Usability and usefulness directly related to the constantly present aspects of time and cost often dominate considerations. Focusing on the limitations can obscure the opportunities presented, as the industrial environment with its limitations can also provide researchers with tools to address a problem. An example of this can

be seen in RAM that utilizes multiple abstraction levels. These abstraction levels were first seen as a problem, then the acceptance of a fact (requirements coming in on multiple levels) was transformed to a possible solution.

Another example can be seen in Chapter 7 where an inspection technology was introduced with the explicit goal of spreading the cost of requirements quality assurance over more than one development phase. Reuse of inspection artifacts, learning effect, and use of already present experts were ways of minimizing the issues of cost and time while at the same time addressing the problem.

But presenting industry with research results addressing relevant challenges is not enough. Prior to piloting surrounding issues need to be addressed. Examples of such issues can be tool support, training, manuals, reference guides and so on. These may not be a part of the research itself, but it is important to realize that it is a part of the overall transfer process, and a prerequisite for securing commitment and industry validation.

An important part in getting commitment is also demonstrating relative value of a new solution over practices already in place. This can be a “Catch-22” as relative value seems to be dependent on industry trials. In the case of RAM academia was used as laboratory to evaluate aspects of the model in a controlled environment. The motivation was of course to test usability and usefulness, but also to gather data which could be used to demonstrate positive aspects, as well as limitations. This controlled evaluation in combination with all other static and dynamic validations (pilots) were used as input to the risk assessment which is a part of incorporating any new practice or model in industry.

The area of market-driven requirements engineering is a field with unique challenges. The fact that it is possible to validate research results addressing these challenges in industry practice is an opportunity which should be exploited. This has been recognized by many contributors to the field over the last decade, resulting in promising and definitely relevant results, many of them used as inspiration to the research presented in this thesis.

The overall goal with the research presented in this thesis was to improve the area of requirements engineering by utilizing industry collaboration. The development of assessment and improvement planning methods allowed us to elicit relevant areas to address, and gain commitment as the limits and opportunities presented by industry practice were incorporated explicitly in the overall research effort.

The conception and gradual refinement of an MDRE model grew out of the collaboration. RAM is aimed at addressing many of the challenges

identified in cooperation with our two industry partners, challenges that have been confirmed by other reports from industry. Initial post-transfer results of RAM show great promise, but also give us information about limitations on which future research endeavors can be based.

## 5.1. FUTURE RESEARCH

There is a real need to actively address the issues of providing models and methods to enable development organizations to improve. There are two parts of this, one general regarding SPI, and one targeted at MDRE.

(i) The process evaluation framework and improvement planning tool presented need to be refined. Assessment and planning are prerequisites for any SPI effort not using a prescriptive approach. Chapters 2 and 3 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.

An important aspect that also needs further research is measurement of improvement of success in general and in the area of MDRE in particular. High cost extensive measurement programs and gathering of metrics have to be adapted to fit the situation of industry with limited resources for anything but core pursuits, this is especially true for small and medium sized enterprises.

The area of MDRE poses additional challenges as the measurement process is confounded by many factors in addition to “just” measuring requirements selection accuracy. For example, the success of a product is a direct effect of the requirements selection quality, but the success of a product also depends on many other aspects, e.g. marketing and sales. In addition there is a need to measure long-term effects (e.g. product success), but enable fast improvement activities continuously.

(ii) The other part of enabling improvement is the development of better methods and models suited for MDRE. Looking at the area there is a need to enable development organizations to handle the extensive and continuous 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 but also addressing long term scalability and tools support. RAM presents new possibilities for using product strategies/roadmaps for early acceptance/dismissal of requirements, but it depends on the formulation of product strategies that are appropriate for the task.

Another area with great potential for enabling improvements is the creation of explicit support for even better communication between technical management and general management, marketing and sales. The requirements can be seen as a least-common-denominator stating what an organization can and should do. This fact can be used to the advantage of everyone. The use of the requirements abstraction concept shows promise, but the current state of RAM has barely begun to investigate the area and gauge the potential.



# **PART I**

## **Process Assessment and Improvement**

---



# Chapter 2

---

## Identification of Improvement Issues Using a Lightweight Triangulation Approach

2

*Tony Gorschek and Claes Wohlin*

European Software Process Improvement Conference (EuroSPI'2003), Verlag der Technischen Universität, Graz, Austria, 2003.

### **Abstract**

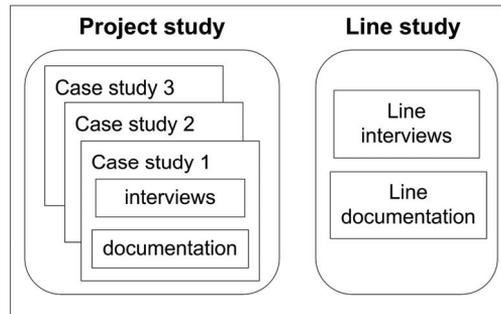
One of the challenges in requirements engineering is the ability to improve the process and establish one that is “good-enough”. The objective of this chapter is to present a lightweight approach to identify process improvement issues. The approach is developed to capture both the views of different 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.

# 1. INDUCTIVE PROCESS ASSESSMENT

## 1.1. 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 [13, 14].

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) [63]. The lightweight approach presented uses data point *triangulation* [131] 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 10. The project study is comprised of three exploratory case studies [131], and each case study is in turn comprised of project specific interviews and documentation (see Section 1.3.2).



**Figure 10.** Investigation description.

The line study is comprised of line interviews and documentation (see Section 1.3.3). The main objective of the investigation was to ascertain state-of-practice at Danaher Motion Särö (DHR) (see Section 1.2) and to identify improvement points in their RE process. There are approaches for RE process improvement targeted at e.g. SME. Sommerville and Sawyer [18] for instance present a series of guidelines as well as a guide for process improvement as a product of the REAIMS project [74].

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 1.3.1), In addition to this the

emphasis of the approach presented in this chapter is on fast and fairly low-cost evaluation.

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) [23]. 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 1.3), (ii) the improvement points identified and triangulated (presented in Section 1.4), and (iii) the dependencies between the improvement points (also presented in Section 1.4). In addition to this there is an overview of state-of-the-art in relation to each improvement point (see Section 1.5). Section 1.6 contains the conclusions drawn in the chapter.

## 1.2. 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 un-certified 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. Require-

ments 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.

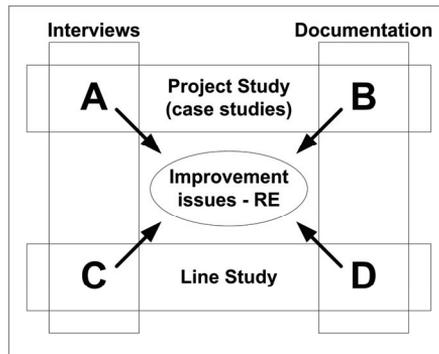
### 1.3. INVESTIGATION DESIGN

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.

#### 1.3.1. 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 [132]. 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 1.3.2). (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 11. 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.

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 1.3.3).



**Figure 11.** *Triangulation using several data sources.*

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 1.3.2.1), 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.

### 1.3.2. 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 [133]. 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 [3] 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.

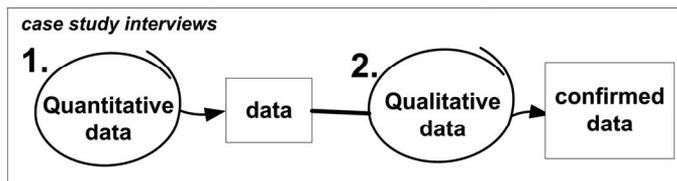
### 1.3.2.1 Case Study Interview Questions

Semi-structured interviews [131] 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 in-

spired primarily by the work of Sommerville & Sawyer [18] and Gorschek et al. [75]. 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 12 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 11).



**Figure 12.** *Pre-validation of case study data.*

---

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.

### 1.3.2.2 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 11).

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

### 1.3.3. 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 [131], 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 1.3.2.1) 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.

### **1.3.3.1 Line Documentation**

Line documentation (D) (see Figure 11) 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.

## **1.3.4. 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. [129]. One type of threats mentioned in [129] 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.

### **1.3.4.1 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.

### **1.3.4.2 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.

### **1.3.4.3 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 and 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.

## **1.4. RESULTS**

### **1.4.1. PROJECT STUDY**

#### **1.4.1.1 Case Study Interviews (A)**

Data from the case study interviews resulted in nine general improvements issues summarized in

Table 4. 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 1.3.2.1).

**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.

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

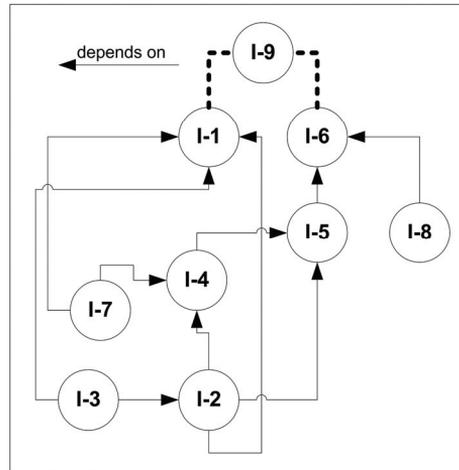
**Table 4.** *Improvement issues case study.*

**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 13 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 13 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 13.** *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 [134]) 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 13), 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 13) as far as the level of abstraction. All is-

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 13.

#### 1.4.1.2 Case Study Project Documentation (B)

Looking at the project documentation in the case study projects several of the improvement issues described in Section 1.4.1.1 are substantiated.

Table 4 shows each issue (sorted by issue id), the cells in the B-column (B for triangulation point B, see Figure 11) 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.

### 1.4.2. LINE STUDY

#### 1.4.2.1 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 6, C standing for triangulation point C, see Figure 11). 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.

**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) [18, 134], and to partially trace requirements through design documents to implemented components (forward-from traceability) [18, 135]. These traceability possibilities are however dependent on individual knowledge by project team members and they are not documented.

B	Issue Id	Note
<input checked="" type="checkbox"/>	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	
<input checked="" type="checkbox"/>	3	No requirements prioritization is documented, and no priority grouping can be observed.
<input checked="" type="checkbox"/>	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	
<input checked="" type="checkbox"/>	7	The problems with Issue-1 and Issue-4 make system test based on requirements difficult.
	8	
<input checked="" type="checkbox"/>	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.

**Table 5.** *Improvement issues documentation.*

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

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

**Table 6.** *Improvement issues line interviews.*

### 1.4.2.2 Line Study Documentation (D)

Several issues identified thus far (see sections 1.4.1.1 and 1.4.1.2) were not substantiated by the line documentation, i.e. Issue-2 and Issue-4 to Issue-8. Table 7 offers an overview of this.

D	Issue Id	Note
<input checked="" type="checkbox"/>	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	
<input checked="" type="checkbox"/>	3	No instructions/help/regulations/method description exists to aid in the prioritization of requirements.
	4	
	5	
	6	
	7	
	8	
<input checked="" type="checkbox"/>	9	see Issue-1, Issue-3
<b>Improvement Issues from line interviews</b>		
<input checked="" type="checkbox"/>	10	No policies for requirements reuse exist.
<input checked="" type="checkbox"/>	11	No policies for requirements traceability exist.
<input checked="" type="checkbox"/>	12	No policies for requirements version handling exist.

**Table 7.** *Improvement issues line documentation.*

---

### 1.4.3. TRIANGULATION OF RESULTS

Table 8 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 1.3.1), and are considered to be triangulated.

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.

Issue Id	A	B	C	D	TOTAL
1	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	4
2	<input checked="" type="checkbox"/>				1
3	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>	3
4	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		3
5	<input checked="" type="checkbox"/>				1
6	<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>		2
7	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		3
8	<input checked="" type="checkbox"/>				1
9	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	4
10			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	2
11			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	2
12			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	2

**Table 8.** *Triangulation of improvement issues.*

## 1.5. RELATION TO STATE-OF-THE-ART

In this section each triangulated issue (see Section 1.4.3) 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 [5, 13, 18, 136]. 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 [18, 137]Gorschek, 2003 #66} 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 [138].

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 [139], to how important linguistics [140] and ethnography [141] are in the context.

Formal specifications [89] 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 [3, 5, 18, 136].

**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 [142, 143], one of the most well known is the Analytic Hierarchy Process or AHP for short [102]. The main issue with this method is scale-up issues, i.e. prioritizing of a lot of requirements is rather time consuming [142]. 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 [98]. 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 14 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 [144]. One issue for RE is to identify the tasks to be conducted [18, 135], 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.

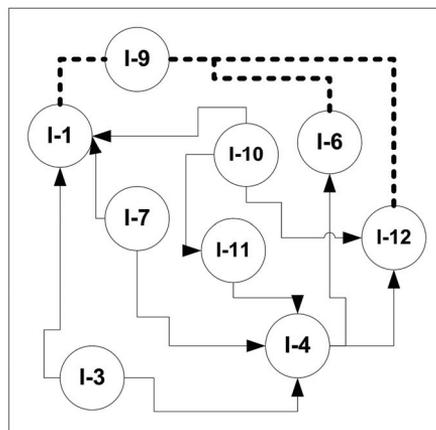
**Issue-7:** *System tests performed against requirements.* Requirements are typically described as *what* is to be delivered to a customer [18]. By performing system tests on requirements it is possible to validate if the implemented features are really based on the requirements (functional requirements) [136], and if the system complies with other requirements (non-functional requirements) [136]. Making test scenarios/cases during

initial requirements engineering can also be a good way of validating the requirements themselves at an early stage [18].

**Issue-9:** *RE process/methods*. All issues are included here. There are several RE processes suggested in literature [18, 135, 136], 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 [145]. 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 14 [146]) everything from the requirement itself to analysis, design, implemented components, test cases, scenarios, and use cases etc. can be reused [18].

**Issue-11:** *Requirements traceability*. Several types of traceability could be identified in RE [135]. 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 [18, 135]. The first is important for verifying what sources there are for a certain requirement [147], and the second for making reuse of more than just the requirements themselves possible. There are however several views on requirements traceability [18, 135, 148] which should be studied before a distinction about what model for traceability should be implemented in an organization.



**Figure 14.** *Dependency diagram (updated).*

**Issue-12: Requirements version handling.** This issue could effectively be sorted under the umbrella of traceability [18, 135], but is separated due to the fact that this issue was identified separately from Issue-11 during the line interviews (see Section 1.4.2.1). 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.

## 1.6. 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 evolution 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 SMEs.

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 im-

proved. 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 SMEs. 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.



# Chapter 3

---

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

# 3

*Tony Gorschek and Claes Wohlin*

Software: Practice & Experience, vol. 34, 2004, pp. 1311-1344.

### **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.

# 1. PROCESS IMPROVEMENT (PRE-) PLANNING

## 1.1. 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<sup>6</sup> to reach their goals.

It stands to reason that continuous evaluation and improvement of an existing process (Software Process Improvement - SPI [3]) 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 1.2.1).

The chapter is structured as follows. Section 1.2 gives some background information pertaining to SPI concerns and the motivation behind the work presented in this chapter. Section 1.3 introduces the “Dependency Adherent Improvement Issue Prioritization Scheme” or “DAIIPS” for short, and the techniques on which it stands. Section 1.4 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 1.5 has the discussion and the conclusions.

---

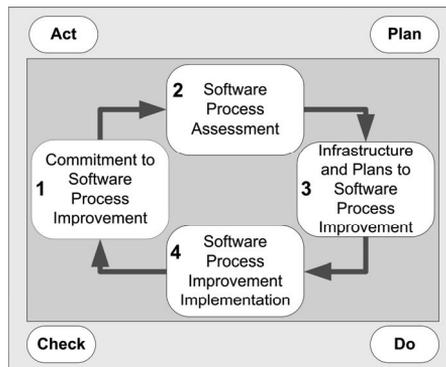
6 A sequence of steps performed for the purpose of producing the software in question (IEEE-STD-610).

## 1.2. 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 15, inspired by [27]).

Most well known process improvement (quality improvement/assurance) frameworks are based on this general principle. From the Shewart–Deming “Plan-Do-Check-Act” paradigm [149] and Basili’s “Quality Improvement Paradigm” [34], to standards like CMM [39], CMMI,[41] ISO/IEC 15504 (a.k.a. SPICE) [45] and the Software Engineering Institutes IDEAL SPI Guide [42].

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 15.** *Generic process improvement scheme.*

The main issues seem to be *cost* and *time*. An assessment-improvement cycle is often rather expensive and time consuming [51]. A typical SPI cycle using e.g. CMM can take anything from 18 to 24 months to complete [30] 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 [27-29].

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) [27, 28]. 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 [64] reports an initiative to use elements from proven technologies like CMMI and SPICE to implement a lightweight SPI framework. Another example is presented in [26] 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.

### 1.2.1. 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 [55-59].

Some of the central are<sup>7</sup> (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 1.4), 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

---

<sup>7</sup> 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.

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-through 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 [51]. 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.

### 1.2.2. **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.

### 1.3. DAIIPS – AN OVERVIEW

DAIIPS consists of three basic steps, (A) prioritization, (B) dependency mapping, and (C) packaging (illustrated in Figure 16). 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.

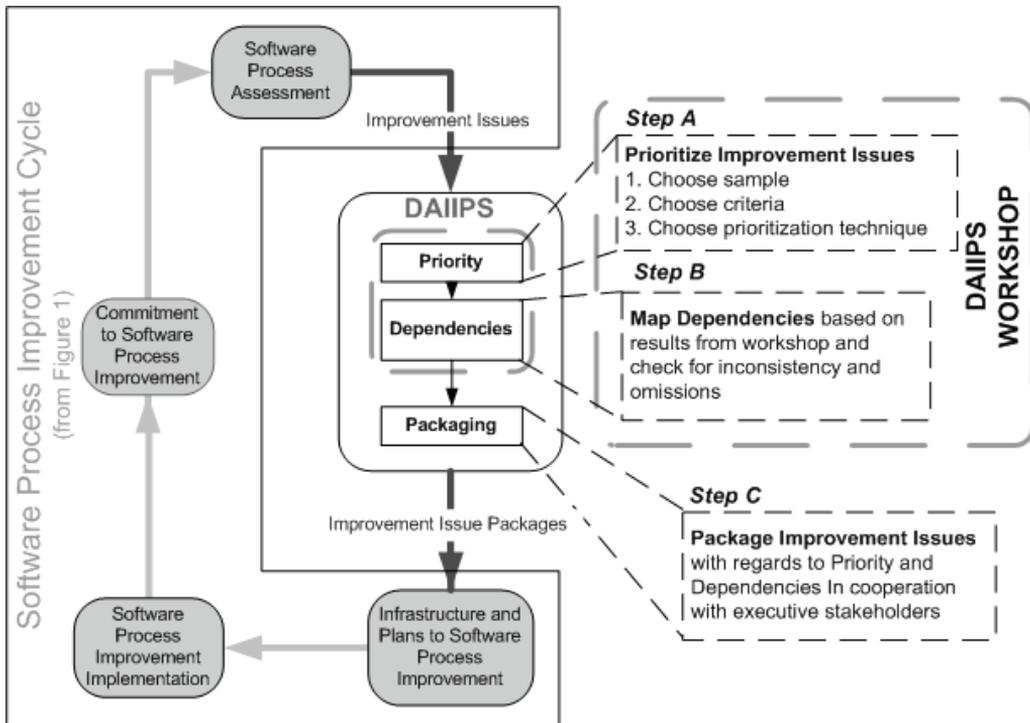


Figure 16. DAIIPS overview.

#### 1.3.1. 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 assess-

ment. A number of sub-steps should be performed in prioritization (see Figure 16), each described below. For an example of the prioritization (Step A) and results from an industry case, see Section 1.4.3.2.

### **1.3.1.1 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. [129, 131] 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 [129] 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 1.4.2.1.1.

### **1.3.1.2 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.

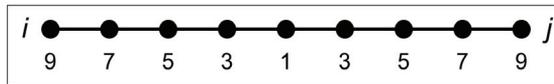
The choice of criteria should be conveyed to the participants beforehand (and consensus reached about what a criterion means) in order for the prioritization to be done with the same focus in mind. For an example of criteria formulation in an industry case, see Section 1.4.2.1.1.

### 1.3.1.3 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 1.4), and mention three more. Note that AHP is just one of several available prioritization techniques that may be used.

**The Analytical Hierarchy Process (AHP)** [150] is a method using scaled pair-wise comparisons between variables, as illustrated in Figure 17.



**Figure 17.** *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 9.

As the variables have been compared the comparisons are transferred into an  $n \times n$  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.

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 $n_{ij} = x$ then $n_{ji} = 1/x$ .	

**Table 9.** *AHP comparison scale.*

AHP uses more comparisons than necessary, i.e.  $n \times (n - 1) / 2$  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  $\leq 0.10$  is considered to be acceptable according to Saaty [150], 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 [151] and [152], and this is an ongoing debate. A rule of thumb is that a CR of  $\leq 0.10$  is optimal, although higher results are often obtained in the real world. Further details about AHP can be found in [150] and [102].

AHP Example (Using AHP to prioritize mobile phone features)														
answer	feature												feature	
1	SMS	9	+	7	+	5	+	3	+	(1)	+	3	+	Color display
2	SMS	9	+	7	+	5	+	3	+	(1)	+	5	+	WAP
3	SMS	9	+	7	+	5	+	3	+	(1)	+	3	+	Vibrating Call Alert
4	WAP	9	+	7	+	5	+	3	+	(1)	+	3	+	Color display
5	WAP	9	+	7	+	5	+	3	+	(1)	+	3	+	Vibrating Call Alert
6	Vibrating Call Alert	9	+	7	+	5	+	3	+	(1)	+	3	+	Color display

Having 4 features [ $n=4$ ] to prioritize against each other would demand a table with  $4 \times (4 - 1) / 2 = 6$  rows [ $n \times (n - 1) / 2$ ], 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 [153] 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 [103].

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  $n \times (n - 1) / 2$  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 [142].

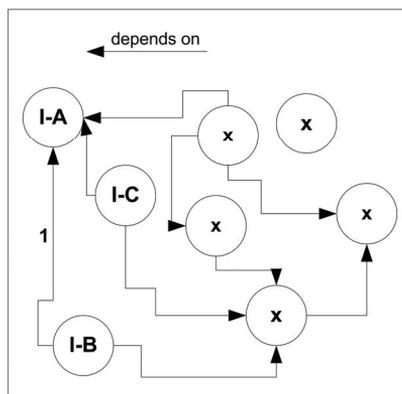
#### 1.3.1.4 Weighing the Priorities after the Fact

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 1.3.1.1, 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.

#### 1.3.2. 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 1.4.3.3.



**Figure 18.** *Dependency diagram example.*

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 1.3.1.1). 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 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 18, and the direction of the dependency can be seen through the direction of the arrow. In Figure 18 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 18 *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.

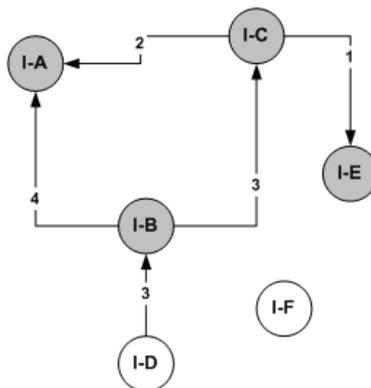
The result of this step should be a list of dependencies between the issues as well as their relative weight, i.e. how many times they are specified by the participants. Table 10 illustrates the results of an example dependency mapping. The dependencies present (e.g. *I-B* on *I-A*) is decided by what dependencies are identified.  $P_1, P_2, \dots, P_n$  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. The weights are compiled and drawn into a dependency diagram, see Figure 19, which is like the one presented earlier in Figure 18, 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.

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					

**Table 10.** *Dependency table.*

If an improvement issue dependency has a low weight, e.g. 1 (*I-C* on *I-E* in Figure 19 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 19.** *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.

### 1.3.3. 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 1.4.3.4.

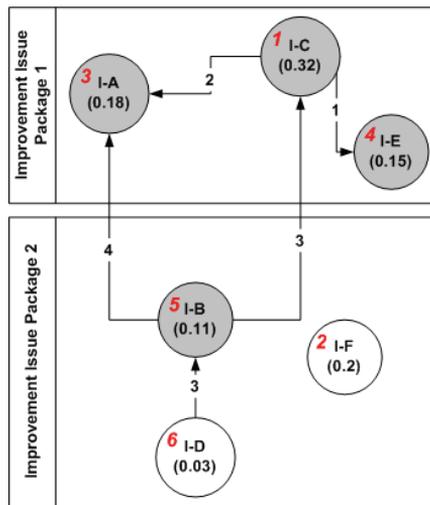


Figure 20. Improvement issue packages.

Figure 20 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 *I-D* the lowest (note that all priority methods do not produce relative values of priorities, AHP is used in this example, see Section 1.3.1.3).

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.

## **1.4. 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 1.4.2 the designs of the studies are described. The results from the industry study are presented in Sub-sections 1.4.3.2 (Step A), 1.4.3.3 (Step B), and 1.4.3.4 (Step C), corresponding to DAIIPS three major steps (see Figure 16). Section 1.4.2.2 presents the academia (validation) study results, and in Sub-section 1.4.5 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.

### **1.4.1. 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.

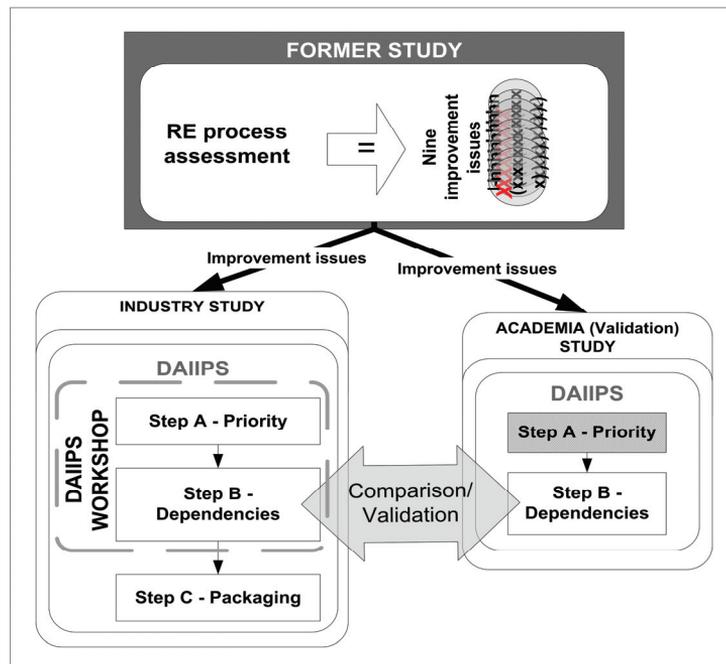
A proceeding process assessment concentrated on the area of requirements engineering was conducted at DHR using the lightweight triangulation approach presented in Chapter 3.

The nine improvement issues (see Section 1.4.3) used as input to DAIIPS (as viewed in Table 13) came from this assessment of the requirements engineering process at DHR.

## 1.4.2. STUDY DESIGN

This section covers the design of the industry study performed. The design is based on the DAIIPS steps described in Section 1.3, 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 1.4.2.3.3), 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 21.



**Figure 21.** Study overview.

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.

### 1.4.2.1 Industry Study Design (Preparation Step A and B)

#### 1.4.2.1.1 *Prioritization Preparation (Step A)*

Sampling (see Section 1.3.1.1) 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 1.4.1). It was based on quota sampling, in an effort to get subjects from different parts of the population [129]. 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 11.

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

**Table 11.** *Participant distribution over roles – prioritization.*

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 1.3.1.2) 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 1.3.1.3) 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 1.3.1.3 – AHP for further information), and [142] for a evaluation of AHP in comparison to other prioritization techniques.

#### 1.4.2.1.2 *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 12.

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
<b>Total</b>	<b>10</b>

**Table 12.** *Participation distribution over roles - dependency mapping.*

---

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 0) 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.

### 1.4.2.2 Academia (Validation) Study Design

**Sampling** (see Section 1.3.1.1) of the subjects for the prioritization (and later the dependency mapping) was based on convenience sampling [131]. The sample consisted of six PhD students from the Department of Software Engineering & Computer Science at Blekinge Institute of Technol-

ogy. 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 0.

### 1.4.2.3 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. [129]. One type of threats mentioned in [129] 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.

#### 1.4.2.3.1 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.

#### 1.4.2.3.2 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.

### 1.4.2.3.3 *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.

## 1.4.3. INDUSTRY STUDY EXECUTION AND RESULTS

The first part of the DAIIPS workshop (see Figure 21 and Section 1.3) 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 13. 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, preceded 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.

<b>Improvement Issues (in no specific order)</b>
<p><b>Issue-1: Abstraction level &amp; Contents of requirements</b> 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.</p>
<p><b>Issue-2: Requirements prioritization</b> This issue suggests a systematic prioritization of all requirements in a standardized way.</p>
<p><b>Issue-3: Requirements upkeep during &amp; post project</b> In order to keep the requirements up to date during and post project the requirements have to be updated as they change.</p>
<p><b>Issue-4: Roles and responsibilities - RE process</b> 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.</p>
<p><b>Issue-5: System tests performed against requirements</b> All system tests should be performed against requirements (e.g. using requirements as a base to construct test-cases).</p>
<p><b>Issue-6: RE process/methods</b> This issue is basically the creation and incorporation of a complete and comprehensive Requirements Engineering process at DHR that is well defined and documented.</p>
<p><b>Issue-7: Requirements reuse</b> By reusing requirements everything from the requirement itself to analysis, design, implemented components, test cases, scenarios, and use cases etc. can be reused.</p>
<p><b>Issue-8: Requirements traceability</b> Policies and support for traceability to and from the requirements are to be established.</p>
<p><b>Issue-9: Requirements version handling</b> Policies and support for version handling of each requirement (not only on requirement's document level) should be established.</p>

**Table 13.** *Improvement issues at DHR.*

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

### 1.4.3.1 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  $\approx 0.22$  (median  $\approx 0.16$ ). A summary of the participants CR is given in Table 14.

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.

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

**Table 14.** CR for DHR participants.

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 14, the grey cells).

In Table 15 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.

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)
<b>TOTAL</b>	<b>14</b>	<b>9 (-5)</b>

**Table 15.** Result of invalidation impact on sample.

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 1.4.3.2 and Figure 15).

### 1.4.3.2 Priority of Improvement Issues (Results - Step A)

The results of the prioritization can be viewed in Table 16 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 16 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 22 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.

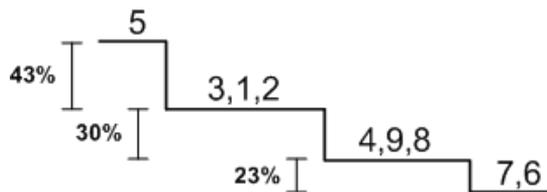


Figure 22. Priority shelves.

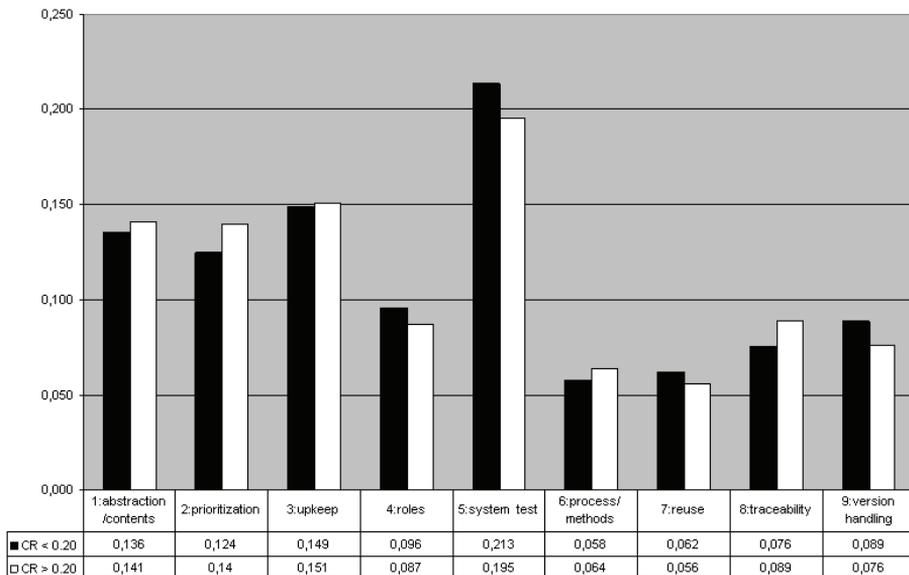
There is a quite large difference in opinion regarding issue 5: *System Test* (ranked number one), i.e. a scattering from a priority of only 0.05 to an extreme 0.44 (i.e. more than eight times as high). The scattering is less in issues 2-4 which lie closer in priority.

Figure 23 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 first 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.

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	

**Table 16.** Priority results (CR<0.20).



**Figure 23.** Priority comparison.

### 1.4.3.3 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 1.3.2).

Dependency ( Issue <i>i</i> on issue <i>j</i> )	Weight in %		Dependency ( Issue <i>i</i> on issue <i>j</i> )	Weight in %
<b>2 on 1</b>	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

**Table 17.** *Dependencies between improvement issues identified at DHR.*

The results from the dependency mapping can be seen in Table 17, 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.

A dependency diagram was drawn (see Figure 24). Here the relations, their weight (e.g. 0.6 is the same as 60% in Table 17), and the relations' direction can be observed. Relative priority (the value inside the parenthesis) and rank (top bold numeral) are also visible.

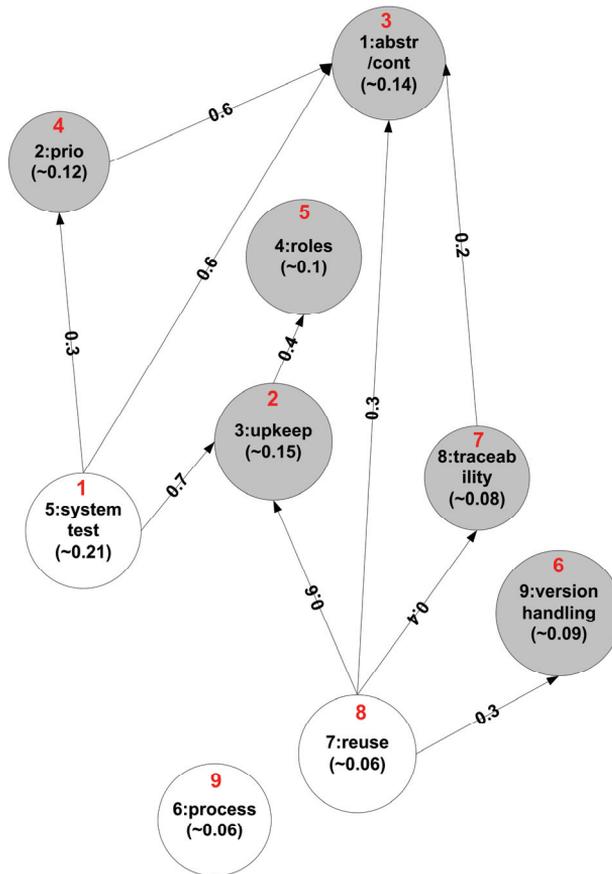


Figure 24. DHR dependency diagram.

### 1.4.3.4 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 25. This was a seemingly optimal choice with regards to priority and the mapped dependencies. However this packaging (as illustrated in Figure 25) 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.

The previously suggested SPI Package A (see Figure 25) was divided into two packages, i.e. 2: *prioritization* and 5: *system test* was broken out to its own package (see Figure 26). This meant that issue 5 (which had the highest priority) was postponed. The reasoning was that SPI Package 1

(see Figure 26) 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 25. 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 26) or not could be made earlier and at a lesser cost in regards to time and resources.

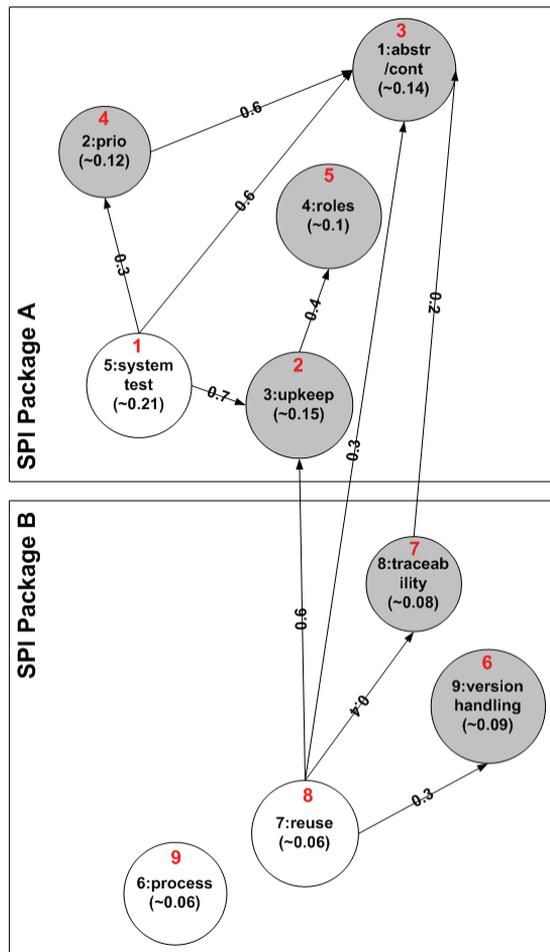
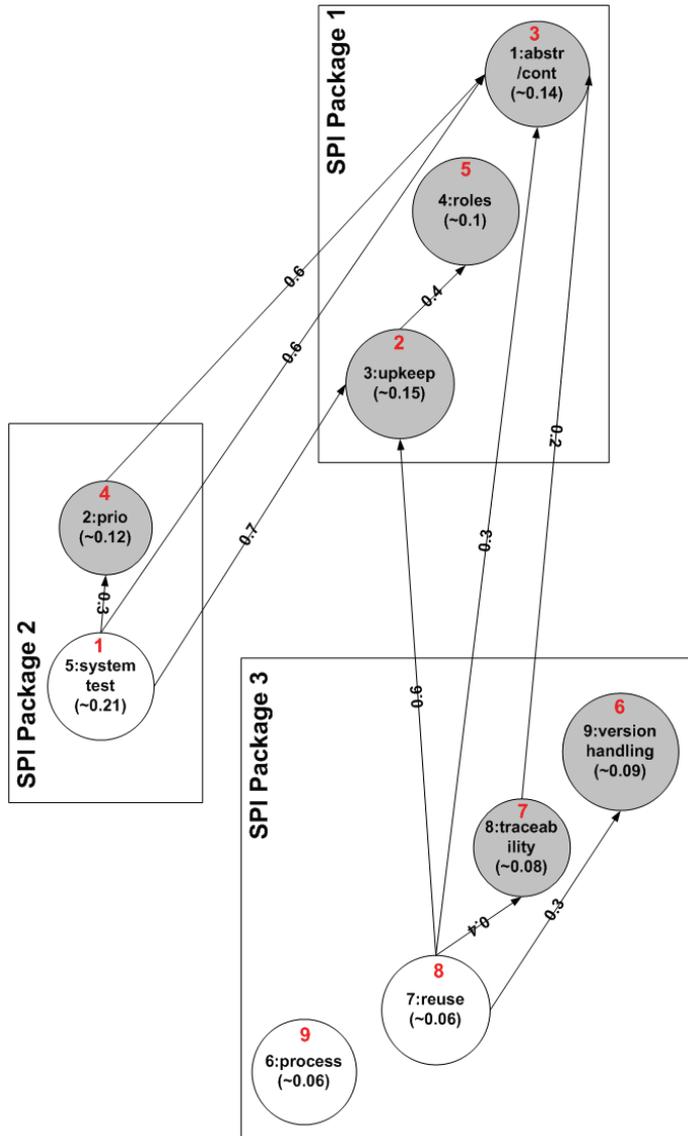


Figure 25. SPI package suggestion at DHR.



**Figure 26.** Final SPI package at DHR.

#### 1.4.4. 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.

#### 1.4.4.1 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  $\approx 0.40$  (median  $\approx 0.35$ ). A summary of the participants CR can be viewed in Table 18.

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

**Table 18.** *CR for academia Participants.*

In Section 1.4.2.3.3 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.

#### 1.4.4.2 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 1.3.2).

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

**Table 19.** Dependencies between improvement issues identified in academia.

The results from the dependency mapping can be seen in Table 19. It is noticeable that no dependencies identified were specified by less than 33% (i.e. 2 participants in this case).

## 1.4.5. COMPARISON – INDUSTRY VS. ACADEMIA

### 1.4.5.1 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 1.4.2.3.3). In addition this section provides a chance to observe some differences between the outlook on dependencies between industry and academia.

Table 20 presents a summation of the dependencies presented previously (i.e. in Table 17 and Table 19). The weights for both industry (column two) and academia (column three) are presented in percent. The dependencies are also presented in Figure 27 where dependency weights are presented on the lines (industry | academia).

### 1.4.5.2 Comparison Analysis

Below some analysis and augmentation is provided on the dependencies listed in Table 20. The note column holds numbers that have corresponding numbers in the text below, i.e. some analysis is offered for each num-

ber. 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 1.4.3.3). The dependencies here are not analyzed or commented upon any further.

Dependency (Issue <i>i</i> on issue <i>j</i> )	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

**Table 20.** *Dependency comparison between DHR and academia.*

**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  $\leq 50\%$ . 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.

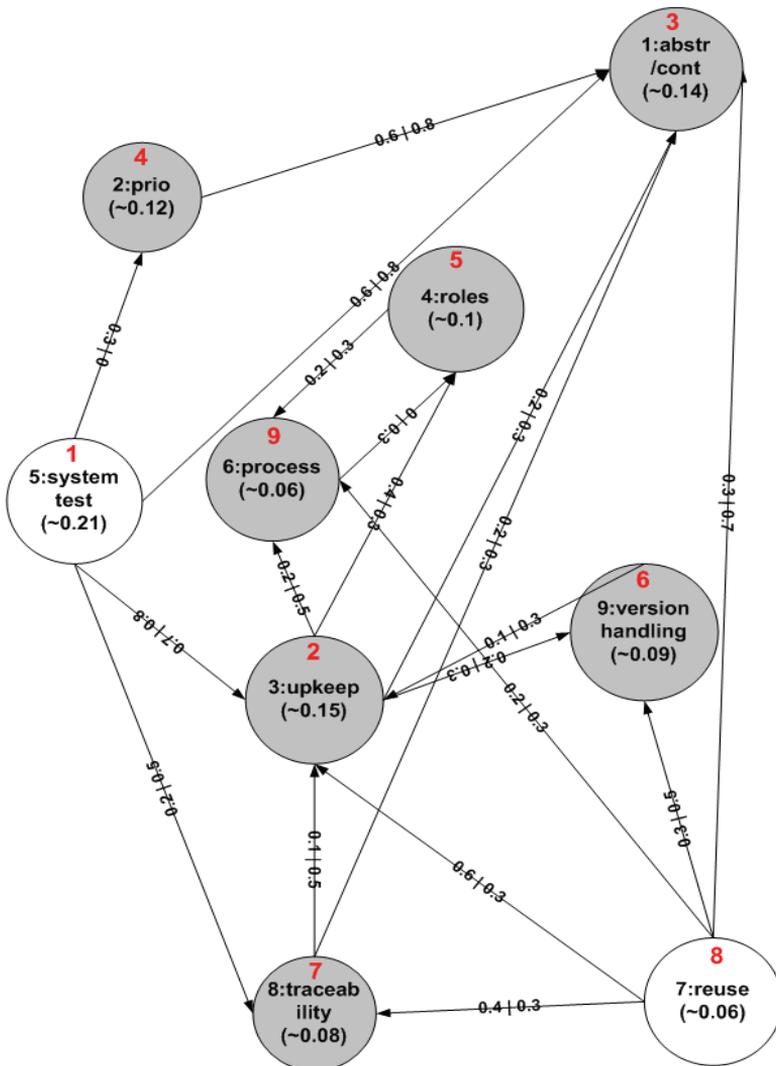


Figure 27. Dependency diagram (DHR Priority | Academia Priority).

**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 ( $\leq 20\%$ ). 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.

**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 1.4.3.3) 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 premised.

**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. up-kept). 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.

### 1.4.5.3 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 26, 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 sel-

dom 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.

## 1.5. 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 1.2.1) 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.

## 1.6. 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 of 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.

## **PART II**

# **MDRE Model Presentation and Validation**

---



# Chapter 4

---

## Requirements Abstraction Model

*Tony Gorschek and Claes Wohlin*

Requirements Engineering journal, vol. 11, 2006, pp. 79-101.

4

### **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 results 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.

## 1. THE REQUIREMENTS ABSTRACTION MODEL

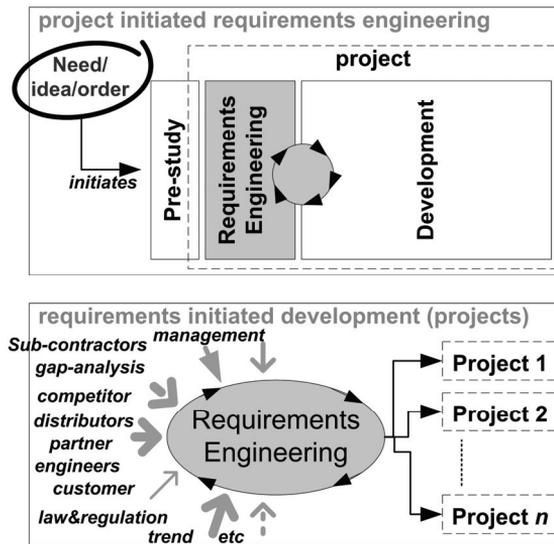
Market driven incremental product development and delivery (release) is becoming increasingly commonplace in software industry [1, 2]. Incremental product development is planned and executed with the goal of delivering an optimal subset of requirements in a certain release (version of a product that 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 [12].

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 28, 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.

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.



**Figure 28.** Requirement's role in the development process.

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 28 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 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 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 4.3).

The benefits of using 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 2 offers an overview of related work. In Section 3 the background and motivation is presented, along with how it relates to both DHR and the general industry case. Section 4 offers an introduction and exemplification of the Requirements Abstraction Model (RAM), and Section 5 presents how RAM was evaluated through static and dynamic validation. Section 6 presents the Conclusions.

## 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 [128]. 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 [12, 101, 154] when planning a release, as well as priority from the customer perspective [98]. 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, 117].

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 [20, 155].

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 where 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 be 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 [20]. The basic notion is that both the abstract (long-term overview) and the detailed (giving context and the short term-view) is important [156, 157], and the two used in combinations offers a better understanding.

The field of goal based requirements engineering (see e.g. [158-160]) focuses on the elicitation of goals that are to become requirements at some point, working from the top down.

Wiegiers [106] and Lauesen [161] 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 [136]) 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 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.

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 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 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.

### **3. RESEARCH CONTEXT – BACKGROUND AND MOTIVATION**

The development of 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 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 2). 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.

### 3.1. PROCESS ASSESSMENT RESULTS

During the process assessment conducted at DHR in total nine major improvement issues were identified and formulated (see Chapter 2 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 3 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 7). Improvement issue package 1 consisted of three issues as can be seen in Table 21<sup>8</sup>. 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 RAM.

#### 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 1 for an example.

**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 [18, 73]. This is further supported by the findings in [20, 75, 162-164], where certain issues were identified as general deficiencies in the requirements engineering process, i.e.

- 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

---

<sup>8</sup> The improvement issues “issue-id” has been altered from their original state for reasons of simplification. Title and description in Table 21 are unaltered.

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

<b>Improvement Issue Package 1</b>
<p><b>Issue-1: Abstraction level &amp; Contents of requirements</b>  <i>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.</i></p>
<p><b>Issue-2: Roles and responsibilities - RE process</b>  <i>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.</i></p>
<p><b>Issue-3: Requirements upkeep during &amp; post project</b>  <i>In order to keep the requirements up to date during and post project the requirements have to be updated as they change.</i></p>

**Table 21.** *Improvement issue package 1.*

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"

**Example 1.** *Abstraction level and level of detail.*

### 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 [144, 165]. Roles and responsibilities needed to handle requirements and in essence manage products often lay outside the focus of traditional quality assurance frameworks like CMM [77] since it is not a direct constituent of the development, rather an initiator of it.

### 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 [86] or as a binding agreement (contract) between customer and developers [161].

In the case of DHR’s continuous product development the requirements sources are many (see Figure 28) 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.

## 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 [10]. 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 RAM. A different organization has shown an interest in applying the model to their organization, due to that they experienced similar challenges as DHR.

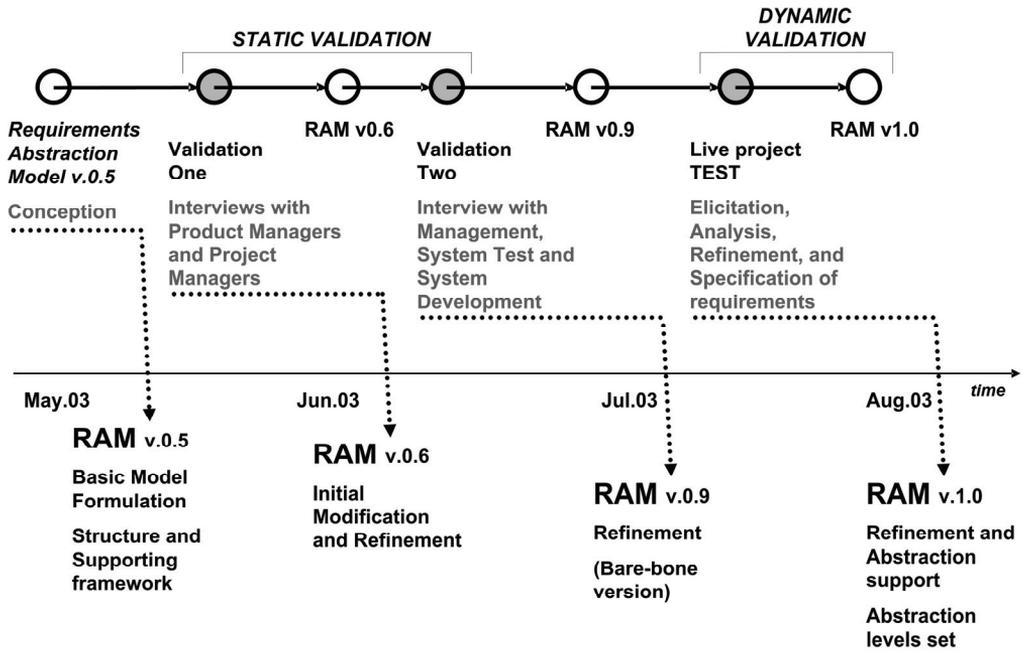
The way of working with requirements was to reflect good practices identified in state-of-the-art, i.e. adopting appropriate good practices (see e.g. [5, 18, 73]) in RAM. Moreover, aspects like repeatability and structure were seen as important, but only to the extent of offering a clear support-framework 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 RAM in a way suitable for organizations faced with certain issues, rather than tailoring the model towards one organization.

### 3.3. EVOLVEMENT OF THE REQUIREMENTS ABSTRACTION MODEL

RAM was developed in several stages as can be viewed in Figure 29.

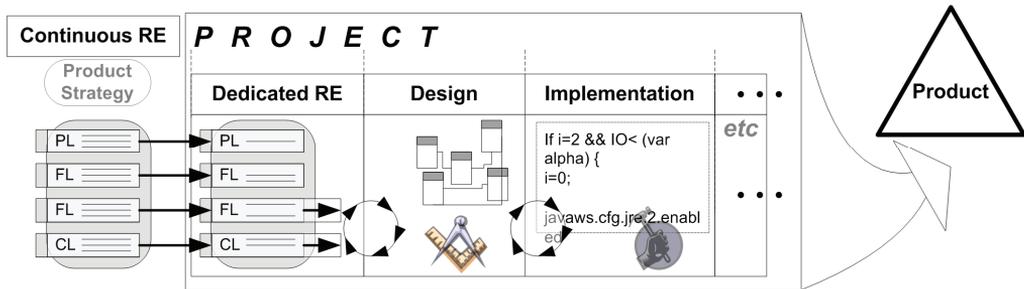
Subsequent to the initial formulation RAM went through two major validations (Validation One and Two are seen as *static validation*) before it was deemed good-enough to be tested in a live industry setting (*dynamic validation*). The *static validation* steps involved brainstorming/interview sessions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



**Figure 29.** *Evolution time-line of RAM.*

The dynamic validation consisted of using 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.

RAM version presented in this chapter is version 1.0 (see Section 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 30 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 30.** RAM overview – continuous vs. dedicated requirements engineering.

## 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 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 RAM for continuous requirements engineering. Developing relevant examples is a part of the Model Tailoring (see Section 4.5), as is ascertaining the number of abstraction levels appropriate, and attributes needed for each requirement. The version of RAM presented in this chapter (Sections 4.1 through 4.3) is based on the one developed at DHR, and it is an example of what 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 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 31. 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.

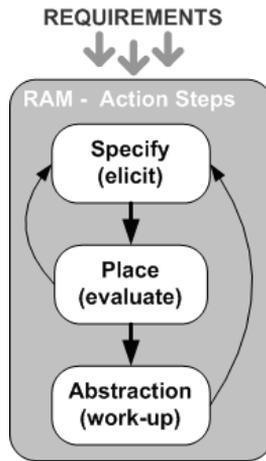


Figure 31. RAM action steps.

#### 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 2 offers examples of this attribute.

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.

Example 2. Description.

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 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 3, Requirement A and C).

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

### Example 3. Reason/Benefit /Rationale.

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 4 offers examples of this attribute.

Requirement A	Requirement B	Requirement C
<p>Using third party standards means a dependency on external organizations.</p> <p>By opening up our system to communication that we don't control we lose control over what other systems may be used with ours.</p> <p>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.</p>	<p>Communication with printers is an issue. The format for this communication has to be defined in more detail.</p>	<p>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.</p>

### Example 4. Restrictions/Risks.

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

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

### Example 5. Title.

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 4.4.

## 4.2. ACTION STEP TWO - PLACE

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



**Figure 32.** RAM abstraction levels.

---

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, [5]), thus the use of the term “requirement” can be 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 4.3), as a part of RAM work-up.

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 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.

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 36 in Section 9.

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.

#### Example 6. *Placing example requirements.*

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 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 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 under Section 9, Figure 36. 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 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 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 RAM to be consistent, i.e. placing requirements of comparable abstraction levels on the same level, and enforcing consistency over time.

### 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 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 32) 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.

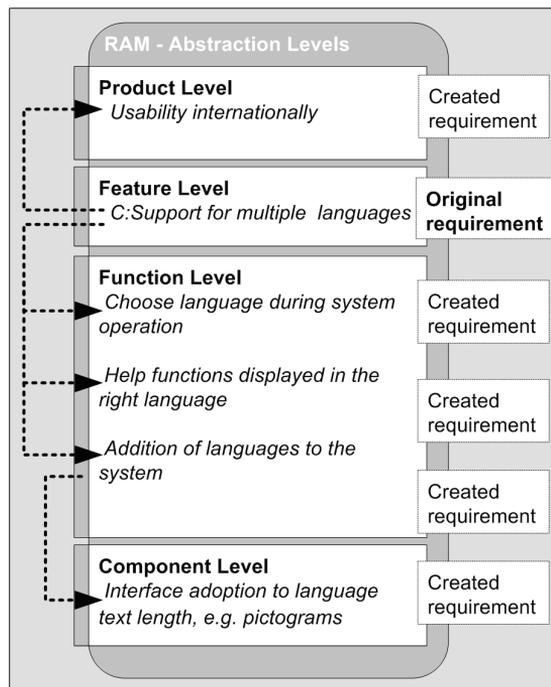
An example of R1 and R2 can be viewed in Example 7. Here the original requirement “C:Support for multiple languages” (placed on Feature Level) is abstracted to Product Level (R1) through the creation of a work-up requirement “Usability internationally”, and broken down to Functional Level (R2) where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

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 Sec-

tion 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 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 37 in Section 9) 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 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.



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

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).

#### 4.3.1. REQUIREMENTS WORK-UP - DISCUSSION

Looking further into the process of work-up using RAM, several potential issues can be identified when applying the work-up rules.

As mentioned before, no requirement may exist without having a connection to the Product Level (R1). This can imply that new work-up requirements are created on upper levels using lower level original requirements as a base. A potential issue during this activity is that an original incoming low-level requirements can give rise to (or be linked to) several requirements on a more abstract level. Two rules of thumb apply here, based on what was learned during the validations described in Section 5 First, staying true to the intention of the original requirement is a priority. Inventing new requirements from scratch outside what is needed to work-up an original requirement can result in overwork of requirements at this early stage (product management). New requirements that are created as a part of the work-up process *are* separate requirements, but not *original* requirements. This distinction can seem somewhat trivial but it helps in keeping focus on expanding the meaning of the original requirements and the work-up of them instead of using the whole process as an

excuse to create large amount of new independent requirements. Like all activities the requirements engineers, e.g. product managers, base this distinction on their expertise and judgment, and they must judge not only the impact on future development but also impact on product strategy. Secondly, a new requirement that cannot be directly attributed to the work-up process, but still invented as a result of an original requirement idea, should not be dismissed, but rather stated as a new original requirement (inventor is the source), and then subjected to work-up of its own. Work-up has to be seen as a part of the analysis, refinement and specification of requirements, as well as a way to compare incoming requirements to product strategies.

Once the work-up is completed and there are distinct links from the original requirement upwards and downwards (if applicable) a requirement in the “chain” cannot be removed without considering the whole structure of the requirements. For example, removing a requirement on Product Level would result in that all requirements linked under it must either be re-linked to another requirement on Product Level, or all requirements in the chain have to be deleted as well.

The same consideration has to be taken when removing requirements on lower levels, i.e. removing a requirement on Function Level would demand an explicit decision stating that the above linked requirement(s) can be satisfied after the removal (by other requirements linked to the Feature Level requirement in question), or the Feature Level requirement should also be removed. Looking at Example 7, this would mean that the removal of e.g. the Function Level requirement “*Addition of languages to the system*” would demand that an explicit decision is made that functionality for adding new ‘language sets’ to the system should not be incorporated as a part of the feature of supporting multiple languages in the system. This in turn has implications. How should languages be added? Are some set of them just added manually at development, and so on? The decision of when (in this case a Feature Level) a requirement is satisfied by lower level requirements is also a judgment call, but an explicit decision has to be made all the way up to Product Level regarding adequacy and satisfaction. Ultimately management (both business and technical) have to decide if a certain chain of requirements are complete and within the product strategies or not.

#### 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 4.1) for each requirement as applicable. Table 22 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 (4.4.1 and 4.4.2) elaborate regarding the attributes and their rationale.

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 RAM can have different states giving information of the status of the requirement, see Figure 33 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)

**Table 22.** RAM requirement’s attributes. (For attribute 1 to 4, see Section 4.1).

#### 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 RAM action steps (see Figure 31) 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.

#### 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 33 offers an overview of the different states. Looking at Figure 33 states *A*, *B* and *C* are a part of RAM action steps (see Figure 31) 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 23 where each separate step and sub-step is described. Observe that states *D* (*dependent on pri-*

oritization) and *F* (dependent on packaging into release packages) are addressed in future work (see Section 7).

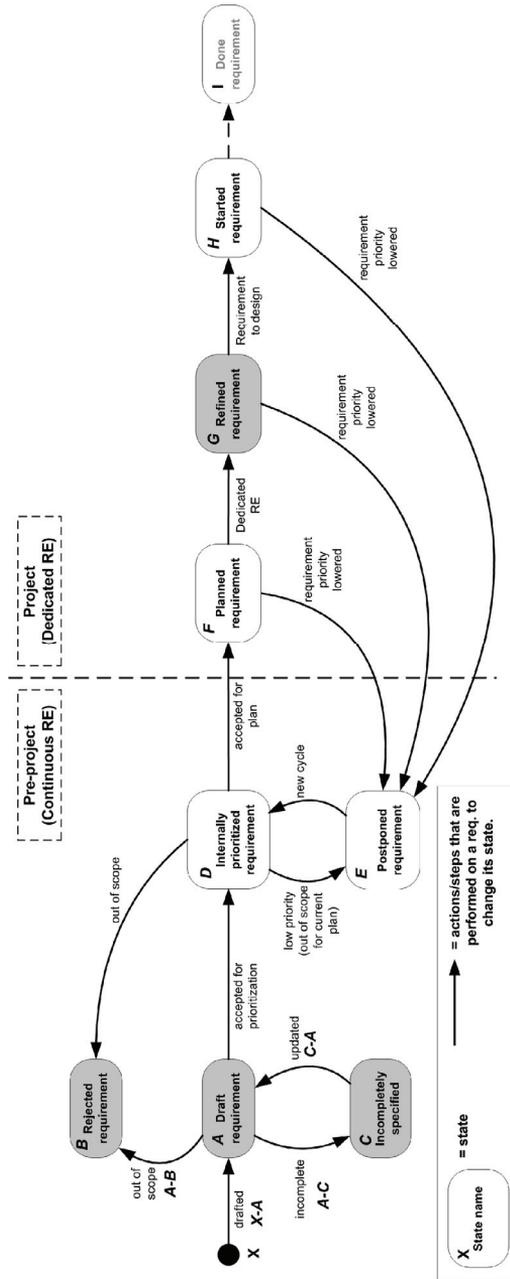


Figure 33. Requirement's states in RAM.

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 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 RAM.

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).	<p>X-A-1: Specify attribute 1 to 4 (Specify).</p> <p>X-A-2: Determine on which level the original requirement is on (Place).</p> <p>X-A-3: Specify all attributes on relevant level.</p> <p>X-A-4: Abstract and/or Breakdown the original requirement according to work-up rule <b>R1</b> and <b>R2</b> (work-up).</p> <p>X-A-5: Validate requirement against Requirement Owner and Requirement Source.</p>
A-B	<p>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.</p> <p>As a requirement is rejected, directly related requirements (work-up requirements) on adjacent abstraction levels either are removed as well or re-linked (enforcing <b>R1</b> and <b>R2</b>).</p>	<p>A-B-1: Compare requirement (directly or indirectly) to product strategies.</p> <p>If the req. is not in line with strategy it is rejected.</p> <p>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 <b>R1</b> and <b>R2</b> to be enforced.</p>
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.	

**Table 23.** RAM state steps (see Figure 33).

## 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. RAM action steps described in Sections 4.1, 4.2, and 4.3 follow this ambition, and the use of attributes (Section 4.4) enables a requirements oriented (not document oriented) way of working.

This does not mean that 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 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 RAM.

Below an overview of the tailoring aspects is offered.

- **Abstraction Levels** (and usage of them) are a fundamental part of 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 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 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 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 RAM are considered outside the scope of this chapter. However, it should be noted that a tailoring effort of RAM is underway at present, which is a prelude to the test and implementation of RAM in a second organization faced with a situation similar to the one identified at DHR.

## 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 requirements engineering environment, and a structured framework for how to work-up these requirements. The main steps of 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 4.1 and 4.4).
2. **Place** the requirement (see Section 4.2) on an appropriate abstraction level (using product relevant examples, see Section 4.5, and the how-to guide in Figure 36 in Section 9).
3. **Work-up** (see Section 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 4.5, and the how-to guide in Figure 37 in Section 9).

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 4.4).

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

## 5. RAM - VALIDATION

This section offers a presentation of the evolutionary development of RAM as it was validated in industry against an organization faced with the improvement issues described in Section 3.

The validation is divided into two main parts, static and dynamic validation (see Figure 29). The static validation consists of reviews and walk-throughs of RAM, and the dynamic validation consists of a live industry requirements engineering effort where the model was put through its phases.

### 5.1. STATIC VALIDATION

As RAM was developed it was considered important to get input from all stakeholders involved with product management and requirements engineering, as well as the ones using requirements as input to their work.

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 RAM (see Section 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 Section 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 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 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 [131] 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 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.

### 5.1.1. **STATIC VALIDATION IMPACT AND LESSONS LEARNED**

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

The initial designs of 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 in Section 9) enhanced usability of the model substantially, especially in comparison to just offering e.g. a list of rules describing RAM.

The structure and rules of 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 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 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 RAM *general process* and gave rise to a division of RAM requirements engineering into two main parts: a *continuous* (pre-project) part and *dedicated* (during project) part. Figure 30 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 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 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).

## 5.2. DYNAMIC VALIDATION

As a candidate model (RAM v.0.9, see Figure 29) 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 to validate its components (e.g. attributes) and if the model was scalable to a real development situation. This involved validating several aspects of 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 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 RAM easy and intuitive enough to use practically?

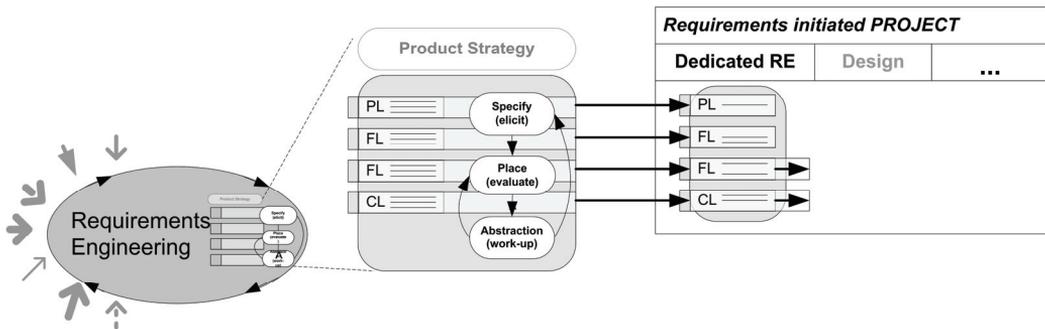
The main factor limiting the effectiveness of the dynamic validation were the difficulties of actually performing continuous requirements engineering over a long time-period, as the dynamic validation was limited in time and scope. This meant that the “continuous” part was not really tested, i.e. only 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 RAM and described in e.g. Figure 28) the dynamic validation would have to be performed over a much longer period eliciting/catching requirements continuously, and initiating several development projects as needed (based on the requirements).

However, the consensus at DHR was that a limited, but nevertheless *real*, test of RAM would produce a clear indication as to the applicability of the model to solve the problems described in Section 4, 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 4.4). Two persons in cooperation conducted the actual requirements engineering work performed with 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 RAM in a requirements engineering situation, and by doing so involve other parts of the organization described by 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 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 34 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 RAM repository for one development project to be initiated. In a continuous requirements situation the requirements engineering with 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 5.1.1, Static Validation Two).



**Figure 34.** *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).

### 5.2.1. DYNAMIC VALIDATION - LESSONS LEARNED

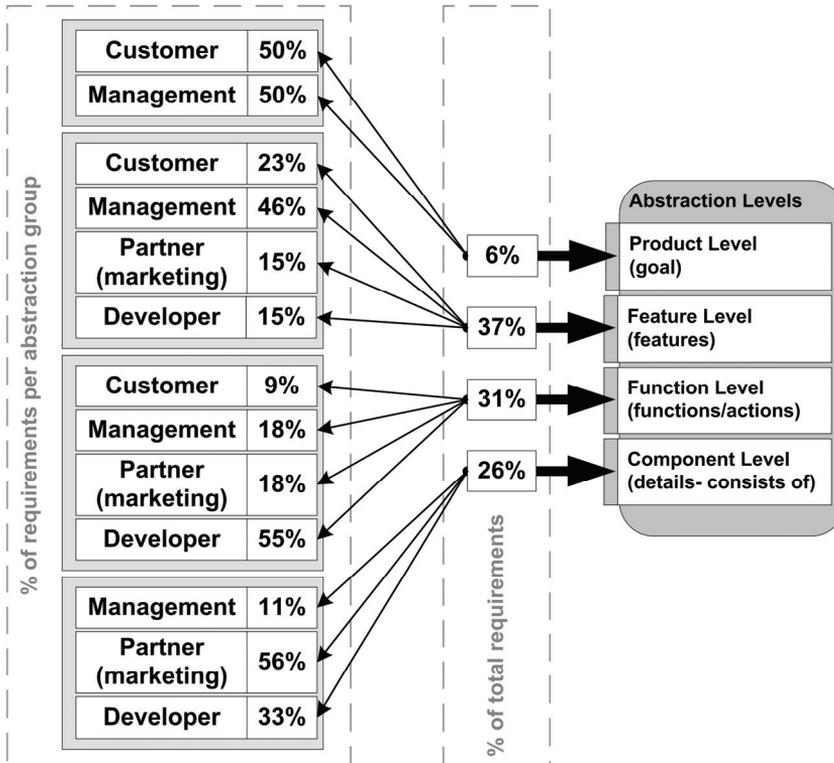
Below each of the questions posed in Section 5.2 (i.e. question 1 to 3 with sub-questions) is discussed based on the experiences from the dynamic validation of RAM.

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 35. In the center of Figure 35 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 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 re-

quirements 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.



**Figure 35.** Requirements distribution over sources and abstraction levels.

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 perceived as more hands-on and not prone to abstract non-verifiable statements. The Partner group consisted of requirements where 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 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 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 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 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 where 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 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 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 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 RAM, but rather the infrastructure needed to support 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 RAM, the overall usability of the model was considered more than adequate by the Product Manager using RAM, as were the results produced by RAM.

### **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 pre-

sented in Wohlin et al. [129]. The validity threats considered are conclusion, internal, external and construct validity threats respectively.

### **5.3.1.1 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.

### **5.3.1.2 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.

### **5.3.1.3 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.

#### 5.3.1.4 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.

## 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.

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 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, 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 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 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, 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 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 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. 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.

## **7. FUTURE WORK – RAM VALIDATION AND EVOLUTION**

RAM v.1.0, as described in this chapter, was aimed at assuring that requirements be specified on multiple abstraction levels in a repeatable and homogenous way, enabling e.g. product managers in early requirements engineering efforts. This was considered a prerequisite for subsequent activities, i.e. requirements estimation, risk analysis, prioritization and packaging of requirements taking e.g. coupling between requirements into consideration. The next phase that will be undertaken in the validation and evolution of RAM can be described as a two-step process. First, a controlled empirical evaluation in a lab environment will be performed. This evaluation will be designed to further test the concepts behind RAM in an experimental setting, prior to industry trials. The goal for this evaluation will be to give important feedback regarding the usability and usefulness of RAM without spending valuable industry resources, but nonetheless enabling further improvements on the model's concepts, identification of potential weaknesses, and collection metrics regarding the usage of RAM. All information gathered during this lab evaluation will be used as input to the next step, the tailoring and large scale piloting of RAM in industry.

The present plan is for RAM to be tailored to, and subsequently piloted in, two organizations, DanaherMotion Särö AB and ABB Automation Technology Products. Data, both qualitative and quantitative, gathered during the pilots will act as decision support material for further model improvements, but also as input to the organizations when considering the full scale adoption of a tailored version of RAM in their product planning and development process.

In addition, the natural evolution of RAM will continue, incorporating explicit support for requirements estimation, risk analysis, prioritization and packaging of requirements into development instances. The work name for this endeavor is RAM v.2.0. Requirements engineering closer to and within projects will be incorporated. The goal will be to create one usable and flexible lightweight model that cover requirements engineering activities performed during product planning and management activities, but also within and post projects, thus effectively addressing most of the issues identified in industry (see e.g. Improvement Package Two and Three in Chapter 3).

## **8. ACKNOWLEDGEMENTS**

The work and results presented in this chapter were done and produced in close and fruitful cooperation between research and industry. The authors would like to thank everyone involved during the SPI work at DHR, and especially the members of the SPI work team for their commitment and tireless work.

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>).



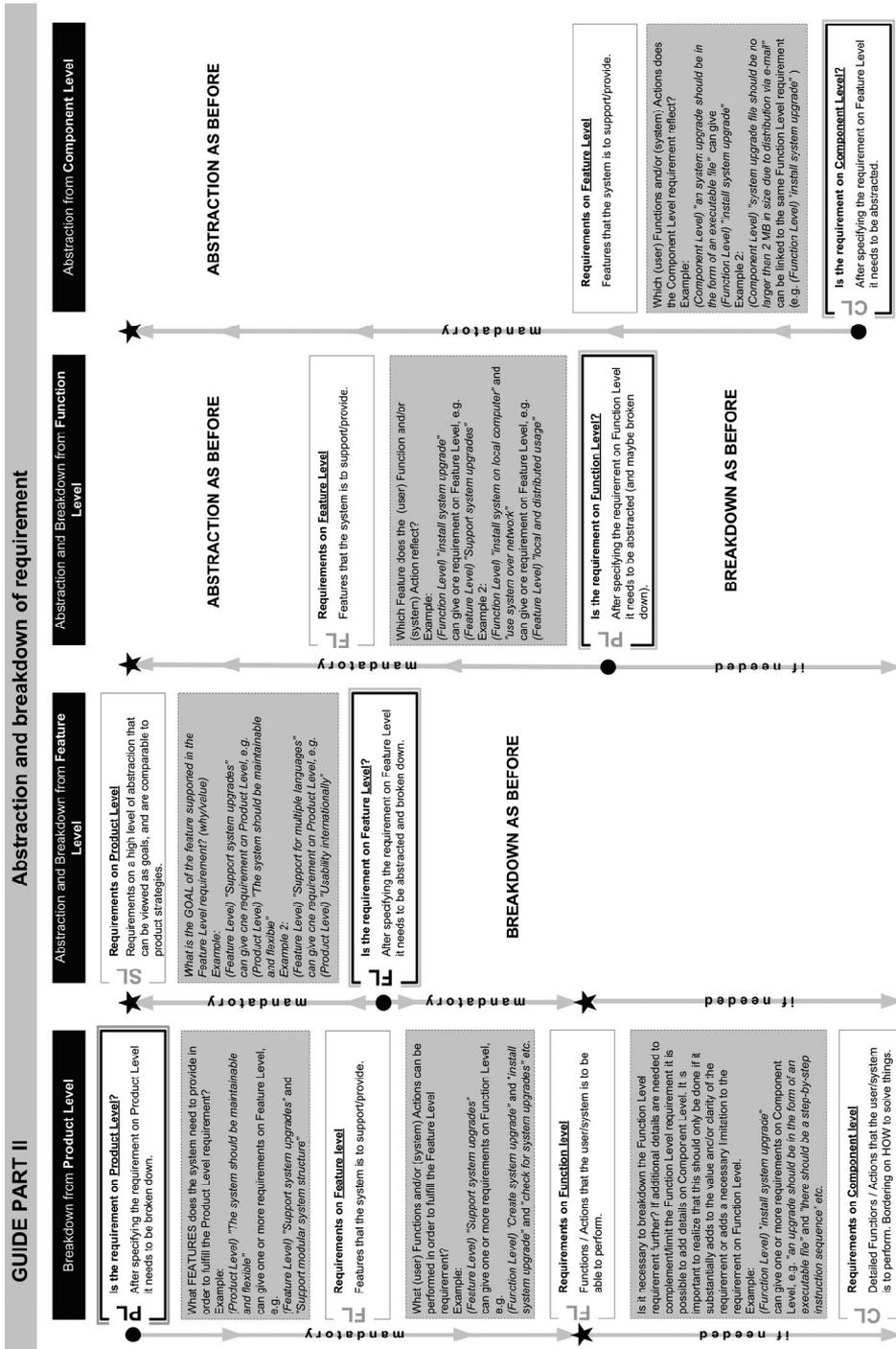


Figure 37. RAM how-to guide part II – Abstraction and breakdown of requirement.



# Chapter 5

---

## A Controlled Empirical Evaluation of a Requirements Abstraction Model

*Tony Gorschek and Mikael Svahnberg*

Submitted to Information and Software Technology, 2005.

### **Abstract**

Requirement engineers in industry are faced with the complexity of handling large amounts of requirements as development moves from the traditional bespoke projects towards market-driven development. There is a need for usable and useful models that recognizes this reality and supports the engineers in the continuous effort of choosing which requirements to accept and which to dismiss off hand using the goals and product strategies put forward by management. This chapter evaluates such a model -built based on needs identified in industry. The evaluation's primary goal is to test the model's usability and usefulness in a lab environment prior to large scale industry piloting, and is a part of a large technology transfer effort. The results indicate that the model is indeed both useful and usable and ready for industry trials.

## 1. INTRODUCTION

Requirements Engineering (RE) more and more transcends projects and project boundaries as market-driven product development is becoming increasingly commonplace in software industry [1-3]. Central activities in RE are performed pre-project as a part of e.g. the product management activities since the requirements flow is continuous and not limited to a specific development instance [6, 7].

In this environment requirements come from several sources both internal (e.g., developers, marketing, sales, support personnel, bug reports etc) and external (e.g., users, customers and competitors, often gathered via surveys, interviews, focus groups, competitor analysis etc) [8, 10, 11]. Large volumes of requirements from multiple sources risk overloading companies unless they can handle incoming requirements in a structured way, dismissing some, and refining some prior to allocating them to a development instance (e.g. one or several projects) [20]. In addition to the volume, the requirements themselves are of varying quality, state of refinement, and level of abstraction. In traditional bespoke development (customer-developer) [3] a requirements engineer can actively elicit requirements and thus hope to control or at least substantially influence these aspects, in a market-driven situation this is seldom the case. Most requirements are already stated in one way or another when they reach the requirements engineer (e.g. a product manager). The knowledge and experience of the developing organization (of which the requirements engineer in this case is instrumental) is central to making sense of the requirements as they are processed [21].

Many requirements engineering best-practices, frameworks, and also tools are adapted to suit a bespoke environment with traditional project focused customer-developer relationships. There is a need for the development and evaluation of RE practices and models that support professionals working with product planning and development (e.g. product managers) in a market-driven environment.

This chapter presents an evaluation of some key aspects of such a model, the Requirements Abstraction Model (RAM). RAM was developed as a response to needs identified in industry during process improvement conducted in cooperation with Danaher Motion Särö AB and ABB Automation Technology Products [134, 166]. The Requirements Abstraction Model (RAM) is designed towards a product perspective, supporting a continuous requirement engineering effort, aimed at taking requirements of multiple types (abstraction levels) as input, and offer a structure for the

work-up of these requirements and the handling of large quantities of requirements (see Section 2.1). The main purpose of the evaluation is to assess the usefulness and usability of RAM in a controlled environment prior to widespread industry piloting. This is achieved by evaluating the model in two parts. Firstly, by using the model engineers should be able to get an overview of an entire product by studying relatively few top level requirements. Secondly, engineers should be able to utilize the concepts of requirements abstraction as they decide if new incoming requirements are in-line with product goals and strategies (this is further detailed in Section 2.2).

Prior to describing the study itself and the results a short introduction to RAM is given in order to increase the understanding of the rationale behind the evaluation. Therefore Section 2 gives an overview of RAM, as well as the problem statement used in this evaluation. Section 3 details the planning, context, and overall design of the evaluation along with a subsection detailing validity issues. The research questions are formally stated in Section 4, and the operation (execution) of the evaluation is detailed in Section 5. Section 6 presents the results and analysis with the help of descriptive statistics, and Section 7 is devoted to revisiting the research questions and discussing them explicitly based on the results and the analysis. Last, Section 8 presents the conclusions drawn, and some plans for further work.

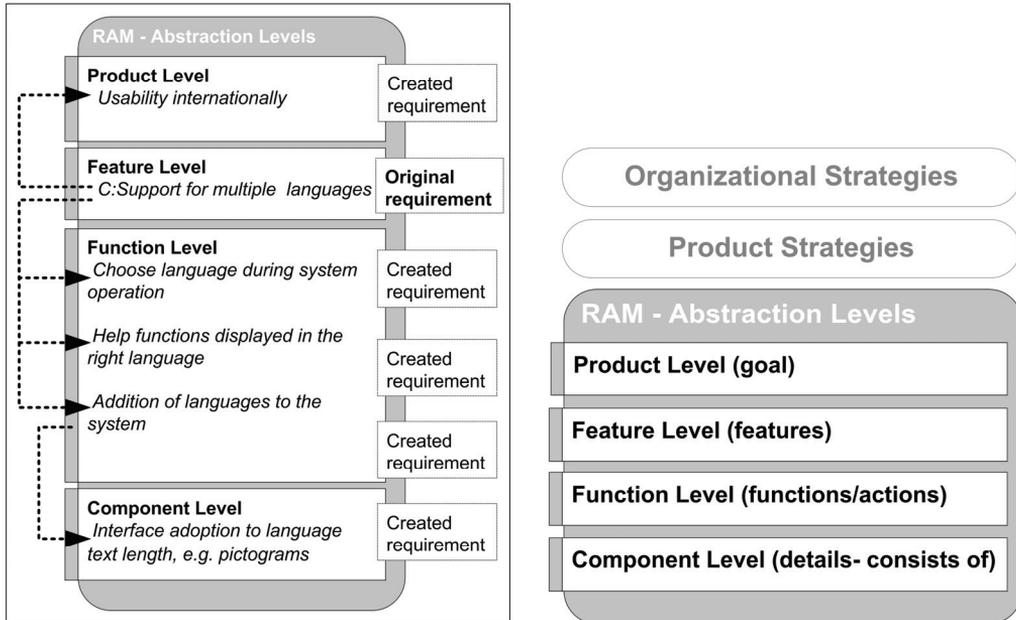
## **2. BACKGROUND AND RELATED WORK**

This section gives a short description of RAM and related work regarding the concepts of abstraction in relation to RE work. For reasons of brevity details not central for the evaluation presented in this chapter have been left out. For details, please see Chapter 4.

### **2.1. THE REQUIREMENTS ABSTRACTION MODEL**

RAM is a hierarchical requirements abstraction model and a method for working with this model based on the concept that requirements come on several levels of abstraction. Instead of flattening all requirements to one abstraction level RAM uses the varying abstractions of requirements, and orders the requirements hierarchically according to abstraction level. Looking at Figure 38, four abstraction levels can be seen, i.e. *Product Level*, *Feature Level*, *Function Level*, and *Component Level*. The Product Level is the most abstract level and requirements here are considered abstract enough to be comparable to the product strategies, and indirectly to the organizational strategies. In the context of RAM, product strategies are e.g. rules,

long and short-term goals, road-maps, and visions pertaining to a product specified by management. As you go down to Feature and Product Level the requirements become concrete enough to be used for estimations and as input to development.



**Figure 38.** RAM abstraction levels and example of Work-up.

Briefly, the process followed with RAM is that when requirements arrive they are placed and specified on an appropriate level by comparing with the existing requirements base (i.e. example-driven). This means that the present mass of requirements is used as decision support material for the inclusion, exclusion, and placement of new requirements. Following this all requirements go through *work-up*. Work-up entails abstracting all requirements up to Product Level and breaking down all requirements to Function Level. This is done by creating new requirements in levels above and below and linking them to the original requirement already present. Figure 38 gives an example of this. Here the original requirement “C:Support for multiple languages” (placed on Feature Level) is abstracted to Product Level through the creation of a new work-up requirement “Usability internationally”, and broken down to Functional Level where three new work-up requirements are created as a part of the breakdown. In some cases requirements already present can be used for abstraction. As an example, in Figure 38, if a new requirement comes in stating

“D:Support BTU units” it could be placed on Feature Level and linked directly to “Usability internationally” as BTU units are used in e.g. the US in addition to SI units (metric system). In this case no new requirement has to be created in Product Level and the new requirement can be linked to an already existing one.

During the work-up process the original requirement is compared to product strategies, offering decision support regarding if the requirement should be specified, refined and kept in the repository, or if the requirement should be dismissed as it e.g. goes against or is not supported by the product strategies. For example, let us assume that “*Usability internationally*” was not accepted but rather the company wanted to limit the product market to the European market. In this case “*Usability internationally*” would be “*Usability Europe*”, and the new requirement “*D:Support BTU units*” would be dismissed as only SI units are used in Europe. In other words, the product level ultimately decides whether or not to include requirements since all requirements are directly linked to this level. If the new requirement is not supported, this may indicate that a new product level requirement needs to be created.

The break-down of the requirements stops on Function Level where the requirements are not perfect, but good enough to be used as decision support for estimation and risk analysis and as input to project(s) for realization. The Component Level is not mandatory in the model, but present due to the fact that requirements in some instances were delivered in a very detailed form, and these requirements also needed to be handled (specified and abstracted to assure that they are in line with the overall goals and strategies). In the example presented in Figure 38 the Component Level requirement acts as extra information (on a detailed technical level) associated with the requirement, for example the interface has to be adapted to pictograms for e.g. the Chinese language. In this case the Component Level sets a restriction that will accompany the requirements into development.

The features of using RAM 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 resources to work on and 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(s)). 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 [5]).

(III) Work-up of a requirement means that additional requirements may have to be created in order to get a connection to the upper most level, e.g. if a incoming new requirement is placed on Function Level and no appropriate requirement exist on the Feature Level, one has to be created. This Feature Level requirement in turn needs to be linked to the Product Level. This ensures that you can follow a requirement through abstraction levels to assure that there is an explicit connection upwards to product strategies. In the same way every requirement is broken down to a level good-enough to serve as a basis for project initiation (Function Level). Requirements on a certain abstraction level are homogenous enough with regards to abstraction to be comparable with each other, 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. Managers can for example study the most abstract levels and get a fast overview of the system, while developers can choose another more detailed view, but still have an explicit connection to the overall goals of the product as detailed requirements are connected upwards through the levels.

As an initial step in preparing RAM for industry use some of the underlying concepts of the model needed to be evaluated in a laboratory setting, this is elaborated in Section 2.2 below.

## **2.2. PROBLEM STATEMENT**

The usefulness and the usability of RAM depend on several concepts that need to be assessed, the ones studied in this controlled evaluation are summarized below in two main parts.

(Part I) Top level (Product Level) requirements should be used to ascertain if a new incoming requirement is in line with product strategies or not (decision support for acceptance or early dismissal of a new requirement). Product Level requirements should therefore convey an overview (“big-picture”) of the product (i.e. Product Level requirements should summarize and represent a considerable amount of requirements on lower levels).

In Part I we evaluate if abstract Product Level requirements are general enough to summarize several requirements on

lower levels, but still concrete enough so that it is possible to determine if they represent the product or not. If this is the case, Product Level requirements can be used to convey the aforementioned big-picture product overview.

(Part II)

In Part II we evaluate if it is possible in practice, with relatively few abstract top-level requirements, to understand what features and attributes a product should and should not consist of. As new requirements come in it should be possible to:

- dismiss or accept a particular requirement using product strategies (explicitly expressed as Product Level requirements in this evaluation), and
- place the requirement on an appropriate level of abstraction using already present requirements on each level as indication of where the new requirement should be placed.

The usability of the model depends on these two points and thus the utilization of requirements abstraction. Requirements abstraction is the ability to use relatively few abstract Product Level requirements to derive decisions across the hierarchy of abstraction with regards to including or excluding new requirements from a product. In addition, the ability to place new requirements on an appropriate level (using already present requirements as decision support) keeps the levels homogenous, i.e. requirements on each level are comparable with other requirements on the same abstraction level. The ability to place requirements in the model also preserves model consistency over time.

The controlled evaluation of RAM presented in this chapter is centered on the aspects described in Part I and Part II, and they are stated formally as research questions in Section 4.

## **2.3. RELATED WORK**

The realization that requirements are often on different levels of abstraction and in varying stages of refinement has been recognized by others in both industry and research [20, 155], and by us during process assessment and improvement efforts conducted at both Danaher Motion Särö AB and ABB Automation Technology Products [21].

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 [20]. The basic notion is that both the abstract (long-term overview) and the detailed (giving context and the short term-view) is important [156, 157], and the two used in combinations offer a better understanding.

The field of goal based requirements engineering (see e.g. [158-160]) focuses on the elicitation of goals that are to become requirements at some point, working from the top down. For example, Wiegers [106] and Lauesen [161] describe that requirements can be of different types pertaining to what they should be used for, not entirely unlike the industry view of dividing requirements of different types into separate documents, from goal (abstract natural language formulations [136]) to technical design like specifications. The focus is on that requirements are specified on different abstraction levels depending on usage, e.g. project type.

RAM is designed to operate in a market driven product centered development environment. The volume of requirements that needs to be handled makes initial screening important in order to decrease the risk of overloading in the evaluation and realization process [20]. RAM is intended to utilize abstraction, in a somewhat similar way to what e.g. Wiegers and Lauesen describe, gathering all requirements for a product in one repository where requirements are linked between levels of abstraction, making it possible for fast screening and dismissal of requirements not in line with product strategies.

The requirements work is performed by the product management organization, using RAM independently in a continuous RE effort designed for long term product development. Projects that realize requirements is a result of the work conducted, i.e. RE activities with RAM are not project initiated, but can rather give rise to any number and type of projects over time as needed.

This entails handling a large and steady stream of requirements continuously before any commitment or plan exists for realization of them. Early screening of requirements (comparing to product strategies) is only one part, another is the need to select requirements for realization in projects. By having requirements on different homogenous abstraction levels comparisons, prioritization, and planning on respective level is made possible, offering support for product management to perform early screening and selection of requirements. This differs from e.g. Wiegers and Lauesen, as they focus more towards requirements engineering in relation to projects.

### **3. PLANNING OF THE EVALUATION**

In this section we describe the overall parameters for the evaluation. These parameters are common for all the different parts of the study and the different research questions, further described in Section 4.

#### **3.1. CONTEXT**

The study is conducted in an academic setting, with the help of graduate students (normally their fourth year) and some undergraduate students at Blekinge Institute of Technology. The evaluation is conducted as a mandatory although non-graded (it was strongly emphasized during the initial presentation that we would not use the results to grade the participants) exercise in a requirements engineering course. All Software Engineering Master's students are required to take this course. Participation was mandatory because despite the ethical issues of forcing subjects to participate in a study, it was decided that the evaluation had several pedagogical benefits in the course. The students were instead given the option to decide whether their results should be allowed in the study. All students gave their consent to this.

The intended target for RAM, however, is product managers with several years of experience in a specific domain and of a specific product. In the study, the subjects have no training on RAM, possess limited domain knowledge, are under time pressure, and have not seen the requirements before. There is thus a considerable gap between the intended target group and the sample used in this study. The subjects in our study can be expected to adapt a more surface oriented approach to the problem than product managers. We argue that this works to our advantage, since any effects that we measure are likely to stem from the instrumentation and the use of RAM, rather than previous experiences of the participants in the study. If RAM proves to be usable in the study, it would indicate that it is able to decrease the dependency on individual persons' experience, knowledge, and methodology.

#### **3.2. SUBJECTS**

As mentioned, the participants in the evaluation are software engineering students at Blekinge Institute of Technology. This group consists mostly of graduate students as well as some undergraduate students. A proportion (8 out of the in total 21 subjects) are international students, the rest are Swedish of which most have performed their undergraduate studies at the software engineering program at the university. The subjects have an av-

erage of 0.83 years of industrial experience, and consider themselves between novices and of moderate experience regarding requirements engineering. The study was run at the end of a 7.5 ECTS<sup>9</sup> merits master's course in requirements engineering. The Swedish subjects have done a large team project (15 persons, 16000 person hours), and for most of them this is their only RE experience. In addition, the subjects consider themselves between "moderate" to "skilled" in English.

### **3.3. DESIGN**

Prior to the controlled evaluation presented in this chapter two test rounds were performed to refine the evaluation design. An initial version of the study was constructed and evaluated with the help of colleagues at Blekinge Institute of Technology. Based on the experience from this test round, the study package was revised. Specifically, it was reduced in size and where the participants were asked to motivate their answers the questionnaires were changed to provide a closed set of answers, to facilitate analysis. This revised study package was then evaluated a second time with the help of a network of Swedish requirements engineering researchers, SiREN<sup>10</sup>.

The evaluation consists of four parts, all run consecutively without any breaks between. The first part is a preparatory lecture where RAM and the concept of a hierarchical requirements specification are introduced, together with a presentation of the study and the research instruments. The remaining three parts of the evaluation, further described in Section 4, consists of materials and questionnaires to answer the research questions. All participants are given the same instrumentation, the same treatment, and the same questionnaires to answer. The study is estimated, after the pilot tests, to take a maximum of three hours if no breaks are allowed.

### **3.4. INSTRUMENTATION**

The instrumentation consists of three different questionnaires, further described in Section 4, and a requirements specification. This requirements specification details an intranet solution for a course management system. The system contains features such as news (information) in courses, file

---

<sup>9</sup> European Credit Transfer System. 7.5 ETCS equals 5 weeks of full time studies.

<sup>10</sup> <http://www.siren.lth.se/>.

archives for courses, course calendars, course discussion forums, and management of course participants. The requirements specification is based on a 16000 person-hour students' project. From this project different features have been added and removed to generate a requirements specification of adequate size. The requirements are constructed to be of moderate to good quality based on the authors' experience as software engineers and requirements engineers as well as requirements engineering researchers, and this was confirmed through the two pilot studies. To ensure that the requirements specification represents a complete system of adequate size (we estimate that it would take 15000 person-hours or more to develop the system) and still be small enough to be usable in the evaluation (i.e. 120 requirements in total) many of the more detailed requirements have been omitted.

The requirements specification is structured hierarchically into four levels, denoted Product (containing 9.6% of the requirements), Feature (19.1%), Function (59.6%), and Component level (11.7%) (the levels are further discussed in Section 2.1), and each requirement contains the fields *Id* (a unique number where also the abstraction level can be discerned), *Title* (title of the requirement), *Description* (the actual requirement), *Rationale* (an explanation of the need for the requirement), *Restrictions* (any risks or restrictions on the requirement), *Relations to* (id and title of requirements that this requirement is related to), and *Relations from* (id and title of requirements that link to this requirement). The requirements are listed in tabular form as the example in Figure 39 illustrates. In this figure we see a selection of requirements on all levels, starting with two product level requirements, two feature level requirements, four function level requirements, and two component level requirements. Each requirement contains the aforementioned fields.

Level	ReqID	Title	Description	Rationale	Restrictions	Relations To	Relations From
Product	1 013	Product Interface	All access to the system must take place via the systems own user interface, i.e. access from third party products is not allowed.	Control look and feel. Avoid security issues and compatibility problems.		1 040 Manage and Conduct a Course	2 084 Web-based User Interface
Product	1 040	Manage and Conduct a Course	The product shall provide functionality that supports a teacher (course administrator) to conduct a course, manage the course, and supports the participants in following course progress, course news, and accessing information related to the course as well as exchanging course related information	This is the core of the product.			1 013 Product Interface  2 032 Course Start Page 2 035 Course News 2 039 Course File Archive 2 067 Course Administration 2 107 Discussion Forum 4 022 Link to Course
Feature	2 032	Course Start Page	Each course shall have a course start page.	Information overview		1 117 Distribute Information about Courses 1 040 Manage and Conduct a Course	3 065 Course Start Page Contents
Feature	2 035	Course News	It shall be possible to attach news items to a course.	Keep the students informed about the course.		1 117 Distribute Information about Courses 1 040 Manage and Conduct a Course	3 038 Remove Course News  3 129 Access to View Course News 3 128 Access to Add, Edit, and Remove Course News Items 3 127 Edit Course News 3 065 Course Start Page Contents 3 037 View Course News 3 036 Add Course News 3 069 Course Administration Contents
Function	3 036	Add Course News	It shall be possible to add news items to a course.			2 035 Course News	3 128 Access to Add, Edit, and Remove Course News Items
Function	3 041	Add Files to Course File Archive	It shall be possible to add files to the file archive.		What if a file already exists with the same name?	2 154 Support Swedish Alphabet 2 039 Course File Archive	3 130 Access to change in Course File Archive
Function	3 042	Remove Files from Course File Archive	It shall be possible to remove files from the course file archive.			2 039 Course File Archive	3 130 Access to change in Course File Archive
Function	3 043	Download files from Course File Archive	It shall be possible to download files from the Course File Archive.			2 039 Course File Archive	3 131 Access to use Course File Archive
Component	4 018	Successful Login	When a user has successfully logged in, the product shall display the user's personal start page.	After user authentication, the user wants to start working.		2 020 Personal Start Page  3 012 Login	
Component	4 019	First login	The first time a user logs into the system the user's personal profile shall be shown.	The user may provide additional information about him/herself.	The user decides not to provide any additional information, which means that the user has an incomplete profile.	2 047 Personal Profile  3 012 Login	

Figure 39. Example of requirements representation.

## **3.5. VALIDITY EVALUATION**

The validity of the study is divided into four areas: conclusion validity, internal validity, construct validity, and external validity. Below we discuss these in further detail. More information about possible threats to studies can be found e.g. in Wohlin et al. 2000 [129] and Robson 2002 [131].

### **3.5.1. CONCLUSION VALIDITY**

To ensure that the research instruments, including the posed questions, are of a good quality, two separate pilot-tests with the material have been executed before the “live” round. Moreover, all participants received the same introductory lecture, were given the same material in the same order, and received the same additional instructions. It is thus unlikely that the instrumentation and the implementation of the treatment influence the results unduly. That being said, since we use the answers of human subjects as measures the gathered measures are of course not 100% repeatable.

### **3.5.2. INTERNAL VALIDITY**

The participants may mature during the study. Specifically, this concerns their understanding of the requirements specification and how to read it. This means that the accuracy of the participants’ answers may improve as they answer more questions. In effect, as the study progresses the participants’ familiarity with the domain and the requirements specification becomes more and more alike the intended target group.

The instrumentation (i.e. the requirements specification described in Section 3.4 and the related artifacts described in Section 4) is designed to be of moderate to good quality to mimic the situation for the intended target group and should not influence the participants unduly. Moreover, to ensure that the participants answer in accordance with the research instruments and not according to their understanding of the domain of the system, we have intentionally left out some specific features that commonly occur in the domain, such as using standard USENET news clients, import and export users and data to other systems, and synchronize calendars with handheld devices. All of the product statements and candidate requirements used in part I and II of the study (further detailed in Section 4) are common in the domain, and the course management systems that the students are in daily contact with have features equivalent to the product statements and candidate requirements. This ensures that the students provide a motivation for their answers based on RAM and the

provided requirements specification and not according to their domain understanding or their opinion.

Since the study is conducted as a mandatory exercise in a mandatory course, the motivation of the subjects may vary considerably. This may actually speak in favor of the study, since there may be less incentive for the participants to perform well.

### 3.5.3. **CONSTRUCT VALIDITY**

Since we are using a single requirements specification in this study, there is a risk of mono-operation bias, i.e. there is a risk that our choice of system and domain influence the results more than the method for working with the requirements specification that we wish to evaluate. As mentioned above, by intentionally leaving out commonly occurring features in the domain from the requirements specification we at least ensure that the participants follow the prescribed method and that we can detect if they do not.

We make no secret of the hypotheses in this study. Since there is only one treatment the only way the participants can influence the result if they guess the hypothesis is by deliberately answering wrong. This increases the risk that the evaluated methodology is deemed to be less useful than it is, which means that we may err on the cautious side when analyzing the results.

To reduce the risk of evaluation apprehension among the participants (i.e. the participants think it is they that are being evaluated and not RAM), we strongly emphasized that we were not evaluating the individual participants.

To avoid the risk that the wording of the questions may provide hints as to what the answer is, the questions in the questionnaires are formulated in a standard way. This is further described in Section 4.

### 3.5.4. **EXTERNAL VALIDITY**

To ensure the external validity and the ability to generalize the results, we use a requirements specification for a relatively large system in a fairly mature domain.

As discussed in Section 3.1, the knowledge and experience of the participants is less than that of the target audience (e.g. product managers in industry). To reduce this gap, we use a system from a domain that is familiar to the subjects. The target audience would also have training in using RAM and specifying requirements according to RAM, which the study participants do not have. In this study, we only focus on a few key aspects

of RAM, and thus a complete training of RAM is not necessary. We also argue that more experience, knowledge, and training should not negatively impact the ability to work with RAM. In other words, any positive effects detectable in this study should be transferable to the target audience.

In the same way, it is our belief that a hierarchical requirements specification works best when used in a software tool. In this study we use paper printouts which may impact the readability and the ease by which the participants may access the information. Hence, any positive effects are also here transferable to the target audience and the target environment.

## 4. RESEARCH QUESTIONS

The study is divided into three parts, with research questions attached to each part. Part I and II directly correspond to the problem statement posed in Section 2.2 and concern the evaluation of RAM. Part III is a post-test designed to gather information about the subjects to ascertain if and how their background influences the results of the study.

### 4.1. PART I

#### (Research Question 1)

*Given a requirement specification ordered hierarchically according to the level of abstraction of the requirements, to what extent do the top-level (most abstract requirements) connect to the rest of the requirements?*

#### (Test)

This research question is evaluated by:

- a) Removing the most abstract requirements (Product Level) from the requirements specification.
- b) Presenting these requirements as statements about the product.
- c) Asking the subjects to determine whether each statement is supported by the remaining requirements specification, and in particular which of the requirements on the lower level in the requirements specification that supports the statement.

#### (Collected Metrics)

The questionnaire for this part of the study collects metrics on:

- 1) Whether the participants consider the statement supported or not by the requirements specification as a defining factor of the system.
- 2) Which feature requirements (if any) the participants thinks support the statement (we use this during the analysis of metric 1, since this motivates the participants' answers).

- 3) How deep (i.e. to which abstraction level) the participants had to look in the requirements specification to reach a decision.
- 4) Time to complete part I.

**(Design)**

The subjects are presented with 12 statements (of which 7 should be included according to study design). For each of these statements the subjects shall answer if the statement is supported by the other requirements (on lower abstraction levels), and motivate their answer by linking the statement to one or several requirements. Being able to correctly connect the statements to the rest of the requirements would indicate that there is a strong connection between the Product Level requirements and the rest of the specification. In addition, the subjects are asked to answer how deep they had to look into the requirements specification to reach their decision.

**(Comment)**

This research question evaluates if the linking of abstract top-level requirements to more detailed ones (Feature Level and down) is possible (i.e. whether it is possible to determine if an abstract statement is in line with the product or not). If this is the case, and if the abstract Product Level requirements are able to summarize lower level requirements, they can be used to get a big-picture product overview.

## 4.2. PART II

**(Research Question 2)**

*Given a requirements specification ordered hierarchically according to the level of abstraction of the requirements, to what extent is it possible to determine if new requirements should be included in the requirements specification based on the limits set by Product Level requirements?*

**(Research Question 3)**

Given a requirements specification ordered hierarchically according to the level of abstraction of the requirements, *to what extent is it possible to determine at what level of abstraction a new requirement shall be inserted?*

**(Test)**

We evaluate these questions by:

- a) Presenting the requirements specification in its entirety.
- b) Providing a number of new candidate requirements, and for each of these asking the participants to determine:
  - on what abstraction level this candidate requirement fits, and

- if it should be included or not in the system based on the already present requirements in the specification (governed by the top-level Product Requirements)

**(Collected Metrics)**

The questionnaire for this part of the study collects metrics on:

- 1) Whether the participants are able to decide (with a plausible motivation) whether to include or not include each requirement.
- 2) Whether the participants are able to place (with a plausible motivation) the requirement on a specific abstraction level.
- 3) Whether the participants are able to link each candidate requirement to the existing requirements in the requirements specification. These links serve as a motivation for why a requirement should be included or not.
- 4) The reason(s) the participants give as motivation when a requirement should not be included.
- 5) Time to complete part II.

**(Design)**

The participants are presented with 10 candidate requirements (of which 5 should be included according to the study design). The participants have the choice of placing each requirement on Feature, Function or Component Level (Product Level is not offered as an alternative as these requirements are considered locked for the sake of the study). Subsequent to placing the requirement on a certain abstraction level the participants are asked to decide if it should be included in the system or not, and to motivate their decision. If they choose to include the candidate requirement it should be linked to the existing requirements on the abstraction level above (giving a direct or indirect link to one or several Product Level requirements). If they choose not to include the candidate requirement the participants are asked to motivate why, e.g. using the categories “Not supported by Product Level”, “Undeterminable Level”, “Cannot be linked to requirements on the level above”, “Contradicts another requirement”, “Unclear requirement”, and “Other.”

### 4.3. PART III

**(Research Question 4)**

*What is the background of the participants in the study?*

**(Research Question 5)**

*Does the participants' background substantially influence the results of Part I and II?*

**(Test)**

Post-test questionnaire with a number of subjective assessment questions.

**(Collected Metrics)**

For each of the participants we gather the following information:

- 1) Years in software engineering curriculum.
- 2) Number of study points.
- 3) Years of industrial experience in software engineering.
- 4) English skills (rated: Novice, Moderate, Skilled, Expert).
- 5) Requirements engineering skills (rated: None, Novice, Moderate, Skilled, Expert).
- 6) Assessment of domain understanding before & after the evaluation.
- 7) Assessment of ease of understanding and use of: abstraction levels, placement and work-up, RAM, and requirements representation.

**(Comment)**

The purpose of this part is to collect background information on the subjects in order to ascertain if aspects like language skills, experience etc. influenced the results obtained in Part I and II. Although the subjects are students, their experience and skills often vary considerably, especially in the Master's programs at Blekinge Institute of Technology. Many have several years of industry experience in their background, their English skills may differ, and some have worked in development projects as requirements engineers. Thus, this post-test collects information that is used to describe the subjects in Section 3.2, and serves as a help to understand their answers during analysis.

## **5. OPERATION**

### **5.1. PREPARATION**

The subjects were not aware of the aspects that we intended to study, and were not given any information regarding research questions in advance. They were aware of that it was a controlled evaluation in the area of requirements engineering that was a part of their requirements engineering course. The evaluation ran over a period of three hours, and all of the subjects were seated in the same room.

Introduction to the study was given during these three hours in the form of a brief slideshow presentation. In this presentation some basic concepts of RAM were shown pertaining to abstraction and levels, and how the requirements are connected etc. The focus of the presentation was put on presenting the instrumentation which consisted of:

- Purpose, e.g. evaluate RAM according to Part I and Part II (as seen in Section 2.2 and Section 4).
- Requirements specification (ordered hierarchically and with connections between the requirements across levels).
- The questionnaires used to gather the results, and how they should be interpreted.

In addition to this it was made very clear to the subjects that the requirements already present in the requirements specification (RAM) should govern their answers. I.e. the statements provided as a part of the questionnaire for Part I should be accepted or dismissed with regards to the requirements present in the specification, and not according to what the individual subjects “thought” or “wanted”. For Part II the same general principle was emphasized, i.e. the placement of new candidate requirements on appropriate abstraction level was to be example driven, using the requirements already present in the specification as “good examples”. The decision to accept a new candidate requirement into the product or dismiss it should be governed by the Product Level requirements and the ability to link the candidate requirement to them directly or indirectly.

## **5.2. EXECUTION**

The evaluation was executed in one day over a four hour session, i.e. the subjects knew that they had at least four hours to complete the study if they needed the time. The requirements specification used in the first part of the evaluation was handed out at the start of the presentation, to give the participants a chance to familiarize themselves with the format. After the presentation, the questionnaire for the first part was handed out. As the participants completed the first part, they brought the questionnaire and the requirements specification to the front of the room where they were given the material (a questionnaire and the complete requirements specification) for part two. The start time and delivery time was noted on the questionnaire. As part two was completed the participants were given the last part of the study to complete, after which they were allowed to leave the room. No breaks were allowed during this time. The mean and median times to complete Part I and Part II were around 30 minutes, the shortest times spent on each part were around 15 minutes and the longest were 50 minutes for Part I and 1 hour for Part II. After 3 hours all subjects were finished and had handed in all material. All subjects completed all parts of the study.

## 6. RESULTS AND ANALYSIS

As a first step in presenting and analyzing the collected data we use descriptive statistics to visualize the results.

### 6.1. PART I

In total 12 statements were given to the subjects, and they had to decide which of these 12 were in accordance with the product. The product was represented (described by) requirements on Feature, Function and Component Level in the specification. 7 out of the 12 statements were statements designed to be supported by the requirements specification. During the analysis of Part I we have considered those answers that are in line with the study design and aptly motivated as “correct”. If the answers are in line with the study design but missing a proper motivation (i.e. not linked to acceptable Feature Level requirements), or if the answer is not in line with the study design, the answer is considered “incorrect”.

Table 24 shows the results for every subject, the number of correct answers (second column), and the depth measured in abstraction levels that the subject had to use in formulating an answer. The average depth is calculated following the levels presented to the subjects in Part I, i.e. 1 = Product Level, 2 = Feature Level, 3 = Function Level, and 4 = Component Level.

The subject average is 11.33 correct answers (out of 12 possible) with an average depth of 2.24 (where minimum = 2, maximum = 4). This implies that most subjects were able to use Feature level requirements to render a mostly correct decision regarding the statements.

Table 25 shows the results from Part I but divided by statement instead of subject. For every statement (S1, S2, etc) the number of correct answers is displayed (maximum is 21 correct answers). In addition the average depth for each statement is displayed.

The depth of the investigations in order to render an answer is better illustrated through a box plot diagram, presented in Figure 40. Looking at the correct answers (left) most are centered on a depth of 2 with the exception of two outliers that have the depth 3 and 4 respectively. Looking at the incorrect answers however, the spread is more pronounced as most subject lie in between depth 2 and 3. This implies that in the instances where the subjects gave an incorrect answer they also looked deeper in the specification, i.e. the Feature level was not enough and they had to look at the Function or even the Component level.

Subject	# Correct	Average Depth
1	12	2,17
2	12	2,17
3	12	2,33
4	12	2,17
5	9	2,45
6	12	2,00
7	12	2,36
8	9	2,33
9	12	2,27
10	12	2,17
11	11	2,20
12	10	2,33
13	12	2,27
14	12	2,00
15	11	2,08
16	11	2,27
17	12	2,08
18	11	2,50
19	11	2,08
20	11	2,56
21	12	2,20
<b>Average</b>	<b>11,33</b>	<b>2,24</b>
<b>Median</b>	<b>12,00</b>	<b>2,20</b>

**Table 24.** *Number of correct answers and depth divided by subject.*

To check the validity of this implication a Mann-Whitney Test was performed comparing the two populations (Correct and Incorrect) to check that this was not a coincidence. A significance of 0.001 was obtained indicating that the differences between how deep the two populations look are not a coincidence.

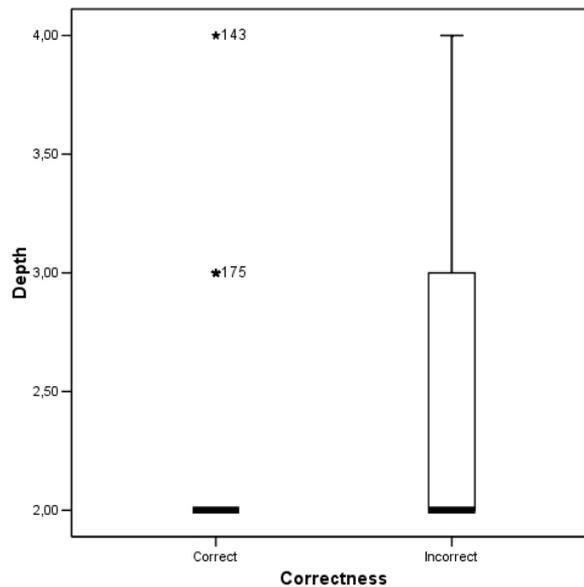
The Mann-Whitney test was chosen for the large differences in group size between correct and incorrect, see Table 26, and due to the fact that the data are not normally distributed (the Shapiro-Wilk normality test gives  $p < 0.00001$ , whereas 0.05 or larger would have indicated a normally distributed data set).

Statement	# Correct	Average Depth
S1	19	2,37
S2	20	2,24
S3	21	2,05
S4	15	2,53
S5	21	2,29
S6	20	2,00
S7	19	2,20
S8	21	2,40
S9	20	2,44
S10	20	2,37
S11	21	2,00
S12	21	2,00
<b>Average</b>	<b>19,83</b>	<b>2,24</b>
<b>Median</b>	<b>20,00</b>	

**Table 25.** *Number of correct answers and depth divided by statement.*

---

The data used in Figure 40 are specified in Table 26, where N denotes amount of answers (as to what depth was used) for correct and incorrect determinations. The data used for Figure 40 can be seen under the “valid” column. In some cases the answers regarding depth used were missing (6.5% of the cases for correct and 16.1% for the incorrect, as can be seen under the “missing” column). In total 203 answers that were correct specified their depth and 26 answers for incorrect specified their depth.



**Figure 40.** Comparison of depth distribution between correct and incorrect answers.

	Valid		Missing		Total	
	N	Percent	N	Percent	N	Percent
<b>Correct</b>	203	93,50%	14	6,50%	217	100,00%
<b>Incorrect</b>	26	83,90%	5	16,10%	31	100,00%

**Table 26.** Answers ordered according to valid and missing answers for correct and incorrect answers – data for Figure 3.

### Summarizing Part I

A clear majority (11.33 average / 12 statements = 94%) of the subjects gave a correct answer regarding the statements. Further, the subjects rendering a correct answer seem to have looked less deep in the specification (requirements hierarchy) than the subjects rendering an incorrect answer, i.e. Feature level (level 2 from the top) was the most used level in the correct answers, while both Feature and Function level were used when rendering the incorrect answers. This implies that it is possible to relatively quickly make a correct decision regarding whether or not a particular statement is supported by the requirements specification. If there is hesitation (and a need for more information), there is a larger risk to make a mistake.

## 6.2. PART II

In total the subjects were given 10 new candidate requirements (CRs). For each of the CRs, the subjects were asked to:

1. Place the CR on an appropriate abstraction level (and motivate this placement through links to one or several requirements on the abstraction level above). We call this “abstraction level placement”.
2. Decide whether or not to include the CR in the product (indicating with links that the CR is in line with other already present requirements in the specification, or motivating their decision for not including the CR). We call this “include decision”.

The “product” in this case was represented and described through the requirements already present in the hierarchically ordered specification.

Table 27 gives an overview of the results divided by each candidate requirement (CR1, CR2, etc). The answers are divided into two main categories, visible in the table as *Correct* and *Fail*. These two main categories are divided into subcategories, further detailed below. By dividing the main categories into several sub-categories, we are able to further scrutinize the answers. It should be observed that in Part II, the subjects have to provide four things for every CR; abstraction level placement, include decision, and motivations for these two. Hence, an answer can be correct or wrong to different degrees, depending on which parts of the question the subjects manage to correctly complete. These distinctions are important for the analysis and the reason for the sub-categories.

### **Correct**

The correct category contains those answers where the subjects have managed to use RAM and the requirements specification in a correct way. This means that the subjects have been able to place the CR on an acceptable abstraction level and motivate this by linking to other requirements, and also, been able to make and adequately motivate an include decision. The correct category is divided into two sub-categories, i.e. *correct according to study design and with proper motivation*, and *correct with proper motivation*.

#### **Correct according to study design and with proper motivation**

This category holds the answers that are exactly as intended during the study design, i.e. not only is the abstraction level placement and the include decision the same as the study design intended, the motivation is also according to study design.

### **Correct with proper motivation**

The answers in this category are nearly the same as the category “correct according to study design and with proper motivation” (i.e. abstraction level placement and include decision are according to the study design), but the links indicated that should motivate the abstraction level placement and the include decision, or the motivations for not including the CR, are not always the same as the ones pre-specified during the study design. The links (and motivation for them) stated by the subjects were however scrutinized and considered to be acceptable. This implies that the subject either stated links that were not caught during the design, or that the requirements themselves are possible to interpret differently than what was done in the study design. However, the central issue with the answers in this category is that the placement of the CR and the include decision are ok, and the motivation for these decisions (in the form of links or exclusion motivations) are completely acceptable.

During the analysis, cases with e.g. correct answers but missing proper and well motivated links (motivations) were *not* given the benefit of the doubt, and were instead put in one of the Fail columns.

In the “correct with proper motivation” column in Table 27 some CRs have an additional number within parenthesis, e.g. CR5 has “(3)”. This indicates that the subjects’ answers in three times out of the eight were missing an abstraction level placement. This is however due to the fact that these CRs were deemed not to be included in the system. The subject motivated this in a satisfactory manner, but assumed that as the CR should not be included there was no reason for stating what abstraction level it should reside on.

*Summarizing the correct category, 113 answers out of a total of 210 are correct in that they have acceptable abstraction level placements and include decisions, using acceptable motivations. Of these, 69 answers are entirely according to study design, and an additional 44 use a different motivation.*

Candidate requirement	Correct		Fail		
	According to study design and with proper motivation	With proper motivation	Include-OK	Level-OK	Not-OK
CR1	10	6	2	2	1(1)
CR2	5	2	9(2)	5	0
CR3	5	4	8(3)	4	0
CR4	5	5	1	7	3(2)
CR5	7	8(3)	2	2	2
CR6	1	5	1	6	8(3)
CR7	9	3	0	5	4(1)
CR8	9	3(3)	4	4	1
CR9	3	5	2	5	6(2)
CR10	15	3(3)	2	0	1
<b>SUM</b>	<b>69</b>	<b>44</b>	<b>31</b>	<b>40</b>	<b>26</b>

**Table 27.** Summary table showing correct and failed answers.

**Fail**

The second category, “Fail”, holds all the answers where some aspects of RAM or the hierarchical requirements specification have been misunderstood (or instances where the participants forgot to fill in information), resulting in answers that do not have an acceptable motivation. Within this category, there are three sub-categories, in order to detail which aspects that have been misunderstood or understood. These three categories, “Include-OK”, “Level-OK”, and “Not-OK”, are further detailed below. It should be stressed again that the analysis has been conducted with the objective of finding as many faults as possible with the answers, and when there has been room for doubt the answer has been placed in the category least favorable for the study (i.e. as far to the right as possible in Table 27).

**Fail(Include-OK)**

Fail(Include-OK) indicates that the include decision is properly motivated by links, but that the abstraction level placement is not sufficiently motivated. Numbers within parenthesis in this case indicate missing information regarding abstraction level placement, e.g. in the case of CR2 nine answers are deemed Fail(Include-OK) and out of these two answers are missing information regarding abstraction level placement.

### **Fail(Level-OK)**

The next sub-category Fail(Level-OK) is the opposite to Fail(Include-OK), i.e. the CR abstraction level placement is sufficiently motivated but the include decision lacks an acceptable motivation.

### **Fail(Not-OK)**

Fail(Not-OK) is as the name indicates when the subjects have not given any acceptable motivation for neither abstraction level placement nor include decision. Numbers within parenthesis in this case indicate missing information regarding abstraction level placement or include decision. As these answers are considered entirely wrong no further detail is given about answers in the Fail(Not-OK) category.

*Summarizing the Fail category, in 31 out of 210 answers the subjects managed to make an acceptably motivated include decision, in 40 answers an acceptably motivated abstraction level placement, and in 26 answers both placement and include decision were missing or not acceptably motivated.*

### **Summary of Table 27**

*Table 27 shows that the subjects answered correctly regarding both abstraction level placement and include decision in  $69 + 44 = 113$  cases (54%). In  $69 + 44 + 31 = 144$  cases (69%) the CR was with a proper motivation accepted as a part of the system or dismissed. In  $69 + 44 + 40 = 153$  cases (73%) the concept of requirements abstraction was correctly used with proper motivations.*

### **Studying Subjects' Consensus**

In this section we compare the subjects' answers with the answers intended by the study design. In this comparison we consider the study design to be produced by domain experts with considerable training in RAM, as well as the particular requirements specification. To this, we compare the subjects' answers in order to ascertain to what extent the lack of domain knowledge or experience has any influence on the results. Where there are differences, we also analyze the reasons for these differences further.

Table 28 shows the average results of the subjects' answers with regards to include decision and abstraction level placement. The first column indicates the answer intended in the study design for every CR, "1" indicating that the CR should be included in the product, and "0" indicating that it should not. Coding the subjects' answers to "1" or "0" in the same way enables us to calculate an average for each CR. For seven of the CRs the subjects' consensus is fairly close to the answer intended in the study design, e.g. in case of CR1 "1" is to be compared with "0.81". This means two things. First, it means that the subjects' answers are in line with

the intentions of the study design (i.e. expert product managers' opinions). Second, the relatively high number means that there is a large consensus among the students that this CR should be included. If this had been a requirements discussion meeting among e.g. managers there is a good chance that the consensus would be to include the requirement in the product.

	Study Design Include Decision 1 = Include 0 = Do not Include	Subject Average Include Decision (Consensus)	Study Design Abstraction Level Placement	Subject Average Abstraction Level Placement (Consensus)
CR1	1	0,81	3	2,90
CR2	0	0,24	2	2,58
CR3	0	0,38	3	2,72
CR4	1	0,33	2	2,21
CR5	0	0,19	3	2,67
CR6	1	0,24	3	3,11
CR7	1	0,57	3	2,65
CR8	0	0,24	3	2,83
CR9	1	0,48	3	3,16
CR10	0	0,05	3	2,94

**Table 28.** Average (consensus) answers in comparison to study design regarding include decision and abstraction level placement.

In three cases such a consensus meeting could render the wrong decision (i.e. for CR4, CR6, and CR9), assuming that the study design is the “correct” answer. This is especially evident with CR4 and CR6 where most (2/3 or more) of the subjects did not want to include the candidate requirement in the product. For CR9 the include decision could go either way as the subjects’ opinions were divided equally on the issue. Below, these three CRs are further discussed.

**CR4**

In the case of CR4 the subjects were more inclined to dismiss requirements than was anticipated during the study design. Although the study design provides opportunities for including a requirement like CR4, the subjects did not agree with this since explicit support for CR4 was missing in the Product Level requirements (according to the views of the subjects).

**CR6**

In the case of CR6 the divergent view of the subjects can be explained to some extent with that several of the subjects misunderstood or misinter-

preted CR6. The requirement is a Function Level requirement and hence rather technically specific. A lack of knowledge regarding web technology resulted in that many subjects thought that the CR contradicted other requirements already present in the requirements specification.

#### **CR9**

In the case of CR9 many subjects not wishing to include the CR argued that there was no support for it in the rest of the product requirements. According to the study design the intention was to have support for CR9, but when analysis was performed explicit support for CR9 could not be found even by the study designers themselves. The tendency is recognized from the CR4 case where dismissal was closer at hand than inclusion, although the study design flaw can also be a contributing factor.

Another way of studying the data is to identify those CRs where there is less than two thirds majority for the decision. This gives us CR3, CR7 and CR9. For these three CRs, there would probably be more discussions during a consensus meeting than for the other CRs, and the ultimate include decision may depend on these discussions.

In the two rightmost columns of Table 28 the intended abstraction level and the average of the levels specified by the subjects is presented. Overall, the subjects' consensus and the study design intentions are comparable, with the exception of CR2, where the subjects' consensus places the CR on level 3 (Function Level) while the intended level was level 2 (Feature Level). Because the requirements model is example-driven, there is often not a definitive answer whether a requirement should be placed on a specific level or on one of the neighboring levels. In this case, there are examples on both the feature and the function level of requirements that can be considered similar to CR2.

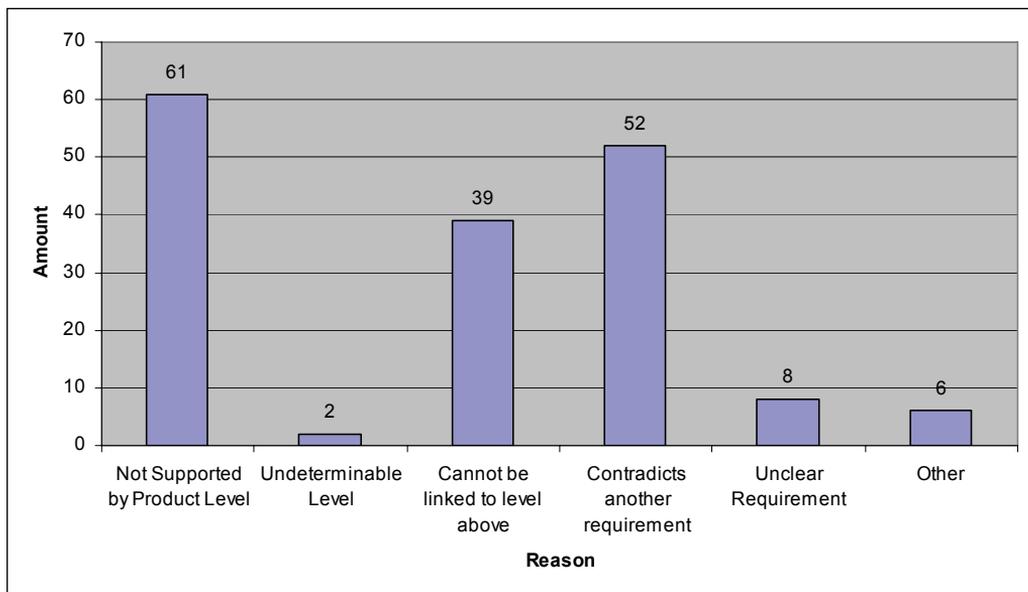
*In summary the consensus whether or not to include a candidate requirement is most of the time comparable to what was intended in the study design. The exceptions to this can be traced to lack of domain knowledge among the subjects, alternatively some insufficiencies in the study design (i.e. the requirements themselves could have been more explicit). There is also a tendency among the subjects to dismiss CRs rather than give them the benefit of the doubt and include them. This was especially visible in the cases where both inclusion and exclusion could be possible.*

*The consensus regarding abstraction level placement is close to the intention of the study design in all cases but one (CR2).*

### Reasons for Dismissing CRs

Figure 41, finally, gives an overview of the reasons given by the subjects when they chose to dismiss a CR rather than to include it in the product. Three motivations are in clear majority, i.e. that the CR is not supported by any requirements on the product level, the CR cannot be linked to any requirements on the abstraction level above, and that the CR contradicts another requirement already present in the requirements specification. These three reasons were also the ones used in the study design.

We also see that the remaining three categories (i.e. undeterminable level, unclear requirement, and other comments (free text comments)), that represent perceived problems in the study design, were rarely used and constitute a total of 13%. It should be noted that the subjects were able to mark more than one reason for excluding a candidate requirement.



**Figure 41.** Reasons given by subjects when they chose not to include a CR.

*In summary, the subjects are able to clearly and in accordance with the opinions of experienced product managers discern the reasons for why a CR should not be included in the product.*

## 7. RESEARCH QUESTIONS REVISITED

### 7.1. PART I

#### (Research Question 1)

*Given a requirement specification ordered hierarchically according to the level of abstraction of the requirements, to what extent do the top-level (most abstract requirements) connect to the rest of the requirements?*

In the evaluation 94% of the answers are correct. The subjects rendering the correct answers used requirements at level 2 (Feature Level) to render decisions, while the subjects that answered incorrectly had a depth of between 2 and 3. This may indicate that in order to get a correct answer you generally only need to go as deep as Feature Level, indicating that Product and Feature Level are tightly coupled, which is a prerequisite for RAM and the idea behind abstraction levels.

The large amount of correct answers indicates that Product Level requirements are indeed general enough to summarize several requirements on lower levels, but still concrete enough so that it is possible to determine if they represent the product or not, and can thus be used to get a system overview.

### 7.2. PART II

#### (Research Question 2)

*Given a requirements specification ordered hierarchically according to the level of abstraction of the requirements, to what extent is it possible to determine if new requirements should be included in the requirements specification based on the limits set by Product Level requirements?*

A total of 69% of the answers include or exclude the candidate requirements using acceptable motivations. In other words, more than 2/3 of the answers are correct in terms of a well motivated include decision. This indicates that the ability to compare requirements with similar requirements on the same abstraction level and the necessity to be able to link a new requirement to the existing hierarchy indeed supports the decision process. Time constraints prevented us from producing a non-hierarchically structured requirements specification to run a control group. However, offering all of the requirements structured according to e.g. functionality would have forced the subjects to read through much more information (requirements) before being able to render an include decision. Looking at

the results regarding the relation between depth and answering correctly in Part I, there is indication that more information (larger amount of requirements) as a basis for a decision does not render a better decision, rather the opposite. In addition there exists a multitude of ways to structure requirements, thus it is not possible to compare RAM structured requirements to a specific “normal” or “default” structured set of requirements.

Strictly speaking we cannot say that RAM structured requirements is better or worse than the current situation, since it depends on what the “current” situation is. However, the results indicate that it is better than answering at random, and in conjunction with the answer to research question 3 we are convinced that it is a real help to product managers.

**(Research Question 3)**

*Given a requirements specification ordered hierarchically according to the level of abstraction of the requirements, to what extent is it possible to determine at what level of abstraction a new requirement shall be inserted?*

A total of 73% of the answers place the candidate requirements on their intended abstraction level using acceptable links to other requirements. It thus seems to be relatively easy to determine the abstraction level of a requirement by studying the “good examples” that the existing requirements specification provides.

**7.3. PART III**

**(Research Question 4)**

*What is the background of the participants in the study?*

**(Research Question 5)**

*Does the participants' background substantially influence the results of Part I and II?*

These questions were mostly used in the analysis of the results. The information gained regarding research question 4 is found in Section 3.2. For research question 5, we were unable to find any difference between the performance of participants with more experience (industry experience or experience with requirements engineering) and those with less experience. It should be noted, however, that the differences in terms of experience between the participants were not very large (a standard deviation of one

year for the industry experience). The homogeneity of the group may be the reason why no difference was found.

## 8. CONCLUSIONS

In this study we evaluate aspects of a hierarchical requirements abstraction model – RAM – as a preliminary step before conducting full scale industry piloting. Using industry as a laboratory through pilots is a powerful tool and a part of building trust and commitment prior to technology transfer (in the form of process improvement activities). However, prior to using valuable and hard to obtain industry resources we consider it prudent to evaluate the ideas in an academic laboratory setting. Initial evaluation of the model in academia intends to investigate whether or not industry piloting is mandated, or if RAM needed further refinement or even redesign before directly involving industry in validation activities. It is thus important to get early indications whether or not the fundamental features inherent in the current model are useful and usable, and if the assumptions implied by the model hold.

RAM is intended to aid product managers (and others involved in requirements engineering) in getting an overview of a product using relatively few abstract Product level requirements. Using the same Product level requirements as a basis (and the requirements linked to them), managers should be able to include or dismiss new incoming requirements in a repeatable way (i.e. not dependent solely on the person performing the work), and place new requirements on appropriate abstraction levels in the model.

The participants in this evaluation were given a relatively large amount of requirements (120 in total) and were asked to accomplish a considerable amount of work in a short amount of time. The participants were expected to form an understanding of the concept of requirements abstraction and hierarchically structured requirements specifications, understand the domain (read and understand the requirements), and then solve the tasks in Part I and II. Very little training of the model was given to the participants, and they also possessed little prior knowledge regarding the domain if compared to e.g. a product manager. Considering these aspects and the time spent the results produced are encouraging and point towards both high usability and usefulness of RAM.

The characteristics of industry are also relevant as real-life usage of RAM should be easier than during the evaluation. In industry, requirements engineering is not performed in isolation (as was the case in the evaluation); regular meetings as well as official and unofficial conversa-

tions and discussions help in sharing views and forming consensus as well as a shared understanding. From this perspective the results obtained in the evaluation are even more promising. The consensus results presented in Table 28 indicate that had there been cooperation and exchange between the participants even better results may have been achieved, since cooperation often yields better results than working individually. In addition, industry product managers are often senior practitioners well versed in both their specific domain and in the field of requirements engineering. Given this some of the mistakes made by the evaluation participants, where lack of domain knowledge was evident, could probably have been avoided in an industry setting.

As indicated by the results of the controlled evaluation in this article, the concept of abstraction and the usage of abstraction levels seems to be fairly intuitive. This is also useful in a scope outside of RAM. For example, the possibility to obtain homogenous requirement abstraction levels using RAM may facilitate requirements prioritization, as requirements on the same level of abstraction are easily compared in contrast to comparing requirements on different levels of abstraction.

A potential issue with the model is that since the abstract requirements on the upper levels (especially Product level) are few, their individual impact on the product is considerable when they are used as the principal decision support (getting a quick overview of the product). This implies that the demands on these requirements are high in terms of being well-formed and unambiguous. In addition they need to be explicit. In this study, lack of explicit support among the product level requirements was a motivation often used by the participants to dismiss a candidate requirement.

In industry we may assume that the users of RAM possess domain knowledge as well as at least minimal training in using RAM together with some tool support. These factors should help ensure even greater usability and usefulness of the model than was observed during the controlled evaluation in this article.

## **8.1. FUTURE WORK**

Although this evaluation has given promising initial results there is a need to run further evaluations on larger populations of participants. In addition there is a need to run control group evaluations performing the same tasks but on requirements specifications structured in several “traditional” ways (e.g. structured according to functionality, stakeholders, or use cases). The comparisons of the results could give further information re-

garding the usefulness and usability of RAM, after which the model may be considered mature enough for industry piloting. Several of these evaluations are planned for the near future and preparations for some of them are already underway.



# Chapter 6

---

## A Replicated Controlled Empirical Evaluation of a Requirements Abstraction Model

*Tony Gorschek, Mikael Svahnberg, Andreas Borg, Jürgen Börstler, Magnus Eriksson, Annabella Loconsole, and Kristian Sandahl*

Submitted to IEEE Transactions on Software Engineering, 2005.

### **Abstract**

Requirement engineers in industry are faced with the complexity of handling large amounts of requirements as development moves from traditional bespoke projects towards market-driven development. There is a need for usable and useful models that recognize this reality and support the engineers in the continuous effort of choosing which requirements to accept and which to dismiss off hand using the goals and product strategies put forward by management. This chapter replicates an evaluation of such a model that is built based on needs identified in industry. The evaluation's primary goal is to test the model's usability and usefulness in a lab environment prior to large scale industry piloting, and is a part of a large technology transfer effort. The replication uses 179 subjects from three different Swedish Universities, which is a large portion of the university students educated in requirements engineering in Sweden during 2004 and 2005. The results match the original study and strongly support that the model is indeed both useful and usable and ready for industry trials.

## 1. INTRODUCTION

Requirements Engineering (RE) more and more transcends project boundaries as market-driven product development is becoming increasingly commonplace in software industry [1-3]. Central activities in RE are performed pre-project as a part of for example the product management activities since the requirements flow is continuous and not limited to a specific development instance [6, 7].

In this environment requirements come from several sources both internal (e.g. developers, marketing, sales, support personnel, bug reports etc) and external (e.g. users, customers and competitors, often gathered via surveys, interviews, focus groups, competitor analysis etc) [8, 10, 11]. Large volumes of requirements from multiple sources risk overloading companies unless they can handle incoming requirements in a structured way, dismissing some, and refining some prior to allocating them to a development instance [20]. In addition to the volume, the requirements themselves are of varying quality, state of refinement, and level of abstraction. In traditional bespoke development (customer-developer) [3] a requirements engineer can actively elicit requirements and thus hope to control or at least substantially influence these aspects. In a market-driven situation this is seldom the case. Most requirements are already stated in one way or another when they reach the requirements engineer (e.g. a product manager). The knowledge and experience of the developing organization, of which the requirements engineer in this case is instrumental, is central to making sense of the requirements as they are processed [21].

Many requirements engineering best-practices, frameworks, and tools are adapted to suit a bespoke environment with traditional, project focused, customer-developer relationships. There is a need for the development and evaluation of RE practices and models that support professionals working with product planning and development (e.g. product managers) in a market-driven environment.

This chapter presents an evaluation of some key aspects of such a model, the Requirements Abstraction Model (RAM). The evaluation is a replication of a previous study, extending it from 21 to 179 subjects from three Universities.

RAM was developed as a response to needs identified in industry during process improvement conducted in cooperation with Danaher Motion Särö AB and ABB Automation Technology Products [134, 166]. The Requirements Abstraction Model is designed towards a product perspec-

tive, supporting a continuous requirement engineering effort. It can handle large quantities of requirements of varying degrees of detail and offers a structure and process for the work-up of these requirements (see Section 2.1). The main purpose of the evaluation in this chapter is to assess the usefulness and usability of RAM in a controlled environment prior to widespread industry piloting. This is achieved by evaluating the model in two parts. Firstly, by using the model engineers should be able to get an overview of an entire product by studying relatively few top level requirements. Secondly, engineers should be able to utilize the concepts of requirements abstraction as they decide whether new incoming requirements are in-line with product goals and strategies. These aspects are further detailed as problem statements in Section 2.2.

Prior to describing the actual study and results a short introduction to RAM is given in order to increase the understanding of the rationale behind the evaluation. Therefore Section 2 gives an overview of RAM, the problem statement used in this evaluation, and related work. Section 3 details the planning, context, and overall design of the evaluation along with a sub-section detailing validity issues. The research questions are formally stated in Section 4, and the operation (execution) of the evaluation is detailed in Section 5. Section 6 presents the results and analysis with the help of descriptive statistics. Section 7 is devoted to revisiting the research questions and discussing them explicitly based on the results and the analysis. In Section 8 conclusions and plans for further work are presented.

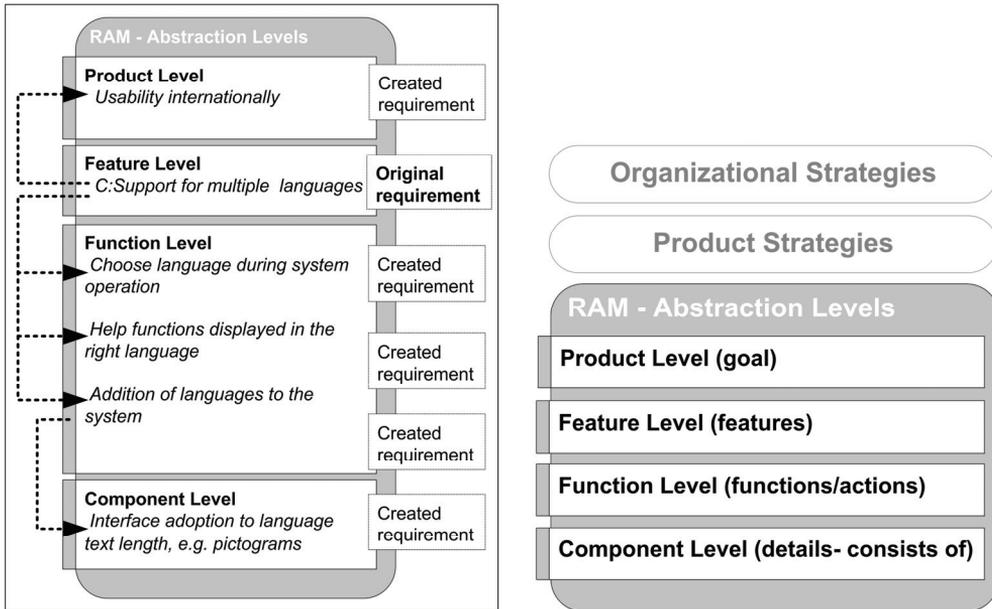
## **2. BACKGROUND AND RELATED WORK**

This section gives a short description of RAM and related work regarding the concepts of abstraction in relation to RE work. For reasons of brevity details not central for the evaluation presented in this chapter have been left out. For details, please see Chapter 4.

### **2.1. THE REQUIREMENTS ABSTRACTION MODEL**

RAM is a hierarchical requirements abstraction model and a method for working with this model based on the concept that requirements come on several levels of abstraction. Instead of flattening all requirements to one abstraction level RAM *uses* the varying abstractions of requirements, and orders the requirements hierarchically according to abstraction level. Figure 38 shows four abstraction levels; *Product Level*, *Feature Level*, *Function Level*, and *Component Level*. The Product Level is the most abstract level and requirements here are considered abstract enough to be compa-

rable to product strategies, and indirectly to organizational strategies. In the context of RAM, product strategies are rules, long and short-term goals, road-maps, visions pertaining to a product specified by management etc. Going down to Feature and Product Level the requirements become concrete enough to be used for estimations and as input to development.



**Figure 42.** RAM Abstraction levels and example of Work-up.

Briefly, the process followed with RAM is that when requirements arrive they are placed and specified on an appropriate level by comparing with the existing requirements base (i.e. example-driven). This means that the present mass of requirements is used as decision support material for the inclusion, exclusion, and placement of new requirements. Following this all requirements go through *work-up*. Work-up entails abstracting low-level requirements up to Product Level and breaking down high-level requirements to Function Level. This is done by creating new requirements in levels above and below and linking them to the original requirement. Figure 38 gives an example of this. The original requirement “C:Support for multiple languages” (placed on Feature Level) is abstracted to Product Level through the creation of a new work-up requirement “Usability internationally”, and broken down to Function Level where three new work-up requirements are created as a part of the breakdown. In some cases require-

ments already present can be used for abstraction. As an example, in Figure 38, if a new requirement comes in stating “*C:Support imperial units*” it could be placed on Feature Level and linked directly to “*Usability internationally*” as imperial units are used in Great Britain and the US in addition to SI units (metric system). In this case no new requirement has to be created on Product Level and the new requirement can be linked to an already existing one.

During the work-up process the original requirement is compared to product strategies. This offers decision support regarding whether the requirement should be specified, refined and kept in the repository, or whether the requirement should be dismissed. For example, let us assume that “*Usability internationally*” was not accepted but rather the company wanted to limit the product market to the Scandinavian market. In this case “*Usability internationally*” would be “*Usability Scandinavia*”, and the new requirement “*C:Support imperial units*” would be dismissed as only SI units are used in Scandinavia. In other words, the product level ultimately decides whether or not to include a requirement, since all requirements are directly linked to this level. If the new requirement is not supported, this may indicate that a new product level requirement needs to be created.

The break-down of the requirements stops on Function Level where the requirements are not perfect, but good enough to be used as decision support for estimation and risk analysis and as input to project(s) for realization. The Component Level is not mandatory in the model, but present since requirements in some instances were delivered in a very detailed form, and these requirements also needed to be handled (i.e. specified and abstracted to assure that they are in line with the overall goals and strategies). In the example presented in Figure 38, the Component Level requirement acts as extra information on a detailed technical level. For example the interface has to be adapted to pictograms for the Chinese language. In this case the Component Level sets a restriction that will accompany the requirements into development.

The features of using RAM can be summarized as follows:

(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 resources to work on and 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(s)). This

assures that projects get good-enough requirements to base their development efforts on (e.g. testable and unambiguous [5]).

(III) Work-up of a requirement means that additional requirements may have to be created in order to get a connection to the upper most level. For example if an incoming new requirement is placed on Function Level and no appropriate requirement exists on the Feature Level, a new one has to be created. This Feature Level requirement in turn needs to be linked to the Product Level. This ensures that it is possible to follow a requirement through abstraction levels and assure that there is an explicit connection upwards to product strategies. In the same way every requirement is broken down to a level good-enough to serve as a basis for project initiation (Function Level). Requirements within a certain abstraction level are homogenous enough to be comparable with each other, which is a prerequisite for 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. Managers can for example study the most abstract levels and get a quick overview of the system, while developers can choose a more detailed view, but still have an explicit connection to the overall goals of the product as detailed requirements are connected upwards through the levels.

As an initial step in preparing RAM for industry use some of the underlying concepts of the model need to be evaluated in a laboratory setting, this is elaborated in Section 2.2 below.

## **2.2. PROBLEM STATEMENT**

The usefulness and the usability of RAM depend on several concepts that need to be assessed. The ones studied in this controlled evaluation are summarized below in two main parts.

(Part I) Top level (Product Level) requirements should be used to ascertain whether a new incoming requirement is in line with product strategies or not, acting as decision support for acceptance or early dismissal of a new requirement. Product Level requirements should therefore convey an overview (“big-picture”) of the product (i.e. Product Level requirements should summarize and represent a considerable amount of requirements on lower levels).

In Part I we evaluate whether abstract Product Level requirements are general enough to summarize several re-

quirements on lower levels, but still concrete enough so that it is possible to determine if they represent the product or not. If this is the case, Product Level requirements can be used to convey the aforementioned big-picture product overview.

(Part II) In Part II we evaluate whether it is possible in practice, with relatively few abstract top-level requirements, to understand what features and attributes a product should and should not consist of. As new requirements come in it should be possible to:

- dismiss or accept a particular requirement using product strategies (explicitly expressed as Product Level requirements in this evaluation), and
- place the requirement on an appropriate level of abstraction using already present requirements on each level as indication of where the new requirement should be placed.

The usability of the model depends on these two points and thus the utilization of requirements abstraction. Requirements abstraction is the ability to use relatively few abstract Product Level requirements to derive decisions across the hierarchy of abstraction with respect to inclusion or exclusion of new requirements. In addition, the ability to place new requirements on an appropriate level using already present requirements as decision support keeps the levels homogenous, i.e. requirements can be easily compared within the same abstraction level. The ability to place requirements in the model also preserves model consistency over time.

The usefulness and the usability of RAM according to Part I and Part II above have previously been evaluated in Chapter 5. In the previous study 21 participants from one university (Blekinge Institute of Technology) participated. In this article the study has been replicated with software engineering and computer science students from three universities (i.e. Blekinge Institute of Technology, Linköping University, and Umeå University). This totals 179 participants from four different education programs, which is a large portion of the university students educated in requirements engineering in Sweden during 2004 and 2005.

The controlled evaluation of RAM presented in this chapter is centered on the aspects described in Part I and Part II, and they are stated formally as research questions in Section 4.

## **2.3. RELATED WORK**

The realization that requirements are often on different levels of abstraction and in varying stages of refinement has been recognized by others in both industry and academia [20, 155]. We have confirmed this during process assessment and improvement efforts conducted at both Danaher Motion Särö AB and ABB Automation Technology Products [21].

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 [20]. The basic notion is that both the abstract (long-term overview) and the detailed (giving context and the short term-view) are important [156, 157], and the two used in combinations offer a better understanding.

The field of goal based requirements engineering (see [158-160]) is focuses on the elicitation of goals intended to become requirements at some point, i.e. working from the top down. For example, Wiegers [106] and Lauesen [161] describe that requirements can be of different types pertaining to what they should be used for. This corresponds to the industry view of dividing requirements of different types into separate documents, from goal-like (abstract natural language formulations [136]) to technical design-like specifications. The focus is on that requirements are specified on different abstraction levels depending on usage, e.g. project type.

RAM is designed to operate in a market-driven product centered development environment. The volume of requirements that needs to be handled makes initial screening important in order to decrease the risk of overloading in the evaluation and realization process [20]. RAM is intended to utilize abstraction, in a somewhat similar way to what Wiegers and Lauesen describe. It gathers all requirements for a product in one repository where requirements are linked between levels of abstraction. This makes it possible to perform fast screening and dismissal of requirements not in line with product strategies.

The requirements work is performed by the product management organization, using RAM independently in a continuous RE effort designed for long term product development. Projects that realize requirements are a result of the work conducted, i.e. RE activities using RAM are not project initiated, but rather *initiates* projects as needed. This entails handling a large and steady stream of requirements continuously before any com-

mitment or plan exists for realization of them. Early screening of requirements (comparing to product strategies) is only one part, another is the need to select requirements for realization in projects. By having requirements on different homogenous abstraction levels comparisons, prioritization, and planning on each level is made possible. This enables product management to perform early screening and selection of requirements. This differs from e.g. Wiegers and Lauesen who focus more on requirements engineering in relation to projects.

### **3. PLANNING OF THE EVALUATION**

In this section we describe the overall parameters for the evaluation. These parameters are common for all parts of the study and research questions described in Section 4.

#### **3.1. CONTEXT**

The study is conducted in an academic setting, with the help of graduate and undergraduate students at Blekinge Institute of Technology, Linköping University and Umeå University. The evaluation was conducted as an exercise in different courses at the universities.

The intended target audience for RAM, however, is product managers with several years of experience in a specific domain and of a specific product. In the study, the subjects have no training on RAM, possess limited domain knowledge, are under time pressure, and have not seen the requirements before. There is thus a considerable gap between the intended target group and the sample used in this study. The subjects in our study can be expected to adapt a more surface oriented approach to the problem than product managers. We argue that this works to our advantage, since any effects that we measure are likely to stem from the instrumentation and the use of RAM, rather than previous experiences of the participants in the study. If RAM proves to be usable in the study, it would indicate that it is able to decrease the dependency on individual persons' experience, knowledge, and methodology.

#### **3.2. SUBJECTS**

As mentioned, the participants in the evaluation are software engineering or computer science students at the three participating universities. This group consists of graduate and undergraduate students. Some of the participants are international students, the rest are Swedish. The subjects have an average of 0.51 years of industrial experience, and consider themselves

between novices and of moderate experience regarding requirements engineering, and between “moderate” to “skilled” in English.

### **3.3. DESIGN**

Prior to the controlled evaluation presented in this chapter two test rounds and one live round were performed. An initial version of the study was constructed and assessed with the help of colleagues at Blekinge Institute of Technology. Based on the experience from this test round, the study package was revised. Specifically, it was reduced in size and the questionnaires were changed to provide a closed set of possible motivations, to facilitate analysis. This revised study package was then assessed a second time with participants from a network of Swedish requirements engineering researchers, SiREN<sup>11</sup>. After this, the controlled evaluation was performed live with participation of Software Engineering Master’s students at Blekinge Institute of Technology (see Chapter 5). The exact same study package was then distributed to colleagues within the SiREN network in order for the controlled evaluation to be replicated at their respective university.

The controlled evaluation consists of four parts, all run consecutively without any breaks between. The first part is a preparatory lecture where RAM and the concept of a hierarchical requirements specification are introduced, together with a presentation of the study and the research instruments. The remaining three parts of the controlled evaluation, further described in Section 4, consists of materials and questionnaires to answer the research questions. All participants are given the same instrumentation, the same treatment, and the same questionnaires to answer. The study is estimated, after the pilot tests, to take a maximum of three hours if no breaks are allowed.

### **3.4. INSTRUMENTATION**

The instrumentation consists of three different questionnaires, further described in Section 4, and a requirements specification. This requirements specification details an intranet solution for a course management system. The system contains features such as news (information) in courses, file archives for courses, course calendars, course discussion forums, and management of course participants. The requirements specification is

---

<sup>11</sup> <http://www.siren.lth.se/>

based on a 16000 person-hour students' project. From this project different features have been added and removed to generate a requirements specification of adequate size. None of the subjects had any knowledge of this actual student project. The requirements are constructed to be of moderate to good quality based on the authors' experience as software engineers and requirements engineers as well as requirements engineering researchers. This was confirmed through the two pilot studies. To ensure that the requirements specification represents a complete system of adequate size (we estimate that it would take 15000 person-hours or more to develop the system) and still be small enough to be usable in the evaluation (i.e. 120 requirements in total) many of the more detailed requirements have been omitted.

The requirements specification is structured hierarchically into four levels, denoted Product (containing 9.6% of the requirements), Feature (19.1%), Function (59.6%), and Component level (11.7%). The levels are further discussed in Section 2.1. Each requirement contains the fields *Id* (a unique number where also the abstraction level can be discerned), *Title* (title of the requirement), *Description* (the actual requirement), *Rationale* (an explanation of the need for the requirement), *Restrictions* (any risks or restrictions on the requirement), *Relations to* (id and title of requirements that this requirement is related to), and *Relations from* (id and title of requirements that link to this requirement). The requirements are listed in tabular form as the example in Figure 39 illustrates. In this figure we see a selection of requirements on all levels, starting with two product level requirements, two feature level requirements, four function level requirements, and two component level requirements. Each requirement contains the aforementioned fields.

Level	ReqID	Title	Description	Rationale	Restrictions	Relations To	Relations From
Product	1 013	Product Interface	All access to the system must take place via the systems own user interface, i.e. access from third party products is not allowed.	Control look and feel. Avoid security issues and compability problems.		1 040 Manage and Conduct a Course	2 084 Web-based User Interface
Product	1 040	Manage and Conduct a Course	The product shall provide functionality that supports a teacher (course administrator) to conduct a course, manage the course, and supports the participants in following course progress, course news, and accessing information related to the course as well as exchanging course related information	This is the core of the product.			1 013 Product Interface  2 032 Course Start Page 2 035 Course News 2 039 Course File Archive 2 067 Course Administration 2 107 Discussion Forum 4 022 Link to Course
Feature	2 032	Course Start Page	Each course shall have a course start page.	Information overview		1 117 Distribute Information about Courses 1 040 Manage and Conduct a Course	
Feature	2 035	Course News	It shall be possible to attach news items to a course.	Keep the students informed about the course.		1 117 Distribute Information about Courses 1 040 Manage and Conduct a Course 3 129 Access to View Course News 3 128 Access to Add, Edit, and Remove Course News Items 3 127 Edit Course News 3 065 Course Start Page Contents 3 037 View Course News 3 036 Add Course News 3 069 Course Administration Contents	3 038 Remove Course News
Function	3 036	Add Course News	It shall be possible to add news items to a course.			2 035 Course News	3 128 Access to Add, Edit, and Remove Course News Items
Function	3 041	Add Files to Course File Archive	It shall be possible to add files to the file archive.			2 154 Support Swedish Alphabet 2 039 Course File Archive	3 130 Access to change in Course File Archive
Function	3 042	Remove Files from Course File Archive	It shall be possible to remove files from the course file archive.			2 039 Course File Archive	3 130 Access to change in Course File Archive
Function	3 043	Download files from Course File Archive	It shall be possible to download files from the Course File Archive.			2 039 Course File Archive	3 131 Access to use Course File Archive
Component	4 018	Successful Login	When a user has successfully logged in, the product shall display the user's personal start page.	After user authentication, the user wants to start working.		2 020 Personal Start Page	
Component	4 019	First login	The first time a user logs into the system the user's personal profile shall be shown.	The user may provide additional information about him/herself.		3 012 Login 2 047 Personal Profile	
				The user decides not to provide any additional information, which means that the user has an incomplete profile.		3 012 Login	

Figure 43. Example of requirements representation.

## **3.5. VALIDITY EVALUATION**

The validity of the study is divided into four areas: conclusion validity, internal validity, construct validity, and external validity. Below we discuss these in further detail. More information about possible threats to studies can be found in Wohlin et al. [129] and Robson [131].

### **3.5.1. CONCLUSION VALIDITY**

To ensure that the research instruments, including the posed questions, are of a good quality, two separate pilot-tests with the material have been executed before the “live” rounds (i.e. the original study at Blekinge Institute of Technology and the replicated study at Linköping University and Umeå University). Moreover, all participants received the same introductory lecture, were given the same material in the same order, and received the same additional instructions. It is thus unlikely that the instrumentation and the implementation of the treatment influence the results unduly. That being said, since we use the answers of human subjects the gathered measures are of course not 100% repeatable.

### **3.5.2. INTERNAL VALIDITY**

The participants may mature during the study. Specifically, this concerns their understanding of the requirements specification and how to read it. This means that the accuracy of the participants’ answers may improve as they answer more questions. In effect, as the study progresses the participants’ familiarity with the domain and the requirements specification becomes more and more alike the intended target group.

The instrumentation (i.e. the requirements specification described in Section 3.4 and the related artifacts described in Section 4) is designed to be of moderate to good quality to mimic the situation for the intended target group and should not influence the participants unduly. Moreover, to ensure that the participants answer in accordance with the research instruments and not according to their understanding of the domain of the system, we have intentionally left out some specific features that commonly occur in the domain, such as using standard USENET news clients, import and export users and data to other systems, and synchronize calendars with handheld devices. All of the product statements and candidate requirements used in part I and II of the study (further detailed in Section 4) are common in the domain, and the course management systems that the students are in daily contact with have features equivalent to the product statements and candidate requirements. This ensures that the

students provide a motivation for their answers based on RAM and the provided requirements specification and not according to their domain understanding or their opinion.

The motivation of the subjects may vary considerably, since they are students and do not have real interest in the product. This may actually speak in favor of the study, since there is less incentive for the participants to perform well.

### 3.5.3. **CONSTRUCT VALIDITY**

Since we are using a single requirements specification in this study, there is a risk of mono-operation bias, i.e. there is a risk that our choice of system and domain influence the results more than the method for working with the requirements specification that we wish to evaluate. As mentioned above, by intentionally leaving out commonly occurring features in the domain from the requirements specification we at least ensure that the participants follow the prescribed method and that we can detect if they do not.

We make no secret of the hypotheses in this study. Since there is only one treatment the only way the participants can influence the result is by deliberately answering wrong. This increases the risk that the evaluated methodology is deemed to be less useful than it is, which means that we may err on the cautious side when analyzing the results.

To avoid the risk that the wording of the questions may provide hints as to what the answer is, the questions in the questionnaires are formulated in a standard way. This is further described in Section 4.

### 3.5.4. **EXTERNAL VALIDITY**

To ensure the external validity and the ability to generalize the results, we use a requirements specification for a relatively large system in a fairly mature domain.

As discussed in Section 3.1, the knowledge and experience of the participants is less than that of the target audience (e.g. product managers in industry). To reduce this gap, we use a system from a domain that is familiar to the subjects. The target audience would also have training in using RAM and specifying requirements according to RAM, which the study participants do not have. In this study, we only focus on a few key aspects of RAM, and thus a complete training of RAM is not necessary. We also argue that more experience, knowledge, and training should not negatively impact the ability to work with RAM. In other words, any positive

effects detectable in this study should be transferable to the target audience.

In the same way, it is our belief that a hierarchical requirements specification works best when used with a software tool. In this study we use paper printouts which may impact the readability and the ease by which the participants may access the information. Hence, also here any positive effects are transferable to the target audience and the target environment.

Moreover, we use subjects from three different universities, with different backgrounds, corresponding to a large portion of the software engineering and computer science students educated in requirements engineering in Sweden during 2004 and 2005.

## **4. RESEARCH QUESTIONS**

The study is divided into three parts, with research questions attached to each part. Part I and II directly correspond to the problem statement posed in Section 2.2 and concern the evaluation of RAM. Part III is a post-test designed to gather information about the subjects to ascertain if and how their background influences the results of the study.

### **4.1. PART I**

#### **(Research Question 1)**

*Given a requirement specification ordered hierarchically according to the level of abstraction of the requirements, to what extent do the top-level (most abstract requirements) connect to the rest of the requirements?*

#### **(Test)**

This research question is evaluated by:

- a) Removing the most abstract requirements (Product Level) from the requirements specification.
- b) Presenting the Product Level requirements as statements about the product.
- c) Asking the subjects to determine whether each statement is supported by the remaining requirements specification, and in particular which of the requirements on the lower level in the requirements specification that supports the statement.

#### **(Collected Metrics)**

The questionnaire for this part of the study collects metrics on:

- 1) Whether or not the participants consider the statement supported by the requirements specification as a defining factor of the system.

- 2) Which feature requirements (if any) the participants think support the statement (we use this during the analysis of metric 1, since this motivates the participants' answers).
- 3) How deep (i.e. to which abstraction level) the participants had to look in the requirements specification to reach a decision.
- 4) Time to complete part I.

**(Design)**

The subjects are presented with 12 statements of which 7 should be included according to the study design. For each of these statements the subjects shall answer whether the statement is supported by the other requirements (on lower abstraction levels), and motivate their answer by linking the statement to one or several requirements. Being able to correctly connect the statements to the rest of the requirements would indicate that there is a strong connection between the Product Level requirements and the rest of the specification. In addition, the subjects are asked to answer how deep they had to look into the requirements specification to reach their decision.

**(Comment)**

This research question evaluates whether the linking of abstract top-level requirements to more detailed ones (Feature Level and down) is possible (i.e. whether it is possible to determine if an abstract statement is in line with the product or not). If this is the case, and if the abstract Product Level requirements are able to summarize lower level requirements, they can be used to get a big-picture product overview.

## 4.2. PART II

**(Research Question 2)**

*Given a requirements specification ordered hierarchically according to the level of abstraction of the requirements, to what extent is it possible to determine if new requirements should be included in the requirements specification based on the limits set by Product Level requirements?*

**(Research Question 3)**

*Given a requirements specification ordered hierarchically according to the level of abstraction of the requirements, to what extent is it possible to determine at what level of abstraction a new requirement shall be inserted?*

**(Test)**

We evaluate these questions by:

- a) Presenting the requirements specification in its entirety, i.e. all four abstraction levels.
- b) Providing a number of new candidate requirements, and for each of these asking the participants to determine:
  - on what abstraction level this candidate requirement fits, and
  - if it should be included or not in the system based on the already present requirements in the specification (governed by the top-level Product Requirements)

**(Collected Metrics)**

The questionnaire for this part of the study collects metrics on:

- 1) Whether the participants are able to decide (with a plausible motivation) whether to include each requirement or not.
- 2) Whether the participants are able to place (with a plausible motivation) the requirement on a specific abstraction level.
- 3) Whether the participants are able to link each candidate requirement to the existing requirements in the requirements specification. These links serve as a motivation for why a requirement should be included or not.
- 4) The reason(s) the participants give as motivation when a requirement should not be included.
- 5) Time to complete part II.

**(Design)**

The participants are presented with 10 candidate requirements of which 5 should be included according to the study design. The participants have the choice of placing each candidate requirement on Feature, Function or Component Level. Product Level is not offered as an alternative as these requirements are considered locked for the sake of the study. Subsequent to placing the requirement on a certain abstraction level the participants are asked to decide whether it should be included in the system or not, and to motivate their decision. If they choose to include the candidate requirement, it should be linked to the existing requirements on the abstraction level above by giving a direct or indirect link to one or several Product Level requirements. If they choose not to include the candidate requirement the participants are asked to motivate why, using the categories "Not supported by Product Level", "Undeterminable Level", "Cannot be linked to requirements on the level above", "Contradicts another requirement", "Unclear requirement", and "Other."

### 4.3. PART III

#### **(Research Question 4)**

*What is the background of the participants in the study?*

#### **(Research Question 5)**

*Does the participants' background substantially influence the results of Part I and II?*

#### **(Test)**

Post-test questionnaire with a number of subjective assessment questions.

#### **(Collected Metrics)**

For each of the participants we gather the following information:

- 1) Years in software engineering curriculum.
- 2) Number of study points.
- 3) Years of industrial experience in software engineering.
- 4) English skills (rated: Novice, Moderate, Skilled, Expert).
- 5) Requirements engineering skills (rated: None, Novice, Moderate, Skilled, Expert).
- 6) Assessment of domain understanding before & after the evaluation.
- 7) Assessment of ease of understanding and use of: abstraction levels, placement and work-up, RAM, and requirements representation.

#### **(Comment)**

The purpose of this part is to collect background information on the subjects in order to ascertain whether aspects like language skills, experience etc. influenced the results obtained in Part I and II. Although the subjects are students, their experience and skills often vary considerably, especially since the students are from three different universities, and some of them are international students. The subjects may have several years of industry experience in their background, their English skills may differ, and some have worked in development projects as requirements engineers. Thus, this post-test collects information that is used to describe the subjects in Section 3.2, and serves as a help to understand their answers during analysis.

## **5. OPERATION**

### **5.1. PREPARATION**

The subjects were not aware of the aspects that we intended to study, and were not given any information regarding research questions in advance. They were aware that it was a controlled evaluation in the area of requirements engineering. The evaluation ran over a period of three hours, and all of the subjects were seated in the same room.

Introduction to the study was given during these three hours in the form of a brief slideshow presentation. In this presentation some basic concepts of RAM were shown pertaining to abstraction and levels, and how the requirements are connected etc. The focus of the presentation was put on presenting the instrumentation which consisted of:

- Purpose, i.e. evaluate RAM according to Part I and Part II (as seen in Section 2.2 and Section 4).
- Requirements specification (ordered hierarchically and with connections between the requirements across levels).
- The questionnaires used to gather the results, and how they should be interpreted.

In addition to this it was made very clear to the subjects that the requirements already present in the requirements specification (RAM) should govern their answers. I.e. the statements provided as a part of the questionnaire for Part I should be accepted or dismissed with regards to the requirements present in the specification, and not according to what the individual subjects “thought” or “wanted”. For Part II the same general principle was emphasized. The placement of new candidate requirements on appropriate abstraction level was to be example driven, using the requirements already present in the specification as “good examples”. The decision to accept a new candidate requirement into the product or dismiss it should be governed by the Product Level requirements and the ability to link the candidate requirement to them directly or indirectly.

### **5.2. EXECUTION**

At each university, the evaluation was executed in one day over a four hour session, i.e. the subjects knew that they had four hours to complete the study if they needed the time. The requirements specification used in the first part of the evaluation was handed out at the start of the presentation, to give the participants a chance to familiarize themselves with the

format. After the presentation, the questionnaire for the first part was handed out. As the participants completed the first part, they brought the questionnaire and the requirements specification to the front of the room where they were given the material for part two (a new questionnaire and the complete requirements specification). The start time and delivery time was noted on the questionnaire. As part two was completed the participants were given the last part of the study to complete, after which they were allowed to leave the room. No breaks were allowed during this time. The mean and median times to complete Part I and Part II were around 30 minutes, the shortest times spent on each part were around 15 minutes and the longest were 50 minutes for Part I and 1 hour for Part II. After 3 hours all subjects were finished and had handed in all material. All subjects completed all parts of the study.

## **6. RESULTS AND ANALYSIS**

As a first step in presenting and analyzing the collected data we use descriptive statistics to visualize the results, presented for each part (Part I and Part II). The results are then analyzed and a summary for each analysis step is presented. In Section 7 the research questions are revisited with the help of the analysis results.

### **6.1. PART I**

In total 12 statements were given to the subjects, and they had to decide which of these 12 were in accordance with the product. The product was represented (described by) requirements on Feature, Function and Component Level in the specification. Seven out of the 12 statements were statements designed to be supported by the requirements specification. During the analysis of Part I we have considered those answers that are in line with the study design and aptly motivated as “correct”. If the answer is in line with the study design but missing a proper motivation (i.e. not linked to acceptable Feature Level requirements) or if the answer is not in line with the study design, the answer is considered “incorrect”.

Table 24 shows the average and median results for all 179 subjects, the number of correct answers (second column), and the depth measured in abstraction levels that the subjects had to use in formulating an answer. The depth is calculated following the levels presented to the subjects in Part I, i.e. 1 = Product Level, 2 = Feature Level, 3 = Function Level, and 4 = Component Level.

The subjects’ average is 11.22 correct answers (out of 12 possible) with an average depth of 2.34 (where minimum = 2, maximum = 4). This im-

plies that most subjects were able to use Feature level requirements to render a mostly correct decision regarding the statements.

Table 25 shows the results from Part I but divided by statement instead of subject. For every statement (S1, S2, etc) the number of correct answers is displayed (maximum is 179 correct answers for every statement). In addition the average depth for each statement is displayed.

	# Correct answers	Depth
<b>Average</b>	11,22	2,34
<b>Median</b>	12,00	2,00
<b>StdDev</b>	1,291	0,5831
<b>Min</b>	4,00	2,00
<b>Max</b>	12,00	4,00

**Table 29.** *Number of Correct Answers and Depth Divided by Subject.*

The depth of the investigations in order to render an answer is better illustrated through a histogram, presented in Figure 40<sup>12</sup>. Looking at the correct answers (black bars) most are centered on a depth of 2 (>73%). Looking at the incorrect answers however, the spread is more pronounced as most subject lie between depth 2 and 3. This implies that in the instances where the subjects gave an incorrect answer they also looked deeper in the specification, i.e. the Feature level was not enough and they had to look at the Function (~26%) or even the Component level (~17%).

To check the validity of this implication a Mann-Whitney Test was performed comparing the two populations (Correct and Incorrect) to check whether this was a coincidence. A significance of 0.000004 was obtained. This indicates that there is a significant difference between the two group means, i.e. indicating that the difference between how deep the two populations look is not a coincidence. An independent sample T Test gave a significance of 0.00006.

---

<sup>12</sup> For the purpose of the histogram the values were normalized to enable comparison.

Statement	# Correct	Average Depth
S1	147	2,48
S2	170	2,42
S3	177	2,06
S4	153	2,71
S5	174	2,25
S6	170	2,06
S7	168	2,43
S8	170	2,43
S9	165	2,50
S10	171	2,51
S11	171	2,11
S12	172	2,06
<b>Average</b>	<b>167,33</b>	<b>2,34</b>
<b>Median</b>	<b>170,00</b>	<b>2,00</b>

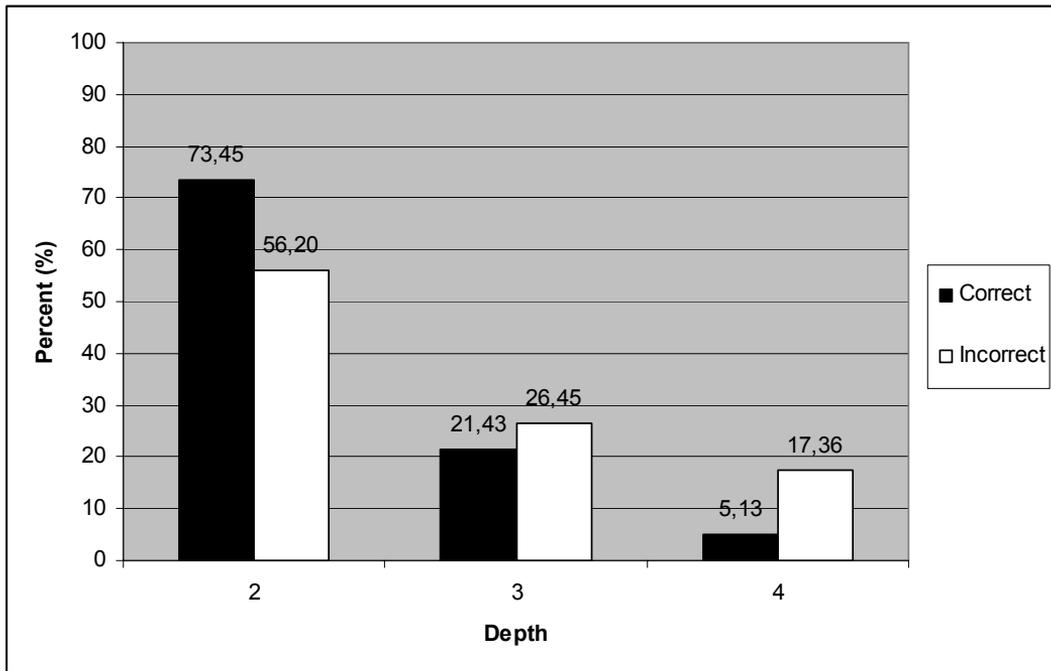
**Table 30.** *Number of Correct Answers and Depth Divided by Statement.*

The Mann-Whitney test was chosen for the large differences in group size between correct and incorrect, see Table 26, and due to the fact that the data are not normally distributed (the Shapiro-Wilk normality test gives  $p < 0.0000000001$ , whereas 0.05 or larger would have indicated a normally distributed data set).

Table 26 presents a summary of the answers regarding depth, where N denotes amount of answers (as to what depth was used) for correct and incorrect determinations. Figure 40 uses the data from the “valid” column. In some cases the answers regarding depth used were missing (8.7% of the cases for correct and 13.6% for the incorrect, as can be seen under the “missing” column). In total 1834 answers that were correct specified their depth and 121 answers for incorrect specified their depth.

	Valid		Missing		Total	
	N	Percent	N	Percent	N	Percent
<b>Correct</b>	1834	91,33	174	8,665	2008	100
<b>Incorrect</b>	121	86,43	19	13,57	140	100

**Table 31.** *Answers ordered according to valid and missing answers for correct and incorrect answers – data for Figure 3.*



**Figure 44.** Comparison of Depth Distribution between Correct and Incorrect Answers.

### Summarizing Part I

A clear majority of the subjects gave a correct answer regarding the statements, i.e. 93.5% of the answers were correct (2008 correct answers / 2148 possible). Further, the subjects rendering a correct answer have looked less deep into the specification (requirements hierarchy) than the subjects rendering an incorrect answer, i.e. Feature level (level 2 from the top) was the most used level for correct answers, while both Feature and Function level were used when rendering incorrect answers. This implies that it is possible to relatively quickly make a correct decision regarding whether or not a particular statement is supported by the requirements specification. If there is hesitation (and a need for more information), there is a larger risk to make a mistake. Or to put it in other words, going down several levels might not improve the correctness of an answer.

## 6.2. PART II

In total the subjects were given 10 new candidate requirements (CRs). For each of the CRs, the subjects were asked to:

1. Place the CR on an appropriate abstraction level *and* motivate this placement through links to one or several requirements on the

abstraction level above. We call this “abstraction level placement”.

2. Decide whether or not to include the CR in the product *and* indicate with links that the CR is in line with other already present requirements in the specification, *or* motivate their decision for not including the CR. We call this “include decision”.

The “product” in this case was represented and described through the requirements already present in the hierarchically ordered specification (on four levels, Product, Feature, Function, and Component).

Table 27 gives an overview of the results divided by candidate requirements (CR1, CR2, etc). The answers are divided into two main categories, visible in the table as *Correct* and *Fail*. These two main categories are divided into subcategories, further detailed below. By dividing the main categories into several sub-categories, we are able to further scrutinize the answers. It should be observed that in Part II, the subjects have to provide four things for every CR; abstraction level placement, include decision, and motivations for these two. Hence, an answer can be correct or wrong to different degrees, depending on which parts of the question the subjects manage to complete correctly. These distinctions are important for the analysis and the reason for the sub-categories.

### **Correct**

The correct category contains those answers where the subjects have managed to use RAM and the requirements specification in a correct way. This means that the subjects have been able to place the CR on an acceptable abstraction level and motivate this by linking to other requirements, and have also been able to make and adequately motivate an include decision. The correct category is divided into two sub-categories, i.e. *correct according to study design and with proper motivation*, and *correct with proper motivation*.

#### **Correct according to study design and with proper motivation**

This category holds the answers that are exactly as intended during the study design, abstraction level placement and the include decision are the same as the study design intended. Also the motivation is according to study design.

#### **Correct with proper motivation**

The answers in this category are close to the category above, but the links used for motivating the answers (i.e. abstraction level placement and include decision) are not always the same as the ones pre-specified during the study design. The links (and motivation for them) stated by the subjects were however scrutinized and considered to be acceptable. This im-

plies that the subject either stated links that were not caught during the design, or that the requirements themselves are possible to interpret differently than what was done in the study design. However, the central issue with the answers in this category is that the placement of the CR and the include decision are ok, and the motivation for these decisions (in the form of links or exclusion motivations) are fully acceptable.

During the analysis, correct answers missing proper and well motivated links were *not* given the benefit of the doubt, and were instead put in one of the Fail category.

*Summarizing the correct category, 884 answers out of a total of 1790 are correct in that they have acceptable abstraction level placements and include decisions, using acceptable motivations. Of these, 637 answers are entirely according to study design, and an additional 247 use a different but proper motivation.*

Candidate requirement	Correct		Fail		
	According to study design and with proper motivation	With proper motivation	Include-OK	Level-OK	Not-OK
CR1	88	47	14	9	21(1)
CR2	27	16	42(7)	56	38(2)
CR3	35	1	51(5)	75	17(1)
CR4	48	31	8	60	32(3)
CR5	56	20	44(7)	41	18(1)
CR6	31	36	33(2)	34	45(7)
CR7	86	44	17	18	14(3)
CR8	88	16	45(8)	21	9
CR9	87	29	17	20	26(3)
CR10	91	7	62(11)	4	15
<b>SUM</b>	<b>637</b>	<b>247</b>	<b>333</b>	<b>338</b>	<b>235</b>

**Table 32.** Summary table showing correct and failed answers.

### Fail

The second category, “Fail”, holds all the answers where some aspects of RAM or the hierarchical requirements specification have been misunderstood (or instances where the participants forgot to fill in information), resulting in answers that do not have an acceptable motivation. Within this category there are three sub-categories, in order to detail which aspects have been misunderstood or understood. These three categories, “Include-OK”, “Level-OK”, and “Not-OK”, are further detailed below. It should be

stressed again that the analysis has been conducted with the objective of finding as many faults as possible with the answers, and when there has been room for doubt the answer has been placed in the category least favorable for the study (i.e. as far to the right as possible in Table 27).

**Fail(Include-OK)**

Fail(Include-OK) indicates that the include decision is properly motivated by links, but that the abstraction level placement is not sufficiently motivated. Numbers within parenthesis in this case indicate missing information regarding abstraction level placement. For example in the case of CR2 42 answers are deemed Fail(Include-OK) and out of these seven answers are missing information regarding abstraction level placement.

**Fail(Level-OK)**

The next sub-category Fail(Level-OK) is the opposite to Fail(Include-OK). The CR abstraction level placement is sufficiently motivated but the include decision lacks an acceptable motivation.

**Fail(Not-OK)**

Fail(Not-OK) includes answers without any acceptable motivation for neither abstraction level placement nor include decision. Numbers within parentheses in this case indicate missing information regarding abstraction level placement or include decision. As these answers are considered entirely wrong no further detail is given about answers in the Fail(Not-OK) category.

*Summarizing the Fail category, in 333 out of 906 answers in the fail category the subjects managed to make an acceptably motivated include decision, in 338 answers an acceptably motivated abstraction level placement, and in 235 answers both placement and include decision were missing or not acceptably motivated.*

**Summary of Table 27**

*Table 27 shows that the subjects answered correctly regarding both abstraction level placement and include decision in  $637 + 247 = 884$  cases (49.4%). In  $637 + 247 + 333 = 1277$  cases (71.3%) the CR was with a proper motivation accepted as a part of the system or dismissed. In  $637 + 247 + 338 = 1222$  cases (68.3%) the concept of requirements abstraction was correctly used with proper motivations.*

**Studying Subjects' Consensus**

In this section we investigate whether or not a general consensus can be seen in the subjects' answers. In an industrial setting the subjects would have cooperated through e.g. consensus meetings, and in this section we discuss possible outcomes of such a work practice. In addition some answers from Part II (CRs) are analyzed further.

For the sake of this discussion we consider the study design to be produced by domain experts with considerable training in RAM, as well as the particular requirements specification. The reason for this is to have a benchmark with which the subjects' consensus can be compared.

Table 28 shows the average results of the subjects' answers with respect to include decision and abstraction level placement. The first column indicates the answer intended in the study design for every CR, "1" indicating that the CR should be included in the product, and "0" indicating that it should not. Coding the subjects' answers to "1" or "0" in the same way enables us to calculate an average for each CR. For six of the CRs the subjects' consensus is fairly close to the answer intended in the study design. For example in case of CR1 "1" is to be compared with "0.93". This means two things. First, it means that the subjects' answers are in line with the intentions of the study design (i.e. expert product managers' opinions). Second, the relatively high number means that there is a large consensus among the students that this CR should be included. If this had been a requirements discussion meeting among managers there is a good chance that the consensus would be to include the requirement in the product.

In four cases such a consensus meeting could render the wrong decision (i.e. for CR2, CR3, CR4 and CR6), since there is less than 2/3 majority for the decision. There is no case where a clear majority of the subjects have an opposite opinion to the study design. Rather, the include decisions could go either way as the subjects' opinions were divided equally on the issues. Below, these four CRs are discussed further.

### **CR2**

In the case of CR2 the subjects that wanted to include the CR motivate their decision through linking to already present requirements in the specification. CR2 is (unintentionally) formulated such that it can be interpreted to fit with already present requirements. The same is true for the opposite, i.e. a subject average of 0.47 could go either way although there is a tendency to dismiss the CR in-line with study design.

	Study Design Include Decision 1 = Include 0 = Do not Include	Subject Average Include Decision (Consensus)		Study Design Abstraction Level Placement	Subject Average Abstraction Level Placement (Consensus)
CR1	1	0,93		3	2,75
CR2	0	0,47		2	2,75
CR3	0	0,50		3	2,60
CR4	1	0,44		2	2,16
CR5	0	0,35		3	2,77
CR6	1	0,43		3	2,99
CR7	1	0,84		3	2,71
CR8	0	0,17		3	2,80
CR9	1	0,80		3	3,07
CR10	0	0,10		3	2,66

**Table 33.** Average (consensus) answers in comparison to study design regarding include decision and abstraction level placement (significant deviations marked yellow).

### CR3

The case of CR3 is similar to the case of CR2. Several subjects linked to already existing requirements motivating their decision to include CR3. Since the formulation of CR3 is such that this was possible, and the fact that there is no explicit requirement contradicting CR3 (e.g. on Product Level) half of the subjects considered it to be included. It should be noted that explicit support for CR3 is missing as well, which is the motivation behind the study design decision that CR3 should not be included. In a consensus discussion (e.g. a requirements meeting) for CR2 and CR3 the issue of uncertainty would be discussed and should optimally yield a decision to either add a requirement on the Product Level explicitly supporting the inclusion of the CRs, or a slight reformulation of existing requirements to explicitly exclude them. The Product Level requirements are of course subject to change in a real product management/requirements engineering situation.

### CR4

In the case of CR4 the subjects were more inclined to dismiss requirements than was anticipated during the study design. Although the study design provides opportunities for including a requirement like CR4, the subjects did not agree with this since explicit support for CR4 was missing in the Product Level requirements (according to the views of the subjects).

## CR6

In the case of CR6 the divergent view of the subjects can be explained to some extent with that several of the subjects misunderstood or misinterpreted CR6. The requirement is a Function Level requirement and hence rather technically specific. A lack of knowledge regarding web technology made many subjects believe that the CR contradicted other requirements already present in the requirements specification, in particular that the use of a web interface to the system precluded the use of IP-address locking, which is not the case.

In the two rightmost columns of Table 28 the intended abstraction level and the average of the levels specified by the subjects is presented. Overall, the subjects' consensus and the study design intentions are comparable, with the exception of CR2, where the subjects' consensus places the CR on level 3 (Function Level) while the intended level was level 2 (Feature Level). Because the requirements model is example-driven, there is often not a definitive answer whether a requirement should be placed on a specific level or on one of the neighboring levels. In this case, there are examples on both the feature and the function level of requirements that can be considered similar to CR2.

*In summary the consensus whether or not to include a candidate requirement is most of the time comparable to what was intended in the study design. The exceptions to this can be traced to lack of domain knowledge among the subjects, alternatively some insufficiencies in the study design (i.e. the requirements themselves could be misinterpreted). In these cases the subjects are divided fairly equally between accepting or dismissing a CR, as could be expected.*

*The consensus regarding abstraction level placement is close to the intention of the study design in all cases but one (CR2).*

## Reasons for Dismissing CRs

Figure 41, finally, gives an overview of the reasons given by the subjects for dismissing a CR rather than including it in the product. Three motivations are in clear majority, i.e. that the CR is not supported by any requirements on the product level, the CR cannot be linked to any requirements on the abstraction level above, and that the CR contradicts another requirement already present in the requirements specification. These three reasons were also the ones used in the study design.

We also see that the remaining three categories (i.e. undeterminable level, unclear requirement, and other comments (free text comments)), that represent perceived problems in the study design, were rarely used and constitute a total of 16%. It should be noted that the subjects were able to mark more than one reason for excluding a candidate requirement.

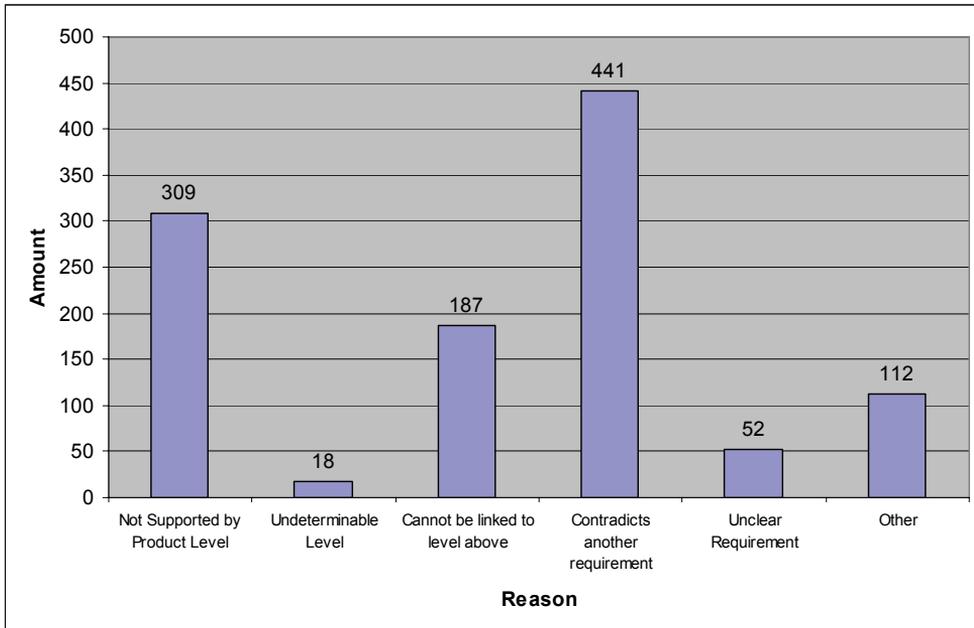


Figure 45. Reasons given by subjects when they chose not to include a CR.

In summary, the subjects are able to clearly and in accordance with the opinions of experienced product managers discern the reasons for dismissing a CR.

## 7. RESEARCH QUESTIONS REVISITED

### 7.1. PART I

#### (Research Question 1)

*Given a requirements specification ordered hierarchically according to the level of abstraction of the requirements, to what extent do the top-level (most abstract requirements) connect to the rest of the requirements?*

In the evaluation 93.5% of the answers are correct. The subjects giving correct answers used requirements at level 2 (Feature Level) to render decisions, while the subjects that answered incorrectly had a depth of between 2 and 3. This indicates that in order to get a correct answer it is in general only necessary to go as deep as Feature Level. This indicates that Product and Feature Level are tightly coupled, which is a prerequisite for RAM and the idea behind abstraction levels.

The large amount of correct answers indicates that Product Level requirements are indeed general enough to summarize several requirements on lower levels. However, they are still concrete enough to determine whether they represent the product or not, and can thus be used to get a system overview.

## 7.2. PART II

### (Research Question 2)

*Given a requirements specification ordered hierarchically according to the level of abstraction of the requirements, to what extent is it possible to determine if new requirements should be included in the requirements specification based on the limits set by Product Level requirements?*

A total of 71.3% of the answers include or exclude the candidate requirements using acceptable motivations. In other words, more than 2/3 of the answers are correct in terms of a well motivated include decision. This indicates that the ability to compare requirements with similar requirements on the same abstraction level and the necessity to be able to link a new requirement to the existing hierarchy indeed supports the decision process. Time constraints prevented us from producing a non-hierarchically structured requirements specification to run a control group. However, offering all of the requirements structured according to for example functionality would have forced the subjects to read through much more information (requirements) before being able to render an include decision. Looking at the results regarding the relation between depth and answering correctly in Part I, there is indication that more information (larger amount of requirements) as a basis for a decision does not render a better decision, rather the opposite. In addition there exists a multitude of ways to structure requirements, thus it is not possible to compare RAM structured requirements to a specific “normal” or “default” structured set of requirements.

Strictly speaking we cannot say that RAM structured requirements are better or worse than the current situation, since it depends on what the “current” situation is. However, the results indicate that it is much better than answering at random<sup>13</sup>, and in conjunction with the answer to re-

---

<sup>13</sup> A rough calculation yields 1/2 chance to make the correct include decision, 1/3 chance for correct abstraction level placement, approximately 1/5 chance to

search question 3 we are convinced that it is a real help for product managers.

**(Research Question 3)**

*Given a requirements specification ordered hierarchically according to the level of abstraction of the requirements, to what extent is it possible to determine at what level of abstraction a new requirement shall be inserted?*

A total of 68.3% of the answers place the candidate requirements on their intended abstraction level using acceptable links to other requirements. It thus seems to be relatively easy to determine the abstraction level of a requirement by studying the “good examples” that the existing requirements specification provides.

**7.3. PART III**

**(Research Question 4)**

*What is the background of the participants in the study?*

**(Research Question 5)**

*Does the participants' background substantially influence the results of Part I and II?*

These questions were mostly used in the analysis of the results. The information gained regarding research question 4 is found in Section 3.2. For research question 5, we were unable to find any difference between the performance of participants with more experience and those with less experience (industry experience or experience with requirements engineering). It should be noted, however, that the differences in terms of experience between the participants were not very large (a standard deviation of one year for the industry experience). The homogeneity of the group may be the reason why no difference was found.

---

link to at least one appropriate requirement on the level above. This would yield about 1 in 30 chance to “guess” correctly.

## 8. CONCLUSIONS

In this study we evaluate aspects of a hierarchical requirements abstraction model – RAM – as a preliminary step before conducting full scale industry piloting. Using industry as a laboratory through pilots is a powerful tool and a part of building trust and commitment prior to technology transfer in the form of process improvement activities. However, prior to using valuable and hard to obtain industry resources we consider it prudent to evaluate the ideas in an academic laboratory setting. Initial evaluation of the model in academia intends to investigate whether or not industry piloting is mandated, or whether RAM needed further refinement or even redesign before directly involving industry in validation activities. It is thus important to get early indications whether or not the fundamental features inherent in the current model are useful and usable, and if the assumptions implied by the model hold.

RAM is intended to aid product managers and others involved in requirements engineering in getting an overview of a product using relatively few abstract Product level requirements. Using the same Product level requirements as a basis (and the requirements linked to them), managers should be able to include or dismiss new incoming requirements in a repeatable way. That is, not dependent solely on the person performing the work, and place new requirements on appropriate abstraction levels in the model.

The participants in this evaluation were given a relatively large amount of requirements (120 in total) and were asked to accomplish a considerable amount of work in a short amount of time. The participants were expected to form an understanding of the concept of requirements abstraction and hierarchically structured requirements specifications, understand the domain (read and understand the requirements), and then solve the tasks in Part I and II. Very little training of the model was given to the participants, and they also possessed little prior knowledge regarding the domain when compared to a product manager. Considering these aspects and the time spent the results produced are encouraging and point towards both high usability and usefulness of RAM.

The characteristics of industry are also relevant as real-life usage of RAM should be easier than during the evaluation. In industry, requirements engineering is not performed in isolation (as was the case in the evaluation); regular meetings as well as official and unofficial conversations and discussions help in sharing views and forming consensus as well as a shared understanding. From this perspective the results obtained in

the evaluation are even more promising. The consensus results presented in Table 28 indicate that had there been cooperation and exchange between the participants even better results may have been achieved, since cooperation often yields better results than working individually. In addition, industry product managers are often senior practitioners well versed in both their specific domain and in the field of requirements engineering. Given this some of the mistakes made by the subjects participating in the controlled evaluation, where lack of domain knowledge was evident, could probably have been avoided in an industry setting.

As indicated by the results of the controlled evaluation in this article, the concept of abstraction and the usage of abstraction levels seems to be fairly intuitive. This is also useful in a scope outside of RAM. For example, the possibility to obtain homogenous requirement abstraction levels using RAM may facilitate requirements prioritization, as requirements on the same level of abstraction are easily compared in contrast to comparing requirements on different levels of abstraction.

A potential issue with the model is that since the abstract requirements on the upper levels (especially Product level) are few, their individual impact on the product is considerable when they are used as the principal decision support (getting a quick overview of the product). This implies that the demands on these requirements are high in terms of being well-formed and unambiguous. In addition they need to be explicit. In this study, lack of explicit support among the product level requirements was a motivation often used by the participants to dismiss a candidate requirement.

In industry we may assume that the users of RAM possess domain knowledge as well as at least minimal training in using RAM together with some tool support. These factors should help ensure even greater usability and usefulness of the model than was observed during the controlled evaluation in this article.

## **8.1. FUTURE WORK**

As a future study we intend to run control group evaluations performing the same tasks but on requirements specifications structured in several “traditional” ways (e.g. structured according to functionality, stakeholders, or use cases). The comparisons of the results could give further information regarding the usefulness and usability of RAM, after which the model may be considered mature enough for industry piloting. Several of these evaluations are planned for the near future and preparations for some of them are already underway.

# Chapter 7

---

## Test-case Driven Inspection of Pre-project Requirements - Process Proposal and Industry Experience Report

*Tony Gorschek, Nina Dzamashvili - Fogelström*

Requirements Engineering Decision Support Workshop held in conjunction with the 13th IEEE International Conference on Requirements Engineering, Paris, 2005.

### **Abstract**

Requirements inspections can be used not only for defect removal in projects but also applied pre-project. This to assure that managers have good-enough requirements for product and subsequent project planning activities, such as requirements selection for realization and estimation. This chapter introduces an inspection process designed to address the needs of companies with limited resources operating in a market-driven environment. The inspection technique presented here utilizes well-known and recognized concepts like perspective based reading, while also introducing new application ideas. The reuse of testing expertise and inspection artifacts help spread the cost and benefit of inspections over several development phases. Initial experiences from industry application report on positive reactions from managers, testers and developers.

## 1. INTRODUCTION

Inadequate<sup>14</sup> (low quality) requirements can have severe consequences for a product development effort. These consequences are mainly due to the fact that problems in requirements filter down to design and implementation [3, 5, 7]. 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 [19], and several other sources indicate that inadequate requirements are the leading source for project failure [13-18].

The importance of producing good-enough requirements can be considered as crucial to the successful development of products, whether it be in a bespoke [3] or market-driven [7] development effort – this is generally not disputed. There are however clear indications that requirements quality is lacking in industry and thus low quality requirements being a main contributor to project failure. Both from the perspective of being inadequate as basis for development (in projects), but also in terms of being inadequate as decision support material for pre-project requirements selection and estimations [13-18, 118, 168].

Inspections, first presented by Fagan in 1976 [169], are recognized as a powerful tool with regards to finding and removing defects, thereby increasing quality in development artifacts (e.g. requirements). There are reports indicating that 50-90% of the defects can be caught [82] using inspections.

Equally importantly – inspections offer the possibility to catch defects in the early stages of development thus reducing cost of rework. This is the main reason why there is a general consensus amongst most experience reports and research that inspections on requirements is very beneficial [83, 86], and recommended as a part of the development process, see e.g. CMMI [22] and ISO/IEC 15504 [45].

---

<sup>14</sup> The word “inadequate” is used as a term denoting requirements that are not good-enough, i.e. of low quality in terms of being incomplete, unambiguous, incorrect, conflicting and so on. All of these issues can be seen as defects. In this chapter increasing “quality” denotes addressing these mentioned issues i.e. removing defects [167]. (1998) IEEE Standard for Software Verification and Validation. IEEE Std. 1012-1998

Furthermore, the recommendation given to organizations with limited resources for reviews and inspections is to prioritize requirements inspections over e.g. design and code inspections [82, 86].

In spite of the seemingly obvious benefits and explicit recommendations inspections are not commonplace in industry, Ciolkowski *et al.* [170] reports that only about 40% perform any type of reviews (on any development artifact). The number for inspections on requirements is even lower.

The explanation to this could be the fact that inspections are often considered as labor intensive and therefore costly process [171]. Moreover the return of investment is not immediate causing skepticism among e.g. industry managers [172].

This chapter offers a proposal for an alternative inspection technology, i.e. *Test-case Driven Inspection of Pre-project Requirements* (TCD Inspections). It utilizes test-cases as a tool for inspection efforts - involving the Verification and Validation department (or more precisely “testers”) at an early stage of the development process (pre-project).

The main idea behind TCD Inspection is reuse, i.e. making the inspection efforts and artifacts beneficial to several development stages (both before and during the projects) and thereby “spread” the cost and time demands of the inspection process.

In summation the TCD Inspection can be characterized by:

- Involving experienced testers in the inspection process, thus reducing the educational and training costs of inspectors.
- Using tester’s competence and effort for double purposes, i.e. testing and inspections. In software development projects it’s common that testers review requirements specification in order to plan and execute testing activities. Thus using testers in pre-project inspections can be seen as an effective usage of company’s resource.
- Producing reusable inspection artifacts: test-cases produced during inspections are to be used in subsequent stages of the project such as during implementation (as an appendix to requirements offering another view to e.g. developers) and testing.
- As managers (writing the requirements) and testers are directly involved in TCD Inspections the learning effect (becoming better at writing and formulating requirements) is achieved.
- By performing inspections pre-project (prior to final selection of requirements for realization) managers get better requirements as input for requirements selection and cost estimations.

This chapter presents both the TCD Inspection process and an experience report from industry application.

The structure of this chapter is as follows. Section 2 gives an overview of traditional requirements inspections. Section 3 provides analysis of benefits and drawbacks of inspection techniques identified by industry and academia, as well as motivation for finding new approaches for performing inspections. In Section 4 the TCD Inspection process is presented, showing how early requirements inspections are performed using test-cases. In Section 5 a discussion is presented where the traditional and alternative inspection technologies are compared and put against each other in order to clarify the contribution of this chapter. The conclusions are in Section 6.

## 2. INSPECTION - GENERAL OVERVIEW

This section starts by presenting the classical inspection process as defined by Fagan in 1976 [169]. Next the basics of the Perspective Based Reading (PBR) technique are explained. The reason for focusing PBR is twofold: PBR is identified as an effective reading techniques while performing requirements inspections [173, 174], and second, the TCD Inspection process presented in this chapter (see Section 4) is related to the ideas behind PBR.

### 2.1. INSPECTION PROCESS

Inspections in general can be defined as a process of visual examination of software products with the goal of identifying defects [167, 173]. Software products, in this context, are different documents produced in software development, such as requirements specifications, high level designs, source code, test plans and so on. Defects cover errors, deviations from specifications, software anomalies and so on.

The inspection process itself can be characterized by inspection team members (who they are and their roles) and inspection team size, inspected artifact(s), utilized reading technique and the number of inspection meetings held. Classical Fagan Inspection process includes steps described below [86, 169, 170, 173].

#### 2.1.1. PLANNING

The inspection team is formed and roles are formulated and assigned. These roles can be summarized in the following manner:

- A. **Moderator** - Team manager and inspection coordinator. Selects team members, sets schedule, and assigns resources and so on.

- B. **Producer** (a.k.a. Author) – The individual who produces the product (e.g. design, requirements, code and so on). It is the producer’s responsibility to assure that the product is ready for inspection. The producer also supports the other parties in the inspection by e.g. answering questions and offering support.
- C. **Reader** – The individual who paraphrases the design or code during the meeting. I.e. reads the product being inspected at the inspection meeting and thus supporting the moderator and the inspection process by taking focus of the producer (e.g. alleviating potential problems with the producer becoming defensive).
- D. **Inspector** – The individual inspecting the product. The inspector has the responsibilities of thoroughly familiarizing him/herself with the product and inspecting it in order to find and document defects.
- E. **Tester** – The individual who inspects the product from a testing point of view.
- F. **Manager** – This role helps establish what products (or maybe what parts of a product) is to be inspected and selects the moderator. In some instances this role also takes some managerial weight of the moderator by taking over issues such as resource allocation and training. In addition to role assignments the overall goal of the inspection is formulated [17, 21].

### 2.1.2. **OVERVIEW MEETING (OPTIONAL)**

This optional step is used for familiarizing the inspection participants with the product if needed.

### 2.1.3. **DEFECT DETECTION (A.K.A. PREPARATION)**

Each team member reviews the material independently to get an understanding of the product. This is the phase were the defects are found and documented. Looking at the roles described above roles C, D and E are the ones performing the actual reviews [86].

Subsequent to Fagan’s introduction of inspections several reading techniques have been examined and developed as a part of the inspection technology. An overview of these can be seen in Section 2.2.

### 2.1.4. **INSPECTION MEETING (A.K.A. EXAMINATION)**

This is the traditional approach for collecting and discussing the defects (not the solutions to a defect) to reach consensus about a joint number of defects. A defect can be accepted as a defect or refused as a false positive. A false positive is an item suspected to be a defect but later on turns out

not to be. The moderator documents the findings to enable all issues be passed along to the rework and follow-up stages.

### 2.1.5. **DEFECT CORRECTION (A.K.A. REWORK)**

The defects are corrected by the author. Some products may need to be re-inspected and reworked several times before all the defects can be considered to be corrected.

### 2.1.6. **FOLLOW-UP**

The moderator verifies that all defects are corrected. During this phase there is generally also feedback to the inspection participants as to the inspection results.

## 2.2. **READING TECHNIQUES**

There are several reading techniques available, and they are mainly used as a way to improve defect finding efficiency in the defect detection step during the inspection. Some of the most well known reading techniques are ad-hoc reading (not really a “technique” per se), checklist-based reading, scenario-based reading and perspective based reading [18, 21, 22, 24, 25].

According to number of studies, out of the above mentioned reading techniques scenario-based reading and perspective-based reading methods have shown to be superior compared to ad-hoc and checklist-based reading, see e.g. [83, 174, 175].

Looking at the specific case of requirements inspections Basili [176] states that two types of reading techniques have been found suitable, i.e. scenario-based reading and perspective-based reading. The reasoning behind this being that the structure and dimensions (perspectives) of these two are superior to checklist-based and ad-hoc reading when it comes to inspections in general, but especially for requirements as they are used by multiple stakeholders (that have multiple perspectives) and are the foundation for the development effort.

Perspective-based Reading focuses on conducting inspections using the perspectives of different stakeholders when inspecting a document [173]. A distinguishing feature of this reading technique is that the reviewers build and use models to uncover defects. For example using this technique an inspector can review a requirements specification utilizing the perspectives of a system developer, a system end-user and/or a system tester. In the case of using the system user perspective the created model would consist of e.g. a user manual or use-cases, in the case of the devel-

oper perspective e.g. a high level design, and in case of the tester perspective e.g. a set of test-cases [175]. Thelin et al. report on the efficiency and effectiveness of using PBR with the perspective of users (thus creating use-cases for the purpose of performing inspections) [177].

TCD inspection, which is presented in this chapter, uses test-cases created by testers for requirements inspection. The detailed description of TCD inspection is found in Section 4.

### **3. BENEFITS AND ISSUES – INSPECTION EXPERIENCES FROM INDUSTRY AND ACADEMIA**

As all technologies, inspections have both benefits and issues (drawbacks/problems). This section presents an analysis of some negative issues and positive benefits related to inspections. The result of this discussion builds a motivation for suggesting TCD inspection, presented in Section 4.

#### **3.1. BENEFITS OF INSPECTIONS**

The overall most significant benefits of inspections is that 50-90% of defects can be caught [82], and equally importantly - they can be caught in the early stages of development thus reducing cost of rework. This is the main reason why there is a general consensus amongst most experience reports and research that requirements inspection is very beneficial [83, 86]. I.e. the recommendation given to organizations with limited resources for reviews and inspections is to prioritize requirements inspections over e.g. design and code inspections [82, 86], this mainly due to the filtering down-effect of defects in requirements. The reasoning being that the earlier problems like incompleteness, ambiguity, errors, conflicts, and so on can be caught the less effort has to be put on fixing the problems and reworking parts that have been influenced by the problems in question. I.e. problems filter down from requirements.

This is not to say that defects cannot be introduced later in development, i.e. design, code and so on may also benefit a great deal from inspection.

Another obvious benefit with inspections is that all products (from requirements to code) can be inspected, whereas other V&V activities, e.g. tests cannot be performed on all products as they demand some sort of executability (e.g. execution of code or a formal specification). In addition to these benefits there is a positive spin-off effect with inspections, namely the learning effect. As individuals in an organization perform (or are a

part of) inspections they learn about defects, and subsequently can avoid making the same mistakes repeatedly [82, 86].

### 3.2. ISSUES WITH INSPECTIONS

Seeing the benefits with inspection one might assume that inspections are used extensively in industry as a way to detect and remove defects, thus increasing quality. This is however not the whole truth. Ciolkowski et. al. [170] reports that only about 40% perform any type of reviews.

The main issues with inspections seem to be related to a number of main points that to some extent may explain the reluctance of organizations to adopt inspections in their regular process [84, 170, 178, 179]:

- **Time pressure and Cost.** Development projects are under constant time pressure, thus performing inspections (which are generally time-consuming) are not perceived as a clear benefit over e.g. decreasing time-to-market for the product.

In addition to being time-consuming inspections are labor intensive. An inspection team consisting of e.g. 5 people tie up crucial resources over an extended period of time. In addition inspection result is often tied to the competences of the inspectors, resulting in that good results demand good inspectors. This means that the best results demand that the best personnel be tied up in an inspection during development, not an ideal situation for organizations with limited resources.

In addition to this the preparation of inspections, documentation and inspection meetings are both labor intensive and time consuming.

- **Difficulty in quantifying the ROI.** Looking at the issues described above the logical response is that the benefit of inspections outweigh the cost regarding needed resources and time for both training and inspection execution. The main issue seems to be that the benefits of inspections are not easily quantifiable. E.g. how is the value of a removed defect in a requirement specification ascertained? There are proponents of stringent metrics collection during inspection as a way to measure benefit [173], but this entails having measurement practices in place over a period of time (which in itself can be very costly [60]) to quantify inspection benefit. There is also the question of organizations with limited resources; on what basis do they initially adopt inspections as a way to increase quality, especially if an alternative investment could be faster time-to-market or the realization of more requirements?

- **Lack of training.** As mentioned earlier, a certain amount of training may be necessary in order for inspectors to master the different reading techniques. In addition to this the moderators need to be trained in the inspection process.

### **3.3. MOTIVATION TO FIND ALTERNATIVES**

High cost and time-intensive work can be seen as issues that create concern when it comes to the adoption of inspection technology, especially for Small and Medium Sized Enterprise (SME). This is supported by the fact that most experience reports from industry describing inspections are from large organizations with (in comparison to SMEs) vast resources, see e.g. Basili's work with SEL and NASA [60, 84, 174, 175].

The difficulty to measure and quantify benefit contributes to raising the bar, i.e. making inspections a harder sell to management. Without the possibility to convince management of the benefits inspections will not be implemented.

Most recommendations to solving this problem in academia are centered on gathering metrics that will prove benefit. The problem is that this in turn implies that measurement programs have to be in place, i.e. the collection and analysis of metrics over an extended period of time. This in itself is costly and not commonplace in SMEs [60].

The alternative inspection technology suggested in this article aims to address the problem connected with inspection costs via effective use of company's resources. The main idea behind this technology is reuse, i.e. making the inspection's results and artifacts beneficial to as many development stages and people as possible.

This makes it possible to "spread" the cost and time demands of the inspection over several development stages, in an attempt to lower the bar for organizations with limited resources to use inspections as a tool.

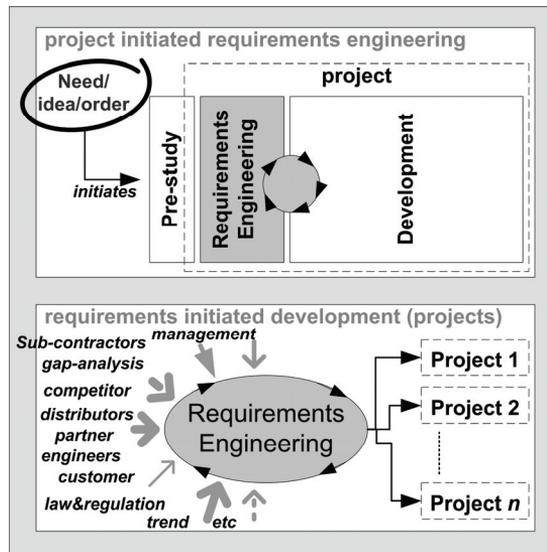
In addition the inspection technique aims at producing better quality requirements and thus providing better decision support that can be used directly by company management.

## **4. TEST-CASE DRIVEN INSPECTION**

TCD inspection was developed as a part of a SPI activity conducted at Danaher Motion Särö AB (DHR).

The DHR development organization was faced with a market-driven product centered development situation (see Figure 46), characterized by the need to have high quality requirements pre-project. This was crucial in

order to provide decision support for requirements selection and estimation of future projects, i.e. enable product planning activities.



**Figure 46.** *Requirements' role in the market driven development process.*

Figure 46 depicts a market-driven development situation where the requirements flow is not limited to a development instance (e.g. a project), but rather is continuous in nature, and the requirements themselves act as the catalyst for initiating development. Requirements engineering in this situation is not limited to projects but also a part of e.g. product management activities which are pre-project in nature.

Introducing traditional requirements inspections at DHR, as described in Section 2, (and performing them pre-project – and not after project initiation), would be connected with high cost due to large number of incoming pre-project requirements. In addition, the requirements at this initial product planning stage are not allocated to a project, implying that the explicit decision to implement the requirement has not yet been taken. In order to meet the above described restrictions an alternative inspection process was developed. The overall goal was to achieve high quality requirements pre-project and at the same time keep inspection costs low through reuse. The following section provides an overview of the inspection process introduced and piloted at DHR (for details on the SPI activity at DHR see Chapter 2, 3 and 4).

## 4.1. TCD INSPECTION

The TCD inspection process introduced and piloted at DHR consists of three steps as can be seen in Figure 47. In Step 1 a product manager(s) initially reviews and selects requirements as an iterative part of the initial formulation/specification. Requirements come from multiple sources (see), and some requirements are discarded during this initial step. This is mostly an ad-hoc process where the product manager utilizes personal as well as coworkers' expert knowledge and experience to perform an initial "trial" of the requirements, deciding which requirements will be passed on to Step 2. As a backdrop to this initial trial the product manager is governed by the limitations and goals set up by management through product strategies. They explicitly or (more commonly) implicitly govern the goals and direction of development activities, see [21].

Formulated/specified requirements are the input to Step 2. These requirements are specified according to a template and each includes attributes such as ID, Title, Description, Benefit/Rationale, Source, and so on. It is during Step 2 that the TCD inspection itself is performed creating test-cases on the requirements. As the inspection is performed some additional requirements may be discarded or (more often) postponed as a result of the refinement of the requirements (thus adding to the decision support for the product manager). This enables him/her to remove requirements that are inappropriate or of obvious low priority at the time of the inspection. Step 2 will be described in further detail in the following sections.

In Step 3 the inspected requirements are prioritized and project planning is performed. Some requirements may be discarded or (more often) postponed during this phase also.

The initial review of the requirements in Step 1 is meant to ensure that mostly viable requirements are inspected in Step 2. The requirements inspected refined through test-case creation, enabling dismissal and postponement of some requirements at an early (pre-project stage), as well as passing requirements of higher quality to be passed to Step 3.

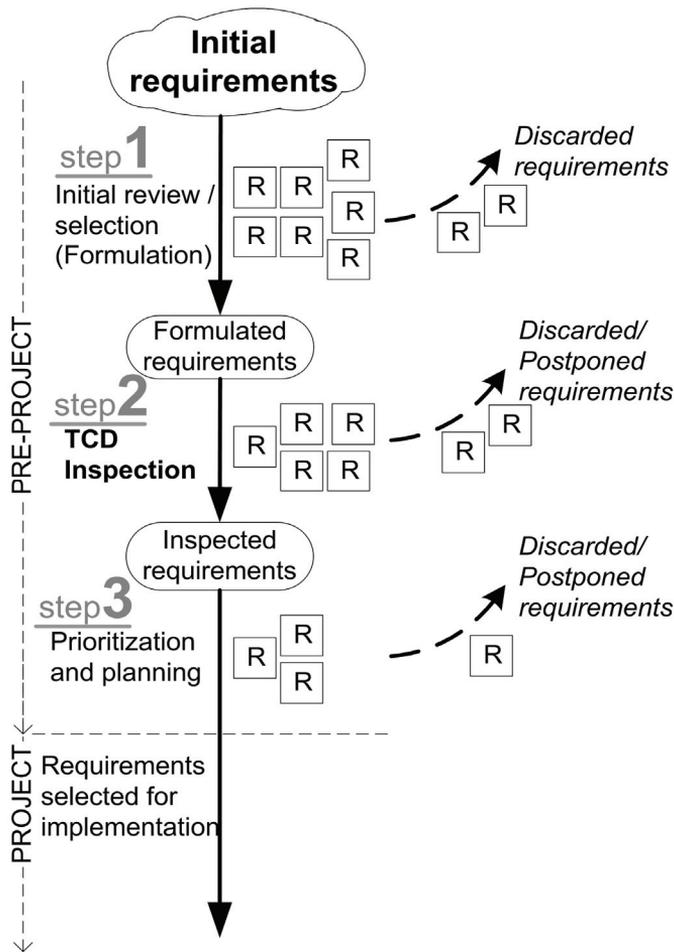


Figure 47. TCD inspection process steps.

#### 4.1.1. TCD INSPECTION ROLES

The following roles are relevant for TCD inspections:

**Product Manager (PM)** – this role was chosen based on the fact that PMs elicit and specify requirements. In addition PMs are product experts with an overview of both technical and non-

The PM role in a TCD inspection is comparable to both the Producer and Manager roles as described in Section 2.1.1. The PM is the “producer” of requirements, and in addition decides what requirements are to be inspected.

**Tester** – this role was chosen due to the fact that testers are also product experts as they test the system continuously and have to understand and possess an overview of it as well as knowledge about details, without which the creation of test-cases and execution of tests is difficult.

An additional major reason for choosing testers to take part in the inspection of requirements is the fact that they are (or at least should be) used to reading requirements as system tests are to be based on requirements [86].

The tester role in a TCD inspection is comparable to both the Inspector and Tester roles as described in Section 2.1.1. The tester inspects the requirements and assures testability.

#### 4.1.2. TCD INSPECTION PROCESS (STEP 2)

The TCD inspection (Figure 47, Step 2) can be divided into several sub-steps comparable to the ones presented in Section 2.1:

**Planning** - This step involves the PM. The main function of this step is to ascertain what requirements are to be inspected (which requirements are passed to Step 2, see Figure 47), and to allocate resources for this inspection, the main resource being a tester.

**Defect Detection** - This step involves the tester(s) that inspects the requirements. The tools for the inspection are the creation of test-cases based on the requirement, thus ascertaining:

- Testability - if it is possible to create test-cases based on the requirement, the requirement can be considered as testable.
- Completeness - as the tester is an expert in usage of the system, missed functionality, whether it is whole requirements or parts in a requirement, can be caught.
- Non-conflicting – the tester goes through each requirement thoroughly as test-cases are created. During this the tester can catch issues such as conflicts and inconsistencies.

As the tester performs the inspection test-cases are produced. In addition an inspection protocol is created documenting defects caught during the inspection.

**Inspection Meeting** - As the inspection is completed a meeting between the inspector (tester) and the owner (PM) takes place. During this meeting the inspection protocol is reviewed and the test-cases created are gone through.

The inspection protocol serves as specification of what is to be corrected, and the defects are confirmed or dismissed.

The test-cases are gone through to establish that a joint understanding of what the requirements entail. The main reasoning behind this is to as-

sure that the requirements are correct, i.e. if both the tester and the PM have the same understanding of a requirement the formulation of it is assumed good-enough. Whether or not it is correct from the requirement's sources view is up to the PM.

**Defect Correction** - The PM corrects the defects agreed upon during the inspection meeting. If there is a need the requirement's source is elicited for additional information.

**Test-case Completion** - As the PM completes the corrections the updated/new requirements are delivered to the tester. The tester then completes the creation of test-cases, effectively confirming the corrections and re-inspecting the requirements in question.

In some instances there might be a need for additional corrections if new defects are caught during this step.

As this step is completed the test-cases are saved as attachments to the relevant requirements. The reason for this is twofold. First, the test-cases can be used to augment the requirements, e.g. a developer can use the test-case in addition to reading the requirement to ascertain that he/she understands what is meant. Second, if a requirement is changed, the likelihood of updating the test-case in question is greater if it is directly visible and assessable.

## 5. DISCUSSION AND STUDY RESULTS

Looking at the TCD inspection there are several main differences from a traditional inspection described in Section 2. These can be summarized as follows:

1. Inspection roles and team size – TCD inspection has two distinct roles, PM and tester.
2. Reusable artifacts – the artifacts produced during TCD inspections are intended to be reused in later project stages.
3. Pre-project requirements – the TCD inspection was initially developed to make inspections of pre-project requirements feasible.

The combined effects of these features can result in lowering inspection costs, whereas contributing to increased quality of inspected requirements.

### 5.1. TCD VS. TRADITIONAL APPROACHES

In this section the main features of TCD inspections are explored and motivated in relation to the generic inspection technology described in Section 2.

### 5.1.1. **INSPECTION ROLES AND TEAM SIZE**

The roles of the TCD inspection are comparable to traditional inspections, but with some differences. The minimal size inspection team in TCD inspection consists of only two persons, PM and a tester. This is not new as such, two-person inspections have been used before [173], here traditionally one person is the Author and one the Inspector. In a TCD inspection the roles of the participants are “double” in nature and explicitly described pertaining to each role’s responsibilities.

The PM is both Author of the requirements and Manager (the PM owns and interprets the requirements and often specifies them as well). This is logical since the PM is a manager responsible for the product planning (what requirements are selected for implementation) and the requirements themselves.

The reading technique used during TCD inspections can be seen as perspective-based. The tester inspects the requirements from the perspectives of a tester, i.e. ascertaining testability of the requirement, but there is an added perspective, namely the end-user perspective. As the tester inspects the requirements through the creation of test-cases, functionality is inspected, giving the perspective of system end-users, the benefit of this has been presented in previous studies, see e.g. [177].

To assign a tester the end-user perspective seems logical as the tester is not only seen as a testability expert, but also as a system (functionality) expert and an expert at reading and interpreting requirements, fully capable of determining other perspectives than just testability.

It is important to highlight that in TCD inspections the role of a tester is occupied by a tester, not e.g. a developer inspecting requirements from a tester’s perspective, which is usually the case in traditional inspections [83]. This is considered as a benefit for the reasons expressed above, i.e. the competence of a tester regarding inspecting requirements, creating test-cases and ascertaining testability was considered superior to that of e.g. a developer.

It should be noted that two-person inspections have been reported to be as effective as traditional inspections, especially if the two-person team can be considered an expert-pair [178]. The PM and the tester do constitute an expert pair in TCD inspections.

### 5.1.2. **CREATION OF REUSABLE ARTIFACTS**

A main benefit of TCD inspections is that the artifacts are not created solely for the purpose of the inspection, but test-cases can be used for several purposes in addition to the inspection. The obvious use of the test-

case is of course during the testing of the system. There is however another potentially positive effect, namely that the test-cases are attached to the requirements as they are sent to implementation. The test-cases can be used to augment the requirements offering a better understanding and perhaps even decreasing the chance of the requirements being interpreted differently than was originally intended by the PM.

In some traditional inspections the creation of models (e.g. UML models) have been reported as beneficial as the modeling itself increased understanding and thus increased defect detection rate [82]. In the case of TCD inspections test-case are created, using the same basic idea of inspection through the creation of “models”. There is however one basic difference, UML models are generally aimed at inspecting *HOW* something is to be done, not *WHAT* as in the case of requirements and test-cases. Attaching a UML diagram to a requirement and offering it to e.g. a designer could be potentially beneficial, but it could also limit the designer as a UML diagram can be seen as a suggestion of how something can/should be realized, as well as taking focus away from the requirement itself, which should be the basis for what is designed. It goes without saying that models in the form of e.g. UML diagrams are not directly usable during system test, as is the case for test-cases.

### 5.1.3. PRE - PROJECT REQUIREMENTS

As mentioned previously, there are several advantages to concentrate inspection effort on requirements. However, the recommendation of inspecting requirements generally applies to requirements within projects, in which case there is commitment to implement the requirements. TCD inspection could be applied within projects as well, but in the case of the SPI activity at DHR (of which the TCD inspection implementation was a part) the requirements inspected were in product planning state (see Step 2 in Figure 47). This had the advantage of increasing the quality of requirements, and complementing them with test-cases improving the decision support material used for both product planning (selection) and project planning (estimations).

The fact that the inspection was performed at an early stage could also be seen as a disadvantage, as effort was put into inspecting requirements that ultimately may not be implemented (not selected for realization). This is however countermanded by the fact that improved requirements quality increased the chance of irrelevant/not appropriate/high risk requirements were discarded or postponed pre-project (instead of later within project).

Whether or not Test-case Driven Inspection of Pre-project Requirements is advantageous depends on two things, pre-project benefit and cost. Benefit is as the name indicates how much benefit there is to having high-quality requirements as decision support for product planning/project planning activities. Product planning activities include e.g. selecting and packaging of requirements into releases with regards to coupling and cohesion (i.e. cohesion within a package should be maximized, while minimizing coupling between packages).

Project planning activities involve estimations of time and effort with regards to implementation of a requirement, as well as estimating e.g. risk etc.

Cost relates to the ability to reuse the inspection artifacts produced by TCD inspections (test-cases) in projects during development and testing, as well as how beneficial it is to get inspected requirements as input to projects.

## **5.2. STUDY RESULTS**

During the pilot study of the TCD inspection the general view of the participants, which included experienced personnel (one PM and one tester), was that the benefit was substantial. The PM felt that the refined requirements were of very high quality in comparison to the requirements before the inspection, offering much better decision support material for product planning and project planning activities. As the completeness and understanding of the requirements was improved during the inspection it was also possible to discard and/or postpone requirements during Step 2 (see Figure 47), enabling fewer unviable requirements to pass to Step 3, and thus wasting less resources on planning and estimation activities.

An additional benefit perceived by the PM was a learning effect, making the PM better at specifying requirements in terms of the perspectives inspected.

The tester felt that the possibility to see and ascertain testability at an early stage was far superior to getting the requirements post-implementation, as this offered the possibility to faithfully base system test on requirements that were good-enough for this purpose. In addition, a positive spin-off effect was that the activity of creating a realistic test-plan could be started earlier as the requirements were available and testable.

A drawback identified was that non-functional requirements were hard to inspect. Not necessarily in terms of testability, but from the perspective of conflicts. It was hard to ascertain if e.g. a performance re-

quirement could be in conflict with a security requirement without actually running tests.

The creation of test-cases at an early stage also entails the potential risk as they are vulnerable to change as the requirements may change during development. The test-cases may thus be subject to re-work. On the other hand, as the requirements are inspected, the chance that they are changed due to e.g. misunderstandings may be smaller.

An additional potential benefit of TCD inspections is that “inspection knowledge” can be passed on to parties not directly participating in the inspection. This is achieved through the attachment of test-cases to the requirements. The test-cases, if used by the developers as a requirement complement, are the means of this inspection knowledge transfer.

## 6. CONCLUSIONS

The main goals of TCD inspections are aimed at enabling pre-project requirements inspections (increasing the quality of the requirements) and at the same time keeping time and cost at a minimum.

The process of TCD inspections may involve just two persons, utilizing already existing expertise (tester and PM) thus minimizing the need for training. But more importantly, the major artifacts of the inspection, test-cases, are manufactured during the inspection and can be reused to augment requirements, spread inspection knowledge, enable test-plans to be created at an early stage, and ultimately for performing system-tests.

A noticeable benefit of TCD inspections is that the early quality and completeness increase of pre-project requirements improves the decision support material for market-driven development organizations.

TCD inspection technology is in its initial stages. There is need for additional pilot tests in industry and collection of metrics to measure and quantify potential benefits. The results from the initial pilot were promising, and the work of improving this inspection technology will continue through cooperation with industry, in an attempt to make inspection technology accessible to organizations with limited resources.

# Chapter 8

---

## Industry Evaluation of the Requirements Abstraction Model

*Tony Gorschek, Per Garre, Stig Larsson and Claes Wohlin*

Submitted to Requirements Engineering journal, 2005.

### **Abstract**

Software requirements are often formulated on different levels and hence they are difficult to compare to each other. To address this issue, a model that allows for placing requirements on different levels has been developed. The model supports both abstraction and refinement of requirements, and hence requirements can both be compared with each other and to product strategies. Comparison between requirements will allow for prioritization of requirements, which in many cases are impossible if the requirements are described on different abstraction levels. Comparison to product strategies will enable early and systematic acceptance or dismissal of requirements, minimizing the risk for overloading. This chapter presents an industrial evaluation of the model. It has been evaluated in two different companies, and the experiences and findings are presented. It is concluded that the Requirements Abstraction Model provides helpful improvements to the industrial requirements engineering process.

## 1. INTRODUCTION

Requirements Engineering (RE) more and more transcends project boundaries as market-driven product development is becoming increasingly commonplace in software industry [1-3]. Central activities in RE are performed pre-project as a part of for example the product management activities since the requirements flow is continuous and not limited to a specific development instance [6, 7].

In this environment, requirements come from several sources both internal (e.g. developers, marketing, sales, support personnel, bug reports etc) and external (e.g. users, customers and competitors, often gathered via surveys, interviews, focus groups, competitor analysis etc) [8, 10, 11]. Large volumes of requirements from multiple sources risk overloading companies unless they can handle incoming requirements in a structured way, dismissing some, and refining some prior to allocating them to a development instance [20]. In addition to the volume, the requirements themselves are of varying quality, state of refinement, and level of abstraction. In traditional bespoke development (customer-developer) [3], a requirements engineer can actively elicit requirements and thus hope to control or at least substantially influence these aspects. In a market-driven situation, this is seldom the case. Most requirements are already stated in one way or another when they reach the requirements engineer (e.g. a product manager). The knowledge and experience of the developing organization, of which the requirements engineer in this case is instrumental, is central to making sense of the requirements as they are processed [21].

Many requirements engineering best practices, frameworks, and tools are adapted to suit a bespoke environment with traditional, project focused, customer-developer relationships. There is a need for the development and evaluation of RE practices and models that support professionals working with product planning and development (e.g. product managers) in a market-driven environment. This was confirmed by results from two separate process assessment efforts conducted in cooperation with Danaher Motion Särö AB and ABB [134, 166].

In response, a market-driven product centered requirements engineering model was developed, the Requirements Abstraction Model (RAM) (see Chapter 4). RAM is designed towards a product perspective, supporting a continuous requirement engineering effort. It can handle large quantities of requirements of varying degrees of detail and offers a

structure and process for the work-up of these requirements. A brief introduction to the model is provided in Section 2.1.

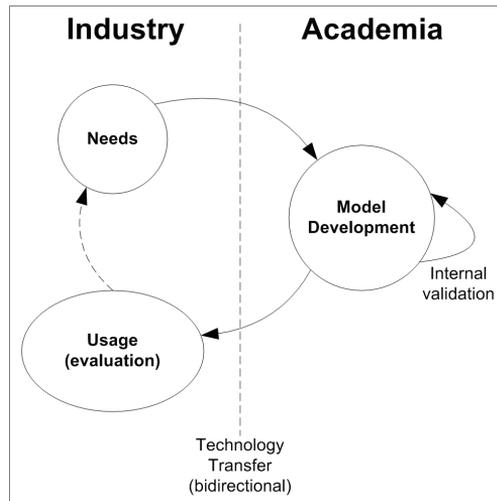
This chapter presents two cases of RAM tailoring, implementation and most important evaluation, conducted at Danaher Motion Särö AB and ABB. The main purpose is to give a brief overview of how RAM was tailored to fit two different organizations, and how the model performed in terms of usability and usefulness based on evaluations performed with professionals using it in their day-to-day work, thus establishing the relative value of using RAM [180].

The chapter is structured as follows. Section 2 gives some background information and a short introduction to RAM to increase the understanding of the rationale behind the tailoring and evaluations performed. Section 3 introduces the companies and the product development situations where RAM is used. The concept of model tailoring and the implementation of RAM at the companies are presented in Section 4. In Section 5, the study design is presented, and the results from the evaluation are given in Section 6. Section 7 presents conclusions drawn and in Section 8 plans for future work are detailed.

## **2. BACKGROUND AND RELATED WORK**

The development of RAM was performed in close collaboration with industry. Figure 48 illustrates this process. Industry needs and possibilities for improvement were identified through several process assessments [134, 166]. The assessment results then acted as a basis for model development. RAM was subsequently validated in several incremental steps, both in industry (small scale pilots) (see Chapter 9) and academia through several experiments using senior students as subjects [181, 182]. Each round of validation was used to refine the model in terms of contents and structure, as well as test issues relating to usability and usefulness.

The large scale industry trials presented in this chapter brings the focus back to industry, allowing us to validate the models usability and usefulness in a non-simulated environment. Feedback obtained here will be used to further refine the model, completing the circle of technology transfer in which RAM was created. More details about the technology transfer process itself can be found in Chapter 9.



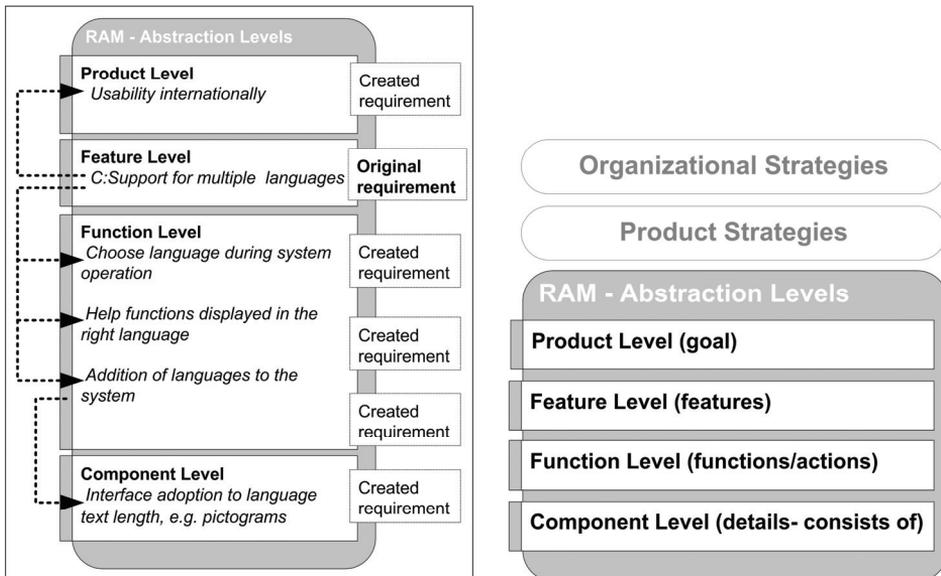
**Figure 48.** Overview of RAM Development.

## 2.1. INTRODUCTION TO THE REQUIREMENTS ABSTRACTION MODEL

RAM is a hierarchical requirements abstraction model, and a method for working with requirements. It is based on the concept that requirements come on several levels of abstraction. Instead of flattening all requirements to one abstraction level RAM *uses* the varying abstractions of requirements, and orders the requirements hierarchically according to abstraction level. Figure 38 shows four abstraction levels; *Product Level*, *Feature Level*, *Function Level*, and *Component Level*. The Product Level is the most abstract level and requirements here are considered abstract enough to be comparable to product strategies, and indirectly to organizational strategies. In the context of RAM, product strategies are rules, long and short-term goals, roadmaps, visions pertaining to a product specified by management etc. Going down to Feature and Function Level the requirements become concrete enough to be used for estimations and as input to development.

Briefly, the process followed using RAM is that when requirements arrive they are placed and specified on an appropriate level by comparing them with the existing requirements base (i.e. example-driven). This means that the present mass of requirements is used as decision support material for the inclusion, exclusion, and placement of new requirements. Following this, all requirements go through *work-up*. Work-up entails abstracting low-level requirements up to Product Level and also breaking down high-level requirements to Function Level. This is done by creating

new requirements in levels above and below and linking them to the original requirement. Figure 38 gives an example of this. The original requirement “C:Support for multiple languages” (placed on Feature Level) is abstracted to Product Level through the creation of a new work-up requirement “Usability internationally”, and broken down to Function Level where three new work-up requirements are created as a part of the breakdown. In some cases requirements already present can be used for abstraction. As an example, in Figure 38, if a new requirement comes in stating “C:Support imperial units” it could be placed on Feature Level and linked directly to “Usability internationally” as imperial units are used in Great Britain and the US in addition to SI units (metric system). In this case, no new requirement has to be created on Product Level and the new requirement can be linked to an already existing one.



**Figure 49.** RAM abstraction levels and example of Work-up.

During the work-up process, the original requirement is compared to product strategies (as it is abstracted). This offers decision support regarding whether the requirement should be specified, refined and kept in the repository, or whether the requirement should be dismissed. For example, let us assume that “Usability internationally” was not accepted but rather the company wanted to limit the product market to the Scandinavian market. In this case “Usability internationally” would be “Usability Scandinavia”, and the new requirement “C:Support imperial units” would be dis-

missed as only SI units are used in Scandinavia. In other words, the product level ultimately decides whether or not to include a requirement, since all requirements are directly linked to this level. If the new requirement is not supported on Product Level, it may indicate that the requirements should be dismissed, but it could of course also mean that a new Product Level requirement needs to be created.

The break-down of the requirements stops on Function Level where the requirements are good enough to be used as decision support for estimation and risk analysis and as input to project(s) for realization. The Component Level is not mandatory in the model, but present since requirements in some instances were delivered in a very detailed form, and these requirements also needed to be handled (i.e. specified and abstracted to assure that they are in line with the overall goals and strategies). In the example presented in Figure 38, the Component Level requirement acts as extra information on a detailed technical level. For example, the interface has to be adapted to pictograms for the Chinese language. In this case, the Component Level sets a restriction that will accompany the requirements into development. The features of using RAM can be summarized as follows:

(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 resources to work on and refine relevant requirements that are in line with the product strategies, minimizing the risk of overloading the organization with irrelevant requirements [20].

(II) All requirements are broken down to an abstraction level where they are good-enough for initiating a development effort (project(s)). This assures that estimations, risk analysis etc. are based on requirements of appropriate abstraction level and contents. In addition, projects get good-enough requirements to base their development efforts on (e.g. testable and unambiguous [5]).

(III) Work-up of a requirement means that additional requirements may have to be created to get a connection to the top level. For example, if an incoming new requirement is placed on Function Level and no appropriate requirement exists on the Feature Level, a new one has to be created. This Feature Level requirement in turn needs to be linked to the Product Level. This ensures that it is possible to follow a requirement through abstraction levels and assure that there is an explicit connection upwards to product strategies. In the same way, every requirement is broken down to a level good-enough to serve as a basis for project initiation (Function Level). Requirements within a certain abstraction level are ho-

mogenous enough to be comparable with each other, which is a prerequisite for 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. Managers can, for example, study the most abstract levels and get a quick overview of the system, while developers can choose a more detailed view, but still have an explicit connection to the overall goals of the product as detailed requirements are connected upwards through the levels.

For reasons of brevity details not central for the tailoring and evaluations presented in this chapter have been left out. For details, please see Chapter 4.

### **3. THE COMPANIES**

The industry trials were conducted at two different companies, Danaher-Motion Särö AB and ABB. Both companies are participating in a joint long-term (six year) research project with Blekinge Institute of Technology in the area of process improvement and requirements engineering. The collaboration in requirements engineering started in late 2002 with DHR, and ABB joined late 2003.

Below each company is described briefly to get an idea of the organizations and the domains in which they operate.

#### **3.1. DANAHERMOTION SÄRÖ AB (DHR)**

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.

Development projects range from six to nine months in calendar time, with a budget of 2000-5000 person-hours.

### 3.2. **ABB (ABB)**

ABB is a leader in power and automation technologies that enable utility and industry customers to improve performance while lowering environmental impact. The ABB Group of companies operates in around 100 countries and employs about 102,000 people. The transfer of new methods for requirement engineering was performed with one of the ABB development centers in Sweden. The product development part of this organization has 200 employees, including development and product management. The organization is primarily working with product development, production and service, supporting ABB sales organizations as well as partners developing solutions for industrial use.

The introduction of RAM was made on the organization developing the controller part of the product offering. The controller includes electronics and software, and project typically involves development of functions in both hardware and software. Projects are divided into release projects with a typical duration of 9 months comprising 20-40 person years, and functional development projects with duration from 3 to 24 months with a large diversity in effort.

Product Management has the responsibility for the functionality of the controller, and orders development from the development organization. Over the last five years, the Product Development organization, including Product Management, has been re-organized several times. This indicates that there is a willingness and need to find improved working methods, also for the requirements engineering.

Process improvement is initiated and managed by process owners that are part of the development organization. Plans and progress are regularly monitored by senior management, and the interest in improving requirement engineering is steadily increasing.

## 4. **MODEL TAILORING**

In the initial stages of model development it was soon realized that one-size-does-not-fit-all. RAM as presented in Section 2.1 is generic in nature and is not intended to act as a prescriptive model with a set of best practices appropriate for all organizations and products. The model is intended to be a framework of principles 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 RAM to fit a specific product (organization), giving the adopting organization's members a chance to clarify critical issues and practices as well as to decide how RAM can be used to best suit their organization and products.

The tailoring of RAM was conducted in workshop format. During the workshop, summarized in Figure 50, a selection of requirement engineers/product managers (Doers), engineers (Users) and managers collaborate in model tailoring and process definitions. These representatives are selected based on their roles and expertise by the local moderator/domain expert (top left in Figure 50). As they are invited, they are asked to prepare for the workshop by reading some initial introductory materials, e.g. introduction to generic RAM, and bring artifacts such as domain specific requirements (bottom left in Figure 50).

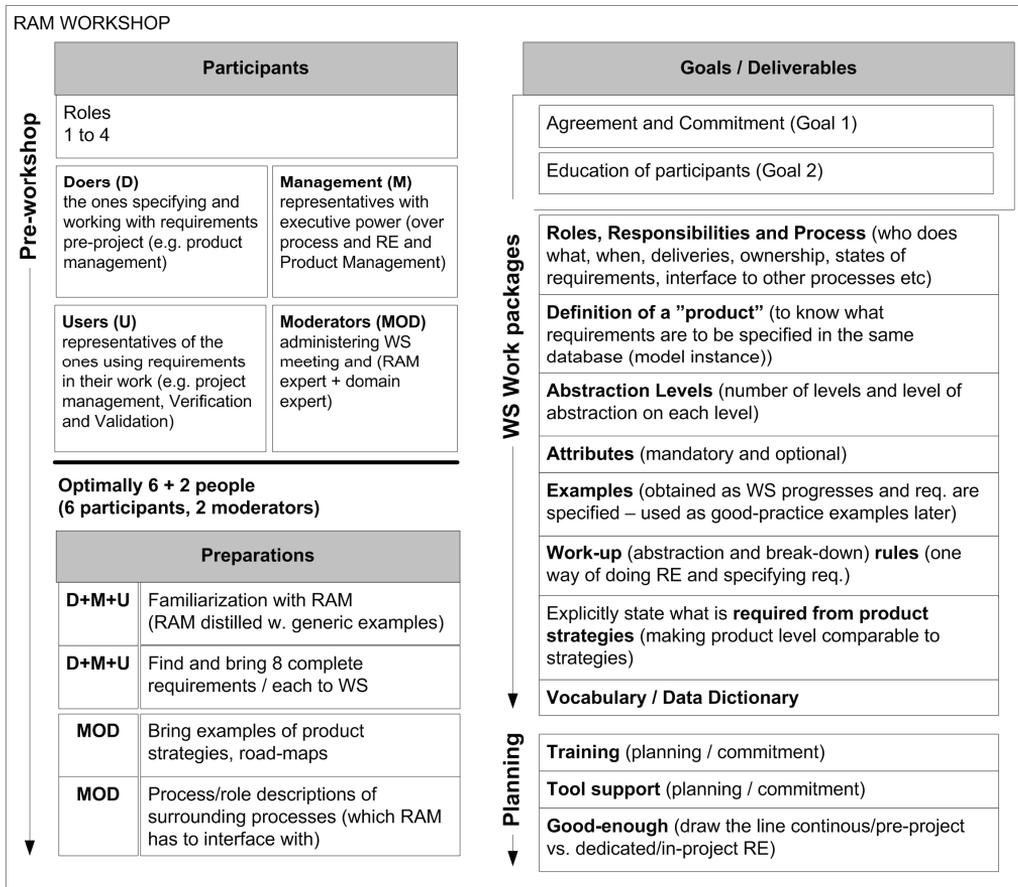


Figure 50. RAM workshop overview.

The main tool used in the workshop is real requirements from the specific domain. The requirements are used as examples driving the work conducted during the workshop. For example, the definition of attributes and abstraction levels are based on discussions generated as actual requirements are specified during the workshop. The workshop activities can be described as a combination of brainstorming session, requirements engineering/specification session, and process formulation.

All ideas and thoughts are written down and scrutinized until a least common denominator agreement is reached. The least common denominator speaks to pragmatics, e.g. if 20 attributes are suggested the number has to be scaled down to the most important ones (set as *mandatory* attributes) and additional ones that can be specified if need be (*optional* attributes), while some are dismissed altogether. In a similar fashion, all decisions reached at the workshop are compromises between what different groups need/want vs. what is considered good-enough for pre-project requirements engineering and what is practical in day-to-day work. The moderators direct the process, i.e. as experts in both the domain and the model they can assure that no critical issues are overlooked.

As the workshop is concluded, the main goal is to achieve agreement regarding the Deliverables and the initial commitment to training and tools support (see Figure 50, right). Post workshop the moderators summarize the results, formalize the model with descriptions, the process, and develop brief reference guides. These guides are light-weight manuals and include domain specific examples. Tool acquisitions/adaptations are also important as well as the planning of training sessions.

One of the main overall goals of the workshop is to obtain commitment from all groups, especially management, as it is crucial for successful process improvement [55-60].

## 4.1. IMPLEMENTATION

It is important to realize that the tailoring resulted in two different instantiations of the RAM model. Each of the companies had their own needs and preferences that dictated the tailoring, but also the implementation of the tailored model. Looking at the model instantiations themselves many of the details, e.g. exactly what attributes are used; their descriptions, and details regarding process and roles and responsibilities are considered proprietary in nature. This implies that these details cannot be shared in this chapter. The implementation of the tailored RAM instances at the two organizations also differed, mainly due to two factors, size of the organizations and time. DHR has a smaller organization than ABB making it

possible to implement RAM in one increment. Further DHR only has one product management organization (for AGVs). ABB on the other hand (as a larger organization) opted to do a stepwise implementation starting with one product management group dealing with one set of products. The implementations turned out to be rather similar despite the organizational differences, i.e. one product management group in each organization.

The time aspect revolved around that DHR had been involved in the development of RAM from the beginning (1 year longer than ABB), resulting in that the model had a somewhat longer time to mature at DHR. In addition, the generic model was largely already formulated when ABB entered the stage, although this was not considered a problem due to RAM's tailorability.

A generic overview of the model as implemented at the companies can be seen in Table 34.

Model/Process Aspect	Company	
	DHR (RAM instance called "DHRRAM")	ABB (RAM instance called "ABBR-RAM")
Abstraction Levels	4 levels (Full work-up as in generic RAM.)	2-3 levels (With work-up in the form of Headers. Headers imply that the hierarchy exists but full requirements are not specified on multiple levels but only on one level and headers created above. A header is similar to a requirements title on upper level).
Attributes	9 Mandatory (3 auto generated) 6 Conditional 5 Optional	10 Mandatory (2 auto generated) 1 Conditional 3 Optional
Possible requirement states	7 Requirement states	5 Requirement states + release status
Tool	CaliberRM	Excel (macro development adapted)
Main users of RAM (specification/analysis)	Process/Model/RE Expert, Product Management, Project Management	Process/Model/RE Expert, Product Management
Roles involved in pre-project RE (using requirements from RAM)	Product Management, Project Management Requirements owner, Technical Expert	Process/Model/RE Expert, Product Management, Requirements owner, Reviewer (development, architecture), Reviewer (Verification & Validation), Technical Expert, Requirements receiver (project)

**Table 34.** *Implementation overview DHR and ABB.*

As mentioned, the technology transfer (process improvement effort) had continued longer at DHR than at ABB. DHR used full work-up of all requirements, in approximately the same manner as described by the generic RAM model described in Section 2.1 (with the exception of having tailored abstraction of the levels) at the time of the evaluation. This was preceded by making sure that tools support was available. ABB on the other hand was at the time of the evaluation not implementing full work-up as they chose to do a stepwise implementation of RAM, starting with the specification of all requirements on *one* abstraction level using RAM attributes, then using feature level and above in a limited fashion. This involved creating headers/titles (not complete work-up requirements) under which requirements could be sorted. The implementation plan at ABB said to wait with the full work-up implementation until the product management organization got used to specifying requirements as objects (with attributes), but also improved tool-support was considered an advantage prior to implementing full work-up.

It is important to notice that the companies have implemented different instantiations of RAM, and to a varying degree. In addition, the initial (pre-RAM) challenges facing the product management and requirements engineering organizations at the companies differed. Although, the common challenge of working in a product centered market-driven environment united them, the immediate improvement potentials were not the same. The focus of this chapter and the evaluations presented is not a comparison between the companies. Rather the evaluation of the process improvements conducted, i.e. the implementation of RAM.

## 5. EVALUATION DESIGN

The first step in the RAM evaluation was to make a selection of interview subjects for participation in the evaluations. The general idea was to select a representative sample of the roles involved in both using RAM to specify and analyze requirements, but also elicit information from the engineers involved in the pre-project requirements engineering. In addition, the receivers of the “RAM requirements” were considered an important source of information as they had a unique opportunity to assess e.g. requirements quality aspects. Table 35 lists the roles represented in the evaluation, as well as their organizational affiliation within the company. The “code” column refers the results presented in Section 6.

The sample chosen (the actual people representing each role) was based on seniority, experience and time spent using RAM. This selection was done in collaboration with on-site expertise. Although the titles of the

roles are similar in the companies, the actual role content and responsibilities varied. The implication, with regards to the evaluation presented in this chapter, being that a certain role may have been involved in performing different actions in the companies. This is discussed when relevant in the presentation of the evaluation results in Section 6.

Role	Organization	Company	Code
<i>Requirements Engineer/Specification</i>	Product Management	DHR + ABB	Subject RE
<i>Product Manager</i>	Product Management	DHR + ABB	Subject PM
<i>Developer/Systems Architect</i>	Project/Development	DHR + ABB	Subject DEV
<i>Verification &amp; Validation (system test)</i>	Project/Development	DHR + ABB	Subject V&V
<i>Development Project Manager</i>	Project/Development	ABB only	Subject PL

**Table 35.** *Roles participating in the evaluation.*

The evaluations were conducted in interview form. As preparation, the subjects were asked to prepare pertinent information and bring relevant documentation to the interview, e.g. test plans, requirements specifications and so on.

The evaluation itself was divided into two main parts, each detailed below.

## 5.1. PART I

The first part focuses on RAM itself and the product management/requirements engineering work conducted using the model. It studies the activities, called 'Actions' in the study, performed in the day-to-day work by primary the product management organization. The Actions evaluated are displayed in Table 36 in the same manner as they were presented to the subjects.

The subjects were asked to grade each Action from three perspectives. First, the *effort* needed to perform a certain Action, for example how much effort did it take to perform 'Estimations' using RAM and RAM requirements. Second, the *accuracy* of each Action was elicited.

The third perspective speaks to the *fulfillment* of each Action when taking both effort and accuracy into account. The idea behind fulfillment is to catch implicit problems and offer a "reality-check". For example, if an Action takes 'more' effort, and the accuracy is 'better' it is not possible to ascertain if the improvement in accuracy is due to simply putting more effort into the Action or if RAM has influenced the outcome. Fulfillment was interpreted as "bang for the buck" or return on investment by the participants. Fulfillment also gauges positive (or negative) spin-off effects in using RAM that are not covered by the concept of accuracy. For example,

specification of requirements using RAM may require more effort and improve accuracy due to e.g. structure, but if using RAM is cumbersome and usability is an issue fulfillment will be low. The opposite is also possible, e.g., the process with RAM encourages requirements engineering meetings, which in itself can lead to catching e.g. dependencies between requirements thus avoiding overlap. In this case the fulfillment might be higher than indicated by just compiling the results of effort and accuracy.

The scale used spans from 'much more' to 'much less' regarding effort, and 'much better' to 'much worse' regarding accuracy and fulfillment, as can be seen in Table 36 (the scales used are identical for every column). In each case, the subject has the opportunity to choose one neutral, four negative, and four positive alternatives. The comparisons performed are based on how product management/requirements engineering Actions were performed prior to the implementation of RAM in the organization.

In addition to grading each Action, every subject was asked to explain his/her answers by commenting on their reasoning, effectively motivating their answers and assuring to the evaluators that the questions and Action were understood as intended. This assured that the different subjects had the same understanding of each Action.

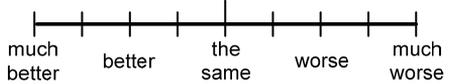
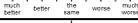
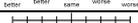
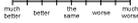
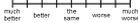
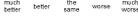
Action	Effort with DHR-RAM compared to before	Accuracy with DHR-RAM compared to before	Fulfillment (do the job good-enough compared to effort and accuracy)
Estimation (resources)			
Risk/problem analysis			
Packaging to project			
Dismiss/accept requirement			
Specifying req.			

Table 36. Evaluation Part I, actions performed using RAM.

## 5.2. PART II

The second part focuses on the quality of the products of RAM, i.e. the requirements themselves. The subjects are asked to compare the requirements generated using RAM with requirements used prior to RAM implementation. Table 37 lists the requirements quality attributes evaluated using a similar scale as in Part I (all scales in Table 37 are identical and only miniaturized to save space). The choice of quality attributes evaluated comes from both academia [5, 18], but also from what was considered important during process assessments performed at the companies in question [134, 166].

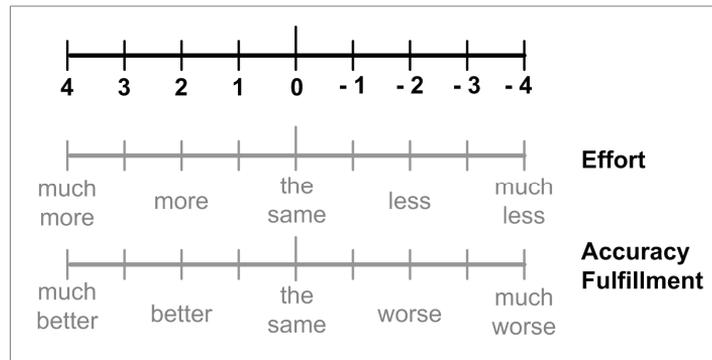
The subjects were also asked to motivate their answers in the same way as in Part I.

Requirement Quality Attribute	
Completeness	
Un-ambiguity	
Testability	
Traceability	
Understandability / readability (why, context, big picture)	
Consistency	
Catching Dependencies Relationships between req.	
Redundancy (double information, unnecessary information)	
Structure (organization) find information etc	
Analysis base (can you base analysis/design on them)	
Conformance to business goals	

**Table 37.** Evaluation Part II, requirements quality using RAM.

## 5.3. ANALYSIS PART I AND PART II

For the purpose of comparison and analysis descriptive statistics are used. The answers obtained on the scales described under Part I and Part II are transformed into numerical values as can be seen in Figure 51. The less effort an Action requires in Part I the lower the value (lower is better). Accuracy and fulfillment are treated the opposite, i.e. better and much better are translated into positive values (higher is better).



**Figure 51.** Conversion of Part I and Part II answers to numerical values.

The results are presented in the form of descriptive statistics (diagrams) and tables with numerical values created as illustrated in Figure 51. An example is shown in Table 38. Effort, Accuracy and Fulfillment are specified by a subject and converted to numerical values as described previously. The last column (MAX:12, MIN:-12) is a compilation of the values calculated with the following formula: Accuracy – Effort + Fulfillment, giving a maximum of 12, and a minimum of -12 per subject with regards to a certain Action. Using the example in Table 38 subject x would get a MAX/MIN of:  $2 - 0 + 3 = 5$ , and in the case of subject n:  $1 - (-2) + 4 = 7$  (less being better in the case of effort to perform an Action). The MAX/MIN compilation is a way to summarize the total level of perceived benefit concerning a certain Action, allowing for easier comparison in diagram form as will be seen in Section 6. The use of numerical values converted from scales for performing operations such as summation can be questioned, as it is not normal in measurement theory. In the case of MAX/MIN the use is strictly illustrative, i.e. enabling concise presentation in diagram form collecting several aspects in one figure (as can be seen in Section 6). Care has been taken to present the actual values in parallel and the analysis of the actions are based on effort, accuracy and fulfillment, not the calculated MAX/MIN values.

In addition, not every subject had the possibility to answer every question or make estimates in all cases. This is due to what Actions the subjects were involved with. For example, the role of Verification & Validation (system test) was not involved in the Action of performing estimations on requirements during the pre-project requirements engineering. In this case, the answers will be grey and marked “N/A” as can be seen for subject y in Table 38.

Action X				
	Effort	Accuracy	Fulfillment	MAX:12, MIN:-12
subject x	0	2	3	5
subject y	N/A	N/A	N/A	N/A
subject z	1	2	3	4
subject n	-2	1	4	7

**Table 38.** Example of results Part I for actions.

---

## 6. EVALUATION RESULTS

The evaluation results are presented by company, and divided according to Part I and II.

### 6.1. DHR

The evaluation at DHR was performed in one session with individual interviews of about 1-1.5h each. Subjects representing RE, PM, and DEV were all involved with aspects of the pre-project RE and the activities performed in relation to product management. The subject representing V&V was only involved in Verification & Validation activities (creation of test plan and test cases) and only answered questions in Part II.

#### 6.1.1. PART I

In total five actions were evaluated (see Table 36) concerning the effort it took to accomplish the work, accuracy of the performed work, and the fulfillment achieved (gauging return on investment taking amongst other things effort and accuracy into account). The individual results are presented in Table 39

A general tendency observed is that although some effort increases are present it is predominantly compensated by increased accuracy and the fulfillment is in all cases positive.

A compact overview of the results is offered in Figure 52. The bars represent the MAX/MIN compilation per subject (see Table 35) and Action. The tanned notes in the bars are the individual results of each subject, 'E' standing for effort, 'A' for accuracy, and 'F' for fulfillment (for example, for the Action of Estimation subject RE has an effort of zero, accuracy of two and fulfillment of two, and a MAX/MIN of four indicated by the bar itself). In the cases where no answer was given (due to a subject not performing a certain Action) an 'N/A' note can be seen, this is to not confuse the absence of a bar with a value of zero (for example, subject PM did

not perform the Action of Estimation). Observe that the y-axis scale in reality goes from -12 to 12, although only 0-10 is displayed in the diagram, as all the values are in that range. Each Action is analyzed in further detail below.

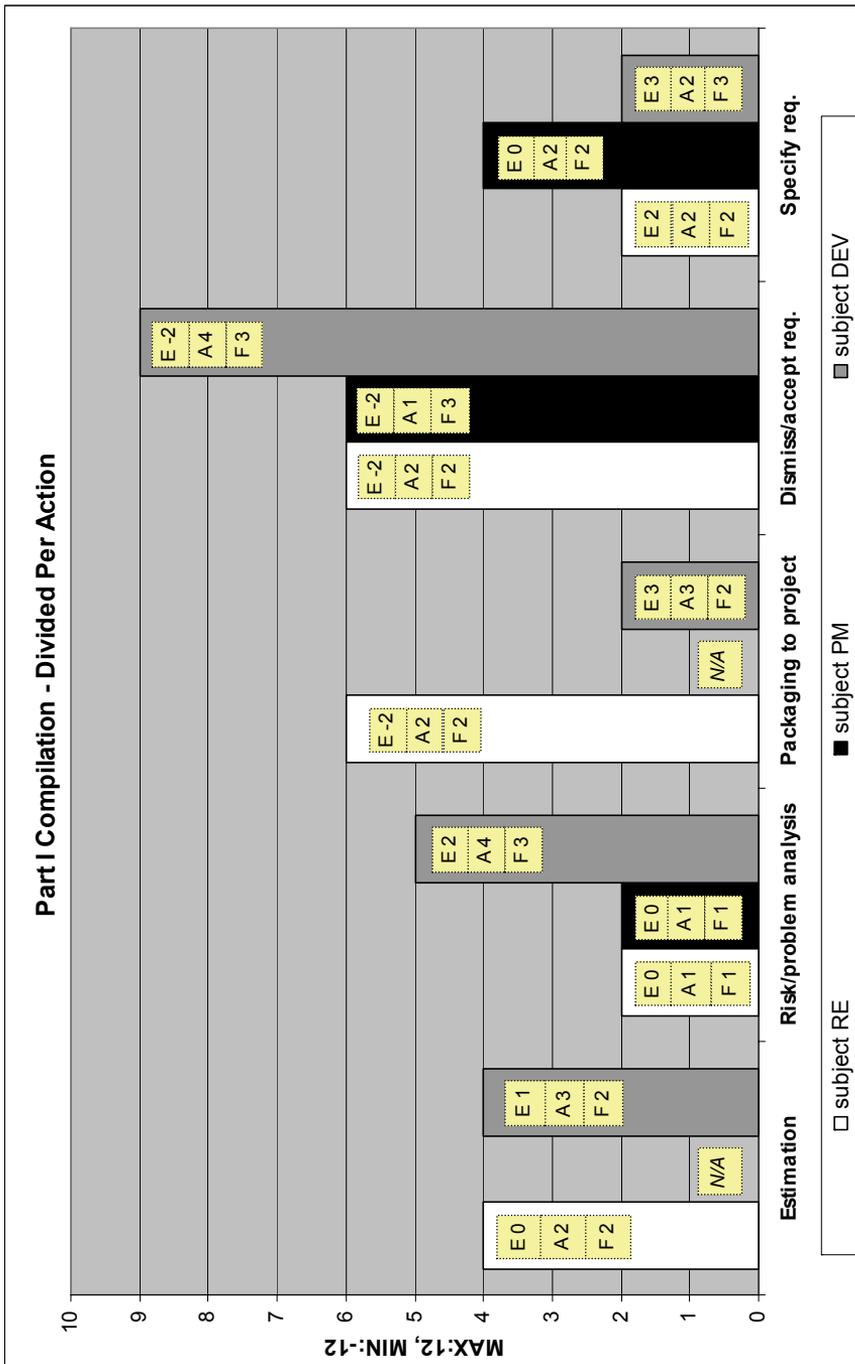
- **Estimation** in the case of DHR is performed by project personnel (DEV) and to some part by the requirements engineer. Representatives for these roles claim improved accuracy and fulfillment. The main motivation is that the hierarchical structure of abstraction gives a better overview, in turn offering more information in total. Being able to traverse the levels gives a big-picture, while *all* requirements are broken down to a level where they are detailed enough to base estimations on.
- **Risk/Problem Analysis** is not performed as a separate activity, rather it is a implicit part of the analysis and specification of requirements (certain attributes specified for each requirements explicitly demands that limitations and risks be a part of the specification). RE and PM felt that there was no real increase in effort as the explicit risk and problem analysis was a part of the specification (the effort increase was put on the Action of specification). DEV felt that the break-down of requirements, especially from feature to function level, gave early analysis on a more detailed level (as attributes e.g. concerning risk/limitation had to be specified on function level and not just for abstract features). Although DEV felt strongly that accuracy was greatly increased, there was also an increase in work effort (E2). The total fulfillment was even greater than just a compilation of effort and accuracy due to that collaboration in RE meetings using the structured RAM requirements made it possible to catch problems not directly associated with requirements being specified, but in already existing products and legacy features.
- **Packaging to Project** did not involve PM in this instance (was involved in the earlier phases of pre-project RE) and from a PM perspective the requirements going to a certain project were already packaged.

There seems to be a large discrepancy between RE and DEV regarding this Action. Both agree that accuracy and fulfillment is better than before, but while RE considers less effort being spent, DEV feels a noticeable increase in effort (which is what brings down the total MAX/MIN value for DEV). The increase in effort can be explained by two factors, both given by the subject as motivation for the answers. First the initiation threshold of using RAM (learning

curve), and second a new practice of creating Implementation Proposals prior to project start. The learning curve is rather obvious in nature and was felt especially by DEV as being an active part of the requirements engineering was new. The practice of creating Implementation Proposals had nothing to do with RAM as such, but the activities were to some extent performed in parallel making the total effort pre-project greater.

<b>ESTIMATION</b>				
	<b>Effort</b>	<b>Accuracy</b>	<b>Fulfillment</b>	<b>MAX:12, MIN:-12</b>
subject RE	0	2	2	4
subject PM	N/A	N/A	N/A	N/A
subject DEV	1	3	2	4
subject V&V	N/A	N/A	N/A	N/A
<b>RISK/PROBLEM ANALYSIS</b>				
	<b>Effort</b>	<b>Accuracy</b>	<b>Fulfillment</b>	<b>MAX:12, MIN:-12</b>
subject RE	0	1	1	2
subject PM	0	1	1	2
subject DEV	2	4	3	5
subject V&V	N/A	N/A	N/A	N/A
<b>PACKAGING TO PROJECT</b>				
	<b>Effort</b>	<b>Accuracy</b>	<b>Fulfillment</b>	<b>MAX:12, MIN:-12</b>
subject RE	-2	2	2	6
subject PM	N/A	N/A	N/A	N/A
subject DEV	3	3	2	2
subject V&V	N/A	N/A	N/A	N/A
<b>DISMISS/ACCEPT REQ.</b>				
	<b>Effort</b>	<b>Accuracy</b>	<b>Fulfillment</b>	<b>MAX:12, MIN:-12</b>
subject RE	-2	2	2	6
subject PM	-2	1	3	6
subject DEV	-2	4	3	9
subject V&V	N/A	N/A	N/A	N/A
<b>SPECIFYING REQ.</b>				
	<b>Effort</b>	<b>Accuracy</b>	<b>Fulfillment</b>	<b>MAX:12, MIN:-12</b>
subject RE	2	2	2	2
subject PM	0	2	2	4
subject DEV	3	2	3	2
subject V&V	N/A	N/A	N/A	N/A

**Table 39.** Results Part I for DHR actions.



**Figure 53.** Action Compilation divided by Action and subject.

- **Dismiss/Accept Requirements** was effort wise felt to be substantially less by all subjects. The main motivation was that the abstraction levels offered much better overview of all requirements, enabling discussions focusing on abstract levels where the amount of requirements were manageable, but with the possibility to look at more detailed levels (following dependencies and relationships downward) when needed to see the implications of the decisions.  
RE and PM mostly agreed with regards to increase in accuracy and a positive fulfillment. DEV felt a substantially greater increase in accuracy mainly due to the traceability of consequence, i.e. dismissing a Feature level requirement was not performed using a gut-feeling when gauging consequences, but rather explicit relationships could be followed down the abstraction levels giving a true picture of the decision being made.
- **Specify Requirement** shows an effort increase for RE and DEV that can be contributed to two factors, learning curve in the case of DEV, but also the fact that more information is specified using RAM, and the use of abstraction levels, following the RAM process demands more effort. However, it should be observed that the effort increase was considered moderate in nature.  
PM agreed with RE and DEV (they have the same level of accuracy and fulfillment increase) but did not feel a great increase in specification effort. This can be attributed to that the subject consciously subtracted the learning threshold when giving an answer to this question, i.e. looking at the comments/motivation offered when answering the learning threshold would have added moderately to the effort just like in the case of RE and DEV.

### 6.1.2. PART II

The subjects were asked to evaluate the requirements quality from 11 different perspectives. In this part, the views of a new subject is introduced, i.e. V&V. V&V did not participate in the requirements engineering activities, but is an expert in the artifacts produced, namely the requirements that are used to create test cases.

All answers for Part II are displayed in Table 40. The rows are summarized giving a total quality attribute value for the RAM requirements per subject (this is of course only to give an illustration, just like in the case of MAX/MIN in Part I, and not to be interpreted as absolute values). All positive values in the table indicate an increase in quality, where a four is maximum, and vice versa for a decrease in quality with a value of negative four in the worst case.

REQUIREMENT QUALITY												
	Completeness	Un-ambiguity	Testability	Traceability	Understandability/ readability	Consistency	Catching dep. Relationships btw req.	Redundancy	Structure	Analysis base	Conformance to Business Goals	SUM per subject
subject RE	2	3	2	2	2	2	3	-1	2	2	0	19
subject PM	2	2	0	4	4	2	0	0	0	2	1	18
subject DEV	2	0	3	4	2	2	4	2	4	2	N/A	25
subject V&V	2	3	3	4	2	3	0	4	3	2	N/A	26
<b>Sum per quality attr.</b> (max 16, min -16)	8	8	8	14	10	9	7	5	9	8	1	

**Table 40.** Results Part II for DHR requirement quality attributes.

It is noticeable that the highest scores are obtained from DEV and V&V, users of the requirements within their development work. The high scores were obtained despite of the fact that they could not answer whether or not the requirements conformed to business goals as they were not well versed in that aspect of product development. The total scores of RE and PM were similar.

The columns are also summarized displaying the overall score for every quality attribute.

Figure 54 gives a different overview of the results divided by quality attribute.

- **Completeness** (total score 8 out of 16). All subjects agree that substantial increase has been achieved with regards to completeness of the requirements. This is due to the use of attributes and more information being available than an individual requirement as the abstraction levels can be traversed offering overall big-picture and details in the form of linked requirements on lower levels.

- **Un-ambiguity** (total score 8 out of 16). All subjects except for DEV agree that the RAM requirements are less ambiguous than the previous ones. DEV scores it the same as before (zero). The main motivation offered is that in spite of more and better information (attributes and abstraction levels) the actual natural language formulation of the requirements needs to be improved.
- **Testability** (total score 8 out of 16). A general increase, although this increase was not confirmed by PM (giving it a zero). The main motivation being that from the PM perspective no direct difference in testability can be seen. This is however strongly opposed by both RE, DEV and especially V&V, who scores it high and sees a significant improvement in testability (3).
- **Traceability** (total score 14 out of 16). All subjects except RE agree that a very significant increase in traceability has been achieved. RE also scores it as a significant improvement but remarks that several traceability aspects are still lacking, due mainly to what perspective one has. For example, traceability to source/origin for DEV and V&V is fulfilled if the internally responsible for a requirement is listed. From the perspective of RE, the real external source is more interesting and not always traceable.
- **Understandability/readability** (total score 10 out of 16). All subjects indicate a significant improvement and they agree with regards to the level, with the exception of PM indicating a very large improvement. This is motivated mainly by the explicit abstraction of requirements and the possibility to gain an overview, and a connection to the product and business level.
- **Consistency** (total score 9 out of 16). All subjects see a substantial improvement in requirement consistency, with V&V indicating a somewhat higher consistency than the other subjects do.

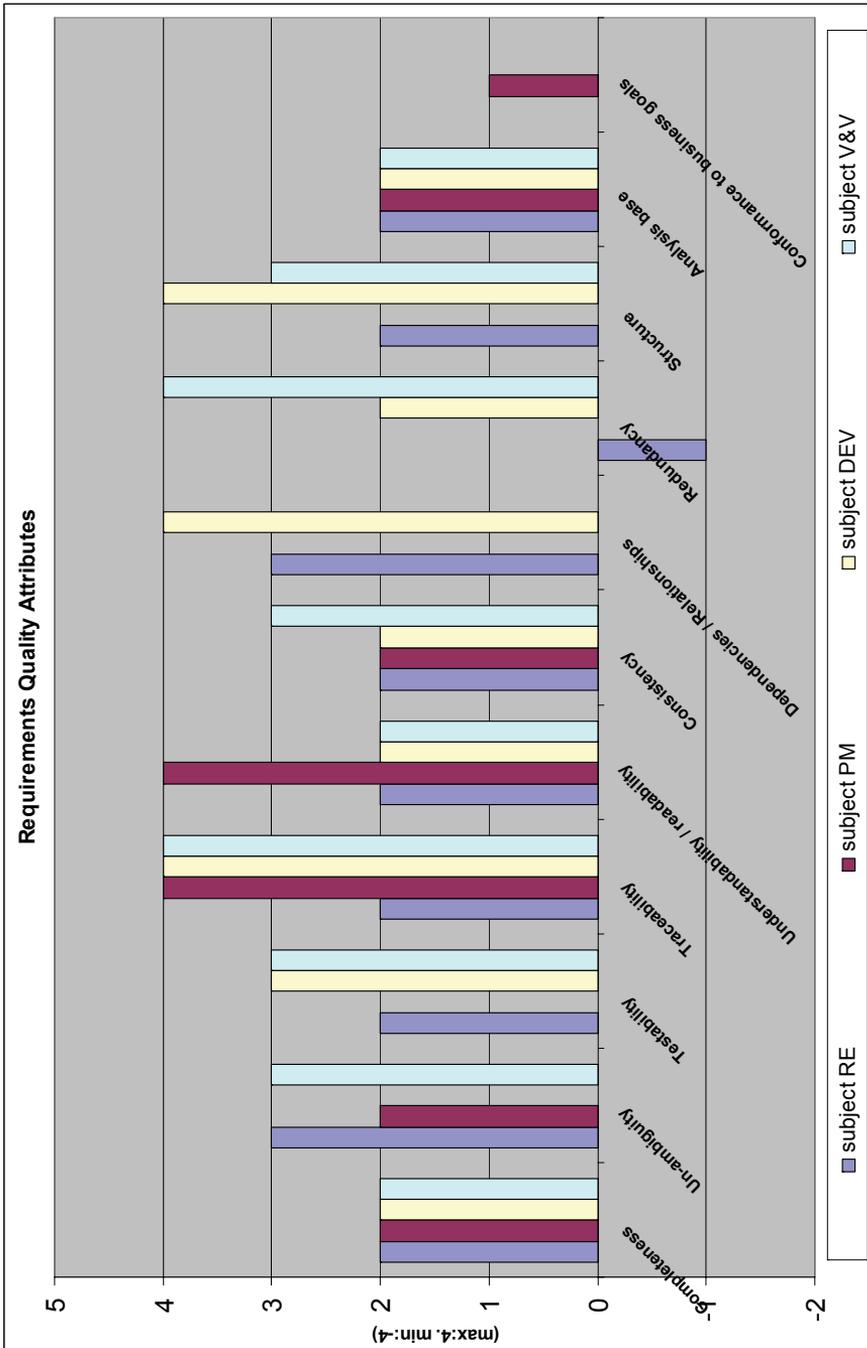


Figure 54. Overview of requirements quality attributes.

- **Catching dependencies/relationships** (total score 7 out of 16). The total score is brought down by PM and V&V who grade the improvement as zero in strong contrast to RE and DEV seeing a great improvement. The explanation for this is that while RE and DEV included dependencies and relations between abstraction levels in their score, PM and V&V did not. Both PM and V&V acknowledge that a very substantial improvement in catching and specifying dependencies exists up-wards and downwards between abstraction levels, no additional effort or analysis has been put in catching dependencies between requirements on the same level (nor has it become worse than before).
- **Redundancy** (total score 5 out of 16). DEV and especially V&V see substantial improvements (less redundancy) in using RAM, while PM sees neither improvement nor degradation. RE however sees some added redundancy. The explanation for the two perspectives may be that RE and PM work with requirements actively, using and especially knowing about all information specified. DEV and V&V on the other hand concentrate and use predominantly parts of the total information specified (e.g. using primarily function level when specifying test cases). Thus, they are isolated from some information they may see as redundant in nature.
- It should be observed that the overall score indicates an improvement with regards to redundancy in spite of the fact that the amount of information specified in total using RAM is much greater than previously.
- **Structure** (total score 9 out of 16). The improvement of this quality attribute is scored high to very high by all subjects except PM. The main motivation behind this is can be attributed to the fact that the PM in this instance was not involved in the actual specification of many of the requirements using RAM, and when being involved the learning curve of using RAM took away from the usability. The main motivation from RE not scoring it higher was attributed to the learning curve of the new tool used for requirements engineering.
- **Analysis base** (total score 8 out of 16). All subjects agree that the use of RAM (both attributes and abstraction levels) produces requirements of significantly better quality with regards to the analysis base.

- **Conformance to business goals** (total score 1 out of 8). At first glance, this score indicates a very low improvement in requirements conformance to business goals. First, it should be observed that the maximum possible score for this quality attribute is eight, not 16 as DEV and V&V did not score it. Still one out of a possible improvement of eight is low, especially since RAM abstraction is intended to explicitly connect requirements to product strategies (through the use of abstraction). PM does score some improvement but both RE and PM remark that there is not an explicit mapping between the most abstract requirements and the product strategies and business goals. The main reason for this is that as RAM is relatively new to the organization the formulation of business and product strategies have not been performed in the manner needed for explicit use for mapping requirements. This work was planned and underway, but in the early stages at the time of the evaluation.

### 6.1.3. SUMMARY AND DISCUSSION - DHR

Part I overall shows some increase in effort being expended to perform the actions, but this is more than compensated by an increase in accuracy, especially when the learning curve of both RAM (new process, concept of abstraction, attributes) and a new requirements engineering tool are taken into consideration. Improvements can be seen across the board, with the ability to have fast dismissal or acceptance of new incoming requirements at the top. Estimation activities also show a large increase in accuracy.

Part II also shows improvements across the board. Traceability scores the highest and Understandability/readability comes in second. Completeness, Un-ambiguity, Testability, Consistency, Analysis base, and Catching dependencies/relationships all score in the mid-range showing an overall substantial increase in requirements quality. The low scoring quality attribute Conformance to business goals can to some extent be attributed to the short operation time of RAM in the organization as the routines for creating and using explicit product strategies and business and mapping to high level requirements is under way. In addition, it is interesting to notice that in many cases agreement is high. For example, looking at Completeness, Traceability, Understandability/readability, Consistency, and Analysis base the broad spectrum of roles all feel high increase in requirements quality. This is especially evident in the case of Completeness and Analysis base where the subjects agree totally, i.e. an increase of two (out of four). This indicates that the implementation of RAM managed to give a significant increase for multiple roles and thus multiple

uses of the requirements. RE and PM use requirements for figuring out *what* is to be done and for product and project planning activities. V&V has the perspective of verification and validation, while DEV uses requirements to base solutions (how) on. An agreement in quality increase could suggest that the requirements on different abstraction levels suit the needs of multiple types of users.

Overall, the use of appropriate attributes and abstraction (work-up) of requirements results in a positive outcome both with regards to actions performed as a part of pre-project requirements engineering/product management and with regards to the requirements quality. It should be noticed that all parts of the evaluation presented are relative in nature, i.e. an increase in accuracy, positive fulfillment, and improvement in quality all indicate the relative benefit achieved using RAM instead of previous practices.

## 6.2. ABB

The evaluation at ABB was performed in two sessions spanning over 2 days. Each individual interview lasted between 1-2h. Subjects representing RE, PM, DEV and V&V were all involved with aspects pre-project requirements engineering and the activities performed in relation to product management. Subject PL represents the project manager who was not involved in pre-project requirements engineering, thus only answered questions in Part II.

### 6.2.1. PART I

The individual results for Part I are presented in Table 41, and a more compact overview can be seen in Figure 55. The bars (Figure 55) represent the MIN/MAX compilation per Action and subject. The tanned notes in the bars are the individual results of each subject, 'E' standing for effort, 'A' for accuracy, and 'F' for fulfillment. In the cases where no answer was given (due to a subject not performing a certain Action) an 'N/A' note can be seen, this to not confuse the absence of a bar with a value of zero. Observe that the y-axis scale in reality goes from -12 to 12, although only 0-9 is displayed in the diagram, as all the values are in that range. Each Action is analyzed in further detail below.

- **Estimation** in the case of ABB is performed by RE and PM with input from DEV and other experts as needed. Both RE and PM claim a substantial improvement in accuracy and fulfillment. In addition RE sees less effort being expended using RAM. The main motivation is that the hierarchical structure gives a better overview and that the re-

quirements are more specified utilizing attributes. DEV sees a moderate increase in effort but none in accuracy. Despite of this, DEV claims positive fulfillment as discussions in relation to estimations solve other small issues and assumptions are questioned.

- **Risk/Problem Analysis** did not involve RE in the case of ABB, but mainly PM, DEV and V&V were used as experts, e.g. consulting them as needed. PM sees a substantial decrease in effort needed to perform the Action, and a moderate increase in accuracy. DEV on the other hand sees a moderate increase in effort. DEV motivates this by the adoption of a structured process for the activity which has an initial learning curve. V&V also sees a moderate increase in effort but a substantial increase in accuracy and fulfillment. This is mainly due to the use of attributes which forces the explicit specification of information that could be missed previously. Another contributing factor is that V&V is involved earlier in the process allowing for aspects such as verifiability to be premiered.
- **Packaging to Project** involved RE and PM. Both of them see an increase in accuracy in using RAM. The only difference is that RE sees an increase in effort. The increase in effort was motivated by the subject and can be attributed to a double learning curve. The first part of the learning curve is due to that RE did not have extensive experience in requirements specification at the time. The other part of the learning curve can be attributed to the implementation of RAM itself. (These two aspects can be seen repeated in the rest of the actions for the subject). It should be observed that although effort and accuracy are equally increased the fulfillment was relatively high, indicating that the subject realized that the issue of effort was not mainly due to problems with usability or usefulness of RAM.

<b>ESTIMATION</b>				
	<b>Effort</b>	<b>Accuracy</b>	<b>Fulfillment</b>	<b>MAX:12, MIN:-12</b>
subject RE	-2	2	4	8
subject PM	1	2	4	5
subject DEV	1	0	1	0
subject V&V	N/A	N/A	N/A	N/A
subject PL	N/A	N/A	N/A	N/A
<b>RISK/PROBLEM ANALYSIS</b>				
	<b>Effort</b>	<b>Accuracy</b>	<b>Fulfillment</b>	<b>MAX:12, MIN:-12</b>
subject RE	N/A	N/A	N/A	N/A
subject PM	-2	1	3	6
subject DEV	1	1	1	1
subject V&V	1	4	4	7
subject PL	N/A	N/A	N/A	N/A
<b>PACKAGING TO PROJECT</b>				
	<b>Effort</b>	<b>Accuracy</b>	<b>Fulfillment</b>	<b>MAX:12, MIN:-12</b>
subject RE	2	2	2	2
subject PM	0	2	2	4
subject DEV	N/A	N/A	N/A	N/A
subject V&V	N/A	N/A	N/A	N/A
subject PL	N/A	N/A	N/A	N/A
<b>DISMISS/ACCEPT REQ.</b>				
	<b>Effort</b>	<b>Accuracy</b>	<b>Fulfillment</b>	<b>MAX:12, MIN:-12</b>
subject RE	2	2	2	2
subject PM	-1	1	1	3
subject DEV	2	2	2	2
subject V&V	N/A	N/A	N/A	N/A
subject PL	N/A	N/A	N/A	N/A
<b>SPECIFYING REQ.</b>				
	<b>Effort</b>	<b>Accuracy</b>	<b>Fulfillment</b>	<b>MAX:12, MIN:-12</b>
subject RE	4	2	4	2
subject PM	1	3	3	5
subject DEV	1	1	1	1
subject V&V	-2	2	2	6
subject PL	N/A	N/A	N/A	N/A

**Table 41.** Results Part I for ABB Actions.

- **Dismiss/Accept Requirements.** RE and DEV both considered that a substantial increase in effort was necessary to systematically dismiss or accept requirements using RAM, although an equivalent increase was perceived with regards to accuracy. The effort increase for RE was mainly due to learning curve, but for DEV the effort increase could be explained by the overall participation in this activity. DEV called the RAM early acceptance/dismissal of requirements a “pre-study”, a new activity from DEV’s perspective. Although the increase in effort and the equivalent increase in accuracy seemed to cancel each other out, RE and DEV both indicated that it was worth it (indicated by fulfillment) as they realized that the second time around effort would probably be less as the learning curve was not as steep. In addition, RE felt that a positive spin-off effect was obtained, in the form of that overlapping requirements were dismissed earlier as the requirement levels were homogenous enabling better comparison between requirements. In the case of PM, the effort was moderately less for early dismissal and an equally moderate increase in accuracy. The positive outcome was mainly motivated by an increase in requirements quality, and that all (a clear majority) of the requirements were comparable with regards to abstraction and detail. This increased requirements understanding, making it easier to dismiss/accept requirements at this early stage.

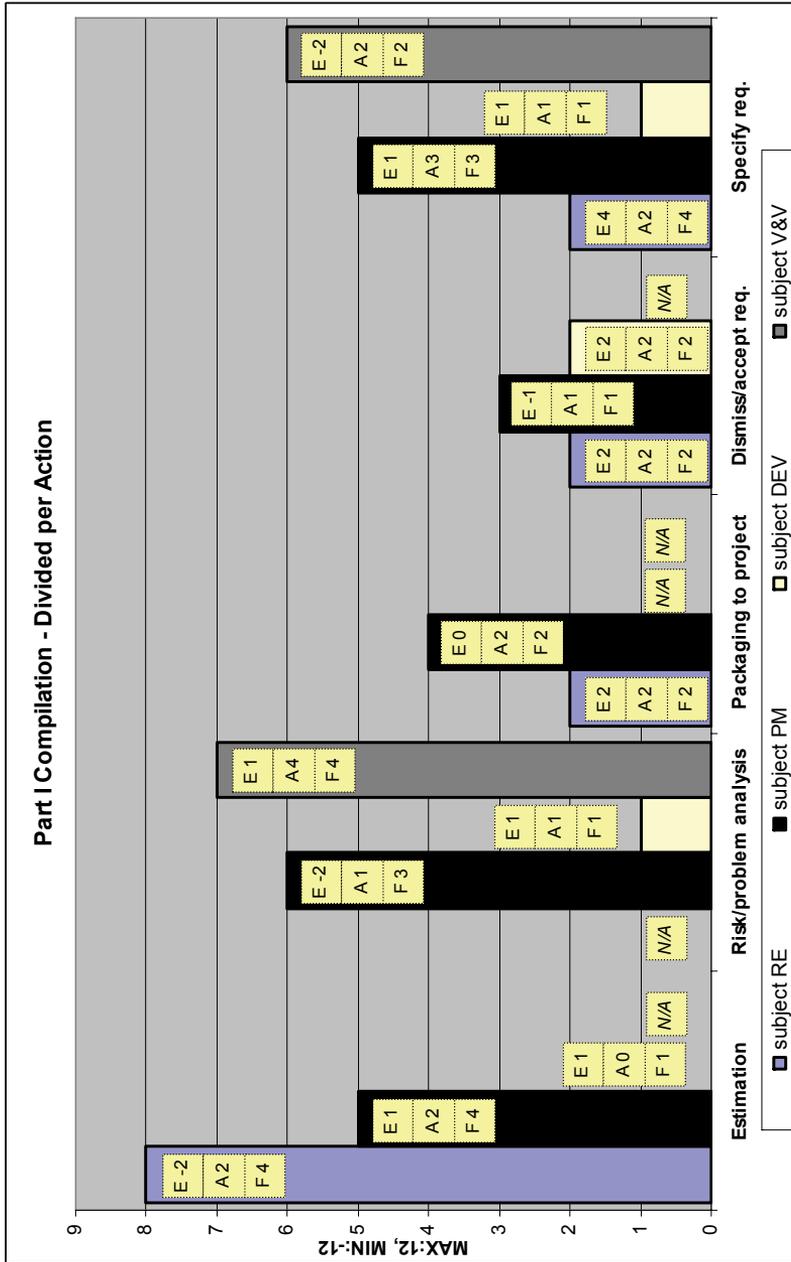


Figure 55. Action compilation divided by Action and subject.

- **Specify Requirement** shows a large effort increase for RE and a moderate increase for PM and DEV. However, V&V indicates an effort decrease. The decrease can be attributed to that better and more

structured specification prior to V&V active involvement decreases effort for V&V.

All subjects indicate a substantial increase in accuracy, except DEV which indicates a more moderate increase.

## 6.2.2. PART II

The subjects at ABB evaluated requirements quality from 11 different perspectives. In this part the views of a new subject is introduced, PL. PL did not participate in the requirements engineering activities, but uses requirements as input to the development projects, which are under PL's responsibility in terms of planning, control and general administration.

All answers for Part II are displayed in Table 42. The rows are summarized giving a total quality attribute value for the RAM requirements per subject. All positive values in the table indicate an increase in quality, where a four is maximum, and vice versa for a decrease in quality with a value of negative four in the worst case.

It is noticeable that the highest score is obtained from PL, which uses the requirements in development efforts. DEV scores lowest values. RE, PM, and V&V score in the same range.

REQUIREMENT QUALITY												
	Completeness	Un-ambiguity	Testability	Traceability	Understandability/readability	Consistency	Catching dep. Relationships btw req.	Redundancy	Structure	Analysis base	Conformance to Business Goals	SUM per subject (max 44, min -44)
subject RE	4	0	3	3	3	0	0	0	0	3	0	16
subject PM	3	1	2	2	2	0	0	-1	0	3	0	12
subject DEV	1	0	N/A	2	1	0	0	0	0	2	N/A	6
subject V&V	2	2	4	3	2	0	1	-1	0	2	0	15
subject PL	4	4	2	4	2	2	2	0	1	2	2	25
<b>Sum per quality attr. (max 20, min -20)</b>	<b>14</b>	<b>7</b>	<b>11</b>	<b>14</b>	<b>10</b>	<b>2</b>	<b>3</b>	<b>-2</b>	<b>1</b>	<b>12</b>	<b>2</b>	

**Table 42.** Results Part II for ABB requirement quality attributes.

The columns are also summarized displaying the overall score for every quality attribute.

Figure 56 gives a different overview of the results divided by quality attribute.

- **Completeness** (total score 14 out of 20). All subjects (except DEV) agree that a substantial increase has been achieved with regards to completeness of the requirements. This is mainly due to the utilization of attributes and to some extent homogenous abstraction of the requirements on an abstraction level suiting RE, PM, V&V, and PL. The exception to this is expressed by DEV, who in general feels that the increase in quality is moderate to none for most quality attributes (barring some exceptions). The main motivation given was that the level of abstraction on which the requirements are specified are not optimal, even if some improvements can be seen. DEV feels that for the purposes of development some improvement can be seen (e.g. analysis base) but overall the requirements can be specified on a level more appropriate for development activities, i.e. broken down further.
- **Un-ambiguity** (total score 7 out of 20). PM, V&V and in particular PL feel that there has been improvement with regards to requirements ambiguity. DEV feels that the requirements are too abstract to say they are less ambiguous. PM and especially RE feels that the natural language specification of the requirements text itself can be improved.
- **Testability** (total score 11 out of 16, not specified by DEV). The general consensus (with exception of DEV who does not feel qualified to answer) is that from the perspective of testability a clear improvement has been obtained. PL comments that although an improvement has been achieved there is still room for more improvement.
- **Traceability** (total score 14 out of 20). All subjects agree that a significant increase in traceability has been achieved. This impression is underlined by PL quoting especially traceability to requirements source/origin and requirement's specifier as an improvement.
- **Understandability/readability** (total score 10 out of 20). All subjects indicate a significant improvement (except DEV). DEV quotes the need to have more general information about the problem and goals (why) of the requirements to improve understandability.
- **Consistency** (total score 2 out of 20). All subjects indicate that no improvement has been obtained with regards to consistency. The main motivation for this is that there is a focus on individual requirements and not a conscious effort to specify requirements in a way that several requirements as a whole are consistent amongst each other. PL feels that some improvement has been achieved, as the requirements are more thoroughly worked-through.

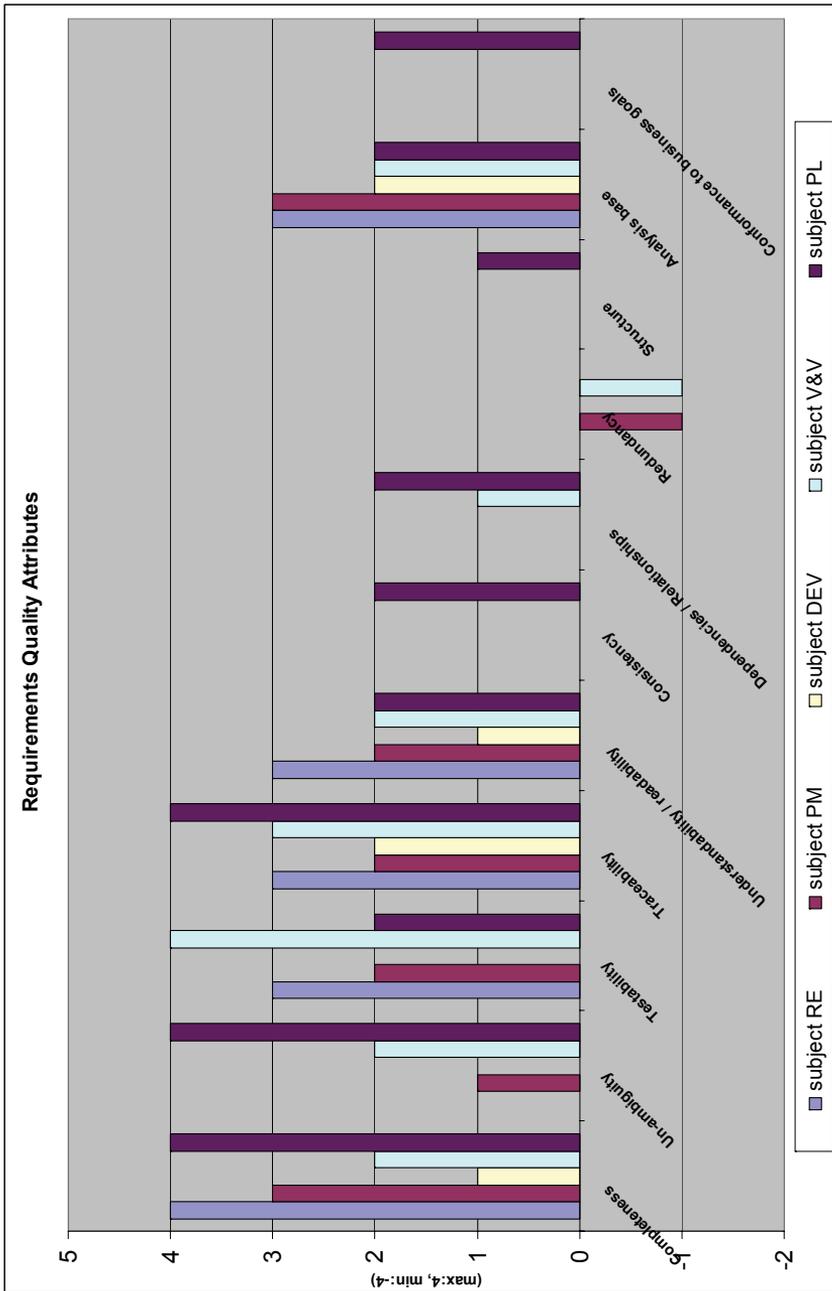


Figure 56. Overview of Requirements Quality Attributes.

- **Catching dependencies/relationships** (total score 3 out of 20). As in the case of consistency the total score is very low (i.e. no great im-

- provement). The reasons are to a large extent the same as in the case of consistency, namely that the focus is on individual requirements.
- **Redundancy** (total score -2 out of 20). Redundancy scores the lowest of all quality attributes, with PM and V&V actually indicating a slight decline (more redundancy) in comparison to pre-RAM. Once again, the main reason given by the subjects revolves around the fact that there is focus on individual requirements, and that the same information sometimes has to be specified in several requirements as it is important for several requirements. This is often information handling goal like information, or general risks and restrictions and so on.
  - **Structure** (total score 1 out of 20). The improvement of this quality attribute is scored very low. In general, the consensus is that no improvement has been made. The reason for this can be traced to less than optimal tool support, and the lack of full sorting of requirements under more abstract ones.
  - **Analysis base** (total score 12 out of 20). All subjects agree that the use of RAM (attributes and more detailed analysis and specification) has produced requirements of significantly better quality in this aspect.
  - **Conformance to business goals** (total score 2 out of 16, not specified by DEV). The main reason for this relatively low score of improvement can be accredited to the fact that the abstraction level of the requirements resides too far from product strategies. This makes comparison very difficult.

### 6.2.3. SUMMARY AND DISCUSSION - ABB

Part I shows an overall increase in Action accuracy. Positive fulfillment is also a general tendency even if the level of effort sometimes rivals the increase in accuracy, indicating that the extra effort is considered a good investment. The learning effort for RAM in general is low, although there is a substantial effort increase in the case of RE with regards to general requirements specification as the person filling the role was new to the task. A general piece of feedback often accompanying the ratings performed was that improved communication between the product management organization and the product organization was achieved using RAM. The positive effects of this were hard to quantify and not necessarily caught by this evaluation.

Part II displays some substantial improvements in requirement quality. Completeness, Traceability, Testability, Analysis base, and Understandability/readability all score high to fair improvements in comparison to previous requirements engineering/product management practices.

The quality attributes of Completeness, Un-ambiguity, Consistency, Catching dependencies/relationships, Redundancy, Structure, and Conformance to business goals all score very low on improvement. The overall explanation for this by the subjects themselves is the fact that full work-up of requirements according to RAM has not yet been implemented (they specify requirements on one level only, and use headers on the levels above). This was a conscious decision by ABB, planning for a stepwise implementation of which the first step is evaluated here. The implication being that the requirements created (one level of abstraction) suited certain roles and thus certain work efforts more than others in this phase of RAM implementation. For example, DEV saw some increase but not at all as high as e.g. PL or RE.

The use of full work-up (abstraction, and break-down) creating new requirements as needed is by no means a silver bullet taking care of all problems. However the predominant feeling amongst the subjects was that this could further increase accuracy and decrease effort needed to complete actions, as well as substantially improve requirements quality. Examples of this are specified below:

- Completeness, Un-ambiguity, Understandability/readability could be improved as requirements of different abstraction (specified on different levels) explicitly linked across levels offer different views of one or several requirements. In total offering a more complete picture, increasing understandability (especially the *why* sought by DEV), as the levels can be traversed. This gives the subject the choice of abstraction, and in total gives more information.
- Looking at Consistency, Dependencies/relations, and Redundancy, the full use of abstraction levels could make it possible to store general information (that is true for several requirements) in one or at least in a few requirements on a more abstract level. This would lessen Redundancy, as similar information does not have to be specified repeatedly. In addition, full use of abstraction levels in RAM enables (and forces) a more holistic view. Related requirements on a lower level all connect to a more abstract requirement on an upper level. For example, a Feature level requirement generally has several requirements on Function level that “belong to it”, i.e. they have to be implemented in order for the Feature level requirement to be fulfilled. This encourages a group view of requirements (lessening the focus on just individual requirements). This may increase Consistency and make Dependencies/relations more explicit.

- It should be noticed that the lack of improvement of certain aspects of the requirements engineering/product management process (as can be seen above) was predicted when the stepwise implementation plan was decided post the RAM tailoring workshop. The evaluation results confirm these predictions (made by the process improvement responsible personnel, not the users of RAM). Improved tool support is also a factor here. The use of an appropriate tool was considered a prerequisite to full RAM implementation. The acquisition and training in tool usage will be a part of the continuation of RAM implementation at ABB.

### **6.3. COMPARISON OF RAM EVALUATION RESULTS AT DHR AND ABB**

The comparison presented here is not aimed at comparing the companies. The objective is solely to compare the experiences in using RAM.

Substantial improvements in Action accuracy and fulfillment, as well as requirements quality can be seen in the evaluation results for both DHR and ABB. However, some interesting differences merit notice. At DHR the different subjects (representing the span from product management to development) are in agreement to a larger extent than at ABB. This is especially evident in Part II where most roles see a very similar increase in requirements quality (see Figure 54). At ABB the rated quality increase fluctuates from generally high when asking RE, PM and PL, to low when asking DEV. The main reason for this can be attributed to the fact that a stepwise implementation of RAM was chosen at ABB. In the first step (evaluated in this chapter), requirements were formed on one abstraction level only, resulting in that the increase in quality was perceived as substantially greater by the roles using requirements on that abstraction level. DEV in ABB's case saw some improvements compared to before, but the requirements were specified on a somewhat too abstract level. At DHR the full work-up of requirements offers requirements on different levels of abstraction, each targeted at a different group, thus offering requirements appropriate for e.g. RE and PM, but also for DEV.

An other noticeable difference between the two is that although DHR uses full work-up and thus in reality specifies more requirements relative to the incoming number, the redundancy is less than at ABB. Intuitively one might draw the conclusion that more specification equals more redundancy, but this does not seem to be the case. The reason for this can be found in the work-up itself. As requirements are abstracted and broken

down the more abstract requirements gather the general information that is true for all detailed requirements under them. This results in that information of a general (but important) nature does not have to be specified in every detailed requirement, but maybe only once or twice in a more abstract requirement related to the detailed ones. Looking at Part I, the effort needed to perform the Actions are in many cases lower at DHR than at ABB, this despite of full work-up.

The differences above do not come as a surprise to the process improvement team, and were expected by all team representatives, and as the second step of implementation is underway further improvements are anticipated.

It is important to notice that the process improvement activities and the evaluation presented in this chapter in no way can be generalized to other areas. Process improvement activities at ABB have a long and positive tradition, and the maturity of other processes cannot be inferred in any way using the results presented here. The improvements seen in each company are only relative within the companies themselves, thus an improvement at DHR cannot be compared to an equal improvement at ABB.

## 7. CONCLUSIONS

The overall results of the evaluations indicate that the implementation of RAM at DHR and ABB has yielded substantial increases in both accuracy of the practices (actions) performed in requirements engineering/product management, and in requirements quality. It is only natural that these improvements have a price, which can be observed in some effort increase over the board, although there are also examples of the opposite. A learning curve effect can be used to explain some of the increase, but for the most part increased accuracy and quality will have some cost as more work is performed. Although in the case of DHR and ABB, these costs are very moderate in total. The implications of improved quality and accuracy should also yield positive effects in the long run. For example, avoiding requirements overload by having improved acceptance/dismissal of requirements at an early stage will almost certainly save resources in development, enabling more "good" requirements to be implemented. One example of this is the possibility to reduce the number of pre-studies performed if rough estimates can be used to prioritize requirements, and enable the organization to focus on a set of requirements that is feasible. Improved estimation accuracy will thus enable better resource planning overall. Higher quality requirements in general should also decrease defects in later stages of development.

It could be argued that the implementation of any good requirement engineering practice would yield the same result as presented in this chapter. Without actually implementing e.g. two models in parallel, it is impossible to dismiss this possibility. However, several things can be said about RAM that could indicate some advantages. The concepts behind RAM are based on using the reality facing industry (e.g. large amounts of requirements, multiple abstraction levels and limited resources) to enable scalability. This is of course not proven totally as of yet, but based on the evaluations performed indications are positive. Actual practitioners used RAM in real development situations.

RAM is based on needs identified in industry but does not push a predefined one-size-fits-all set of practices and rules on all organizations looking to adopt it. Rather the model is tailorable, maintaining certain concepts (use of abstraction and attributes) but adaptable enough to fit different environments. In addition, RAM is tool independent, but requires tool support to be scalable and practical. The evaluation of RAM at both ABB and DHR indicates that the tailoring aspect of RAM has worked, although it should be realized that the environments are not totally heterogeneous (nor are they homogenous).

It is very important to notice that both DHR and ABB are very successful companies in terms of their domain, have excellent engineering practices, and are ultimately populated by professionals. In addition to this, they have the maturity to realize that improvement is always possible. These facts actually reflect positively on the evaluations as both companies had good requirements engineering/product management practices prior to RAM implementation. This implies that RAM was benchmarked against mature and working practices.

As with any process improvement activity, there are confounding factors when evaluating an improvement. As improvement activities commence (e.g. the workshop, presenting RAM etc), it gives rise to added awareness, increases knowledge, and creates curiosity. These things in themselves can improve practices independent of the actual process improvement. Although all of this can be seen as a part of the technology transfer process, and ultimately any improvement is a good thing.

## **8. FUTURE WORK**

There are several efforts presently underway at both DHR and ABB, and plans for the continuing evolution of RAM as a framework for requirements engineering and product management support. They are described briefly below.

At DHR, RAM is maturing and the initial learning curve with regards to RAM and the use of new tool support is subsiding, allowing for a natural adoption of the new way of working. A part of this is of course a constant monitoring of the process and RAM in order to tweak and adapt the model to the needs. A part of this is to refine the measurement programs at the company to minimize the effort needed to monitor the process improvement. The next step at DHR is to include explicit support for requirements prioritization and packaging of requirements in RAM. In addition, in-project requirements engineering (e.g. technical specification and general requirements maintenance) will be assessed and streamlined to suit the needs of the organization.

At ABB, the next step consists of tool acquisition and implementation of full RAM work-up at the PM units where it is used today. A second part is to implement RAM in several other neighboring PM groups (an activity preceded by additional RAM tailoring). Monitoring and stepwise refinement of all RAM instances implemented is also crucial and will be performed continuously, based on formal and informal evaluations.

RAM in general will continue to evolve. This will be achieved through lessons learned from industry implementation, but also through several experiments conducted in laboratory environment testing new concepts prior to industrial trials.



## **PART III**

# **Knowledge and Tech- nology Transfer**

---



# Chapter 9

---

## Technology and Knowledge Transfer in Practice – Requirements Engineering Process Improvement Implementation

*Tony Gorschek, Per Garre, Stig Larsson and Claes Wohlin*

IEEE Software, in print, 2006.

### **Abstract**

The successful transfer of knowledge and technology from research to practice implies symbiosis. Close cooperation and collaboration between researchers and practitioners benefit both parties. Researchers get the opportunity to observe first-hand the challenges facing industry, making it possible for research not only to be based on interesting issues, but industry relevant ones. This entails on-site observations and process assessments, as well as collaboration with industry champions when designing and validating new methods, tools and techniques. Industry practitioners benefit as new technology (e.g. methods, tools and techniques) are developed based on tangible and real issues identified on-site, and step-wise collaborative validation helps minimize technology transfer risks and maximize potential benefit. This partnership between researchers and practitioners builds trust, and ultimately makes technology and knowledge transfer possible.

We present a technology transfer process and experiences in using it. The process was devised and used during research conducted as a partnership between Blekinge Institute of Technology and two companies, Danaher Motion Särö AB (DHR) and ABB.

## 1. INTRODUCTION

The successful transfer of knowledge and technology from research to practice implies symbiosis. Close cooperation and collaboration between researchers and practitioners benefit both parties. Researchers get the opportunity to observe first-hand the challenges facing industry, making it possible for research not only to be based on interesting issues, but industry relevant ones. This entails on-site observations and process assessments, as well as collaboration with industry champions when designing and validating new methods, tools and techniques. Industry practitioners benefit as new technology (e.g. methods, tools and techniques) are developed based on tangible and real issues identified on-site, and step-wise collaborative validation helps minimize technology transfer risks and maximize potential benefit. This partnership between researchers and practitioners builds trust, and ultimately makes technology and knowledge transfer<sup>15</sup> possible.

We present a technology transfer process and experiences in using it. The process was devised and used during research conducted as a partnership between Blekinge Institute of Technology and two companies, Danaher Motion Särö AB (DHR) and ABB. The partnership was initiated with the intent of conducting industry relevant research in the area of requirements engineering. Technology transfer in this context is a prerequisite. From a research perspective technology transfer is needed to enable validation of research results in a real setting, while from an industry perspective technology transfer is a way to improve development and business processes.

From our experiences technology transfer (and industry relevant research) is not about producing research results and handing them over in the form of publications and technical reports. The process we used involved several steps, one building on the other, carried out during a long-term joint commitment. These steps were devised in close collaboration with industry, and although inspired by previous transfer models, see e.g. Pfleeger[183], it was created in an evolutionary manner, adding steps as needed. This evolution also dictated what each step of the process contained, e.g. how validation is performed depends on the needs of the company (what they trust), as well as the needs of the researchers to vali-

---

<sup>15</sup> We call it "technology transfer" or "transfer" for short.

date new technology for academic purposes. Taking industry preference into account when performing validation (as a part of technology transfer) is important for success[184].

The contribution of this article is to present a technology transfer model that was devised on-demand in collaboration with industry and equally important, to report experiences and lessons-learned from two cases.

Figure 57 gives an overview of our process divided into seven fairly straightforward steps, all which are relevant and interdependent for overall transfer success.

## 1.1. STEP 1 - BASING RESEARCH AGENDA ON INDUSTRY NEEDS

We started by assessing current practices and observing domain and business settings, getting an overview of the demands posed on industry[185]. Observation of the real world before we formulated our research questions was considered natural, and it is also crucial for the improvement endeavor[60]. Without a strong connection to needs perceived on-site by the practitioners themselves commitment can be difficult to obtain.

During the assessments and observations (see Figure 57, step 1) several potential areas for improvement within the fields of product management and requirements engineering were identified. These were subsequently analyzed and prioritized according to perceived importance, and dependency (see Chapter 3). Dependency in this case meant that future improvements (needs) should be transferred in a certain order, avoiding “hitching the cart in front of the horse”.

### Lessons learned – Step 1

- ⇒ Researchers’ on-site presence helps base the research agenda on real industry relevant issues, but also builds technical, organizational, and social/cultural understanding necessary to do a good job.
- ⇒ Becoming a friendly and easily recognizable presence gives the researchers extensive access to all groups of practitioners.
- ⇒ Information gathered should be balanced, i.e. there is a risk that certain groups of practitioners be premiered if they are vocal in comparison to others.
- ⇒ It is very important that all affected parts of the organization be involved, including upper management, middle management, engineers, support, as well as marketing and sales.

## **1.2. STEP 2 – PROBLEM FORMULATION**

Based on the prioritized needs a research agenda was formulated in close cooperation with industry contact persons, i.e. middle management practitioners. At this time these “contact persons” increased their committed and moved from participants to active “champions”, contributing substantially with input and facts as well as making easy and fast access to the company organizations possible. The researchers also had a regular presence on-site, at first learning and observing, then as time went by more actively exchanging ideas and concepts with multiple practitioners, not only the champions. Doing your home-work and learning the domain establishes a common understanding and vocabulary. Close proximity between researchers and practitioners is considered a crucial factor for success[60].

The main needs identified as high priority revolved around moving from bespoke “customer-developer” development to market-driven product-centered development. One of the main challenges in a market-driven environment is handling large volumes of requirements from multiple sources, both internal (e.g., developers, marketing, sales, support personnel and bug reports) and external (e.g., users, customers and competitors, often gathered via surveys, interviews, focus groups and competitor analysis)[10]. In this situation the volume of requirements to be handled makes initial screening important to decrease the risk of overloading in the evaluation and realization process[20]. In addition to this, the requirements themselves come in all different shapes and sizes, and on different levels of abstraction. Some of them very general and formulated like goals rather than requirements, while others very detailed and technical. There is a need to be able to handle all of them, make comparisons to product strategies and roadmaps, and to compare them in a prioritization situation. In our case both companies had similar needs, and thus a vested interest in addressing largely the same issues.

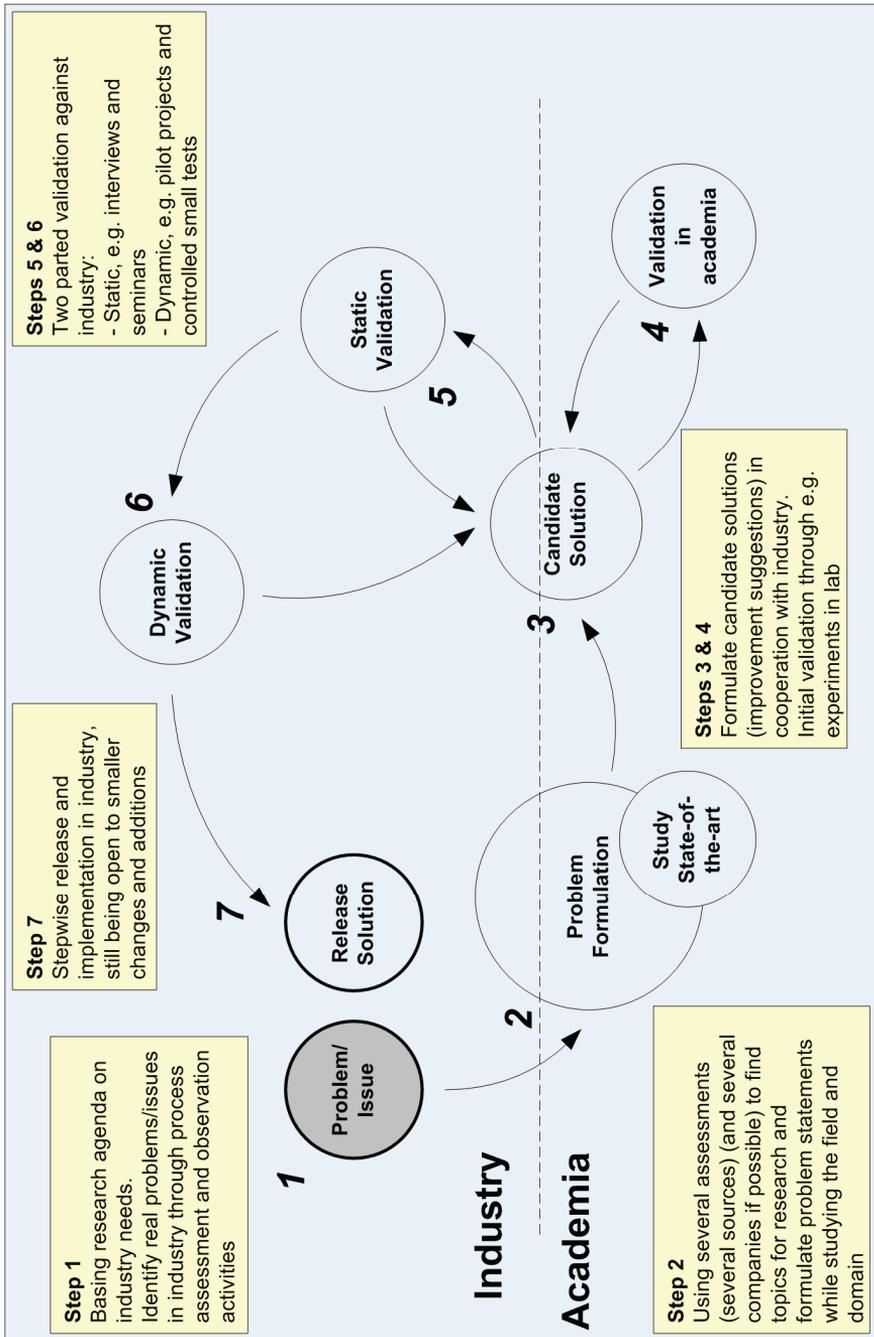


Figure 57. Research approach and technology transfer overview.

**Lessons learned – Step 2**

- ⇒ Doing your “homework” implies learning company and domain specific vocabulary, as well as understanding practitioners’ situation. This is needed to build trust.
- ⇒ The role of the industry champion(s) is formalized and long-term commitment is obtained as practitioners take part in the problem formulation – which is the basis for the research agenda, i.e. what is to be done and improved.

### **1.3. STEP 3 – FORMULATE A CANDIDATE SOLUTION**

Subsequent to establishing a research agenda the collaboration with industry continued with the design of candidate solutions (improvements). Being specific, a requirement engineering model was designed (called RAM – The Requirements Abstraction Model)[21]. The model is intended to incorporate possible solutions for many of the needs identified during the assessments at DHR and ABB, and primarily offers product planning and product management support. RAM is a multilevel requirements abstraction model, with a supporting process that aids practitioners in handling requirements in a structured and repeatable way, during requirements elicitation, analysis, refinement and management. The nature of the model is to *use* the fact that requirements come on different levels of abstraction instead of trying to flatten all or mixing different types in a document. Using RAM makes abstraction (comparing to strategies), and breakdown (until testable) of all requirements a part of the analysis and refinement work. As all requirements are abstracted and broken down it is also possible to compare them, making prioritization a realistic undertaking. (The RAM model itself is not detailed any further, for more information about RAM please see Gorschek and Wohlin[21].)

The candidate solution was created in collaboration with practitioners. The main responsibility of the researchers was to keep track of state-of-the-art in research and apply this in combination with new ideas and angles. Another aspect of having practitioners as collaborators is the fact that they keep it real and compatible with the surrounding environment on-site. One problem often encountered as research is to be transferred is that solutions don’t fit with the already present business/development methods[180, 186], increasing cost and effectively raising the bar for technology transfer.

**Lessons learned – Step 3**

- ⇒ Practitioners, in addition to being a resource, can provide a reality check, making sure that candidate solutions are realistic and to largest possible extent fit current practices and the situation at the company.
- ⇒ As candidate solutions are developed in collaboration with practitioners, commitment and trust are issues premiered. In addition the champions communicate and share ideas and information with colleagues, preparing for change in the mindset of the people in the organization.
- ⇒ Creating new solutions to issues identified is tempting. It is important that the researchers be the link to state-of-the-art in research, i.e. seeing to it that techniques, processes, and tools already developed and validated aren't ignored. In our case it meant building and refining some research results obtained by others, as well as invention of several new parts.

## **2. EVOLUTION AND TRANSFER PREPARATION THROUGH VALIDATION**

As the model (candidate solution) was formulated there was a need for evaluation. Several validation steps (and iterations of these steps) were used to accomplish this. The idea is to refine the candidate solution, test it for usability, scalability and if it actually addresses the needs in a satisfactory way. In addition, the validation steps are meant to gradually prepare for technology transfer. In this case both the solution itself needs to evolve (based on feedback from validation), but there is also a need to prepare the company. Preparation speaks to showing that there is a relative advantage to using the new solution in comparison to doing business as usual. This is crucial for getting commitment to transfer, something that researchers often miss[180].

### **2.1. STEP 4 – LAB VALIDATION**

First, the model was evaluated in a university lab environment prior to (and to some extent parallel with) static validation in industry (Figure 57, step 5). The evaluation was conducted in an experimental setting using master program students in software engineering as subjects. They were asked to use the model, performing requirements engineering activities as we envisioned them being performed in industry. The results from the evaluation made it possible to catch some issues without using industry resources. In addition we got early input as to the usability and scalability of the model. One motivation for early validation in academia is to ensure that pressure to transfer[31] does not take the upper hand, i.e. trying to transfer all research results indiscriminately.

**Lessons learned – Step 4**

- ⇒ An initial practical test of the candidate solution in a lab environment can give valuable and fast feedback, i.e. finding (in hindsight) obvious flaws and fixing them prior to industry piloting.
- ⇒ The results from a lab validation can be used when the candidate solution is presented widely to practitioners and management. Having initial results of using a candidate solution helps convince management of manageable risk and the potential benefit.
- ⇒ It is important to realize that lab validation is not the same as live industry usage. Realizing the limitations adds to the validity of the lab evaluation when presenting it to management in industry.

## 2.2. STEP 5 – STATIC VALIDATION

Static validation (Figure 57, step 5) involved widespread presentation of the model in industry, and at the same time collecting feedback from practitioners. First, the model was presented to all industry personnel that were involved in the initial process assessment (Figure 57, step 1). Practitioners representing developers, designers, project managers, product managers, and sales & marketing got to see the candidate solution through several seminars. In addition to getting feedback from practitioners (which further improved the model), the main goal was to give the practitioners feedback too. It was crucial to us that the practitioners that were to use the model got to voice their opinion early. Involvement and two-way communication between researchers and practitioners not only has the potential of improving results, but also laying the foundation for crucial issues such as shared commitment between researchers and practitioners, i.e. getting support for process change on all levels in the organization[60]. As an extension of the seminars several follow-up interviews were conducted.

The other main part of the static validation was presenting the research results and model to upper-management exclusively in a series of interactive seminars. Upper-management controlled resources needed for taking the next steps, basically implying that if they were not convinced of the benefit, the quality of the research could be irrelevant in terms of transferring it to practice. High quality research is not sufficient.

Management support is crucial[60, 186] and this fact cuts both ways. On the one hand, selling an idea (in this case the model) to management often refines arguments and can be a good control-point when it comes to devoting resources for technology transfer. On the other hand, at this early stage the research is unproven in real practice, thus it's hard to show ob-

jective facts (e.g. metrics) to motivate the transfer. These seminars also showed that the model had the support of operative personnel and middle management in the organization.

The bottom-line of the static validation was not just selling the model to industry representatives. The seminars and interviews gave invaluable feedback regarding usability and scalability of the model. For example about 15% of the contents of the model (e.g. attributes for requirements, validation steps of requirements in the process etc.) were stripped away during this stage. What had worked in theory (and gotten past lab evaluations in step 4) needed to be slimmed down. Very early during the static validation we realized that a smaller model that produced good-enough requirements and decision support materials was preferable to a larger model that risked not being used at all.

**Lessons learned – Step 5**

- ⇒ Widespread presentation of candidate solutions in the organization has several purposes: getting feedback and ideas for improvements, validating understanding and coverage, as well as giving feedback to the practitioners involved in the assessment phase in Step 1.
- ⇒ Anchoring future change (transfer) in all levels of an organization is crucial.
- ⇒ Upper-management seminars and interactive sessions are crucial to assess risk and potential relative benefit of the changes proposed through the transfer of the candidate solution. It is important to show that the candidate solution was developed in cooperation with and has the support of practitioners in the organization.
- ⇒ It is very important that researchers are not afraid of tailoring (even some down-scaling) the candidate solution at this stage. Change should be seen as a part of the candidate solution validation and refinement, and is natural as the ideas presented mature and grow over time.

### 2.3. STEP 6 – DYNAMIC VALIDATION (PILOTING)

Dynamic validation was performed through two pilot projects, both performed at DHR. The first one was limited in scope, using the model to elicit, analyze, refine, and manage about 75 requirements in total. The work in this first pilot was carried out by the researchers and an industry project manager (champion). The product being specified was an in-house product configuration tool and the customers were also situated in-house.

The success of this first pilot gave motivation and commitment by the company to continue the dynamic validation with a larger pilot. The second pilot was carried out using the model in a real development project, without the direct involvement of the researchers or industry champions, although the model used in the first pilot was reused to some extent. The

project had 18 developers and ran for about 4 months (3500 person-hours). The project manager was the primary requirements engineer; the developers used the requirements (produced with the model) in their work.

Both pilots gave valuable feedback, although not many new critical problems or potential difficulties were reported. The primary feedback we got was that tool support, as well as tool adaptation was important for model scalability. In addition there was a need to incorporate training efforts as a part of future model implementation. The main reason for there not being any big “surprises” during piloting was felt to be due to the static validation work and the continuous collaboration between the researchers and the industry champions. The relative success of the dynamic validations increased the credibility of the model, and built trust for us working with the model (researchers and champions), in the eyes of practitioners and management.

**Lessons learned – Step 6**

- ⇒ Piloting in industry gives the possibility to perform realistic evaluations of candidate solutions without giving up control (minimizing risk as the pilot is a limited test).
- ⇒ Pilots can be used to get input for further improvements, and indications of what is needed during the full scale transfer.
- ⇒ As pilots are limited in scope they may not catch all potential problems like scalability issues.
- ⇒ As pilots to some extent precede “proper” transfer (e.g. training is not formalized and the solution itself is not released) the usage of the candidate solution may differ somewhat from what was intended.

## 2.4. STEP 7 – RELEASE SOLUTION

Gauging the results from the validations (static and dynamic) a decision was made to give the go-ahead with the actual implementation of the model, i.e. official release. At DHR the model was to be implemented fully and incorporated in the official development and management process. At ABB the implementation was trialed and initially limited to one large release project, full implementation at ABB pending results from this “trial-release”. Despite of this difference, the technology transfer process and the techniques used to accomplish it were largely the same for both companies. It consisted of a tailoring of the model to fit each company specifically. RAM had been refined through the validations and pilots, but it was rather general in nature, a sort of least common denominator suiting both companies. A central part of the rationale behind the model was that we knew that one-size-does-not-fit-all, an important lesson on research from

working with two companies. Thus RAM can be tailored to fit different organizations taking into account e.g. differences in types of requirements, specific domains, and so on, catering to a specific organization's needs.

### 2.4.1. **MODEL TAILORING**

Tailoring was necessary for several reasons. The two companies had different definitions and vocabulary pertaining to products and development, and specific practices and processes already in place had to be taken into account to align the RAM model for transfer into a specific organization.

#### **One day workshop - tailoring and consensus**

The first stage in model tailoring consisted of a workshop conducted at each company over one entire day. Management, developers, and especially product and project managers (future primary users of the model) participated. Process owners were also present. The focus was on adapting the model not only to suit current practices, but e.g. roles and responsibilities set forward by the model were mapped to existing ones where possible. Concepts like good-enough were explored, establishing some examples of requirements through actual specification of requirements during the workshop.

The requirements that were jointly specified, refined and analyzed (using the model) were taken from within the company keeping it as "real" as possible. As we worked with the example requirements the researchers and the industry champion took turns, one leading the workshop and the other documenting results. The work with requirements proved to have a grounding-effect, keeping things tangible and discussions from becoming too abstract and theoretical.

Obtaining consensus among all groups represented and assuring that compromises made didn't impede usability, was the main objective of the workshop – thus also promoting future commitment.

#### **One week alignment – formalization and planning**

The information gathered during the workshop was crucial for the formalization and planning of the model implementation into the official development and business process. In short this meant converting workshop results into practice, starting with the development of needed documentation, user guides, example databases of requirements (reusing requirements from the workshop and creating additional ones), and tool support. Very early in the process of developing and validating the model itself we

realized that a document heavy, large, and complex model demanding too much from the practitioner may have been transferred to the official development and business process, but it would probably stay on the shelf collecting dust, next to other heavy frameworks and certifications.

**Documentation** consisted of two things that we separated to the largest possible extent. Formal documentation implied creating all mandatory official process descriptions, such as role descriptions, process maps, detailed requirement state diagrams, mapping to current processes, and glossaries etc. These official documents were needed, primarily as reference material, publicly accessible on a need-to-use basis. The user documentation was most important. It consisted of a very thin (fewer than 5 pages) quick reference guide.

**Training** was like in most process improvement efforts considered very important[180, 186], and we prioritized it. A positive spin-off effect of the collaborative research process utilized in general was that many aspects of the model were already to some part assimilated by practitioners in the companies as many of them had a part in creating or validating them. This meant that training was made easier, not least in terms of commitment.

The main parts of the training were performed utilizing a learning-by-doing strategy. The champion led training sessions where one or two practitioners used the model to engineer real new requirements. We nicknamed this “pair-requirements engineering” (as in pair-programming). The idea behind it was twofold; first as a good way of learning and getting experience, second, by performing actual work during training sessions the cost of training would decrease.

**Example driven** was one of the main points with the model and thus the training. The company specific (relevant) requirements developed during the workshop and the alignment week were used as good-practice examples (notice that we don’t use “best-practice”). The examples were in themselves an important part of the “documentation”, i.e. practitioners could use them in their training and later in their day-to-day work as an information source. For this reason we tried to categorize some different types of requirements (most common), and create a database of examples of each category.

**Tool support** was important. During the week we adapted the tool chosen by respective company. As the model is a way of working with requirements, it is largely tool-independent. This was evident as the companies chose to use different tools to support the model.

**Technical support** is a factor that can greatly influence the success of a technology transfer effort[31]. It is important that practitioners have access to continuous support from an expert, or at least an expert-user, enabling

continuous support as needed. Practitioners frustrated and stuck need help fast, or there is a risk that they go back to performing requirements engineering as they did previously. Both DHR and ABB have champions filling the role of expert.

**Owner and champion** is a role to some extent shared by the researchers and the champions during the research and validation of the solution candidate. However, as the candidate evolves and is released ownership has to be (and was) transferred to the company, and relevant champions on-site. This is necessary as the researchers cannot be present and active in supporting practitioners to the extent necessary. Optimally, with time and usage the model will be owned and championed collectively by all practitioners using it. This is however not the same as saying that the researchers' presence or participation is at an end, rather the opposite, as follow-up studies have to be performed.

**Metrics**, i.e. the collection of data regarding the usage of the model is a crucial part of a technology transfer process enabling measurement and follow-up studies. During the alignment week a plan for how to measure (what metrics to collect) was formulated. It was important to enable measurement of the model usage for several reasons. First, after a time metrics collected could indicate benefit and/or hint at potential problems.

In addition to quantitative measurement qualitative follow-up was also planned, e.g. interviews and group discussions. The combination of qualitative and quantitative follow-up is intended to provide feedback for future changes and modifications.

**Lessons learned – Step 7**

- ⇒ One-size-does-not-fit-all. A general candidate solution devised as research results cannot be expected to fit directly, adaptation and tailoring is often a prerequisite to transfer.
- ⇒ Tailoring as a way to get consensus ensures further commitment.
- ⇒ It is not possible to satisfy every need (or solve every problem).
- ⇒ Several things need to be developed to support candidate solutions:
  - Light-weight documentation and reference guides (example driven and domain specific)
  - Training strategy and materials (e.g. performing real work when training cuts costs) – Pair-RE
  - Tools need to be tested, chosen, acquired, tailored and training offered (as needed)
  - Technical support has to be made available to practitioners (e.g. a role that can be filled by champions)
- ⇒ Measurement programs need to be planned and realized in order to gather relevant information regarding the candidate solution in operation.

#### 2.4.2. **PROCESS OBSERVATIONS**

The actual process used was formulated in close collaboration between the researchers and the industry partners. This was crucial to obtain commitment and trust from everybody involved. However, it is interesting to note that the actual process formulated for the two companies in Sweden resembles the levels presented as the Technology Readiness Levels[187] as formulated by government bodies in the US. This implies that there is a growing consensus regarding which steps or levels are needed for successful transfer and adoption new technology.

### 3. **CONCLUSIONS IN PERFECT HINDSIGHT**

Looking back several issues come to mind as “critical success factors”. Commitment, having champions, collective ownership, building trust, and training are all examples. We will not try to cover all of them here, rather mention a few things in addition to the experiences shared this far - all in perfect hindsight of course.

#### **Long-term commitment – Short-term-long-term benefits**

Neither research nor technology transfer is a one-shot deal. It requires long-term commitment from both researchers and practitioners. Technology transfer happens over time as small bits break-off and is adopted by practitioners continuously. This can be frustrating, from a research perspective, as it is hard to measure the impact of such “improvements”. The collaborative research process itself can be seen as implicit technology transfer under favorable conditions.

Long-term in this case applies to the lead-time required. This is partially due to that practitioners perform their daily work in addition to working with the researchers. This is not the same thing as lack of commitment, rather that practitioners are pressed for time as they operate in a competitive environment[31]. Performing lab validations prior to industry trials is a good idea as it can help catch issues without consuming resources.

#### **Risk minimization**

The ability to maximize potential benefit and minimize the potential risk is of paramount importance to companies. The validation in iterations improved the model and helped us convince management and practitioners (and ourselves) that the risks were acceptable in comparison to the potential benefit. This said, the researchers’ job isn’t just research, its making transfer happen. Active participation in some activities that by themselves

don't produce research results (papers) can be considered as overhead from a research perspective, but a necessity from an industry perspective.

We were fortunate in our endeavors as both researchers and practitioners have a pragmatic take on things. Research results were considered important, but from our point of view the value of these results were directly linked to usability in industry.

### **Technology transfer**

The model used for technology transfer was not prepackaged. The steps and the contents of each step were added on-demand to be appropriate for the situation. In some cases this meant additional work to provide sufficient evidence for e.g. management showing the relative value of a new way of working. In terms of future research this suggests flexibility – on-demand modification and addition of steps and iterations to satisfy both industry demands of risk minimization and relative value evidence, as well as the needs of the researchers. Our technology transfer model should be seen as an instantiation of a technology transfer process, which can be used for inspiration rather than prescription.

# References

---

1. Ruhe G, Greer D (2003) Quantitative Studies in Software Release Planning under Risk and Resource Constraints. In Proceedings of the 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. Regnell B, Brinkkemper S (2005) *Market-Driven Requirements Engineering for Software Products*. Engineering and Managing Software Requirements. Springer, New York NY.
5. Kotonya G, Sommerville I (1998) *Requirements Engineering: Processes and Techniques*. John Wiley, New York NY.
6. Wieringa R, Ebert C (2004) Guest Editors' Introduction: RE'03: Practical Requirements Engineering Solutions. *IEEE Software* 21(2):16-18.
7. 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.
8. Kotler P, Armstrong G (2001) *Principles of Marketing*. Prentice Hall, Upper Saddle River NJ.
9. Gorschek T, Davis A (2005) Assessing the Quality of Requirements Process Changes. In Proceedings of the Eleventh International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'05), Universität Duisburg-Essen, Porto, Portugal. Download at: <http://www.bth.se/fou/Forskinfor/nsf/> . pp. 101-112.
10. Lehmann DR, Winer RS (2002) *Product Management*. McGraw-Hill, Boston MA.
11. Mintzberg H, Ahlstrand BW, Lampel J (1998) *Strategy Safari : A Guided Tour through the Wilds of Strategic Management*. Free Press, New York NY.
12. Carlshamre P (2002) Release Planning in Market-Driven Software Product Development: Provoking an Understanding. *Requirements Engineering* 7(3):139-151.
13. Bray IK (2002) *An Introduction to Requirements Engineering*. Addison-Wesley, Dorset.

14. Glass RL (1998) *Software Runaways*. Prentice Hall, Upper Saddle River NJ.
15. Connolly TM, Begg CE (1998) *Database Systems: A Practical Approach to Design, Implementation and Management*. Addison-Wesley, Harlow.
16. Jirotko M, Goguen JA (1994) *Requirements Engineering Social and Technical Issues*. Academic Press, London.
17. 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.
18. Sommerville I, Sawyer P (1999) *Requirements Engineering: A Good Practice Guide*. John Wiley & Sons, Chichester.
19. Davis AM (1993) *Software Requirements: Objects, Functions, and States*. Prentice Hall, Englewood Cliffs NJ.
20. Weber M, Weisbrod J (2003) Requirements Engineering in Automotive Development: Experiences and Challenges. *IEEE Software* 20(1):16-24.
21. Gorschek T (2004) *Software Process Assessment & Improvement in Industrial Requirements Engineering*. Licentiate Thesis No. 2004:07, ISBN 91-7295-041-2. Blekinge Institute of Technology, Ronneby. <http://www.ipd.bth.se/tgo/licentiate/papers/Licentiate>
22. 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)*. Pittsburgh PA.
23. (2002) <http://www.tickit.org/>. Last Accessed: 2002-10-01.
24. IEEE (1984) *Recommended Practice for Software Requirements Specifications (Standard 830-1984)*. IEEE Press, New York NY.
25. Sawyer P, Kotonya G (2001) *Software Requirements. Guide to the Software Engineering Body of Knowledge SWEBOK, Trial Version 1.0*. IEEE, Los Alamitos CA.
26. 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.
27. 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.

28. Kuilboer JP, Ashrafi N (2000) Software Process and Product Improvement: An Empirical Assessment. *Information and Software Technology* 42(1):27-34.
29. Reifer DJ (2000) The CMMI: It's Formidable. *Journal of Systems and Software* 50(2):97-98.
30. Zahran S (1998) *Software Process Improvement: Practical Guidelines for Business Success*. Addison-Wesley, Reading MA.
31. 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.
32. Glass RL (1994) The Software-Research Crisis. *IEEE Software* 11(6):42-47.
33. Shewhart WA, Deming WE (1986) *Statistical Method from the Viewpoint of Quality Control*. Dover, New York NY.
34. Basili VR (1985) *Quantitative Evaluation of Software Methodology*. University of Maryland, College Park, Maryland.
35. El Emam KE, Madhavji NHE (1999) *Elements of Software Process Assessment & Improvement*. Wiley-IEEE, Los Alamitos CA.
36. 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.
37. (2004) <http://herkules.oulu.fi/isbn9514265084/html/x287.html>. Last Accessed: 2004-04-11.
38. 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. N/A.
39. Paulk MC, Curtis B, Chrissis MB, Weber CV (1995) *The Capability Maturity Model: Guidelines for Improving the Software Process*. Addison-Wesley, Reading MA.
40. Paulk MC (1995) *The Capability Maturity Model : Guidelines for Improving the Software Process*. Addison-Wesley, Reading MA.
41. CMMI Product Development 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)*.

42. McFeeley B (1996) *Idealsm: A User's Guide for Software Process Improvement*. TR-CMU/SEI-92-TR004. SEI, Pittsburgh.
43. (2004) <http://www.sei.cmu.edu/cmm/cmm.html>. Last Accessed: 2004-04-11.
44. (2004) <http://www.sei.cmu.edu/cmm/ipd-cmm.html>. Last Accessed: 2004-04-11.
45. (1998) <http://www.sei.cmu.edu/iso-15504/>. Last Accessed: 2004-01-07.
46. Ahern DM, Clouse A, Turner R (2003) *CMMI Distilled: A Practical Introduction to Integrated Process Improvement*. Addison-Wesley, Boston MA.
47. PDT S (2001) *Standard CMMI Appraisal Method for Process Improvement (SCAMPI) Version 1.1* (CMU/SEI-2001-Hb-001). Carnegie Mellon SEI. Pittsburgh, Pa.
48. CMMI-PDT (2001) *Appraisal Requirements for CMMISM, Version 1.1* (ARC, V1.1).
49. 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.
50. (2003) <http://www.sqi.gu.edu.au/spice/>. Last Accessed: 2003-09-11.
51. Wiegers KE, Sturzenberger DC (2000) A Modular Software Process Mini-Assessment Method. *IEEE Software* 17(1):62-70.
52. Beecham S, Hall T, Rainer A (2003) Software Process Improvement Problems in Twelve Software Companies: An Empirical Analysis. *Empirical Software Engineering* 8(1):7-42.
53. Schneider K (2000) Active Probes Synergy in Experience-Based Process Improvement. *Lecture Notes in Computer Science* 1840 6-19.
54. Niazi M, Wilson D, Zowghi D (2005) A Maturity Model for the Implementation of Software Process Improvement: An Empirical Study. *Journal of Systems and Software* 74(2):155-172.
55. 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.
56. 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.
57. 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.

58. 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.
59. Conradi R, Fuggetta A (2002) Improving Software Process Improvement. IEEE Software 19(4):92-100.
60. 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.
61. 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.
62. Basili V, Green S (1994) Software Process Evolution at the SEL. IEEE Software 11(4):58-66.
63. (2003) <http://sme.cordis.lu/home/index.cfm>. Last Accessed: 2003-05-05.
64. 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.
65. 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.
66. 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.
67. 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.
68. 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.
69. Haley TJ (1996) Software Process Improvement at Raytheon. IEEE Software 13(6):33-41.

70. 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.
71. 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.
72. 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.
73. Sawyer P, Sommerville I, Viller S (1999) Capturing the Benefits of Requirements Engineering. *IEEE Software* 16(2):78-85.
74. (2003)  
<http://www.comp.lancs.ac.uk/computing/research/cseg/projects/realms/index.html>. Last Accessed: 2003-05-01.
75. Gorschek T, Svahnberg M, Tejle K (2003) Introduction and Application of a Lightweight Requirements Engineering Process Evaluation Method. In *Proceedings of the Ninth International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'03)*, Universität Duisburg-Essen, Essen, Germany. Download at: <http://www.bth.se/fou/Forskinforso.nsf/> . pp. 101-112.
76. Beecham S, Hall T, Rainer A (2005) Defining a Requirements Process Improvement Model. *Software Quality Journal* 13(3):247-279.
77. 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.
78. Nguyen L, Swatman PA (2003) Managing the Requirements Engineering Process. *Requirements Engineering* 8(1):55 - 68.
79. Coakes JM, Coakes EW (2000) Specifications in Context: Stakeholders, Systems and Modeling of Conflict. *Requirements Engineering* 5(2):103-113.
80. Berander P, Andrews A (2005) Requirements Prioritization. *Engineering and Managing Software Requirements*. Springer, New York NY.
81. Wohlin C, Aurum A, (editors) (2005) *Engineering and Managing Software Requirements*. Springer, New York, NY.

82. Laitenberger O, Beil T, Schwinn T (2002) An Industrial Case Study to Examine a Non-Traditional Inspection Implementation for Requirements Specifications. In Proceedings of the Eighth IEEE Symposium on Software Metrics, IEEE Computer Society, Los Alamitos CA, pp. 97-106.
83. Shull F, Rus I, Basili V (2000) How Perspective-Based Reading Can Improve Requirements Inspections. *Computer* 33(7):73-79.
84. Porter AA, Votta LGJ, Basili VR (1995) Comparing Detection Methods for Software Requirements Inspections: A Replicated Experiment. *IEEE Transactions on Software Engineering* 21(6):563-576.
85. Gorschek T, Dzamashvili - Fogelström N (2005) Test-Case Driven Inspection of Pre-Project Requirements - Process Proposal and Industry Experience Report. In the Requirements Engineering Decision Support Workshop held in Conjunction with the 13th IEEE International Conference on Requirements Engineering, IEEE, Paris, pp. N/A.
86. Rakitin SR (2001) *Software Verification and Validation for Practitioners and Managers*. Artech House, Boston MA.
87. Basili VR, Laitenberger O, Shull F, Rus I (2000) Improving Software Inspections by Using Reading Techniques. In Proceedings of the 2000 International Conference on Software Engineering, IEEE, Los Alamitos CA, pp. 836.
88. Maiden N (2005) What Has Requirements Research Ever Done for Us. *IEEE Software* 22(4):104-105.
89. Potter B, Sinclair J, Till D (1996) *An Introduction to Formal Specification and Z*. Prentice Hall, London.
90. Potts C (1995) Invented Requirements and Imagined Customers: Requirements Engineering for Off-the-Shelf Software. In Proceedings of the Second IEEE International Symposium on Requirements Engineering IEEE, Los Alamitos, pp. 128-130.
91. Regnell B, Beremark P, Eklundh O (1998) A Market-Driven Requirements Engineering Process - Results from an Industrial Process Improvement Programme. *Requirements Engineering* 3(2):121-129.
92. Balm GJ (1996) Benchmarking and Gap Analysis: What Is the Next Milestone? *Benchmarking for Quality Management & Technology* 3(4):28-33.
93. Loucopoulos P, Karakostas V (1995) *System Requirements Engineering*. McGraw-Hill, New York NY.

94. Soffer P, Goldin L, Kuflik T (2005) A Unified RE Approach for Software Product Evolution: Challenges and Research Agenda. In the International Workshop SREP'05, held in conjunction with the IEEE International Requirements Engineering Conference 2005., IEEE, Los Alamitos CA, pp. N/A.
95. Simmons E (2004) Requirements Triage: What Can We Learn from A "Medical" Approach? *IEEE Software* 21(04):86-88.
96. Davis AM (2003) The Art of Requirements Triage. *IEEE Computer* 36(3):42-49.
97. Dahlstedt A, Karlsson L, Persson A, NattochDag J, Regnell B (2003) Market-Driven Requirements Engineering Processes for Software Products – a Report on Current Practices. In the International Workshop on COTS and Product Software RECOTS, held in conjunction with the 11th IEEE International Requirements Engineering Conference, IEEE, Los Alamitis CA, pp. N/A.
98. 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.
99. Albrecht AJ, Gaffney JE (1983) Software Function, Source Lines of Code, and Development Effort Prediction. *IEEE Transactions Software Engineering* SE-9(6):639 - 647.
100. Hihn J, Habib-agahi H (1991) Cost Estimation of Software Intensive Projects: A Survey of Current Practices. In *Proceedings of the 13th International Conference on Software Engineering*, IEEE, Los Alamitos CA, pp. 276-287.
101. 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 the Fifth IEEE International Symposium on Requirements Engineering*, IEEE, Los Alamitos CA, pp. 84-92.
102. Saaty TL, Vargas LG (2001) *Models, Methods, Concepts & Applications of the Analytic Hierarchy Process*. Kluwer Academic Publishers, Boston MA.
103. Leffingwell D, Widrig D (2000) *Managing Software Requirements: A Unified Approach*. Addison-Wesley, Reading MA.
104. Feather MS, Menzies T (2002) Converging on the Optimal Attainment of Requirements. In *IEEE Joint International Conference on Requirements Engineering*, IEEE, Los Alamitos CA, pp. 263-270.

105. Karlsson L, Berander P, Regnell B, Wohlin C (2004) Requirements Prioritisation: An Experiment on Exhaustive Pair-Wise Comparisons Versus Planning Game Partitioning. In Proceedings of the 8th International Conference on Empirical Assessment in Software Engineering (EASE 2004) - (In proceedings of ICSE 2004), IEEE, Los Alamitos, pp. 145-154.
106. Wiegers KE (1999) Software Requirements. Microsoft Press, Redmond WA.
107. Kostoff RN, Schaller RR (2001) Science and Technology Roadmaps. Engineering Management, IEEE Transactions on 48(2):132-143.
108. Kappel TA (2001) Perspectives on Roadmaps: How Organizations Talk About the Future. Journal of Product Innovation Management 18(1):39-50.
109. Gorchels L (2000) The Product Manager's Handbook : The Complete Product Management Resource. NTC Business Books, Lincolnwood Ill.
110. Ross SA, Westerfield R, Jordan BD (2001) Essentials of Corporate Finance. McGraw-Hill, Boston MA.
111. Teece DJ (2000) Managing Intellectual Capital : Organizational, Strategic, and Policy Dimensions. Oxford University Press, Oxford.
112. Gorschek T, Davis A (2005) Requirements Engineering: In Search of the Dependent Variables. Submitted to Information and Software Technology (can be obtained by email) N/A(N/A):N/A.
113. Wohlin C, Aurum A (2005) What Is Important When Deciding to Include a Software Requirement in a Project or Release? Empirical Software Engineering, 2005. 2005 International Symposium on 237-246.
114. Sasaki T, Nagata A, Toyama R, Hirata T, Hasegawa K (2001) Coevolution of Patent Strategy and Product Strategy. In Portland International Conference on Management of Engineering and Technology PICMET '01, IEEE, Los Alamitos CA, pp. 481-484.
115. Vähäniitty J, Lassenius C, Rautiainen K (2002) An Approach to Product Roadmapping in Small Software Product Businesses. In 7th European Conference on Software Quality, Conference notes, pp. 12-13.
116. Carlshamre P, Regnell B (2000) Requirements Lifecycle Management and Release Planning in Market-Driven Requirements Engineering Processes. In Proceedings of the 11th International Workshop on Database and Expert Systems Applications, IEEE, Los Alamitos CA, pp. 961-965.

117. Greer D, Ruhe G (2004) Software Release Planning: An Evolutionary and Iterative Approach. *Information and Software Technology* 46(4):243-253.
118. Regnell B, Karlsson L, Host M (2003) An Analytical Model for Requirements Selection Quality Evaluation in Product Software Development. In *Proceedings of the 11th International Conference on Requirements Engineering, IEEE, Los Alamitos CA*, pp. 254-263.
119. Karlsson L, Regnell B, Karlsson J, Olsson S (2003) Post -Release Analysis of Requirements Selection Quality – an Industrial Case Study. In *Proceedings of the Ninth International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'03), Universität Duisburg-Essen, Essen, Germany*. Download at: <http://www.bth.se/fou/Forskininfo.nsf/> . pp. 47-56.
120. Redling TJ (2003) A Methodology for Developing New Product Line Requirements through Gap Analysis. In *Proceedings 22nd Digital Avionics Systems Conference, IEEE, Indianapolis*, pp. 10-1.
121. Zairi M (1999) *Best Practice : Process Innovation Management*. Butterworth-Heinemann, Boston MA.
122. Hill N, Brierley J, MacDougall R (1999) *How to Measure Customer Satisfaction*. Gower, Aldershot ; Brookfield.
123. Kotler P, Armstrong G, Saunders J, Wong V (2002) *Principles of Marketing*. Prentice Hall, New York NY.
124. Davis A, Bersoff E, Comer E (1998) A Strategy for Comparing Alternative Software Development Life Cycle Models. *IEEE Transactions on Software Engineering* 14(10):1453-1461.
125. Deifel B (1999) A Process Model for Requirements Engineering of Ccots. In *10th International Workshop on Database & Expert Systems Applications, IEEE, Los Alamitos*, pp. 316-321.
126. Hass AMJ (2003) *Configuration Management Principles and Practice*. Addison-Wesley, Reading MA.
127. Höst M, Regnell B, Natt och Dag J, Nedstam J, Nyberg C (2000) Exploring Bottlenecks in Market-Driven Requirements Management Processes with Discrete Event Simulation. In *Proceedings of PROSIM2000, London*, pp.
128. 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.
129. Wohlin C, Runeson P, Höst M, Ohlson MC, Regnell B, Wesslén A (2000) *Experimentation in Software Engineering: An Introduction*. Kluwer Academic, Boston MA.

130. Stake RE (1995) *The Art of Case Study Research*. Sage Publications, Thousand Oaks.
131. Robson C (2002) *Real World Research: A Resource for Social Scientists and Practitioner-Researchers*. Blackwell Publishers, Oxford.
132. Bratthall L, Joergensen M (2002) Can You Trust a Single Data Source Exploratory Software Engineering Case Study? *Empirical Software Engineering* 7(1):9-26.
133. Kitchenham B, Pfleeger SL, Pickard L (1995) Case Studies for Method and Tool Evaluation. *IEEE Software* 12(4):52-63.
134. Gorschek T, Wohlin C (2003) Identification of Improvement Issues Using a Lightweight Triangulation Approach. In *Proceedings of the European Software Process Improvement Conference (EuroSPI'2003)*, Verlag der Technischen Universität, Graz, Austria. Download at: <http://www.bth.se/fou/Forskinfo.nsf/>, pp. VI.1-VI.14.
135. 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.
136. Robertson S, Robertson J (1999) *Mastering the Requirements Process*. Addison-Wesley, Harlow.
137. 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.
138. 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.
139. Rupp C (2002) Requirements and Psychology. *IEEE Software* 19(3):16-19.
140. Potts C (2001) Metaphors of Intent. In *Proceedings of the Fifth IEEE International Symposium on Requirements Engineering*, IEEE, Los Alamitos CA, pp. 31-38.
141. Sommerville I, Rodden T, Sawyer P, Bentley R, Twidale M (1992) Integrating Ethnography into the Requirements Engineering Process. In *Proceedings of the IEEE International Symposium on Requirements Engineering*, IEEE, Los Alamitos CA, pp. 165-173.
142. Karlsson J, Wohlin C, Regnell B (1998) An Evaluation of Methods for Prioritizing Software Requirements. *Information and Software Technology* 39(14-15):939-947.

143. Karlsson J (1996) Software Requirements Prioritizing. In Proceedings of the Second International Conference on Requirements Engineering, IEEE, Los Alamitos CA, pp. 110-116.
144. 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.
145. Lam W (1998) A Case-Study of Requirements Reuse through Product Families. *Annals of Software Engineering* 5(X):253-277.
  
146. Roudies O, Fredj M (2001) A Reuse Based Approach for Requirements Engineering. In Proceedings of the ACS/IEEE International Conference on Computer Systems and Applications, IEEE, Los Alamitos CA, pp. 448-450.
147. 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.
148. Ramesh B, Jarke M (2001) Toward Reference Models for Requirements Traceability. *IEEE Transactions on Software Engineering* 27(1):58-94.
149. Deming WE (1986) *Out of the Crisis*. Massachusetts Institute of Technology Center for Advanced Engineering Study, Cambridge.
150. Saaty TL (1980) *The Analytic Hierarchy Process: Planning, Priority Setting, Resource Allocation*. McGraw-Hill, London.
151. Chu P, Liu JK-H (2002) Note on Consistency Ratio. *Mathematical and Computer Modeling* 35(9-10):1077-1080.
152. Apostolou B, Hassell JM (2002) Note on Consistency Ratio: A Reply. *Mathematical and Computer Modeling* 35(9-10):1081-1083.
153. Beck K, Fowler M (2001) *Planning Extreme Programming*. Addison-Wesley, Boston MA.
154. Dahlstedt Å, Persson A (2003) Requirements Interdependencies - Molding the State of Research into a Research Agenda. In the Ninth International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'03), Universität Duisburg-Essen, Essen, Germany, pp. 71-80.
155. Higgins SA, de Laat M, Gieles PMC (2003) Managing Requirements for Medical It Products. *IEEE Software* 20(1):26-34.
156. 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.

157. Potts C, Hsi I (1997) Abstraction and Context in Requirements Engineering: Toward a Synthesis. *Annals of Software Engineering* 3(X):23-61.
158. Anton AI (1996) Goal-Based Requirements Analysis. In *Proceedings of the Second International Conference on Requirements Engineering*, IEEE, Los Alamitos CA, pp. 136-144.
159. Kavakli E, Loucopoulos P, Filippidou D (1996) Using Scenarios to Systematically Support Goal-Directed Elaboration for Information System Requirements. In *Proceedings of the IEEE Symposium and Workshop on Engineering of Computer-Based Systems*, IEEE, Los Alamitos CA, pp. 308-314.
160. Castro J, Kolp M, Mylopoulos J (2002) Towards Requirements-Driven Information Systems Engineering: The Tropos Project. *Information Systems* 27(6):365-389.
161. Lauesen S (2000) *Software Requirements: Styles and Techniques*. Samfundslitteratur, Fredriksberg.
162. Pfleeger SL (1998) *Software Engineering: Theory and Practice*. Prentice-Hall, Upper Saddle River NJ.
163. 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.
164. Juristo N, Moreno AM, Silva A (2002) Is the European Industry Moving toward Solving Requirements Engineering Problems? *IEEE Software* 19(6):70-78.
165. Humphrey WS (2000) *Introduction to the Team Software Process*. Addison-Wesley, Reading MA.
166. Gorschek T, Wohlin C (2004) Packaging Software Process Improvement Issues - a Method and a Case Study. *Software: Practice & Experience* 34(14):1311-1344.
167. (1998) *IEEE Standard for Software Verification and Validation*. IEEE Std. 1012-1998
168. Clegg C (1996) *The Performance of Information Technology and the Role of Human and Organizational Factors - the OASIG Study*. University of Sheffield, Sheffield.
169. Fagan ME (1976) Design and Code Inspection to Reduce Errors in Program Development. *IBM Systems Journal* 15(3):182-211.
170. Ciolkowski M, Laitenberger O, Biffl S (2003) *Software Reviews: The State of the Practice*. IEEE Software 20(6):46-51.
171. Russell GW (1991) Experience with Inspection in Ultralarge-Scale Developments. *IEEE Software* 8(1):25-32.

172. Wheeler DA, Brykczynski B, Meeson RN (1996) Software Inspection : An Industry Best Practice. IEEE, Los Alamitos CA.
173. Aurum A, Petersson H, Wohlin C (2002) State-of-the-Art: Software Inspections after 25 Years. *Software Testing Verification and Reliability* 12(3):93-122.
174. Basili V, Laitenberger O, Shuffl F, Rus I (2000) Improving Software Inspections by Using Reading Techniques. In *Proceedings of the IEEE International Conference on Software Engineering*, IEEE, Los Alamitos CA, pp. 836.
175. Basili VR, Green S, Laitenberger O, Lanubile F, Shull F, Sorumgard S, Zelkowitz MV (1996) The Empirical Investigation of Perspective-Based Reading. *Empirical Software Engineering Journal* 1(2):133-164.
176. Basili VR (1997) Evolving and Packaging Reading Technologies. *Journal of Systems and Software* 38(1):3-12.
177. Thelin T, Runeson P, Wohlin C, Olsson T, Andersson C (2004) Evaluation of Usage-Based Reading-Conclusions after Three Experiments. *Empirical Software Engineering* 9(1-2):77-110.
178. Sauer C, Jeffery DR, Land L, Yetton P (2000) The Effectiveness of Software Development Technical Reviews: A Behaviorally Motivated Program of Research. *IEEE Transactions on Software Engineering* 26(1):1-14.
179. Votta LGJ (1993) Does Every Inspection Need a Meeting? In *Proceedings of the 1st ACM SIGSOFT Symposium on Foundations of Software Engineering*, ACM, New York NY, pp. 107-114.
180. Morris P, Masera M, Wilikens M (1998) Requirements Engineering and Industrial Uptake. In *Proceedings of the Third International Conference on Requirements Engineering*, IEEE, Los Alamitos, CA, pp. 130-137.
181. Gorschek T, Svahnberg M (2005) A Controlled Empirical Evaluation of a Requirements Abstraction Model. Submitted to *Information and Software Technology* (can be obtained by email) N/A(N/A):N/A.
182. Gorschek T, Svahnberg M, Borg A, Börstler J, Eriksson M, Lonconsole A, Sandahl K (2005) A Replicated Controlled Empirical Evaluation of a Requirements Abstraction Model. Submitted to *IEEE Transactions on Software Engineering* (can be obtained by email) N/A(N/A):N/A.
183. Pfleeger SL (1999) Understanding and Improving Technology Transfer in Software Engineering. *Journal of Systems and Software* 47(2-3):111-124.
184. Pfleeger SL, Menezes W (2000) Marketing Technology to Software Practitioners. *IEEE Software* 17(1):27-33.

185. Gorschek T, Wohlin C (2003) Identification of Improvement Issues Using a Lightweight Triangulation Approach. In Proceedings of European Software Process Improvement Conference (EuroSPI'2003), Verlag der Technischen Universität, Graz, Austria. Download at: <http://www.bth.se/fou/Forskinfo.nsf/>, pp. VI.1-VI.14.
186. Miller S (1997) How Can Requirements Engineering Research Become Requirements Engineering Practice? In Proceedings of the Third IEEE International Symposium on Requirements Engineering, IEEE, Los Alamitos CA, pp. 260.
187. Mankins JC (1995) Technology Readiness Levels - a White Paper. Office of Space Access and Technology, NASA, <http://advtech.jsc.nasa.gov/downloads/TRLs.pdf>





