Integrated Requirements Engineering – Understanding and Bridging Gaps in Software Development

Bjarnason, Elizabeth

2013

Integrated Requirements Engineering – Understanding and Bridging Gaps within Software Development

Elizabeth Bjarnason

Doctoral Dissertation, 2013

Department of Computer Science
Lund University

Dissertation 43, 2013
LU-CS-DISS: 2013-02

ISBN 978-91-7473-732-5 (printed version)
ISBN 978-91-7473-733-2 (electronic version)
ISSN 1404-1219

Department of Computer Science
Lund University
Box 118
SE-221 00 Lund
Sweden

Email: elizabeth@cs.lth.se

Printed in Sweden by Tryckeriet i E-huset, Lund, 2013

You're blessed when you can show people how to cooperate instead of compete or fight.

Jesus in Matthew 5:9. The Message, NavPress Publishing Group

ABSTRACT

Software systems are becoming increasingly ubiquitous and can be found in devices we use every day from mobile phones to cars. As our reliance on software-based systems increases, our tolerance with software that is ill-fitted to our needs decreases. We expect these devices to function whenever and however we need them to. However, developing these (often) large and complex software systems to meet our needs and to be usable and robust is non-trivial. Yes, it requires good tools and methods, and competent software engineers that are good at design, development, testing and debugging. But, equally important is that the engineers can coordinate their activities and work together on developing *the right thing*. Figuring out what *the right thing* is, i.e. defining the requirements, and then ensuring that the whole development team joins together in realising this is a major challenge.

When there are gaps between requirements and other development activities these gaps have a negative impact on the success of a project and a product. Similarly when there are no gaps, or they are effectively managed and bridged, the development process can run more smoothly and the project stands a better chance of delivering the required functionality, with good quality and on time.

The main topic of this thesis is the collaboration and alignment of requirements within software development, and how this can enable a development company to consistently develop and deliver products that are well received by their users. A number of challenges and practices including factors contributing to these have been identified through industrial case studies. Furthermore, this thesis includes two methods for supporting project teams in improving on their work practices. Both methods have been applied in live development projects and found to enable teams to reflect on their practices and consider what gaps there are between people, between activities and between artefacts. By doing so, they can gain insight into how to improve on their coordination and alignment of requirements.

The main conclusion of this thesis is that development as a whole can be improved with an integrated requirements engineering (iRE) approach. Understanding and bridging gaps, or the level of integration between requirements and other development activities, helps development teams in achieving the necessary collaboration to be aligned within the development projects. This then enhances the efficiency and effectiveness of the development process by ensuring that *the right thing* is developed.

CONTENTS

PAPER IV: Distances between Requirements Engineering and Later Software Development Activities: A Systematic Map

PAPER V: Variations on the Evidence-Based Timeline Retrospective Method - A Comparison of Two Cases

PAPER VI: Gap Finder: Assessing and Improving the Integration of Requirements and Testing

PREFACE

This dissertation consists of two parts. The first part introduces the topic and outlines contributions and conclusions. The second part consists of papers supporting these claims.

List of Included Papers

I. Requirements are Slipping through the Gaps – A Case Study on Causes & Effects of Communication Gaps in Large-Scale Software Development.

E. Bjarnason, K. Wnuk and B. Regnell.
Proc of 19th IEEE International Requirements Engineering Conference, 2011, pp.37-46.

II. Are You Biting Off More Than You Can Chew? A Case Study on Causes and Effects of Overscoping in Large-Scale Software Engineering.

E. Bjarnason, K. Wnuk and B. Regnell.
Journal of *Information and Software Technology*, 54(10): 1107-1124, 2012.

III. Challenges and Practices in Aligning Requirements with Verification and Validation: A Case Study of Six Companies.

E. Bjarnason, P. Runeson, M. Borg, M. Unterkalmsteiner, E. Engström, B. Regnell, G. Sabaliauskaite, A. Loconsole, T. Gorschek, R. Feldt.
Journal of Empirical Software Engineering, July 2013.

IV. Distances between Requirements Engineering and Later Software Development Activities: A Systematic Map.

E. Bjarnason.
Proc. of 19th Int. Working Conf. Requirements Engineering: Foundation for Software Quality (REFSQ'13), pp. 292-307. 2013.

V. Variations on the Evidence-Based Timeline Retrospective Method. A Comparison of Two Cases.

E. Bjarnason, A. Hess, J. Doerr and B. Regnell.
Proceedings of 39th Euromicro Conf. Series on Software Engineering and Advanced Applications, 2013, pp. 37-44.

VI. Gap Finder: Assessing and Improving the Integration of Requirements and Testing.

E. Bjarnason, H. Sharp and B. Regnell

To be submitted.

Contribution Statement

I am the main author of all included papers and as such responsible for the research, dividing the work between co-authors and performing most of the writing. The ideas, design, data collection and data analysis for the included papers are largely mine, although I have enjoyed good collaboration with my co-authors who have contributed as follows.

The *ideas* of the different studies originate from many sources of which my co-authors have their share. For example, the study on requirements-test alignment (paper III) was originated by authors 2, 6, 9 and 10, and the idea to quantify distances with the Gap Finder (paper VI) originated from the third author.

The *design* of the research study for paper I (Communication Gaps) and paper II (Overscoping) was made in cooperation with the third author, with input from the second author. For paper III (Requirements-Test Alignment) the initial study design (overall, and up to the design and data collection phase) was performed by authors 2, 4 to 10, while I took an active part in the design of the data analysis together with all the other authors. Paper IV was design by me. The other studies (papers V and VI) were mainly designed by me with input from my co-authors.

I have carried out the main part of the *data collection* for papers I-II, and VI, however with support from other people, e.g. contacts within the studied organisations. For paper III (Requirements-Test Alignment) I performed 2 of the 30 interviews, while the rest were performed by other co-authors. The systematic mapping study (paper IV) was performed by me. For paper V (Evidence-Based Timeline Retrospective Method) the data collection for case 1 was performed by me with support from other researchers and company contacts, while the data collected for case 2 was performed by the second author.

Analyses of data in all the studies are primarily my analyses, however, validated by the co-authors. For paper III (Requirements-Test alignment) the analysis work was shared with the second author, and we each validated the analysis of the other. For paper V (Evidence-Based Timeline Retrospectives) the comparative analysis of the cases was done in collaboration with the second author.

Furthermore, I have authored four papers which are related to but not included in the thesis. These are listed below.

List of Related Papers

The following papers are related to this dissertation, but are not include:

Overscoping: Reasons and Consequences – A Case Study in Decision Making in Software Product Management.

E. Bjarnason, K. Wnuk and B. Regnell.

Proc. of 4th *Int. Workshop on Software Product Management (IWSPM)*, IEEE Computer Society, September 2010, pp. 30-39.

A Case Study on Benefits and Side-Effects of Agile Practices in Large-Scale Requirements Engineering.

E. Bjarnason, K. Wnuk and B. Regnell.

Proc.of 1st Workshop on Agile Requirements Engineering (AREW '11). ACM, 2011.

Evidence-Based Timelines for Agile Project Retrospectives – A Method Proposal

E. Bjarnason and B. Regnell, B.

Proc of *Agile Processes in Software Engineering and Extreme Programming* (XP'12), pp. 177-184. Springer Berlin Heidelberg. 2012

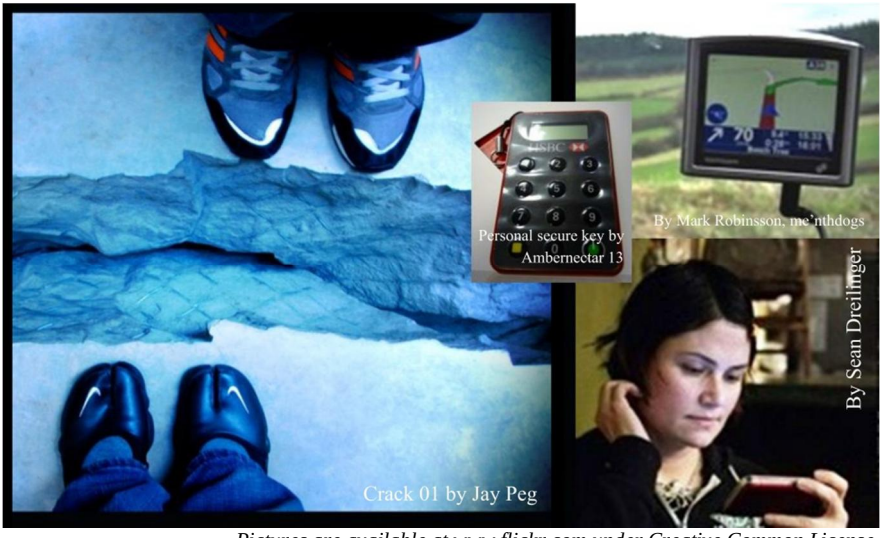
Evidence-Based Timelines for Project Retrospectives – A Method for Assessing Requirements Engineering in Context

E. Bjarnason, R. Berntsson Svensson, B. Regnell

Proc. of 2nd Int. Workshop on Empirical Requirements Engineering (EmpiRE), pp. 17-24, IEEE, 2012.

POPULAR SCIENCE SUMMARY (IN SWEDISH)

Se upp för gap i kravflödet!



Pictures are available at www.flickr.com under Creative Common License.

Av Elizabeth Bjarnason
Institutionen för datavetenskap
Lunds universitet

Vi använder mjukvara hela tiden och förväntar oss att den fungerar som vi vill. Att stavningshjälpen på mobiltelefonen väljer det ord vi tänkt oss, att vi kan be GPSen att undvika en viss väg där det pågår vägarbete, och att datorbanken hjälper oss att betala räkningar till rätt mottagare.

Mjukvaruprodukter blir smartare och mera kompetenta. En smarttelefon har idag lika mycket kapacitet som en PC för 10 år sen och innehåller nästan lika mycket funktionalitet. Samtidigt förväntar vi oss att den fungerar utan problem och hjälper oss i vår vardag.

Men, vilka av användarnas alla förväntningar och krav ska produkten stödja? Och hur ska de förmedlas till de hundratals ingenjörer som är inblandade i att ta fram mjukvaran? De olika delarna av en organisation som utvecklar mjukvara behöver koordineras och ha en samstämmig bild av vad produkten ska uppfylla för att kunna arbeta mot samma mål.

Missförstånd är dyra

Missförstånd kring vad produktkraven innebär ställer till problem och kan leda till att kunderna får en produkt med annat beteende än

vad de har förväntat sig. I bästa fall fångas dessa problem innan kunden får produkten men det krävs ändå extra arbete för att korrigera missförstånden.

Målet för vår forskning är förbättrad precision och effektivitet i mjukvaruutveckling genom att upptäcka och undvika kommunikationsgap och brister i samordningen av kravställningen mellan olika ingenjörer. Specifikt fokuserar denna avhandling på metoder som kan synliggöra potentiellts problematiska gap och därigenom hjälpa organisationer som utvecklare mjukvara att stärka samordningen mellan krav- och test-aktiviteter.

Komplex samordning

De mjukvarukrav som en ny produkt ska uppfylla mejslas gradvis fram genom att jämka samman kundernas förväntningar med affärsstrategier och planer, och väga dessa mot tekniska möjligheter och utvecklingskostnader. Det är en komplex process som fortlöper under hela utvecklingstiden och där många detaljer utformas av ingenjörerna själva. Att koordinera och samordna en kravbild som kontinuerligt förändras kräver god samarbets- och kommunikations-förmåga såväl som kompetens, rutiner och verktyg för att hantera och dokumentera informationen.

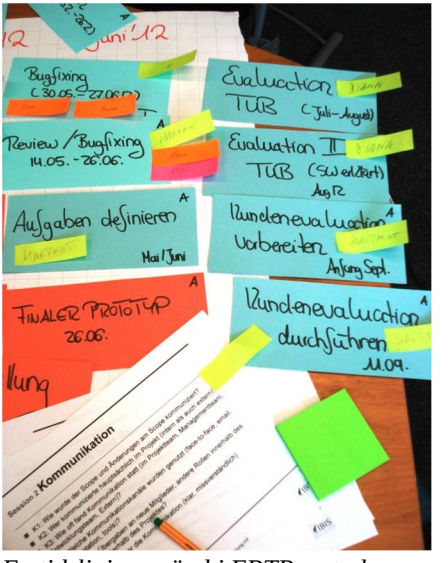
När samordningen mellan produktkraven och ingenjörerna brister kan detta leda till mjukvara som avviker från de överenskomna kraven och kanske också från kundernas förväntningar. Att korrigera dessa brister kräver förändringar i mjukvaran och kan leda till både förseningar och ökade utvecklingskostnader.

Om testingenjörerna också har missuppfattat kraven så finns det en risk att produkten innehåller avvikelserna när den når kunden. Produkten kan då behöva dras tillbaka från marknaden vilket medföra stora kostnader och kan även leda till minskat förtroendet för tillverkaren.

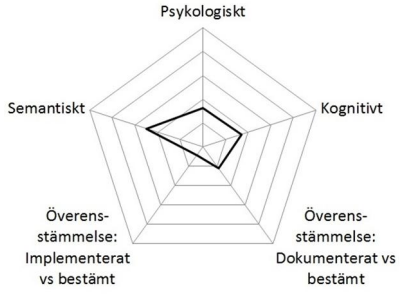
Synliggörande av gap

Det är dessa kommunikationsgap mellan kravställare och ingenjörer som är i fokus för denna avhandling. Målet är att kunna erbjuda metoder som synliggör gapen och därigenom hjälper utvecklingsprojekt att inse var flödet brister och hur gapen kan överbryggas.

Avhandlingen innehåller en metod för gruppreflektioner (*EBTR metoden*) omkring ett avklarat projekt vilket kan leda till nya insikter hos ingenjörerna om styrkor och brister i interaktionen mellan projektmedlemmarna. Visualisering av projekthändelser förser gruppen med en gemensam bild av projektet och stödjer en objektiv diskussion.



En tidslinje använd i EBTR metoden.



En visualisering av olika avstånd använd i Gapfinnar metoden.

Vi har dessutom utvecklat *Gapfinnaren*, en metod för att mäta olika typer av avstånd mellan krav- och test aktiviteter. Gap mellan olika ingenjörer och mellan olika dokument kan tyda på missförstånd i kravkommunikationen. Till exempel, stora skillnader (eller gap) mellan krav- och test-ingenjörens kunskaper om produkten och dess kunder innebär en stor risk för missförstånd omkring mjukvarukraven. Detta kan förbättras genom att införa nya rutiner som t ex att kravingenjören testar delar av mjukvarufunktionaliteten.

Insikt stärker samordningen

EBTR metoden för grupp-reflektioner och Gapfinnar metoden har båda applicerats på flera mjukvaru-utvecklingsprojekt och utvärderats genom en kombination av enkäter och samtalsgrupper. Visualiseringen av projekthändelser initierade relevanta och värdefulla gruppdiskussioner. Speciellt ledde detta till nya insikter hos testingenjörerna om projektets kravflöde. Gapfinnar metoden påvisade ett antal gap i kravflödet inom ett utvecklingsteam vilka då kunde adresseras och därigenom förbättra samordningen inom gruppen.

I grunden handlar det om att spara tid och pengar genom att utveckla och testa mjukvara enligt en gemensam förståelse för produktkraven. Mängden mjukvara kommer troligtvis att fortsätta att växa framöver. Samtidigt ställer vi som konsumenter allt högre krav på funktionalitet, och har mindre tolerans för dåligt fungerande mjukvara. Med ökad synlighet av avstånd mellan produktkrav och mjukvaruutvecklingen får ingenjörerna hjälp att identifiera och brygga gap i kommunikationsflödet. Samordningen inom organisationer som utvecklar mjukvara kan därmed förbättras. Genom att implementera enligt en korrekt förståelse för mjukvarukraven kan senare omarbetningar och förändringar undvikas, vilket leder till ökad produktivet.

ACKNOWLEDGEMENTS

Similar to the old African proverb *'It takes a whole village to raise a child'*, it takes a whole community to raise a researcher. I am very grateful for all that I have learnt both from my academic and my industrial colleagues both in Lund and internationally. Working with you and being asked constructive questions have made me think and think again and lead to something better than if I had done it on my own. Better solutions, better designs, clearer descriptions and presentations, and sharper conclusions.

Among my research colleagues, I in particular want to thank my supervisor, Professor Björn Regnell, for his (always) positive and encouraging attitude, support and introduction into the world of academic research. I also want to thank my assistant supervisor Professor Per Runeson for teaching me about case study research and for widening my perspectives on empirical research. I am indebted to the EASE Theme D team (co-authors of Paper III) for providing me with a good team environment in which I got my first practical introduction into performing interview studies and Krzysztof Wnuk who got me going on writing papers for this area (on an early version of Paper II). I want to thank the SERG group and the Department of Computer Science at Lund University for providing a stimulating and supportive research environment. A special *thank you* goes to Professor Helen Sharp at the Open University, UK for introducing me to research in the wild and for letting me loose on 'your' development organisation. It was fun and we all learnt a lot!

Over the years I have worked with very many different people in industry and I want to thank all of you, including those who have participated in our research studies. In particular I want to thank my colleague Yolanda Perdomo who (probably without realising this) encouraged me in believing that I could contribute to research with my experience and insights from software development. I also want to thank my managers Mats Pettersson and Susanne Engberg for agreeing to send me off on this research journey four years ago.

Finally, I owe a huge *THANK YOU* to my family and friends for their encouragement. The support of my family has allowed me to pursue this research; periodically working long hours, being away on conferences etc. In particular I want to thank you, Hannah, for joining me on my UK adventure (which resulted in the final paper VI of this thesis); Bjarni for keeping things afloat while I was gone; Jonathan for looking after my cats; and Michaela for doing more than your share of the cooking. You are great!

This research was funded by Stiftelsen för Strategisk Forskning (stratresearch.se), EASE (ease.cs.lth.se) and Ericsson Research.

INTRODUCTION

Requirements run like a red thread through software development, or at least they should in order to facilitate developing 'the right software for the customer' (Aurum 2005); software that meets the expectations and needs, or the requirements of the end users and other stakeholders. Communication including knowledge share between organisational units, roles and individuals is vital for enabling an efficient and timely development of competitive software, in particular for large software development companies (Curtis 1988, Kraut 1995, Karlsson 2007). *Gaps* in this communication flow can lead to a range of issues including delays, wasted effort, software quality issues (Curtis 1988) and ultimately dissatisfied customers. Companies operating in a market-driven domain face the added challenge of balancing high requirements volatility and uncertain cost estimates (Karlsson 2007) with releasing software within a critical market window (Sawyer 2000). Delays and increased lead times, e.g. caused by miscommunication, can lead to missing the optimal market window and have serious implications on both sales and brand value (Novorita 1996). Furthermore, the importance and challenge of requirements is continuously highlighted through reports such as the Standish CHAOS survey and research results (Boehm 1991, Tesch 2007, Kamata 2007) that point to requirements-related issues as the top risks and causes of problems in software development projects.

A closer integration of requirements engineering (RE) within the development process can mitigate some of these problems by supporting alignment and coordination of the requirements with later software development activities (see Figure 1) by reducing *gaps* in the communication flow between roles (Damian 2013, Stapel 2012). This is the approach taken in agile software development to address the challenges of a high rate of requirements change and the need for rapid software delivery (Sommerville 2005). In agile development, requirements are defined iteratively and in close cooperation within cross-functional teams (Ramesh 2010). This approach can support development efficiency and effectiveness (Dybå 2009), e.g. by avoiding the waste caused by developing and testing software based on unrealistic or unclear requirements. However, there are also challenges and risks with agile software development (Dybå 2009, Ramesh 2010) thus indicating that there are more factors involved than merely applying an agile or non-agile development model.

Even though the interaction and coordination of RE with other software development activities is vital the bulk of RE research focuses on methods and techniques within RE (Cheng 2007). Although improving the outcome of the RE process itself (e.g. the requirements specification) has benefits, it has little value if RE is not well connected and coordinated with the rest of development. More

attention is needed on the impact that RE has on the success of the development projects and the resulting products (Gorschek 2008).

The aim of this research is three fold: (A) to contribute with empirical insight into the coordination of requirements within development and (B) to provide support for improving the integration of requirements, in particular with testing. This research also contributes with (C) theory of factors influencing the integration and alignment of requirements and testing. The new knowledge and methods presented in this thesis can be used to detect potential *gaps* in development projects and identify work practices that may bridge these *gaps*. This optimisation of the RE integration can increase the alignment between RE and later development activities, thus improving the overall efficiency and effectiveness of a software development project. Ultimately, this can increase a company's ability to successfully develop and launch profitable products to the market.

The remainder of this chapter is organised as follows: Section 1 describes the background and related work of the research presented in this thesis. Section 2 presents our research focus including research questions, contributions and thesis outline. The applied research methodology is described in Section 3. While the details of the research contribution is presented in the second part of this thesis, one of the contributions is presented in Section 4 of this chapter, namely a theoretical framework. Finally, Section 5 contains a synthesis of the research results, while future research directions are outlined in Section 6 and main conclusions in Section 7.

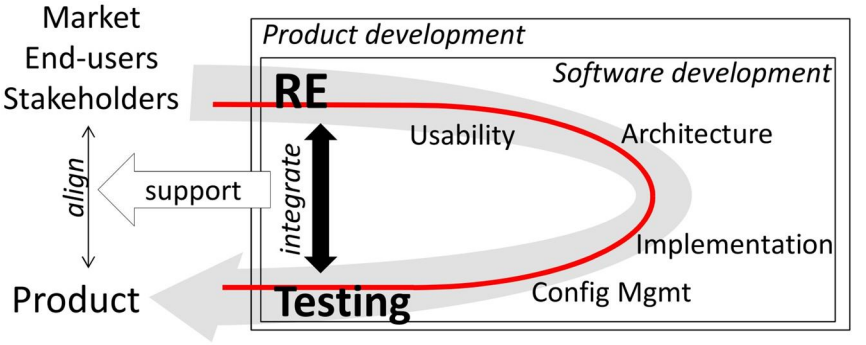


Figure 1. An overview of the problem domain and solution approach, i.e. a closer integration of requirements engineering (RE, the red thread) with other development activities, primarily testing. This can decrease *gaps* in the information flow between customer expectations and product behaviour and strengthen the alignment.

1 The Role of Requirements in Development

Requirements engineering (RE), and its wider context of software engineering, can be viewed from different angles. It can either be seen as a formal and structured transformation of information or as a collaborative effort relying on the creativity and competence of the involved engineers. Process engineering is an expression of the former, while agile development and research into coordination and communication within software development is an example of the latter.

1.1 RE in the Development Process

Traditionally the focus has been on practices, methods and techniques for enabling development through defining work flows, or processes, as a sequence of steps that transform information from an idea or a request through design, implementation and testing to a software solution. Viewed from this angle, process engineering is concerned with prescribing how to achieve a flow from input to output by defining roles, activities and artefacts that will produce the desired end result. Furthermore, from this perspective the various software engineering disciplines (e.g. RE, design, implementation and testing) tend to be defined as separate processes. The interaction between these processes is then defined by artefacts and prescribed activities involving interfacing roles.

Damian et al. (2005) found that a well-defined RE process can augment project planning and the stakeholders' ability to negotiate project scope, as well as, support increased developer productivity. RE, thus, has the ability to enhance the ability of the software development life cycle to produce software that matches the targeted market window. However, this requires the RE activities and roles to be well coordinated with the other software development activities (Curtis 1988, Damian 2006, Kraut 1995), e.g. architectural design, implementation and testing. In particular, RE plays a vital role in decision making concerning which behaviour to implement (Aurum 2003) and communication of these requirements to the relevant development roles (see overview of problem domain in Figure 1). For this reason, requirements that are not aligned with the technical design and with the amount of available resources are likely to cause problems and lead to delays, wasted effort and issues with the quality of the developed software (Damian 2006). Furthermore, frequent requirements changes can lead to similar problems (Curtis 1988, Boehm 1991).

Research into how RE may enhance the overall development process includes work on requirements communication (Damian 2013, Stapel 2009, 2011, 2012, Marczak 2008, 2011), the alignment of requirements and testing (Uusitalo 2008, Kukkanen 2009, Sabaliauskaute 2010) and taking an agile approach in software development (Layman 2006, Dybå 2009, Ramesh 2010).

1.2 Requirements Communication

Communication and coordination of requirements is a challenge (Flemming 1978) common to development in a range of different contexts including market-driven (Karlsson 2007), large-scale (Curtis 1988) and distributed (Damian 2001, Calefato

2007, Stapel 2009) development. Coordination between marketing and development roles within the market-driven domain entails a range of difficulties including lack of common views on the role and need of requirements details, common vocabulary, responsibility for requirements specification and analysis, dependencies on individuals, and suitable combination of sequential and iterative development (Karlsson 2007). For large and complex development these challenges increase and Kraut at al. (1995) argue that a combination of formal and information communication is required to be able to cope with uncertainties and changes. For distributed development where the informal communication channels are reduced most problems have been found to be related to communication, in particular missing context, awareness and missing document information (Stapel 2009). Awareness of one another's work is important since it affects the coordination, which in turn leads to information sharing and knowledge gain (Damian 2010). Lack of knowledge of on-going activities hinders a correct assessment of the impact of changes, and can lead to misunderstandings about requirements, as well as, reduced trust and productivity in a development team (Damian 2003).

Bridging the *communication gaps* between RE and other development roles and activities, in particular for distributed development, has been identified as an important area for future RE research by Cheng and Atlee (2007). The need for increased insight in this area is further highlighted by Marczak et al. (2011) who found that the communication structure in a development team did not adhere to the one prescribed by the organisation. Some approaches to provide such 'bridges' have been researched including use of computer-aided communication for requirements elicitation and negotiation (Calefato 2007, Damian 2001).

Increased awareness of communication paths has been suggested to improve the information transfer in a geographically distributed setting. Stapel et al. (2009) suggest having 'ambassadors' physically present at the different sites and to document fluid information. Marczak et al. (2008) found that the information flow to a large extent is controlled by a few key people, a.k.a. *information brokers*. They pose that extensive experience of an organisation and familiarity with its members may enable certain people to bridge communication gaps also on behalf of their fellow team members. Kwan et al. (2007) propose enhancing awareness by visualising inter-dependent requirements and the people working on them using a requirements-dependency diagram.

Matching the communication patterns with the technical dependencies between requirements and work items is an approach investigated by several researchers. Cataldo et al. (2008) evaluated a framework for assessing the socio-technical congruence and found that the resolution time for a modification request was reduced by a third (on average) when the developers' communication patterns were synchronised with the technical dependencies between the work items they were assigned to performed. A related approach to planning and managing information flows named FLOW Mapping is proposed by Stapel et al. (2009, 2011). FLOW Mapping entails capturing the information needs of a project, and based on these needs develop and implement a communication strategy covering both formal and information channels. The final step is to monitor and measure adherence to this communication strategy (Stapel 2011).

1.3 Agile RE: Concurrent and Integrated

Agile software development applies a concurrent approach by integrating the processes for requirements, design and implementation. Within concurrent engineering (Lawson 1994) product development is performed by concurrently carrying out the multiple engineering processes with extensive feedback and iteration between them (Sommerville 2005). Thus, the developers are to consider all aspects of the development cycle from requirements to cost and quality. The gains reported for concurrent engineering include increased efficiency, productivity and quality, and reduced waste and shortened lead times (Lawson 1994). Similar gains are claimed for agile software development including increased responsiveness to change (Sommerville 2005, Layman 2006).

For agile RE six industrial practices used in agile development and seven challenges connected to these have been identified by Ramesh et al. (2010). One of the agile RE practices is *prioritising face-to-face communication over written documentation*, which goes back to one of the basic agile principles (Beck 2001). However, weak customer and project-level communication within agile projects also leads to challenges with cost and project-level schedule estimations and customer participation (Ramesh 2010). In addition, these *gaps* in communication, in combination with the minimal amount of documentation produced in agile development projects, have been reported to cause problems with scaling and evolving the software and with including new project members (Ramesh 2010).

Another identified agile RE practice is *review meetings and acceptance tests* which are used to validate and verify the requirements (Ramesh 2010). In some organisations the acceptance tests are viewed and used as requirements, thereby fully integrating these two artefacts. The acceptance tests are used to determine whether or not the system is acceptable from the customers' perspective and used as the basis for customer discussions, thus reducing the risk of building the wrong system. However, the communication then occurs on a more technical level and may require more technical insight of the customer. Melnik et al. (2006) found that customers in partnership with software engineers could communicate and validate business requirements through executable acceptance tests, although there is an initial learning curve.

The approach of having an executable specification of the system by defining requirements as test cases is called behaviour-driven development, BDD (North 2006). The test cases are defined with a domain-specific language (DSL) containing terms from the business domain. The DSL provides the customers and developers with a common language that reduces ambiguities and misunderstandings. Solis and Wang (2011) reviewed the available BDD literature and a number of BDD toolkits and found that the area is still under development. In addition, they found that the toolkits available at that time were limited to only support the development phase and did not provide the possibility to add domain-specific concepts to the DSL.

1.4 RE and Test (RET) Alignment

Aligning, coordinating and avoiding *gaps* between RE and testing (RET) is a challenge for software development. This challenge relate to a wide range of issues including organization, process, people, tools, requirement changes, traceability and measurements (Sabaliauskaute 2010). Practices applied in industry to address these challenges include traceability and increased communication, e.g. by involving testers early in the project and in requirement reviews (Uusitalo 2008). Similarly, Marczak et al. (2011) found that in requirements-driven collaboration there is often close communication between requirements and testing roles; key roles which when absent cause disruptions within the development team.

Research into improving RET alignment has primarily focused on model-based testing, formal approaches and traceability (Barmi 2010), which all cause issues when implementing them in practice. Hasling et al. (2008) report on positive experiences, but also challenges, of linking the requirements process to the testing process by applying model-based testing to a UML use case model of the requirements. Similar mixed findings from case studies of applying model-driven engineering are reported by Mohagheghi and Dehlen (2008) based on a literature review. Traceability between requirements and test cases or source code aims to align changing stakeholder needs with on-going development (Jarke 1998). This is a practice seen to improve the quality of the system under development, support clearer documentation, increased system understanding and impact analysis (Lindvall 1996). Research into this field has primarily focused on tools and techniques for supporting traceability. One such technique is trace recovery, which aims at supporting traceability through automatic or semi-automatic identification of related entities through information retrieval algorithms on natural language (Huffman Hayes 2007, De Lucia 2007, Borg 2013). However, issues related to organisational and social context have been reported as challenges in implementing tracing in practice (Gotel 1994, Ramesh 2001). Apart from technical challenges connected to tool environments, reported impediments include lack of organisational commitment, weak insight into the cost-benefit balance, lack of processes for tracing, inadequate training and use of external staff (Ramesh 2001).

Improving the requirements process has been found to support improvements also of later development activities, such as testing. Damian and Chisan (2006) performed a longitudinal case study of a company undergoing requirements process improvements and found that this improved the efficiency of downstream processes including testing. The improved requirements process provided the engineers including testers with requirements details and dependencies early on and, thus supported them in making informed decisions. The decrease in amount of rework is partly attributed to this improvement. Furthermore, a significant change in requirements communication is reported leading to improved communication between functional teams within the organisation and reducing the need for seeking later information concerning clarification and reiterations of requirements.

RET alignment may be enhanced by concurrently improving the requirements and the testing processes. Kukkanen et al. (2009) report on overall improvements when integrating the requirements and testing processes with the aim of ensuring the information flow between these two processes throughout the development life cycle. This integration formed the basis for optimised usage of resources and led to increased visibility of project status, removed risk of double work and increased overall effectiveness of the project (Kukkanen 2009).

Paci and Bouquet (2012) proposed a contrasting approach where changes between the two areas are propagated through strict interfaces defined between the requirements and the testing processes (rather than integration). Model-based traceability is used to connect the two areas, which do not need insight into the other. Rather, the roles of each area adhere to their separate process and rely on them being orchestrated so that information is propagated to the other side as needed. The approach is proposed in particular for security testing and has yet to be evaluated in a real-life setting.

The REST-bench framework (Unterkalmsteiner 2013) takes a somewhat similar stance in assessing RET alignment by mapping the information flow between requirements and testing by using an artefact map. When applying the method on a one-year project at Ericsson AB, a number of misunderstandings between requirements and testing roles were uncovered and subsequently resolved at a joint workshop. At the workshop bottlenecks and sub optimisations in the RET interaction were also identified by analysing the artefact map for the project.

1.5 Software Process Improvement

Software process improvement (SPI) views ‘the software process as the set of tools, methods, and practices we use to produce a software product’ (Humphrey, 1989, p.3) and by improving on the processes the development can become more efficient, reliable and repeatable, and result in software of a higher quality. The traditional and most wide-spread SPI frameworks and methods such as CMMI (Chrissis 2008) and SPICE (ISO/IEC 2004-2011) base their improvement suggestions on a wide set of best practices. These frameworks have been characterised as being prescriptive, or top-down, (Pettersson 2008), since they start by comparing the overall picture of one company to the summarised known set of best practices irrespective of domain, size or other case-specific characteristics. In contrast an inductive, or bottom-up, SPI framework such as QIP (Basili 1985) and Lean Six Sigma (George 2002) start by considering the organisational situation and context of the specific organisation when identifying potential improvements.

Most SPI frameworks and methods share the same main steps of first evaluating the current process, and then identifying, implementing and evaluating suitable process improvements. However, there are a number of different techniques that can be used for assessing a process and identifying improvements. One of them is *retrospective reflection* of past events and experiences with the aim of identifying issues and improvements (Collier 1996, Deby 2006, Drury 2011). Within agile development, iteration retrospectives are a common practice and are strongly connected to the concept of self-governing teams (Drury 2011). The team

then discusses the past iteration and agrees on process improvements to apply for the next one, thus reflecting on a short period of time. In contrast reflecting on the whole period of a project's duration through project retrospectives (a.k.a. lessons learnt or project post-mortems) is more commonly applied within traditional development. These retrospectives can then range from general brainstorming sessions to structured meetings with prepared input concerning project events.

Other approaches to SPI include *analysis of information flows*, *process modelling* and *process simulation*. The technique of analysing information flows (Cataldo 2008, Stapel 2011) strives to identify bottle-necks and optimise the flow through a project based on the idea that efficient and effective development relies on transformation of information. This approach has been applied to requirements information, see Section 1.2. Similarly, *process modelling* (Yu 1994) can be used to construct a model of an existing or an improved process including its information flow. Such a model of an existing process can facilitate group communication and understanding of that process, and thereby support improving and managing it. Furthermore, modelling can enable the implementation of process guidance and steering in the tool environment, thereby enforcing the process prescribed by the model. In addition, a process model can be simulated (Kellner 1999) in order to investigate various modifications of the process.

2 Research Focus

The research within the described problem domain (see Figure 1) was addressed in three main parts. First, the aim was to (A) seek increased insight into the coordination of RE with later development activities, and then to (B) research SPI methods for improving RE coordination and integration. In order to provide a solid foundation for the research on new methods we (C) generated a theory for integrated RE (iRE). This theory was based on empirical data from part A and used as the basis for designing the Gap Finder method in part B. Thus, this iRE theory (generated in part C) connects the descriptive research performed in part A with the prescriptive research performed in part B, see Figure 2.

2.1 Research Questions

In order to further focus and define the scope of the research, research questions were defined for each of the three parts. The 7 main research questions addressed in this thesis are listed in Table 1. An overview of the relationships between these questions is depicted in Figure 3 and described in the following sections.

Table 1. Main research questions covered by this thesis and in which part.

Research Question (RQ)			Addressed in
Insight	**A1**	What causes the challenge of gaps in requirements communication and what effects can these gaps have?	Paper I
	A2	What causes the challenge of overscoping of requirements and what effects can this have?	Paper II
	A3	What are the challenges and practices in aligning requirements engineering and testing (RET)?	Paper III
Improvem.	**B1**	How can the evidence-based timeline retrospective (EBTR) method support new insights and learning concerning the interaction between requirements and other development activities?	Paper V
	B2	How can Gap Finder support project teams in improving RET alignment?	Paper VI
Theory	**C1**	What kind of distances between requirements engineering and later development activities are reported in peer-reviewed literature?	Paper IV
	C2	How are RET alignment practices related to RE distances and gaps?	Introduction

2.1.1 Part A: Gaining Insight into Practice

The first three research questions (RQ.A1-A3) were asked in order to gain insight into challenges and practices in the problem domain. The topics of *RQ.A1 Communication gaps* and *RQ.A2 Overscoping* were selected based on long experience of working with in industrial software projects, in combination with other research insight into RE research. Through RQ.A1 a range of causes and effects of communication gaps were identified. Connections were found to overscoping, i.e. RQ.A2. An additional outcome of RQ.A1 was an expressed need for supporting development projects in gaining new insights and learning concerning how requirements are communicated and coordinated within the project, i.e. RQ.B1. Furthermore, a need for investigating distances between people (individuals, roles, organisational units etc.) and between activities over time arose from the insights into communication gaps and thus led to RQ.C1.

RQ.A3 RET (Requirements Engineering and Testing) alignment was defined based on insight into the research area and on a literature review. In addition, the scope of this thesis was delimited by selecting testing as the 'later development activity' to mainly focus on. RQ.A3 was defined with the aim of gaining insight into challenges and practices in coordinating and aligning RE with testing. This research question led to identifying a potential connection of RET alignment to distances between RE and testing. This resulted in defining RQ.C1 *RE Distances* and RQ.C2 *How RET alignment practices are related to RE distances*. The empirical data collected to address RQ.A3 was also to investigate RQ.C2.

2.1.2 Part B: SPI Methods for Improving Practice

Two of the research questions focus on software process improvement (SPI) methods aimed at the coordination and integration of RE with other development activities. The first one, *RQ.B1 New insights through the EBTR (Evidence-Based Timeline Retrospective) method*, evolved from RQ.A1 Communication gaps and practitioners who expressed a need for supporting development projects in gaining insight and learning concerning how well requirements were managed and communicated throughout the development life cycle.

RQ.B2 Gap finder was defined to investigate if a SPI method based on the knowledge and theory gained from RQ.C2 *Gaps for RET* was applicable for identifying RE gaps and suitable improvement practices.

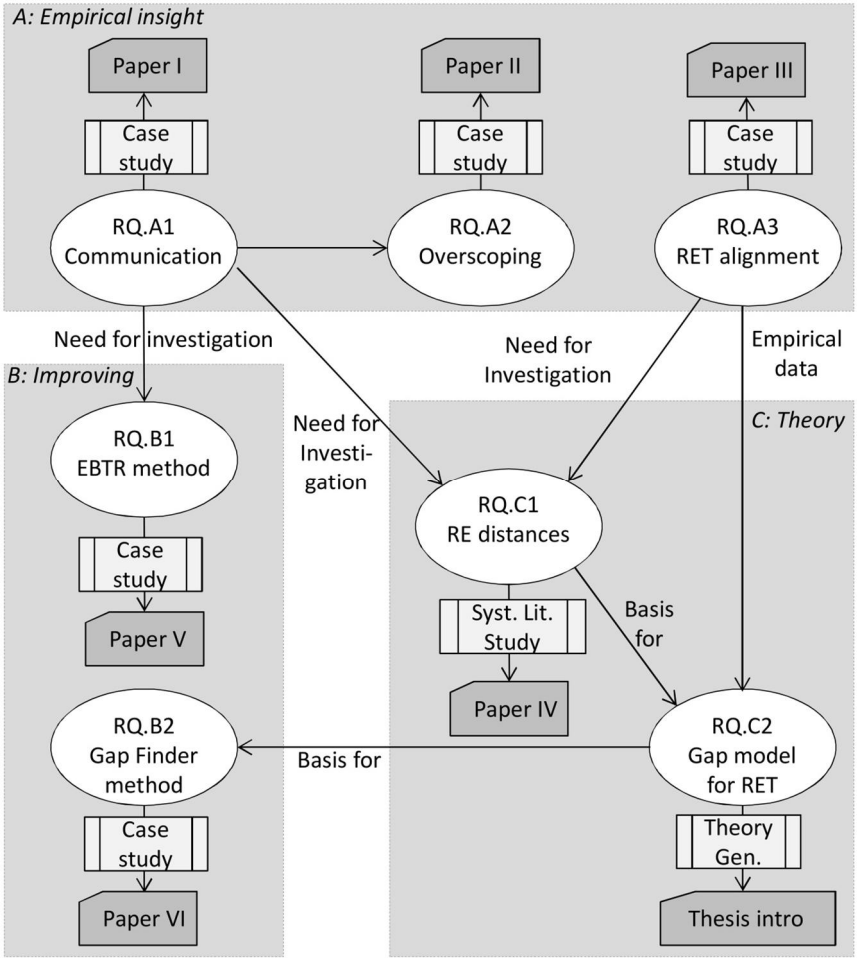


Figure 3. An overview of the main research questions (RQ) defined for each part (A-C), how they relate to each other and the main research method used to address each question.

2.1.3 Part C: Theory of Integrated RE (iRE)

The research questions defined for part C were aimed at identifying a theoretical basis for iRE including the phenomena explored in part A and SPI methods investigated in part B. *RQ.C1 RE Distances* was defined to investigate which distances between requirements and later development activities had been reported in peer-reviewed literature. This question was initiated based on insights and results gained from part A. In particular, the results from *RQ.A1 Communication* and *RQ.A3 RET Alignment* around gaps between people, artefacts and between activities over time. The results identified through RQ.C1 then posed an important foundation for generating further theory in response to *RQ.C2 Relationships between RET practices and RE distances* and resulted in the Gap Model (see Section 4). This research questions was posed to investigate generating a theory of gaps (or distances) that could explain the RET challenges and practices identified through RQ.A3. In addition to using the thirteen RE distances (identified through RQ.C1) as the basic framework of distances, RQ.C2 was investigated based on the empirical data collected for RQ.A3 RET alignment.

2.2 Outline of Thesis

This thesis is composed of two parts, namely this introduction chapter and a six research articles. This first part of the thesis describes the applied research methods (Section 3), the Gap Model (Section 4), the main results and contributions (Section 5), future research directions (Section 6) and conclusions (Section 7). The articles in the second part describe the details of all the performed studies and underpin the contributions and conclusions made in the first part. The exception is the study for RQ.C2, which is described in Section 4 of the first part of the thesis.

3 Research Methodology

This thesis includes five case studies (Papers I-III, V-VI), one literature study (Paper IV) and one theory-building study (see Section 4). Figure 3 contains an overview of the main research method used to address each research question. The overall research approach is discussed below followed by a description of the methods applied to each main category of study, i.e. case studies, systematic literature study and theory-generating study.

3.1 Research Approach

The overall aim has been to perform sound empirical research that can support our industrial partners in improving their software development processes. This is the reason for performing the majority of the work as case studies of industrial software development projects at our partners' sites. The industrial relevance of this research was thus strengthened both by validating the relevance of the research questions with our partners and by basing our findings on empirical data

gathered from industrial projects both during execution and after project completion.

The included studies are predominantly of a qualitative nature although some quantitative data have been collected, e.g. through surveys. This springs from the aim of gaining insight into factors affecting the coordination of requirements within development projects and organisations. Since development is a complex process both from a software engineering perspective and when considering human and psychological factors we believe that 'to truly understand software engineering, it is imperative to study people – software practitioners as they solve real software engineering problems in real environments' (Lethbridge 2005, p. 311). Thus, insight into the full picture requires studying real-life situations using a qualitative approach. Once factors have been identified and sufficient data points have been collected quantitative methods may be suitable for confirming these, however this remains as future work to consider.

Applying a flexible design approach (Robson 2002) in planning and performing the studies has been particularly suitable for the case studies of live development project. This approach has enabled us to adapt the study if and when it is affected by changes and delays in the development projects. In addition, this approach has allowed for iterating and gradually improving on the research design including the analysis process as new knowledge is gained.

3.2 Case Studies: Papers I-III, V-VI

Five case studies of development projects have been performed to gain insight (part A) into factors at play in software development and to evaluate the improvement methods (part B) by applying these in a real-life setting. For these studies all three of the criteria mentioned by Yin are relevant, i.e. (1) 'how' or 'why' research questions, (2) little control over events and (3) focus on contemporary phenomena in a real-life context (Yin 2009, abstract). For research questions RQ.A1-A2 the investigations focused on the 'how' of the challenges in the targeted area was caused by underlying factors. For RQ.A3, the 'how' concerned how the area of RET alignment was experienced and addressed in industry. For the case studies performed to evaluate the SPI methods EBTR (Evidence-Based Timeline Retrospective) and Gap Finder the questions were related to 'how' these methods can support project teams in improving on practice.

For both category RQ.A and RQ.B the phenomena and the SPI methods needed to be studied in a real-life development project and organisation to ensure that the findings are derived from a realistic case with the full complexity of factors involved in software development. Furthermore, by studying particular cases (projects and organisations) these results are of direct interest and readily applicable to those cases and thus to our industrial partners.

When studying a real-life project and development organisation (Yin's third criteria) it is a fact that the researcher has little control of events (Yin's second criteria). However, research can aim to be more of less intrusive and influential on the case under study. The case studies performed to address RQ.A1-3 were designed purely for gaining empirical insight, and not to influence or control

events during the investigations. In contrast, for the method evaluation studies (RQ.B1-2) the researchers controlled events to some degree by applying a new SPI method, as agreed with case representatives. However, the intention was not to influence the participants' reactions and responses to the methods but rather study these in order to obtain insight into strengths and weaknesses of the evaluated methods. Furthermore, since the intention of the SPI methods is to improve on practice there is (hopefully) a more long term effect on events for these studies. However, evaluating such effects of the SPI methods remains as future work and is not within the scope of this thesis.

The applicability of the results obtained through a case study needs to be considered for both internal and external generalizability (Robson 2002, p. 176-177). The internal generalizability, i.e. the extent to which the results are valid within the case context is affected by how representative the performed sampling is. For all the case studies in this thesis this was mitigated by performing a controlled convenience sampling of participants. Although availability was required, suitability was always the prime consideration in order to ensure a representative sample of the roles and experiences for the research questions at hand and thereby increase the internal generalizability of the results.

External generalizability has been considered primarily in connection with reporting of the results rather than in the study design. The findings and the characteristics of the case context have then been appraised to assess to which degree the results may be applicable also to other contexts (Runeson 2012, chapter 5). To further support this analytical generalization the context and characteristics of the cases have been reported together with the results. This enables the reader to evaluate the generalizability compare to other cases (Runeson 2012, chapter 5).

The purpose of the included case studies varied and was either explanatory or exploratory (Robson 2002, p. 59). The case studies on communication gaps (RQ.A1) and overscoping (RQ.A2) were explanatory and sought to explain these phenomena by identify causes and effects. In contrast the RET study (RQ.A3) and the evaluation studies (RQ.B1 and RQ.B2) were performed to explore and seek new insights. In addition, the evaluation studies sought to assess the new SPI methods in the case context rather than merely describing the application of them.

A mixed method approach was applied for all case studies and a range of different research methods were combined as was suitable for the research questions and case context at hand (Robson 2002, p. 370). These include interviews, focus groups, and document studies. This approach allowed us to study a phenomenon from multiple angles and enabled applying triangulation.

For each case study their main purpose and approach are further described below (Sections 3.2.1-3.2.3) including how the main risks to validity were mitigated in the research design. In addition, the various research methods and how they were used in the various case studies is discussed in Sections 3.2.4-3.2.8.

3.2.1 Explanatory Case Studies: Papers I and II

Two of the case studies, namely the ones for RQ.A1 *Communication Gaps* and RQ.A2 *Overscoping* had an explanatory purpose and were designed to explain these phenomena. The starting point of these studies was an assumption of the

factors at work derived from previous experience and insight into software development. The aim and main challenge of the studies was to investigate the phenomena using these assumptions while avoiding biases. Semi-structure interviews around questions defined in-line with the assumptions were performed. The risk of biasing the participants by imposing the assumption on them was mitigated by starting the interviews with open and un-biased questions before asking about the pre-assumed factors. In addition, in the reported results a separation was made between answers to the open questions and those given in response to the more specific questions. The risk of researcher bias was mitigated by involving multiple researchers in designing the study and validating interview transcripts. Furthermore, the outcome of the analysis was validated with practitioners through a survey.

3.2.2 Exploratory Case Study: Paper III

The exploratory case study for RQ.A3 RET alignment was performed as a semi-structured interview study where the questions were based on the researchers' insight into the area and on a literature review. The size of this case study both concerning the number of interviews (30) and companies (6), and the number of researchers involved (10) posed a challenge in coordinating and ensuring consistency between the different parts of the study and in managing the large amounts of qualitative data. The risks included dispersed research objectives and unintended variability in the data collection and data analysis (Runeson 2012, section 7.6). This challenge was partly mitigated by applying a rigorous case study process (Runeson 2012, section 2.6) and by having regular meetings to coordinate and synchronise the activities. Furthermore, a chain of evidence was maintained through traces between the interview data (transcripts), the intermediate analysis material, and the reported results. These traces thus made it feasible for other researchers to validate the results and intermediate analysis steps against the interview transcripts, thereby strengthening the reliability of the results (Yin 2009, chapter 4, pp. 122-124). In addition, the output of each step in the process from research design to reporting was reviewed and validated by more than one researcher, thus applying triangulation throughout the research process.

3.2.3 Exploratory Case Studies for Evaluation: Papers V and VI

The main purpose of the two case studies for part B was to explore and evaluate the proposed SPI methods, i.e. EBTR (Paper V) and Gap Finder (Paper VI), by applying them to live development projects. Both of the evaluated methods contain two parts, namely a) exploring and assessing the project under study without influencing events and b) interacting with the development team by presenting and discussing these findings with them with the intention of supporting, and thus influencing them to learn and improve on practice. The researcher influence for part a) was minimised in different ways for these two studies.

For the case study on the EBTR method, the project was explored (part a) mainly through document studies post fact and resulted in a representation of project history in the form of a visualised timeline, a.k.a. *evidence-based timeline*

(EBT). The risk of researcher bias or misunderstandings was then mitigated by reviewing this timeline with a company representative and during part b) with the involved practitioners.

For the other case study, i.e. the evaluation of the Gap Finder (RQ.B2), the exploration of the project (part a) was done through observations and surveys and took place in parallel to project execution. There was thus a greater risk of researcher influence for this case. This risk was mitigated by choosing an ethnographically-informed approach (Robinson 2007) for the observations, i.e. the team was observed without interfering or actively interacting with team members concerning project issues, e.g. during team meetings or discussions. The ethnographical approach also mitigated the risk of misunderstanding or misinterpreting the data collected through interviews and surveys. Ethnographically-informed observations are further discussed in Section 3.2.7.

3.2.4 Interviews

Interviews have been used to collect data for all of the included case studies except for the EBTR study (RQ.B1). All interviews have been semi structured in order to focus the discussions while still keeping them open. For each interview study an interview instrument with a structured set of questions was designed prior to the interviews. During the interviews this instrument acted as a guide and both the interviewer and the interviewee were free to ask for clarification and follow-on questions. In this way it has been possible to perform wide and deep enquiries into the rich reality of software development for the specific topics of interest. For example, for the explanatory case studies of communication gaps (RQ.A1) and overscoping (RQ.A2) the interview instrument was structured around the phenomena under study and the factors believed to be involved. However, the leading questions for each topic were consciously design to be open and thus encourage a free exchange of experiences and viewpoints. This approach mitigated the risk of biasing the participants and led to identifying factors not previously considered by the researchers.

For RQ.A1-A3 the main aim was to obtain a rich and wide picture of the area under study, which is why interviews were used as the main data collection method for these case studies. For the study on evaluating the Gap Finder (RQ.B2) interviews were used for a slightly different purpose, namely to support the data collection through surveys. The surveys were then administered as semi-structured interview using the survey questions as the interview instrument.

In order to mitigate the risk of researcher bias and misunderstanding what the interviewee meant all interviews were audio recorded and triangulation applied in the subsequent steps of the research process. For each interview either the transcript was reviewed by another researcher (done for RQ.A1-A3) or the practitioners validated the findings through a follow-up survey (as for RQ.A1-A2) or through a focus group (as for RQ.B2).

The level of detail in the transcripts has varied over the studies. For the interview study performed for RQ.A3 where 30 interviews were performed by 10 researchers, the interviews were transcribed word-by-word in order to minimise the risk of introducing variations due to the many researchers involved. For the

other studies, one main researcher was involved in the data collection and analysis, and the amount of interview data was more manageable. For these reasons, a slightly higher level of detail was captured in the interview transcripts. For example, stuttering and repetitions were omitted and rephrased to capture the meaning rather than each word. Furthermore, timestamps were interspersed in the transcripts to facilitate going back to specific parts of the audio recordings if necessary.

3.2.5 Surveys

As insight into the area under study increased so did the need to focus the investigations. We found surveys to be a good way of achieving this. Surveys have been used in two ways, namely to validate findings within a case study (as for RQ.A1-A2) and to study specific factors identified through previous research (as for RQ.B1-B2). An example of the first application is the study on overscoping (Paper II) where a set of factors were identified through interviews and then validated through a survey (or questionnaire) on the relevance and impact of these factors. An example of the second use of surveys is the Gap Finder study (Paper VI) where surveys acted as a starting point and were designed based on previous empirical knowledge represented by the theoretical framework of the Gap Model (see Section 4). The outcome of these surveys then provided data that was used both to understand the case and as part of the method which was being evaluated.

For the case study on the EBTR method (RQ.B1) self-administered surveys were used as a complement to focus groups for collecting data on participants' experience of the EBTR method. These surveys enabled us to collect more specific feedback and offered participants a way to provide individual feedback. This mitigated the risk of participants not freely expressing their viewpoints at the focus groups (Robson 2002, p. 234), which in this case included their peers and managers.

The main threats to validity for surveys are that the respondents do not understand and answers the questions in a uniform way and that the answers do not provide any depth (Robson 2002, p. 233), i.e. explanations or reasons for the given opinion. The risk of misunderstood questions was mitigated by carefully designing the survey questions based on previous empirical insight either from our own studies or from related work. In addition, the survey questions were reviewed and refined in collaboration with other researchers before being used. Furthermore, a researcher with good insight into the specific case was always involved in designing the survey questions to ensure that the questions were in-line with case terminology.

Combining surveys with interviews (as described in Section 3.2.4) for several of the studies further mitigated the risk of survey questions being misinterpreted. This approach also allowed the researchers to ask for clarification of answers when these were not readily understood. In conclusion, the surveys allowed us to focus the enquiries, while combining them with interviews provided a safety net in which unclear questions and answers could be caught and resolved.

Finally, due to the small numbers of respondents to each survey it has not been possible to perform statistical hypothesis-testing. Instead a qualitative approach has been taken in analysing descriptive statistics of the responses.

3.2.6 Focus Groups

The SPI methods EBTR (Paper V) and Gap Finder (Paper VI) both include focus group sessions as the main method to stimulate and support project teams in learning and gaining insight into process improvements through group reflection. In addition, focus groups were used for gauging the practitioners experience and viewpoints concerning these SPI methods in a light-weight yet in-depth way.

The focus group sessions were led by a moderator in a similar way to semi-structured interviews, i.e. predefined questions were used as a guiding framework in leading the discussions. All participants were allowed to freely express their opinions within the scope of the targeted topics for the session and the researchers could ask follow-up questions. For example, to clarify a point or try to ascertain what caused a mentioned event. Furthermore, an additional researcher was also present to support the moderator in ensuring that the discussions kept to the topics and that all topics were covered.

The main challenges of the focus group sessions in our case studies were to encourage all participants to share openly and with an equal airtime, and to ensure sufficient moderator competence and insight into the case. For both of the studies, the focus group participants were existing project teams and thus had a previous good working relationship. This entails openness when meeting together. However, in both cases there were participants who attended together with their managers, which may have limited their openness. For all sessions the moderators actively strived to encourage all participants to share, although there were sessions were certain participants were less active. One way to mitigate this was applied in the EBTR evaluation where the participants were given reflection time for each question and later took it in turns to share their reflections with the group.

The risk of the moderator and other researchers misunderstanding the participants was mitigated by ensuring that the moderators of the focus group sessions had good insight into both the topic area and the case context. In addition, a summary was written for each focus group session. This summary was reviewed by the involved researchers and then distributed to the participants to validate that it reflected their view of what had been discussed and concluded at the meeting.

3.2.7 Ethnographically-Informed Observations

An ethnographically-informed approach was applied during the first part of the Gap Finder study (Paper VI) and in particularly in the observations of the development team. The purpose of these observations was to gain a rich insight into the day-to-day work practices of the team members and their interactions with each other. The ethnographical approach entailed seeking to understand the team's work practices apart from the researcher's assumptions about software development (Robinson 2007). Although the long term aim of the Gap Finder evaluation was to improve on practice, the ethnographically-informed approach meant that the influence and interference with the development team was kept to a minimum during the four weeks that the first parts of the Gap Finder method was applied and during which the observations took place. For example, the researcher

did not contribute to any group discussions or meetings during this time period, merely observed and took notes.

Since the researcher who undertook the observations for the Gap Finder study had just recently been introduced to the case there was a risk of misinterpreting the observed interactions. This risk was mitigated by obtaining some initial insight through an interview and document studies of process descriptions and of development artefacts. Furthermore, an initial observation period of three days was undertaken a few weeks before the main observations began. This warm-up period also allowed the team members to become familiar with the researcher and mitigated the risk of them feeling uncomfortable and not acting as normal when being observed.

3.2.8 Document Study

In the evaluation studies of the EBTR method (Paper V) and the Gap Finder (VI) document studies were used as an unobtrusive and objective way (Yin 2009, Chapter 4) to gain insight into the cases. For the EBTR study the document studies were also a key component in preparing the evidence-based timelines that were later used as the basis for the focus group sessions with the project teams.

There were two main risks associated with these investigations, namely the risk of studying documents that were not representative of the case and the risk of bias both in the information and in the interpretation of it. For both case studies these risks were partly mitigated by initially obtaining an overall picture of the process including the used artefacts. Furthermore, the risk of studying a skewed and non-representative sample was mitigated by asking key project members for pointers to relevant artefacts. In order to avoid restricting the data source, e.g. by limiting the studies to artefacts directly provided by practitioners, the researchers had direct access to all of the relevant repositories.

For both studies the risk of bias was mitigated through triangulation. For the EBTR study, the timelines based on the document studies were reviewed by a case representative and further validated by the project teams at the focus group meetings. The majority of the information covered by the timelines was found to be correct, although details not found in the artefacts were missing, e.g. which testers had been involved throughout the project. Similarly for the Gap Finder study, the outcome of the document studies, i.e. the design of the artefact survey, was reviewed by an additional researcher with insight into the case.

3.3 Systematic Literature Study: Paper IV

A systematic literature study was the main method chosen to investigate research question RQ.C1 (Paper IV). However, each of the other studies has also included studying related literature either as a starting point for designing the study or as a validation step to compare the outcome against existing knowledge. The difference lies in that for those studies a less systematic process was applied in searching and analysing related literature than for Paper IV.

For RQ.C1 we were interested in investigating the concept of distance between RE and later development activities, in particular which distances that had been researched and for what context. The specific method chosen was a systematic

mapping study that can be used to classify and structure an area and assess the coverage and, thus gaps in the current research (Petersen 2008). Relevant literature was searched for by a defined and systematic process and the literature found to be relevant was categorised. Furthermore, for each distance types the literature was reviewed and a synthesis of the research performed so far was produced.

In comparison, the related method systematic literature review (SLR) aims to provide a summary of a topic by 'evaluating and interpreting' all available relevant research (Brereton 2007). We interpret this to mean that an SLR focuses on synthesising the findings of the literature, while a mapping study primarily focuses on providing an overview and a map of the area.

Since this mapping study was performed by one single person there is a risk of researcher bias in all steps of the process, i.e. in the selection of the included literature and in the analysis. This risk was mitigated to some degree by adhering to a predefined process (Brereton 2007). In addition, the full list of identified literature including the categorisation was published thereby allowing other researcher to validate the map.

3.4 Theory Generation Study: Gap Model

Theory has been generated mainly in the RET alignment study (Paper III) and in the Gap Model study (Section 4 of Introduction), although the Gap Model is the only pure theory-generation study. For both of the studies a theoretical framework was constructed by analysing empirical data against existing theoretical knowledge, thereby extending or modifying the theory by constant comparison to the data (Seaman 1999). Connection between factors and phenomena were identified and generalised into theory by coding the data, identifying patterns in and between these codes, and finally comparing and incorporating these patterns into the evolving theory.

For the RET alignment study the resulting framework consists of RET challenges and practices, and connections between them. Together with the set of RE distances (Paper IV), this RET framework provided the starting point when generating the Gap Model. In particular, the distances, challenges and practices provided the initial set of codes used in the analysis of the interview data. This set was gradually modified and refined and finally consisted of the RE distances and RET practices found to be relevant to RET alignment for the analysed data. The Gap Model was then constructed from these codes and the relationships between them.

The 'appropriately developed theory is the level at which the generalization of the case results will occur' (Yin 2009, Chapter 2). Thus, the external validity of a generated theory can be assessed in the same way as external generalizability for a case study, i.e. by applying analytical generalisation. Since the resulting theory is grounded in the analysed data and the initial theory or hypothesis used to generate it, the generalizability of these need to be considered. The theory can be strengthened by iterating the analysis and gradually extending the base of empirical knowledge on which the theory rests thereby building up a weight of evidence for the theory (Seaman 1999). One way of achieving this is to perform

cross-case analysis (Seaman 1999) which was applied for the RET study data where six different cases were analysed. Since the case companies all operated in the embedded domain with a range of different process and development models, it is feasible to assume that the resulting theory can be applicable to software development within the embedded domain.

Although data from only one company was used to generate the Gap Model, the theoretical frameworks that acted as the starting point for that analysis was based on a wider base, i.e. a systematic literature study and the six-case study of RET alignment. In addition, the results from the Gap Finder study (which was based on the Gap Model) indicate that the theory is generalizable beyond the one case for which data has been analysed so far. Namely, that is may be applicable to small and medium sized software development projects in a co-located setting and applying an iterative or agile development model. However, the systematic analysis required for generating sound theory needs to be performed before the Gap Model can be extended and generalised in this way.

The main threats to internal validity for the RET framework and the Gap Model are the risk of researcher bias and the risk of failing to systematically and rigorously manage the large amounts of empirical data. As pointed out by Seamann (1999, p. 567) since the coding and pattern matching is largely a creative process, there is a risk that the researcher is tempted to rely on hunches and start writing too early, rather than systematically analyse the data and the codes. Furthermore, there is a risk that the prejudice and pre-assumptions influences the analysis process. To mitigate this in the RET study, two researchers were involved in the theory generation and reviewed each other's analysis including coding and identification of patterns. For the Gap Model, this remains an open threat to be addressed in future work, although as previously mentioned the model was preliminary validated through the Gap Finder study.

4 The Gap Model: A Theory for iRE

The Gap Model is a theoretical framework for iRE (integrated RE) that describes how iRE, in the form of RE distances (Paper IV), relates to RET alignment. The model was constructed to answer the question *How are RET alignment practices related to RE distances?* (RQ7) For each RET practice, the model contains information of which RE distances that can be affected by applying the practice.

The model was derived by exploring the assumption that alignment challenges occur due to RE distances and that alignment practices found to address these challenges do so by affecting those distances. The method of constant comparison (Seaman 1999) was applied and the aforementioned assumption was compared against empirical data on RET alignment from the study reported in Paper III. Furthermore, this data was compared against the theoretical frameworks of RET challenges and practices (see Paper III) and of RE distances (see Paper IV).

The rest of this section describes the company from which the underlying empirical data was collected (see Section 4.1), the research method used to construct the model (see Section 4.2), the actual Gap Model (see Section 4.3) and the limitations of this model including future work (see Section 4.4).

4.1 The Case Company

The initial version of the model is based on the interviews from one of the six companies involved in the RET alignment study (see Paper III), namely Company A. This company develops computer networking equipment consisting of both hardware and software. The software development unit (which is the part of the company covered by the interview study) has around 150 employees and applies an iterative development model. A typical software project has a lead time of 6-18 months, around 10 co-located members and approximately 100 requirements and 1,000 test cases. A market-driven requirement engineering process is applied. The quality focus for the software is availability, performance and security. Furthermore, the company applies a product-line approach and uses open-source software in their development.

Three people were interviewed at Company A. This included a test engineer and a product manager who had both worked at the company for more than three years. In addition, a project manager was interviewed.

4.2 Research Method

The Gap Model was constructed by an iterative analysis process through which the set of RE distances and RET alignment practices included in the model were gradually refined, see Figure 4. A step-wise approach was applied and two instances of an analysis step were performed on a gradually extended set of transcript chunks and with multiple levels of analysis within each step. Both the preliminary and the main analysis step was performed using the same analysis process (see Section 4.2.1) and resulted in a refined set of RE distances and RET practices.

The preliminary analysis step was performed on the parts of the interview transcripts that relate to the quality of requirements. This sub-set was selected in order to provide a starting point and a focus for the initial analysis. The coding performed during the previous RET alignment study includes full traceability from results to transcripts. This traceability was used to identify the parts of the interview data relevant to the selected starting point, namely the RET challenge

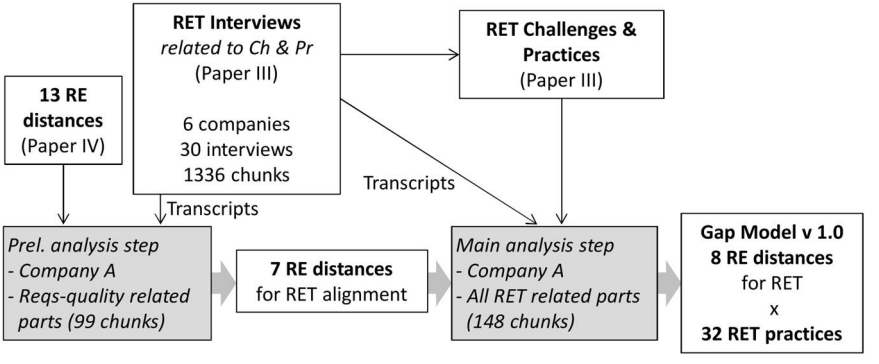


Figure 4. Overview of the research method used to generate the Gap Model.

Ch3 *Quality of requirements specification* and the practices connected to this (see Paper III). In addition, the chunks in immediate connection to the coded ones were included in the analysis if needed in order to understand the intention of the interviewee.

The thirteen RE distance types derived through a systematic literature study (Paper IV) provided the initial set of distances for this analysis step. Through the preliminary analysis an initial set of RE distances specifically related to RET alignment was identified. This set consisted of seven RE distances: abstraction, adherence, cognitive, geographical, navigational, organisational and semantic. This set was used as input to the main analysis step.

The main analysis step was performed on the full set of transcript chunks relevant to all of the identified RET challenges and practices. Through the applied analysis process (see Section 4.2.1) the set of RE distances identified through the preliminary analysis was refined and a set of RET practices was derived from the interview data. Furthermore, connections between these RET practices and RE distances were identified. The Gap Model is the output of this final analysis step, which consists of eight RE distances and seven categories of RET practices, in total 32 practices.

4.2.1 The Analysis Process

The same multi-level analysis process was applied to both the preliminary and the main analysis step. Level 1 consisted of *Coding of the transcripts* and was followed by level 2 A*bstraction and grouping* of these codes to identify relationships between them. In addition, the main analysis step included level 3 *Validation* to ensure consistency of the model. An overview of the analysis process is shown in Figure 5. A number of distances, challenges and practices, and chunks of interview transcript was provided as input for each analysis step (see the previous section for the specific input to each analysis step).

Coding The analysis of the interview transcripts was focused on identifying RE distances affecting or being affected by RET alignment challenges and practices. When such information was identified in the transcripts these parts were coded. The initial codes were provided as input to the analysis step and consisted of RE distances and RET alignment challenges and practices. This set was extended and modified during the analysis as additional distances and practices were identified in the transcripts. For example, in the preliminary analysis step the difference in effort to navigate between related parts of artefacts was mentioned by an interviewee. Based on this a new code and thus a new RE distance type called navigational distance was defined to cover this concept.

Abstraction and Grouping In the second level of analysis relationships between the codes for distances, challenges and practices were identified and abstracted based on the transcripts. These relationships were modelled and visualised in a bi-directional graph, similar to the approach applied in the RET alignment study (see Paper III, Section 3.4). Through analysis of this representation of the interview data RE distances relevant for RET alignment were identified. For the initial

analysis step the main outcome was this refined set of RE distances. For the main analysis step, the output also consisted of RET practices. Furthermore, for each practice the final output includes which RE distance the practice addresses and how, i.e. can the practice bridge, increase or decrease the distance.

Validation was performed for the main analysis step to mitigate the risk of inconsistencies in the bi-directional graph, and in relation to previously published results on RET alignment (Paper III). The graph consists of RE distances, RET practices and RET challenges. In addition, the graph contains information concerning which RE distances that contribute to a RET challenges, which RE distances that are affected by a RET practice and which RET practice addresses each RET challenge. Thus, the graph contains a triangular relationship between challenges, practices and distances (see Figure 5). In addition, relationships between RET practices and RET challenges were identified in the previous RET study. All of these relationships were reviewed to ensure internal consistency between them and to increase the reliability of the resulting Gap Model. This validation was done by comparing the challenge-practice connections in the graph with the ones identified through the RET study. In addition, for each practice addressing a certain challenge, the set of distances contributing to this challenge was reviewed against the set of distances affected by the practices. Ideally these two sets should be the same, although it is possible that additional factors not covered by the identified practices also contribute to the challenge. Whenever discrepancies were identified in the graph the related parts of the transcripts were re-analysed to resolve the inconsistencies by updating the graph accordingly.

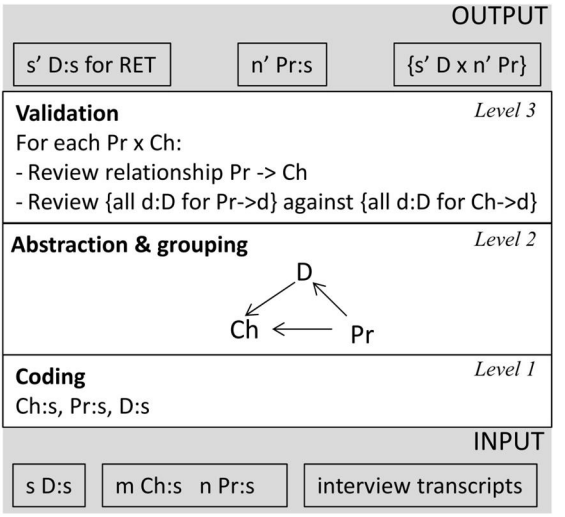


Figure 5. Overview of the process applied for each analysis step in order to construct the Gap Model.

4.3 Gap Model: RET Practices x RE Distances

This initial version of Gap Model consists of eight RE distances and seven categories of RET practices. The RE distances are D1 *Geographical*, D2 *Organisational*, D3 *Psychological* and D4 *Cognitive* distances between people; D5 *Adherence* distances to artefacts, D6 *Semantic* and D7 *Navigational* distances between artefacts; and D8 *Temporal* distance between activities. The RET practices are an extended sub-set of the ones reported in Paper III. This set for the Gap Model consists of the following categories (numbering kept consistent with the one used in Paper III): P1 *RE practices*, P2 *Validation practices*, P3 *Verification practices*, P4 *Change management practices*, P6 *Tracing practices*, P8 *Tool practices* and the newly derived category P11 *Development process*. Four categories of practices from the original list are not included in the Gap Model due to not being mentioned in the interviews for Company A. These non-included categories are P5 *Process enforcement*, P7 *Traceability responsibility roles*, P9 *Alignment metrics* and P10 *Job rotation*. Furthermore, 17 additional practices were identified through the analysis performed to construct the Gap Model. These are marked with a+ in the full list of included RET practices provided in Table 2.

Each of the included RET practices support alignment by addressing one or more RE distances. Each practice either changes a distance (decreases or increases it) or bridges it, i.e. does not alter the distance but reduces its negative effect. For example, organisational distance between requirements engineers and testers can be bridged with P1.1 *Cross-role requirements reviews* by bringing these roles together in a common meeting and around a common task without changing the organisational distance between these roles. This practice (P1.1) can also decrease adherence distance between the agreed and the documented requirements by identifying and resolving this distance through the review meeting. There are two practices that increase organisational distance by introducing additional organisational units, namely P3.2 *Independent testing* and P3.6 *Separate testing team for quality requirements*. An overview of the Gap Model and connections between RET practices and RE distances is shown in Table 3.

To further illustrate the knowledge captured by the Gap Model seven of the included RET practices and their impact on RE distances will now be discussed in more detail. This set of example practices have been selected to illustrate both a simple and a more complex impact on distance. Furthermore, the practices were selected to obtain examples that cover all of the RE distance types and all but one of the practice categories.

Table 2. Overview of RET practices included in the Gap Model. Practices marked with + are added compared to those reported in Paper III. Practices marked in bold are described further in the text.

P1 RE Practices	P1.1 Customer communication at all requirements levels and phases
	P1.2 Development involved in detailing requirements
	P1.3 Cross-role requirements review
	+P1.7 Use of a customer proxy role
	+P1.8 Feature requirements documentation
	+P1.9 Product manager physically present to developers & testers
	+P1.10 Informal communication within organisation
	+P1.11 Product manager involved in development project
	+P1.12 Same process for QRs
	+P1.13 Structure requirements artefacts according to type
	+P1.14 Upfront definition of quality requirements
P2 Validation Practices	P2.1 Test cases reviewed against requirements
	P2.3 Product manager reviews prototype
	P2.5 User / Customer testing
	+P2.6 Early test involvement in development projects
P3 Verification Practices	**P3.2 Independent testing**
	P3.3 Testers re-use customer feedback
	+P3.5 Feature-based test plan
	+P3.6 Separate testing team for quality requirements
	+P3.7 Test-impact analysis
	+P3.8 Close cooperation between Test and Development unit and roles
P4 Change	**P4.1 Process for requirements changes involving Test**
	P4.2 Product-line requirements practices
P5 Tracing Practices	P6.1 Document-level traces
	P6.2 Requirements-test case traces
	P6.3 Test cases as requirements
	+P6.5 Conceptual tracing
	+P6.6 Traces between people/roles
P8 Tool Practices	P8.1 Tool support for requirements and testing
	P8.2 Tool support for requirements-test case tracing
+P11 Process	+P11.1 Incremental development
	+P11.2 Small-scale development

Table 3. Overview of Gap Model: how RET practices (see Table 2) can affect each RE distance, i.e. B=Bridge, D=Decrease or I=Increase. The impact of practices marked in bold are further described in the text.

	P1 RE	P2 Validation	P3 Verification	P4 Change	P6 Tracing	P8 Tools	P11 Process & Size
D1 Geographical	D: **1.9**						
D2 Organisational	B: 1.2, 1.3, **1.9**-1.11, **1.14**	B: 2.1, 2.3	I: **3.2**, 3.6	B: **4.1**	B: **6.2**, 6.6		D: **11.2**
D3 Psychological	D: 1.10						D: **11.2**
D4 Cognitive	BD: 1.1-1.3, **1.9**, 1.11 B: 1.7, 1.8, **1.14**	B: 2.1, 2.3, **2.5** BD: 2.6	BD: 3.6 B: **3.2**, 3.3	B: **4.1**			BD: 11.1, **11.2**
D5 Adherence	D: 1.1-1.3, 1.8, 1.10-1.11	D: 2.1, 2.3, **2.5**		D**: 4.1**	D: **6.2**, 6.5 IB: 6.3	D: 8.2	D: 11.1
D6 Semantic	D: 1.1, 1.2, 1.8	D: 2.1, 2.3, 2.6	B: 3.6 D: **3.2**, 3.5, 3.7-3.8	D: 4.2	D: 6.1-**6.2**, 6.5	D: 8.2	D: 11.1
D7 Navigational			D: 3.5		D: **6.2**-6.3	D: 8.1, 8.2	
D8 Temporal		D: 2.6		B: **4.1**			D: 11.1

4.3.1 Product Manager Physically Present to Developers and Testers (P1.9)

This practice relates to the physical location of the product manager relative the office space where other roles of a development project have their desks, in particular the developers and the testers. The product owner can either be re-located to a desk closer to this area, or an agreement can be made that the product owner will spend more time at this location.

Impact on Geographical Distance (D1): By allocating the product owner a desk closer to the developers and the testers, the geographical distance between these roles is decreased. Similarly, this distance is also decreased, at least part of the time, by having the product owner attend project meetings and spend more time in the office space of the rest of the development team. The increased physical

proximity of the product owner increases the availability of this role to the rest of the development team. This in turn encourages more frequent and efficient communication of requirements, e.g. clarifications, detecting misunderstandings and conflicts.

Impact on Organisational Distance (D2): This practice can bridge organisational distance between the product owner and the roles to which the physical proximity is increased by providing a more direct communication path between these roles rather than traversing the hierarchical structure of the line organisation.

Impact on Cognitive Distance (D4): Cognitive distances can be bridged and eventually decreased between co-located people. The increased communication and awareness caused by being physically close to each other contributes to sharing knowledge of the domain, process and organisation, and different views on priorities for the system under development. This increased knowledge share, can bridge cognitive distance in situations where a high degree of domain knowledge is required. For example, it can ensure a common understanding of user requirements between the product owner and a tester. Furthermore, over time the cognitive differences (distance) between these roles can also decrease as the knowledge and perspectives are shared and discussed.

4.3.2 Upfront Definition of Quality Requirements (P1.14)

Quality requirements including usability aspects are identified and defined upfront before development starts. All stakeholders necessary for the relevant quality aspects are involved in defining these requirements.

Impact on Organisational Distance (D2): This practice brings stakeholders and project members together to define quality requirements and, thus provides them with a common objective. Through providing this objective of defining and agreeing to quality requirements the practice can bridge potential organisational distance and provide a forum that shortcuts any organisational hierarchy.

Impact on Cognitive Distance (D4): The priority aspect of cognitive distance can be bridged between project members and stakeholders by bringing these roles together to jointly discuss and identify the quality requirements to aim for in the project. By sharing different views on the priorities for the system cognitive distance between roles can be bridged by understanding the different perspectives and reaching a common agreement on the quality requirements for a project.

4.3.3 User/Customer Testing (P2.5)

Delivering executable code at regular intervals allows customers and/or end-users to test and validate the product under development, and then provide feedback to the development project.

Impact on Cognitive Distance (D4): This practice can bridge cognitive distance concerning domain knowledge between the customer/user (or a customer proxy)

and the developers and testers within the project. By utilising the customers' or the users' knowledge of the domain in validating the produced software missed or misinterpreted requirements can be identified.

Impact on Adherence Distance (D5): A distance in adherence between the agreed requirements and the produced software can be decreased through this practice. This is achieved by detecting missed or misunderstood requirements and then addressing these, ideally at an early stage in the development cycle.

4.3.4 Independent Testing (P3.2)

This practice entails having a testing team that is separate and independent from the developers, thereby ensuring that testers are not biased by the developers' interpretation of the requirements.

Impact on Organisational Distance (D2): Introducing a separate testing unit will increase the organisational distance between the testers of this team and the roles in the development team. This increased distance will decrease the communication between these testers and the developers (which is the aim of the practice).

Impact on Cognitive Distance (D4): This practice can bridge cognitive distance for the aspects of technical skill in testing and priorities by introducing a testing team focused on testing. The strong competence and focus on testing within this team can then compensate for weaknesses in this area in other project roles.

Impact on Semantic Distance (D6): The semantic distance between requirements and test artefacts for the aspect of coverage can be decreased by this practice. The test competence and clear responsibility for testing prescribed by this practice can lead to improved test coverage of requirements.

4.3.5 Process for Requirements Change Involving Test (P4.1)

Involving testing roles in the decision making and in the communication of changes to the requirements supports alignment through increased communication and coordination of these changes to the test organisation.

Impact on Organisational Distance (D2): Organisational distance between the testers and other project roles, e.g. product owner and developers, is bridged by this practice by introducing direct communication and decision making channels for requirements changes.

Impact on Cognitive Distance (D4): Cognitive distance for the aspect of domain knowledge can be bridged with this practice. This is achieved by ensuring that roles with a high level of domain knowledge are involved in the decision making process for requirements changes.

Impact on Adherence Distance (D5): This practice can decrease adherence distance between delivered software and agreed requirements, both by ensuring

that there is a common view on what the agreed requirements are and by initiating changes to the software that brings it closer to what the user has requested.

4.3.6 Requirements-Test Case Traces (P6.2)

Tracing between individual requirements and the test cases that verify these supports a number of activities like impact analysis, ensuring sufficient test coverage and reviewing test cases against requirements.

Impact on Organisational Distance (D2): This practice can bridge organisational distance, e.g. between the roles defining requirements and the testers, by providing direct pointers into relevant entities in the requirements versus test specification. In this way, information is made more readily available to other parts of the organisation and not just to the local unit responsible for maintaining the documentation.

Impact on Adherence Distance (D5): Requirements-test case tracing can decrease the adherence distance between agreed and documented requirements. The practice leads to a more active use of the documented requirements (for tracing to test cases) and can thereby also catch differences (distance) in the requirements between what has been agreed and what is documented. This is particularly relevant to a development process where a combination of face-to-face and document-based requirements communication is applied.

Impact on Semantic Distance (D6): The semantic distance between requirements and test cases for the aspect of coverage can be decreased by tracing between the two entities. The tracing simplifies measuring and reviewing the test coverage for requirements to ensure that it is sufficient and that all requirements are tested.

Impact on Navigational Distance (D7): The navigational distance between requirements and test cases is decreased by this practice for the simple reason that the effort required to locate the corresponding entity is decreased. However, the effort required to create and maintain the traces is substantial.

4.3.7 Small-Scale Development (P11.2)

This practice entails organising software development in such a way that it simulates small-scale development. Development is then performed with a small and tight-knit development team, thus avoiding the overhead and complexity related to large organisational structures.

Impact on Organisational Distance (D2): Organisational distance is per default minimised with this practice since organisational structure is consciously removed to construct a small organisation and project.

Impact on Psychological Distance (D3): Psychological distance between individuals can be decreased in small-scale development due to the close day-to-day working relationship within the small development team. However, if there is

larger psychological distance between people this may also become more apparent in a small-scale setting, whereas in a larger context these people can avoid each other to a greater extent.

Impact on Cognitive Distance (D4): Cognitive distance between roles and individuals can be expected to be bridged in a small project by the close collaboration and frequent interaction within the project team. These distances may eventually be decreased by sharing knowledge and perspectives, and learning from each other.

4.4 Limitations of Gap Model and Future Work

There are two main limitations of the current version of Gap Model both of which require additional work to address. These limitations concern the extent of the scope and the internal validity of the analysis. The limitations concerning scope primarily affect the generalizability of Gap Model, which currently is based on empirical data for one company. It may be applicable also to other cases with similar contextual characteristics, i.e. small projects in medium sized companies where all employees are co-located at one site and for which an iterative or agile development model is applied. However, this needs to be assessed on a case-by-case basis. Future work includes extending the empirical base of the Gap Model with additional case companies and findings from related literature. In addition, the model may be extended to also include contextual factors found to affect the relationships between practices and distances. Furthermore, the current Gap Model is limited to integration of RE with testing. An interesting future addition would be to extend the model to cover other development activities, such as usability design and architecture.

The fact that the analysis on which the current version of the Gap Model is based was performed by one researcher poses a threat to internal validity due to researcher bias. This was partly mitigated through the iterative analysis approach which included analysing the empirical data twice thereby reassessing and adjusting the initial analysis in the main analysis step. The consistency of the model was also checked against itself and against the outcome of the RET alignment study through a validation activity in the final analysis step. This added review further strengthened the reliability of the Gap Model. We believe the rigour and the applied analysis process to a large extent mitigated the risk of researcher bias. Even so internal validity could be further strengthened by involving another researcher.

5 Research Synthesis

The main results and contributions of this thesis are here summarised per reported study and main research question (as outlined in Section 2.1). In addition, the main limitations to the validity of the results are discussed for each study. More detailed descriptions of the results and threats to validity for each study can be found in the respective paper or, for the Gap Model, in Section 4 of this chapter.

5.1 Paper I: Communication Gaps

RQ.A1: What causes the challenge of gaps in requirements communication and what effects can these gaps have?

Main findings: The study confirmed that communication is a challenging part of RE and may cause a situation where requirements *slip through the gaps*; are misinterpreted or overlooked. These gaps can then result in failure to meet customers' expectations both concerning functionality and quality. These issues can be both costly and time consuming to alleviate at a late stage in the development cycle, thus indicating the importance of detecting and mitigating these gaps early on. Four main factors were identified as affecting requirements communication, namely scale, temporal aspects, common views and decision structure. The results indicate that communication gaps are common in large-scale and complex development with multiple roles involved throughout the development cycle. Temporal aspects of the applied process were found to have an effect on requirements communication. In particular, discontinuity in the requirements flow, e.g. around hand-over points of the responsibility, was found to increase the risk of missing vital requirements knowledge and awareness. In addition, lack of common views and mutual understanding between the various roles was found to cause gaps in the communication. Similarly, lack of decision structures including clearly defined and communicated goals and vision for software development was found to negatively affect the communication, primarily between those defining the requirements and the development unit.

Potential application of results: An increased awareness of the factors related to communication gaps can support practitioners in identifying issues related to these. The requirements communication can then be improved and ultimately contribute to more efficient and successful software development. By closing the communication gaps the requirements may continue all the way through the project life-cycle and be more likely to result in software that meets the customers' expectations.

Main validity issues: The main validity issues for the results of this study concern description validity, i.e. the risk of misinterpreting what the interviewees really meant, and generalizability. Description validity was mitigated both by asking the interviewees to review the transcripts and by performing a survey on the results of the data analysis. The results can be generalized by applying analytical generalization and thereby assessing the validity of the results for other cases that share similar characteristics. Furthermore, even though the results are based on data from one company several of the effects are confirmed by other related literature indicating that at least parts of the results are generalizable also to other cases.

5.2 Paper II: Overscoping

RQ.A2: What causes the challenges of overscoping of requirements and what effects can this challenge have?

Main findings: The study provided an increased understanding of scoping as a complex and continuous activity, including an analysis of the causes and effects of overscoping. The possible impact of agile RE practices to the issue of overscoping is also discussed. The results show that when operating in a fast-moving market-driven domain this ever-changing inflow of requirements must be managed effectively to avoid overscoping the development organisation. A weak awareness within the organisation of overall goals, in combination with low development involvement in early phases can contribute to 'biting off' more than a project can 'chew'. This can then lead to a number of consequences, several of which are potentially serious and expensive to alleviate, e.g. quality issues, delays and failure to meet customer expectations. Finally, the results indicate that overscoping occurs also when applying agile RE practices. However, in the agile case the overload is perceived as more manageable and lead to less wasted effort due to applying the practices of continuous scope prioritisation, gradual requirements detailing and close collaboration within cross-functional teams.

Potential application of results: The results provided by this study can be used to recognise an overscoping situation and to identify factors to address in order to achieve a more realistic project scope. By avoiding overscoping a development organisation can avoid wasting time and effort on managing and investigating requirements that are later cancelled due to having overloaded a project.

Main validity issues: The main validity issues for the results of this study are identical to the ones for Paper I since the same basic research design was applied. Thus, description validity was mitigated by interviewees reviewing the transcripts and by performing a survey on the results of the data analysis. Furthermore, the issue of generalizability of the results was addressed by analytical generalization by applying the *making a case* strategy (Robson 2002, p. 107) through analysis of related work and reporting found similarities, differences and disagreements to our results. This analysis builds a supporting argument for external validity of the results of this study.

5.3 Paper III: Challenges and Practices of RET Alignment

RQ.A3: What are the challenges and practices in aligning requirements engineering and testing (RET)?

Main findings: The contributions of this study include a framework of RET challenges and practices, and how these relate to each other. Four high-level factors found to greatly affect RET alignment were also identified. These factors are the human aspects of development, the quality of requirements, the size of the

development, and the incentives for implementing alignment. The human side of software development including communication and coordination between people was found be vital for alignment in general, so also between requirements engineers and testers. Further, the results indicate that the quality and accuracy of the requirements is a crucial starting point for testing the produced software in-line with the defined and agreed requirements. The size of the development organisation and its projects were found to be key variation factor that affect both which challenges that are faced and which tools and practices are suitable for the specific company, size and domain. Finally, the incentive for applying alignment practices such as good requirements documentation and tracing was found to vary. For companies with safety-critical development this incentive is externally motivated, while it is purely internal for non-safety critical cases. This internal motivation for RET practices is often weak due to low awareness of the cost vs. benefit of RET alignment.

Potential application of results: The results provide practical means for recognising challenges and problems in aligning requirements and testing, and for matching them with potential improvement practices. The studied case companies touch on several industrial domains and the result can therefore be relevant for a range of different companies within these domains. For these companies, the findings can provide knowledge and insight to their process improvement efforts and thereby enable strengthening their RET alignment

Main validity issues: The large number of interviews and researchers involved in this study poses the main validity issue and concerns internal validity due to individual variation in how interviews were performed. For example, different follow-on questions at the interviews. This risk was partly mitigated by defining a common case study process and by involving at least two researchers for each interview. Despite these mitigating strategies, the fact remains that there are variations in the detailed avenues of questioning for the different interviews and thereby variations in coverage of sub-topics within the interview material.

5.4 Paper IV: RE Distances

RQ.C1: What kind of distances between requirements engineering and later development activities are reported in peer-reviewed literature?

Main findings: Thirteen RE distance types were identified in the study; distances between people and between artefacts. In addition, the results provide an overview of the areas for which RE distances has been researched so far and identifies areas for which further research is needed. The most mature research was found on distances between people, i.e. geographical, temporal and socio-cultural. These types of distance have primarily been researched within the context of global software development. Even so, contradicting literature was found concerning the impact of geographical distance indicating that there are additional undiscovered factors at play. Furthermore, since most of the identified distance types are between people it is relevant to consider findings also from fields like psychology

and cognitive science in future research. The results reveal that RE distances in relationship to later development activities, e.g. design, implementation and testing, is largely un-researched. This is so despite the potential for using distance to measure coverage and consistency between related artefacts, e.g. requirements specifications and test cases. Further research into these areas could thus provide valuable contributions to software development practices and enable optimization of RE methods and practices for eliciting, negotiating and communicating requirements.

Potential application of results: The results can provide practitioners with an increased general awareness of the existence and impact of distance in software development, even though the main aim was to provide a building block for further research.

Main validity issues: The main validity issue concerns the possibility that there are additional undetected types of distance. Relevant literature and thereby distances might have been missed due to limiting the search string to the term 'distance', rather than include synonyms such as gap, proximity etc. This risk of missing relevant papers was partly addressed by broad searches for other aspects and by applying a general inclusion policy. Even though this remains an open issue, we believe it is only of minor concern considering that the purpose of the study was to act as a starting point for investigating the use of distance in software development. This is illustrated by the fact that in subsequent studies specifically for RET alignment (Gap Model generation and Gap Finder study) the set of distances has been modified by both adding and removing distances.

5.5 Thesis Introduction: Gap Model

RQ.C2: How are RET alignment practices related to RE distances?

Main findings: The main contribution of this study consists of the actual Gap Model; a theoretical framework that describes which RE distances the included RET practices affect and how. The framework thus represents knowledge of how the RET practices impact the alignment of requirements and testing by influencing underlying factors.

Potential application of results: The knowledge represented by the Gap Model could be used directly by practitioners to identify improvement practices even though the main intention of the model is be used as part of the Gap Finder method.

Main validity issues: There are two main limitations to the resulting Gap Model, namely the extent of the scope and internal validity of the analysis. It is a fact that the scope of the current Gap Model as per design is limited to the topic of integration of RE and testing. This may be extended to other development activities through future work. Furthermore, there is a risk of bias in the analysis due to one sole researcher performing the study. This was partly mitigated through an iterative analysis approach which included analysing the empirical data twice.

In addition, in the final analysis round the consistency of the model was checked against itself and previous results from the RET study. This strengthened the reliability of the Gap Model. Even though we believe the rigour and the applied analysis process to a large extent mitigates the risk of researcher bias internal validity should be further strengthened by involving another researcher. Furthermore, the fact that the Gap Model is based on data for one company poses issues concerning generalizability of the model beyond this one case. This needs to be assessed on a case-by-case basis through analytical generalization. The generalizability of the theory can in the future be extended through continued development of it by comparing it with additional cases.

5.6 Paper V: Evidence-Based Timeline Retrospective Method (EBTR)

RQ.B1: How can the EBTR method support new insights and learning concerning the interaction between requirements and other development activities?

Main findings: The evaluation showed that team reflections can be supported by visualising project history as timelines thereby providing facts as a basis for group discussions. The results indicate that this approach supports memory recall and factual discussions of project events. This visualisation of project history can then enhance group reflections around project events and support teams in gaining insight into issues and potential improvements. Furthermore, a set of variation points of the method have been identified and evaluated. These variation points enable tailoring EBTRs for different contexts and retrospective goals. The results indicate that by selecting certain variations EBTRs can be configured to support either wide assessments (e.g. the overall impact of a new process) or assessments of a specific process area. The reflections are directed accordingly through using open or semi-structured discussions, or by varying the applied timeline technique.

Potential application of results: The proposed method and the insights gained from applying it can be used to enhance the retrospective practices within software development organisations and projects. By considering a specific organisation or project the method can be tailored to match their improvement goals and adapted to the specific context of that development organisation.

Main validity issues: There are two main issues connected to construct validity, namely the risk of missed or misinterpreted factors and that the long term impact of applying the method has not been assessed. The risk of misinterpreting the impact and missing factors was partly mitigated by involving several researchers in the data collection and in the analysis. The impact of the method was assessed by investigating the participants' experiences, while the long term impact on development remains to be investigated. Furthermore, there is a risk of bias among the participants towards the method, which may have resulted in overly positive responses at the focus group. This bias was partly mitigated by triangulating the focus group data with survey data collected a while after the retrospective

meeting, thus allowing the respondents to reflect and to provide answers individually. Furthermore, for the case for which the most positive responses were given the participants were not aware that the EBTR method was the focus of the research. This decreased the risk that they would be positive about the method in order to please its stakeholders, i.e. the researchers.

5.7 Paper VI: Gap Finder Method

RQ.B2: How can Gap Finder support project teams in improving RET alignment?

Main findings: The formative evaluation demonstrated that the Gap Finder method can help teams to detect gaps and identity suitable RET practices for mitigating these. In addition, a number of improvements to the method were identified, e.g. assessing the set of existing practices and agreeing to an improvement plan in the final step. The visualisation of distances was found to enable a constructive group discussion around potentially harmful gaps and be a vehicle for supporting the project team in identifying new improvement areas. In particular, this was enabled by the concept of distance providing the team with a new perspective and potential explanation of experienced issues. Furthermore, the results indicate that some distance types have an effect on RET alignment while others are indicators of the degree of alignment and some may be used to characterise the applied development process.

Potential application of results: Practitioners can apply the proposed approach of measuring distance. By discussing gaps and distances, team members can identify issues and areas for improvement of practice. Furthermore, the reported findings of the impact of gaps on RET alignment can increase practitioners awareness of these factors and thereby lead to them noticing gaps within their own development organisations.

Main validity issues: There are two main issues with validity of the results, these concern internal validity and construct validity. The issue with internal validity concerns potential incorrect gauging of the impact of certain factors or missing other impacting factors. This risk was partly mitigated by deciding to focus the study on one development team during a specific period of development and by discussing found factors at a focus group session with the team members, thereby triangulating the viewpoints. However, it remains an open risk that study participants and/or researchers have missed or incorrectly identified the factors involved, e.g. concerning the effect of an RE distance. Finally, the impact of the method was assessed by investigating the participants' experiences of it. The long term impact on the development project remains to be investigated.

6 Future Research Directions

There are a number of directions and areas of further research that have emerged as the investigations have progressed. It is apparent that 'the more I learn, the more I learn how little I know' (attributed to Socrates as described by Plato). The insights gained through the performed research indicate several directions to take in order to explore what I now have learnt that I know little of. Apart from further improving on the proposed SPI methods (EBTR and Gap Finder) the future directions include visualisation techniques that can support reflective analysis and enhanced techniques for measuring RE distance. There are also several interesting paths to explore concerning the concept of iRE (integrated RE). These include extending the theory derived so far to other areas besides testing and to investigate how the RE integration level of a development project or organisation can be characterised by RE distances and thereby used as an indicator of the degree of integration and alignment.

6.1 Supporting Reflection through Visualisation

One of the main insights from the EBTR study (Paper V) and the Gap Finder study (Paper VI) was that visualisation of project data acted as a powerful focal point and facilitated constructive discussions within a project team. Presenting large amounts of data in a compact and illustrative way supported teams in reflecting of their practices. The project teams gained new insights concerning factors causing problems and identified suitable improvement practices through these fact-based reflections. Although the timelines and radar diagrams used in the performed studies supported reflection, a number of issues and limitations concerning the applied visualisation techniques were identified. The issues for timelines include the balance between aggregating multiple data points in one visualisation compared to showing sufficient details for reflection, how to visualise multiple data values as they change over time and compare these to an estimated level (e.g. in a plan). The issues identified concerning the visualisation of RE distances include how to visualise multi-dimensional measures (i.e. a combination of distance types and between entities) and combining multiple measures that do not have a uniform scale. Furthermore, there are issues concerned with how to visualise changes over time and enable analysing potential relationship between different factors. Finally, for both studies the large amounts of data were a challenge when designing, presenting and manipulating the visualisations.

Research within this area should cover two parts, namely (a) improved visualisation techniques and (b) evaluating the cognitive aspects of using visualisation for reflective practice. Concerning *techniques for visualisation*, the issues identified through the existing studies can act as a starting point for exploring new or adapted techniques. In addition, through reviewing existing research publications within the field of visualisation relevant related work can be identified. This most likely includes results and techniques that may be applicable in supporting reflective practice.

Evaluating the cognitive aspects of using visualisations for reflective practices is ideally performed in a multi-disciplinary forum where empirical software engineering researchers collaborate with researchers from cognitive- and social psychology. Through an initial literature study, existing research within these fields can be reviewed and potentially applicable and transferable results identified. By combining the related work within psychology and within the field of visualisation with our previous research, potential similarities, contradictions and relationships between findings can be analysed. This can then form the basis for defining a theoretical framework based on current knowledge. Through exploring new or adapted visualization techniques and evaluating the application of these in industrial projects, new knowledge can then be gained with which the framework can be extended. These enquiries can be further focused by defining a specific set of areas and scenarios for reflective practice through visualisation. These scenarios can then provide both target, structure and scope boundaries for the enquiries.

6.2 Enhanced Distance Measures

Although the distance measures applied in the Gap Finder study (Paper VI) were found to support identifying gaps and improvement practices, the study also showed that there are issues concerning the validity of these measurements. In particular, the current artefact-related distance measurements were found to be prone to biases indicating that self-assessment is less suitable for these types of distance. We propose investigating the use of techniques and algorithms from the areas of *natural language processing NLP* (Chantree 2006) and *information retrieval IR* (Borg 2013) for estimating the semantic distance between different artefacts. Techniques from this field may be candidates for distance measures for RET alignment and RE integration. These could then be evaluated by selecting a number of existing requirements specifications and the test cases designed to verify these and qualitatively assessing the semantic distance between these. If a correlation is then found between the qualitative measurements and those derived through applying the candidate NLP/IR algorithms, this would imply that these algorithms are suitable for measuring semantic distance. These measurements could then be used as indicators of the degree of alignment between the artefacts and the level of test coverage. As such these measurements would be useful as early warnings of misunderstandings or missing requirements within a development project.

For improving the measurements of the people-related distances, termed *psychometrics* (Feldt 2008), it would be very interesting to consider research from the fields within psychology. For example, the terms psychological distance (Liberman 2007) and cognitive distance (Montello 1991) are both established terms used within social and cognitive psychology. An initial literature study would identify relevant research to draw on and potentially present measurements and techniques to evaluate in the context of software development. Similar to the direction on enhanced support of reflection through visualisation (Section 6.1), this is also an area where it would be beneficial to perform a multi-disciplinary research study. By drawing from these specialist competences, improved

measurements for assessing cognitive and psychological distance within software development projects could be designed. Furthermore, combining the perspectives of these multiple disciplines is likely to provide a deeper understanding of how human factors affect software engineering. Such insight could be utilised to better accommodate these factors in designing and improving processes, methods and techniques for software development.

6.3 Further Explorations of Integrated RE

The Gap Finder study (Paper VI) indicates that integrated RE as characterised by RE distances is a viable concept. A concept that triggers team reflection of experienced issues and identification of improvements relevant to alignment and coordination. Considering the importance of these aspects for software development, it is an area well worth investigating further both in depth and in width. Concerning depth, to further explore RE distances within the concept of integrated RE. The width of the research would entail extending the scope of the area under study to also cover the integration with other development activities besides testing.

The Gap Finder study indicated that while some RE distance types directly affect alignment, others are indicators of the current state of a project, e.g. concerning alignment, or applied development model. This leads to an interesting research question concerning how a set of RE distances can be used to characterise a project concerning RE integration level, weak or strong alignment, degree of agility etc. Such indicators could be very valuable for software process assessment and in assessing the current state of a project. Research into this area could be pursued by profiling a number of projects using RE distance measures and other parameters concerning project size, length, process model, accuracy of delivery plans, number of bugs etc. This data could then be analysed to identify potential patterns and similarities, e.g. between agile projects, or between projects with long delays or large amounts of issue reports. An initial qualitative approach would be desirable in order to understand the cases in depth, and identify the factors involved and potential casual connections between them. This knowledge could then be incorporated into the existing theory for integrated RE. The found connections could be further validated by selecting additional cases to include, and thereby extend the basis for the theory. In addition, when a sufficient amount of data points have been gathered quantitative analysis could be applied and statistical tests applied to the qualitatively identified connections.

Finally, the width of our results and theory concerning integrated RE and the Gap Finder could be extended through research into integrating RE with other SE areas besides testing, e.g. usability design and architecture. Similarly to the RET alignment study (involving researchers within both fields), this should be done in collaboration with researchers specialising in the added field (e.g. usability design) to ensure sufficient competence and insight into the area. Parts of the previous studies (i.e. RET alignment, Gap Model construction and Gap Finder) would need to be performed also for this new area, e.g. a literature study, investigating challenges and practices specific to the area, designing measurements for the specific aspects related to the new area. However, the study would also need to

consider and evaluate which parts of the existing theoretical frameworks are relevant to this new area and which parts are not. For example, the results concerning collaboration and communication are likely similar, while the findings related to reviewing test cases against requirements might be transferrable to usability designs. Furthermore, the overall picture of how all three areas, i.e. RE, testing and the new area work together would also need to be investigated.

7 Conclusions and Main Contributions

Even though software development is a technical field, coordination and communication between software engineers is vital in the creative and complex process that is required to produce software. Projects where requirements are weakly communicated and aligned with other development activities run the risk of developing software that does not match the customers' and the users' requests and expectations. A closer integration of requirements engineering (RE) with the other development activities throughout the entire development life cycle can alleviate this risk of missing or misinterpreting requirements and, thus, avoid costly rework and delays.

The research presented in this thesis consists of contributions within the area of iRE (integrated RE). This contribution consists of empirical knowledge into the interaction and coordination of RE with other development activities, software process improvement methods for assessing and improving on the integration and alignment of RE, and a theory of the impact of RE distances on integrated RE.

Empirically-based insight into causes and effects of requirements communication gaps and overscoping, and challenges and practices of aligning requirements and testing is one of the novel findings. Various factors that influence coordination and alignment have been identified and range from geographical distance between key roles and temporal distance between activities to cognitive and psychological distances within project teams. These factors can cause miscommunication of requirements and delays in scoping and requirements decisions.

Two new software process improvements methods, i.e. EBTR and Gap Finder are proposed. Both method have supported project teams in identifying new areas for improved coordination and integration of RE. The Gap Finder can detect specific gaps and suggest suitable practices for mitigating these. Both apply visualisation of objective information as a mechanism to stimulate team reflection of underlying factors and thereby contribute to new insights and learning.

The presented Gap Model theory covers factors involved in integrating RE and testing, and has been applied and validated through the Gap Finder method. The theory contributes to the knowledge of integrated RE by connecting the framework of RE distances with the one for RET alignment. Gap Model, thus explains how RET alignment practices affect RE distances and the degree of RE integration.

In conclusion, through case studies, systematic literature study and theory generation a picture emerges of distances as key factors which influence the communication and coordination within software development and ultimately the

success of a product and a company. Through increased awareness of these factors and their effects, software development organisation can take steps to bridge and decrease the gaps in their information flow. The SPI methods provided in this thesis can be used by these organisations to detect gaps and empower their development teams to reflect, learn and improve on their practices and thereby the integration of RE within software development. The requirements can then be the red thread that coordinates various development activities in efficiently implementing software matching customers' and users' expectations and needs.

Acknowledgement I want to thank all the researchers involved in the EASE Theme D interview study, M. Borg, E. Engström, R. Feldt, T. Gorschek, A. Loconsole, B. Regnell, P. Runeson, G. Sabaliauskaite and M. Unterkalmsteiner, for their rigorous work in collecting and transcribing the interview material on RET alignment on which the Gap Model is based.

References

Aurum A, Wohlin C (2005), Requirements Engineering: Setting the Context, Ch 1 of Aurum, Wohlin (Eds), Managing and Engineering Software Requirements, Springer-Verlag, Germany, 2005, pp. 1-15.

Aurum A, Wohlin C (2003) The fundamental nature of requirements engineering activities as a decision-making process. Inf. Softw Technol 45:945–954

Barmi ZA, Ebrahimi AH, Feldt R (2011) Alignment of Requirements Specification and Testing: A Systematic Mapping Study. Proc 4th Int. Conf. on Software Testing, Verificiation and Validation Workshops (ICSTW), pp 476-485, Mar 2011

Basili VR (1985) Quantitative Evaluation of Software Methodology. Tech. report TR-1519, University of Maryland, College Park, Maryland.

Beck K, Beedle M, van Bennekum A et al. (2001) Manifesto for Agile Development, http://agilemanifesto.org/ (latest access: 2013-06-18)

Boehm BW (1991) Software Risk Management: Principles and Practices, IEEE Software, vol.8, no.1, pp.32-41, Jan. 1991. doi: 10.1109/52.62930

Borg M, Runeson P, Ardö A (2013) Recovering from a Decade: a Systematic Mapping of Information Retrieval Approaches to Software Traceability. Empirical Software Engineering, May 2013. doi: 10.1007/s10664-013-9255-y

Brereton P, Kitchenham BA, Budgen D et al. (2007) Lessons from Applying the Systematic Literature Review Process within the Software Engineering Domain. Journal of Systems and Softw., 80(4), pp. 571-583.

Calefato F Damian D Lanubile F (2007) An Empirical Investigation on Text-Based Communication in Distributed Requirements Workshops. Proc of 2nd Int. Conf. on Global Software Engineering (ICGSE 2007), pp.3-11. doi: 10.1109/ICGSE.2007.9

Cataldo M, Herbsleb J, Carley K (2008) Socio-Technical Congruence: a Framework for Assessing the Impact of Technical and Work Dependencies on Software Development Productivity. Proc. of 2nd ACM-IEEE Int. Symp. on Empirical Softw. Engineering and Measurements (ESEM '08)

Chantree F, Nuseibeh B, De Roeck A, Willis A (2006) Identifying Nocuous Ambiguities in Natural Language Requirements. Proc. Of 14th Int. Conf. on Reqs Engin., pp.59-68.

Cheng BH, Atlee JM (2007) Research Directions in Requirement Engineering. Proc. Future of Software Engineering (FOSE), pp 285-303, May 2007

Chrissis MB, Konrad M, Shrum S (2007) CCMI for Development, v 1.2. Guidelines for Process Integration and Product Improvement (2nd edition), SEI Series in Software Engineering, Addison-Wesley.

Collier B, DeMarco T, Fearey P (1996) A Defined Process for Project Postmortem Review, IEEE Software, vol. 13, issue 4, pp. 65-72.

Curtis B, Krasner H, Iscoe N (1988) A Field Study of the Software Design Process for Large Systems. Commun. ACM, vol. Nov. 1988, 1268-1287.

Damian D (2001) An Empirical Study of Requirements Engineering in Distributed Software Projects: Is Distance Negotiation More Effective? Proc of 8th Asia-Pacific Software Engineering Conference (APSEC 2001), pp. 149- 152.

Damian DE, Zowghi D (2003) Requirements Engineering Challenges in Multi-Site Software Development Organizations. Requirements Engineering Journal 8: 149–160.

Damian D, Chisan J, Vaidyanathasamy L, Pal Y (2005) Requirements Engineering and Downstream Software Development: Findings from a Case Study. Empirical Software Engineering, 10, 255–283, 2005.

Damian D, Chisan J (2006) An Empirical Study of the Complex Relationships between Requirements Engineering Processes and Other Processes that Lead to Payoffs in Productivity, Quality, and Risk Management. IEEE Transactions on Software Engineering, vol 43, no 7, pp 433-453. (2006)

Damian D, Kwan I, Marczak S (2010). Requirements-Driven Collaboration: Leveraging the Invisible Relationships Between Requirements and People. Collaborative Software Engineering (pp. 57-76). Springer.

Damian D, Helms R, Kwan I, Marczak S, Koelewijn B (2013) The Role of Domain Knowledge and Cross-Functional Communication in Socio-Technical Coordination. Proc. of Int. Conf. on Software Engineering 2013, pp. 442-451.

De Lucia A, Fasano F, Oliveto R, Tortora G (2007) Recovering Traceability Links in Software Artifact Management Systems using Information Retrieval Methods. ACM Transactions on Software Engineering and Methodology, Vol. 16, No. 4, Article 13

Derby E, Larsen D (2006) Agile Retrospectives: Making Good Teams Great! Pragmatic Bookshelf, 2006.

Drury M, Conboy K, Power K (2011) Decision making in agile development: A Focus Group Study of Decisions and Obstacles. Proc. Of Agile Conference 2011, pp. 39-47.

Dybå T, Dingsoyr T (2009) What Do We Know about Agile Software Development? IEEE Software, vol.26, no.5, pp.6-9, Sept.-Oct. 2009. doi: 10.1109/MS.2009.145

Feldt R, Torkar R, Angelis L, Samuelsson M (2008) Towards Individualized Software Engineering: Empirical Studies Should Collect Psychometrics. Proc. of Int. Workshop on Cooperative and Human Aspects of Software Engineering (CHASE '08), pp. 49-52

Flemming WR (1978) Requirements communication. Int. Conf. Automatic Testing (AUTOMTESTCON'78), pp. 228-229, 1978, doi: 10.1109/AUTEST.1978.764370

George M (2002) Lean Six Sigma: Combining Six Sigma Quality with Lean Production Speed. McGraw-Hill.

Gorschek T, Davis AM (2008) Requirements Engineering: In Search of the Dependent Variables. Information and Software Technology, Volume 50, Issues 1–2, January 2008, pp. 67-75, ISSN 0950-5849, 10.1016/j.infsof.2007.10.003.

Gotel OCZ, Finkelstein CW (1994) An Analysis of the Requirements Traceability Problem. 1st Int Conf on Requirements Engineering, pp 94-101.

Hasling B, Goetz H, Beetz K (2008) Model Based Testing of System Requirements using UML Use Case Models. 1st Int Conf on Softw. Testing, Verif. and Valid., pp 367-376.

Humphrey WS (1989) Managing the Software Process. SEI Series in Software Engineering, Addison-Wesley.

Huffman Hayes J, Dekhtyar A, Sundaram SK, Holbrook EA, Vadlamudi S, April A (2007) REquirements TRacing On target (RETRO): Improving Software Maintenance through

Traceability Recovery. Innovations in Systems and Software Engineering, vol. 3, no. 3, Springer, 2007, pp. 193-202.

ISO/IEC (2004-2011) ISO/IEC 15504 Information Technology – Process Assessment, parts 1-10.

Jarke M (1998) Requirements Traceability. Comm. ACM, vol. 41, no. 12, pp. 32-36.

Kamata MI, Tamai T (2007) How Does Requirements Quality Relate to Project Success or Failure? 15th IEEE Int. Requirements Engineering Conf., 2007. pp.69-78.

Karlsson L, Dahlstedt AG, Regnell B, Natt och Dag J, Persson A. (2007) Requirements Engineering Challenges in Market-Driven Software Development-An Interview Study with Practitioners. Information and Software Technology, Vol. 49, issue 6, pp 588-604.

Kellner MI, Madachy RJ, Raffo DM (1999) Software Process Simulation Modeling: Why? What? How? Journal of Systems and Software, Volume 46, Issues 2–3, 15 April 1999, pp. 91-105.

Kraut RE, Streeter L (1995) Coordination in Software Development. Communications of the ACM, vol. 38, no. 3, 69--81.

Kwan I, Marczak S, Damian D (2007) Viewing Project Collaborators Who Work on Interrelated Requirements. Proc. of 15th IEEE Int. Requirements Engineering Conference (RE'07). pp. 369-370.

Kukkanen J, Vakevainen K, Kauppinen M, Uusitalo E (2009) Applying a Systematic Approach to Link Requirements and Testing: A Case Study. Proc of Asia-Pacific Software Engineering Conference 2009 (APSEC '09), pp. 482-488. doi: 10.1109/APSEC.2009.62

Lawson M, Karandikar HM (1994) A Survey of Concurrent Engineering. Concurrent Engineering 1994 2:1. doi: 10.1177/1063293X9400200101

Layman L, Williams L, Cunningham L (2006) Motivations and Measurements in an Agile Case Study. Journal of Systems Architecture 52, pp. 654–667.

Lethbridge T, Sim SE, Singer J (2005) Studying Software Engineers: Data Collection Techniques for Software Field Studies, Empirical Software Engineering 10, 2005, pp. 311–341.

Liberman N, Trope Y, Stephan E (2007). Psychological distance. Chapter 2 of Social Psychology: Handbook of Basic Principles, 2nd edition, pp. 353-383. Guilford Press.

Lindvall M, Sandahl K (1996) Practical Implications of Traceability. Software—Practice and Experience 26, 10 (1996), pp. 1161–1180.

Marczak S, Damian D, Stege U, Schröter A (2008) Information Brokers in Requirement-Dependency Social Networks. Proc of 16th IEEE Int. Conf. on Requirements Engineering (RE'08), pp. 53-62

Marczak S, Damian D (2011) How Interaction between Roles Shapes the Communication Structure in Requirements-Driven Collaboration. 19th IEEE Int Requirements Engineering Conf., pp. 47-56.

Melnik G, Maurer F, Chiasson M (2006) Executable Acceptance Tests for Communicating Business Requirements: Customer Perspective. Proc of Agile Conference (Minneapolis, USA), pp. 12-46, July 2006

Mohagheghi P, Dehlen V (2008) Where Is the Proof? - A Review of Experiences from Applying MDE in Industry. Model Driven Architecture – Foundations and Applications, LNCS vol 5095, Schieferdecker, I and Hartman, A (eds), pp. 432-443.

Montello DR (1991) The Measurement of Cognitive Distance: Methods and Construct Validity. Journal of Environmental Psychology, 11(2), pp. 101-122.

North D (2006) Behavior Modification The Evolution of Behavior-Driven Development. Better Software, Issue March 2006.

Novorita R, Grube G (1996) Benefits of Structured Requirements Methods for Market-Based Enterprises. Proc. of the Int. Council on Systems Engineering (INCOSE) 6th Annual Int. Symposium on Systems Engineering: Practices and Tools.

Paci F, Massacci F, Bouquet F, Debricon S (2012) Managing Evolution by Orchestrating Requirements and Testing Engineering Processes. Proc of 5th Int. Conf. on Software Testing, Verification and Validation (ICST12), pp. 834 - 841

Petersen K, Feldt R, Mujtaba S, et al. (2008) Systematic Mapping Studies in Software Engineering. 12th Int. Conf. on Evaluation and Assessm. in Software Eng., pp.71-80

Pettersson F, Ivarsson M, Gorschek T (2008) A Practitioner's Guide to Light Weight Software Process Assessment and Improvement Planning. Journal of Systems and Software 81(6):972-995

Ramesh B, Jarke M (2001) Toward reference models for requirements traceability. IEEE Transactions on Software Engineering, vol 27, pp 58-93, Jan 2001

Ramesh B, Cao L, Baskerville R (2010) Agile Requirements Engineering Practices and Challenges: An Empirical Study. Inform. Systems Journal, vol 20, issue 5, 449-280.

Robinson H, Segal J, Sharp H (2007) Ethnographically-Informed Empirical Studies of Software Practice. Information and Software Technology, 49, pp. 540-551.

Robson C (2002) Real World Research. Blackwell Publishing. (2002)

Runeson P, Höst M, Rainer A, Regnell B (2012) Case Study Research in Software Engineering Guidelines and Examples. Wiley.

Sabaliauskaite G, Loconsole A., Engstrom E, Unterkalmsteiner M, Regnell B, Runeson P, Gorschek T, Feldt R (2010) Challenges in Aligning Requirements Engineering and Verification in a Large-Scale Industrial Context. 16th Int Working Conf on Requirements Eng. Foundation for Software Quality (REFSQ), pp. 128-142.

Sawyer P (2000) Packaged Software: Challenges for RE. Proc. of the 6th Int. Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'2000).

Seaman CB (1999) Qualitative Methods in Empirical Studies of Software Engineering. IEEE Transactions on Software Engineering, vol. 25, issue 4, pp 557-572.

Solis C, Wang X (2011) A Study of the Characteristics of Behaviour Driven Development, 37th EUROMICRO Conf. on Softw. Eng. and Advanced Applications (SEAA), pp.383-387.

Sommerville I (2005) Integrated Requirements Engineering: A Tutorial. IEEE Software, Vol. 22, issue 1, 16-23.

Stapel K, Knauss E, Schneider K (2009): Using FLOW to Improve Communication of Requirements in Globally Distributed Software Projects. IEEE Proc. Int. Workshop on Collab. and Intercult. Issues on Req.: Comm. Understanding and Softskills. pp. 5-14.

Stapel K, Knauss E, Schneider K, Zazworka N (2011) FLOW Mapping: Planning and Managing Communication in Distributed Teams. Proc. of 6th IEEE Int. Conf. On Global Software Engineering (ICGSE), pp 190-199.

Stapel K, Schneider K (2012) Managing Knowledge on Communication and Information Flow in Global Software Projects. Expert Systems, September 2012, Blackwell Publishing. doi: 10.1111/j.1468-0394.2012.00649.x

Tesch D, Kloppenborg TJ, Erolick MN (2007) IT Project Risk Factors: The Project Management Professionals Perspective. Journal of Computer Information Systems, 47.4, 61, 2007.

Unterkalmsteiner M, Feldt R, Gorschek T (2013) A Taxonomy for Requirements Engineering and Software Test Alignment. Accepted for publication in ACM Transactions on Software Engineering and Methodology.

Uusitalo EJ, Komssi M, Kauppinen M et al. (2008) Linking Requirements and Testing in Practice. 16th IEEE Int Requirements Engineering Conf, NJ, USA, 265-270.

Yin RK (2009) Case Study Research Design and Methods (4th edition). Sage Publications.

Yu ESK, Mylopoulos J (1994) Understanding "Why" in Software Process Modelling, Analysis, and Design. Proc. of 16th Int. Conf. on Software engineering (ICSE '94). IEEE Computer Society Press, Los Alamitos, CA, USA, pp. 159-168.

PAPER I

REQUIREMENTS ARE SLIPPING THROUGH THE GAPS - A CASE STUDY ON CAUSES & EFFECTS OF COMMUNICATION GAPS IN LARGE-SCALE SOFTWARE DEVELOPMENT[1]

Communication is essential for software development as its efficiency throughout the entire project life-cycle is a key factor in developing and releasing successful software products to the market. This paper reports on findings from an explanatory case study aiming at a deeper understanding of the causes and effects of communication gaps in a large-scale industrial set up. Based on an assumption of what causes gaps in communication of requirements and what effects such gaps have a semi-structured interview study was performed with nine practitioners at a large market-driven software company. We found four main factors that affect the requirements communication, namely scale, temporal aspects, common views and decision structures. The results also show that communication gaps lead to failure to meet the customers' expectations, quality issues and wasted effort. An increased awareness of these factors is a help in identifying what to address to achieve a more efficient requirements management, and ultimately more efficient and successful software development. By closing the communication gaps the requirements may continue all the way through the project life-cycle and be more likely to result in software that meets the customers' expectations.

[1] By Elizabeth Bjarnason, Krzysztof Wnuk and Björn Regnell, In Proc of 19th IEEE International Requirements Engineering Conference, 2011, pp.37-46.

1 Introduction

The requirements communication starts with the customer and continues throughout a development project, involving many different roles. The initially elicited requirements need to be communicated, and changes to those requirements negotiated and communicated between all affected roles, e.g. requirements engineers, developers, and testers. Since change occurs throughout the project, requirements communication must also continue during the entire life cycle (Berry 2010). For a software project to be successful, methods and tools must be supplemented with interpersonal communication across functional boundaries, but this needs to be balanced with cost and effectiveness of such communication (Kraut 1995). Despite this, the bulk of RE processes and research is mainly concerned with requirements in the early project phases, while the ultimate goal of any software project is to efficiently produce successful products; the requirements are just a means to an end. Already in the 1970s, the problem of inefficient and incorrect communication, increased as the requirements ripple through a project involving more people, was reported to lead to overly complex and badly functioning systems (Flemming 1978). Studies (Cataldo 2007, Lubars 1993, Karlsson 2007, Piri 2008) have shown that many of the RE challenges facing large-scale software development are of an organisational and social character, rather than a technical one and projects need to be organised to ensure co-ordination and communication of requirements from marketing to engineering.

Previous studies on communication focus mainly on communication paths (Marczak 2008, 2009, Stapel 2009), models (Al-Ani 2008), tools (Niinimaki 2010) and methods for improved requirements communication (Fricker 2010), rather than investigating what factors cause weak communication of requirements and what effects this has on the final software. To address this gap we report on a case study conducted in the context of large-scale market-driven software development with the following main research questions: (RQ1) what causes gaps that hinder the communication of requirements? and (RQ2) what are the effects of these gaps?

We have performed an explanatory case study at a large market-driven software development company, where we have interviewed nine practitioners. We found a number of communication gaps that affect requirements, mainly in the communication to and from the requirements engineers, but also between roles within development. Four main factors that cause communication gaps have been identified, namely scale, temporal aspects, common views and decision structures. In addition, nine effects that are a consequence of communication gaps were found, e.g. failure to meet customers' expectations, quality issues and wasted effort.

Section 2 describes related work. Section 3 provides a description of the case company. Section 4 describes the research method used in this study. Section 5 contains the results from the interview study, while Section 6 describes the outcome of the validation questionnaire on these results. In Section 7 we interpret and discuss the results, as well as, limitations of the study. Section 8 contains conclusions and further work.

2 Related Work

Curtis et al. (1988) studied the upstream part of software development and found that communication between customers, requirements engineers and the development teams is a crucial part in enabling stable requirements and a correct understanding of them, but that for large systems organisational boundaries hinder the communication. It was also found that the communication need is not reduced by documentation (Curtis 1988). Since communication and interaction with other people is a vital part of requirements engineering (apart from technical skills) soft skills are required to be successful. Based on literature and experience, a classification of such soft skills per requirements engineering activity (e.g. elicitation) has been proposed (Penzenstadler 2009).

Communication has also been reported as challenging for distributed software projects that operate in a Global Software Engineering context (Stapel 2009, Urdangarin 2008) as it can impede the understanding of requirements, and possibly lead to delays and project failures. Stapel et al. (2009) found that many problems for global development can be related to communication, and consist of missing context for interpreting requirements, awareness or documented information. Holmstrom et al. (2006) mention temporal distance as challenging in everyday communication in global software development context. Furthermore, even in global software development projects where agile practices were used communication has also been reported as challenging (Urdangarin 2008). On the other hand, Kotlarsky and Oshri (2005) reported that challenges involved in sharing knowledge across globally distributed teams are still widespread. Finally, Piri (2008) reports that many of the common problems encountered in software development projects can be traced back to social factors of the project with special challenges to communicate among distributed teams.

Al-Ani and Edwards (2008) investigated communication models adopted in large-scale software engineering projects. Others, such as Lutz (2009) investigated linguistic challenges in a global software engineering context, while Niinimaki et al. (2010) report on findings on communication tools in twelve distributed software projects. The communication flow between different development teams (Marczak 2009) and teams located at different sites (Marczak 2008, 2009) have been investigated. The interactions between individuals with different roles in cross-functional development teams have been studied and the majority of missing communication edges was found between people performing roles that were not supposed to be communicating according to the formal organisational structure (Marczak 2009). For communication around changes that affect multiple development teams it has been reported that there are a handful of key people (called information brokers) (Marczak 2008) that can both facilitate and enable efficient requirements communication, as well as, hinder and/or introduce noise, i.e. misconceptions or erroneous requirements into the requirements communication process.

3 The Case Company

Our results are based on empirical data from industrial projects at a large company that is using a product line approach (Pohl 2005). The company operates in a market-driven requirements-engineering (Karlsson 2007) context that can be characterised by lack of actual customers that can agree to requirements and the continuous inflow of requirements from multiple channels. The company has around 5000 employees and develops embedded systems for a global market. There are several consecutive releases of a platform (a common code base of the product line) where each of them is the basis for one or more products that reuse the platform's functionality. A major platform release has a lead time of approximately two years and is focused on functionality growth and quality enhancements for a product portfolio. For such projects, typically around 60-80 new features are added, for which approximately 700-1000 system requirements are produced. These are then implemented by 20-25 development teams with around 40-80 developers per team, assigned to different projects. The requirements legacy database amounts to a very complex and large set of requirements at various abstraction levels in the order of magnitude of 20,000 entities, making it an example of the Very-Large Scale Requirements Engineering context (Regnell 2008).

A number of different organisational units within the company are involved in the development. For this study, the relevant units are the *Requirements Unit* that is responsible for scope planning and requirements management, the *Software Unit* that develops the software for the platform and the *Product Unit* that develops products based on the platform. Within each unit there are several groups of specialists for different technical areas that are responsible for the work in various stages of the development process. For this case, the most essential groups are the *Requirements Teams* (RTs) (part of the Requirements Unit) that elicit and specify system requirements for a specific technical area, and *Design Teams* (DTs) (part of the Software Unit) that design, develop and maintain software. Each RT has a team leader who manages the team. Another role belonging to the Requirements Unit is the *Requirements Architect* who manages the scope at the high level and also coordinates the RTs. In the DTs there are several different roles, namely

- *Design Team Leader* who leads and plans the team's work for the implementation and maintenance phase
- *Design Team Requirements Coordinator* who leads the teams during the requirements management and design phase, and coordinates the requirements with the RTs
- *Developer* designs, develops and maintains the software
- *Tester* verifies the software

The software unit has a project management team consisting of among others *Quality Managers* who set the target quality levels and *Software Project Managers* that monitor and coordinate the DTs and interact with the Requirements Architects. The product unit is responsible for of the products, for this study *System Testing* is relevant.

The company uses a stage-gate model with several increments. There are *Milestones* (MSs) and *Tollgates* (TGs) for controlling and monitoring the project progress. In particular, there are four milestones for the requirements management and design before implementation starts: MS1, MS2, MS3, and MS4, and three

milestones for implementation and maintenance: MS5, MS6, MS7. For each of these milestones, the project scope is updated and baselined. The milestone criteria are as follows:

- **MS1**: At the beginning of each project, long-term RT roadmap documents are extracted to formulate a set of features for an upcoming platform project. A feature in this case is a concept of grouping requirements that constitute a new functional enhancement to the platform. At this stage, the features usually contain a description, their market value and effort estimates. The features are reviewed, prioritised and approved. The initial scope is decided and baselined per RT, guided by a project directive and based on initial resource estimates from the relevant DT. The scope is then maintained and regularly updated each week at a meeting of the *Change Control Board* (CCB). The role of the CCB is to decide upon adding or removing features.
- **MS2**: Features are refined into requirements by the RTs. One feature usually consists of ten or more requirements which are expressed in domain-specific, natural language including many special terms that require contextual knowledge to be understood. Each feature is assigned to a main DT that is responsible for its design, implementation and effort estimates. The requirements for a feature are reviewed together with its main DT and approved.
- **MS3**: DTs refine system requirements and start designing the system. The effort estimates are refined, and the scope is updated and baselined.
- **MS4**: The requirements refinement work and the system design are finished, and implementation plans are made. The final scope is decided and agreed with the software unit.
- **MS5**: All requirements are developed and delivered.
- **MS6**: The software is stabilised prior to customer testing.
- **MS7**: Customer-reported issues are handled. The software is updated and ready to be released.

4 Research Method

The research was conducted using a qualitative research approach, which is appropriate when individual perceptions of a complex phenomenon in its context is to be studied, using a series of interviews (Robson 2002). The results reported in this paper are part of a larger study that contains five different RE challenges: 1) Communication gaps, 2) Overscoping, 3) Keeping SRS updated, 4) Monitoring development work from requirements perspective, 5) Manual selection of requirements for release/product. Partial results for challenge 2) Overscoping, were published as a workshop publication (Bjarnason 2010a). In this paper, we present the results around challenge 1) Communication Gaps. The study has been conducted in three stages, outlined in the sections that follow.

4.1 Phase One: Pre-study Investigation & Preparations

In order to seek an explanation and more insight into the challenges around communication of requirements, we selected to perform an explanatory case study

(Robson 2002) where we start by focusing on a specific case. For this approach, we used the experience of one of the authors (from working with requirements, development and processes at the case company) as input in identifying a number of assumed requirements engineering challenges in industry (of which Communication gaps was one), as well as, possible causes and effects of these challenges. In order to avoid selecting a set of assumptions biased by only one person, these assumptions have been iterated upon in a series of brainstorming session with the other authors, and the outcome used as the main input when creating the interview study instrument (which can be accessed online, Bjarnason 2010b). The following assumed causes of communication gaps were identified in this phase (code within parenthesis denotes the cause to which it is classified in the compiled result, see Section 5.1):

- Complex product and large organisation (C1)
- Low understanding of the roles of others (C2)
- Low involvement by RT after requirements definition (C3)
- Low involvement by DT during requirements definition (C3)
- Overlapping requirements processes (C3)

4.2 Phase Two: Interview Study at the Case Company

To facilitate the discussion regarding requirements communication, and support exploring and enriching the understanding of this complex phenomenon, the qualitative interview study method has been utilised. The interview instrument (Bjarnason 2010b) produced in phase I (see Section 4.1) was designed to be semi-structured with a high degree of discussion between the interviewer and the interviewee. For each of the main challenges (including communication gaps) an open ended question about the challenge was asked: if it was a challenge, what causes it and what effects it has. This was done to find the causes and root causes of the main challenges without imposing the assumptions made during the pre-study on the interviewee. If the interviewee did not explicitly mention an assumed cause they were specifically asked about their view on it. The resulting theory related to communication challenges has thus been grounded in the empirical data gathered from interviewee with minimised bias from researchers (Strauss 1990).

The interviews were scheduled for 90 minutes each with the possibility to reduce time or prolong it. All interviews were recorded and transcribed, and the transcripts sent back to the interviewees for validation. The coding and analysis was done in an integrated and iterative fashion. The underlying structure of the interview instrument was used for categorising the views of the interviewees. For each interview, the transcribed chunks of text were placed within the relevant sections and, if so needed, copied to multiple sections. The used sections, or categories, correspond to the challenges, causes and effects (both assumed and mentioned during the interviews). These were numbered to facilitate consolidating between the interviews. Relationships were captured by noting dependencies to and from each category in specific columns.

In order to cover the full project life cycle from requirements definition through development to the end product people from all relevant organisational units (Requirements, Software and Product, see Section 3) were selected. Nine persons were selected (by the researchers) to be interviewed. Two of the interviewees (with

identical roles) requested to have their interview together. The roles, organisational belongings, and length of experience for each interviewee can be found in Table 1.

4.3 Phase Three: Validation of Results with Practitioners

In the third phase of the case study, the results from the interviews were presented to (another) seven practitioners who were asked to state their view on the results of the study via a questionnaire (see Section 6). The following practitioners were selected (by the researchers): four people from the software unit (a Software project manager and, from the Development teams, a team leader, a requirements coordinator, and a tester), 2 people from the requirements unit (Requirements team leader and Requirements architect) and one person from the product unit (System test manager). These 7 practitioners have worked within the company for a range of 4 to 13 years. At a meeting, the results around communication gaps (see Section 5) were presented, briefly discussed (especially around disagreements and additional viewpoints not covered in the results), and the participants filled out a questionnaire (available online Bjarnason 2010b) stating to which degree they agree to the results, and if they see additional, causes, root causes, and effects for communication gaps and connections to other RE challenges. The session was scheduled for 90 minutes with the possibility to extend or decrease the time as needed. Due to scheduling difficulties two sessions were required to cover all participants.

Table 1. Interviewee code (first letter denotes organisational belonging), unit and role(s) (see Section 3).

Code	Organisational unit	Role (experience in years)
Ra	Requirements	RT leader (5 years)
Rb	Requirements	RT leader (2 years)
Rc	Requirements	Requirements architect (3 years)
Pd	Product	System test manager (7 years)
Se	Software	Tester (3 years)
Sf	Software	Software project manager (2 y), DT leader (2 y), Developer (2 y)
Sg	Software	Quality manager (3 years)
Sh	Software	DT requirements coordinator (0,5 y), Developer (2 y), DT leader (1 y)
Si	Software	DT requirements coordinator (7 years)

5 Results

The results of the interview study are divided into four parts. Section 5.1 covers the causes of communication gaps, Section 5.2 contains the root causes of the main causes, Section 5.3 describes the effects of communication gaps, and Section 5.4 covers the connections found between communication gaps and the other

challenges covered by the study. The results of the questionnaire (study phase three, see Section 4.3) are reported in Section 5.

5.1 Causes of Communication Gaps

While analysing the results, we identified three of the assumed causes (see Section 4.1) as exhibiting a temporal aspect, i.e. some roles are available at different times and phases throughout the lifecycle. These assumed causes were grouped into the (new) cause *Gaps between roles over time* (C3). In addition, a fourth main cause was identified based on three of the eight interviewees mentioning issues related to company-wide strategy and unclear business priority of scope, which affects the requirements communication. The cause *Unclear vision of overall goal* (C4) was added to cover this. For each of the causes, the interviewees' viewpoints were categorised per organisation. The results are presented in Table 2, using the following classification:

- *Experienced*: cause (occurrence and impact on challenge) is experienced and was mentioned without prompting
- *Agreed*: cause not directly mentioned, but derived, agreed to direct question, observed or heard from others
- Partly agreed: partly Experienced or partly Agreed
- *Disagreed*: does not agree that this causes the challenge
- *Not mentioned*: not within expected experience for role

All of the nine interviewees had *Experienced*, *Agreed* or *Partly Agreed* to communication gaps being a challenge, and a majority of the interviewees have *Experienced* or *Agreed* to causes 1 (9 of 9) and 2 (5 of 9) contributing to gaps in communication of requirements.

Table 2. No of interviewees who mentioned each cause of communication gaps per organisational unit (R=requirements, S=software, P=product)

	Communication gaps			**C1 Complx product & large org**			**C2 Low understanding of roles**			**C3 Gaps bt roles over time**			**C4 Unclear vision of goals**		
Organisational unit	R	S	P	R	S	P	R	S	P	R	S	P	R	S	P
Experienced	2	2	1	3	4	1	1	2	1		2	1	2	1	
Agreed	1				1		1			1					1
Partly agreed		3					1	2		2	2				
Disagreed													1		
Not mentioned								1			1			4	

Communication gaps (as a challenge) Three of the interviewees (Sg, Sh, Si) *Partly Agreed*, with the motivation that the communication gaps vary between teams; for some there is close communication, for others the requirements are not communication to the affected people.

Complex product & large organisation (C1) All interviewees mentioned that size impacts both agreeing on requirements and communicating them to others. For

example, Rc said 'There are many people who need to be involved and have an opinion on things.' While Sh said: 'No-one knows the full extent of what the product can do, not even within the company.' Interviewee Rb believes that the organisational structure has a huge impact on the communication and the result of development projects.

Low understanding of roles of others (C2) Sh and Si (*Partly Agreed*) both mentioned that the understanding of requirements-related roles is weak within the development teams, with the exception of the DT requirements coordinator. The DT tester (Se) *Experienced* weak understanding of testers potential to contribute to requirements work, e.g. ensuring verifiability. Interviewee Pd *Experienced* lack of consideration of system aspects by the RTs and DTs due to a weak understanding of the role of system test. Rc (*Partly agreed*) stated that communication between RT and DT teams improved with increased understanding of each other's roles.

Gaps between roles over time (C3) One of the RT leaders (Ra) *Agreed* to this cause, and has experienced that direct communication with the DT throughout the life cycle (i.e. no gaps in time) results in more insight into and control of what is implemented. Four of the interviewees *Partly agreed* to this cause; Ra, Rc and Sh mentioned both, periods in time when requirements communication between RT and DT was sufficient (e.g. Requirements architect continuously involved via change management process), and periods when it was not so (e.g. lack of tester involvement in early phases). Si mentioned that the Requirements Teams do not always provide requirements in a timely fashion.

Unclear vision of overall goal (C4) Both RT leaders (Ra and Rb) expressed a lack of clear vision and strategies that can be used in a practical way when defining requirements for new products. This leads to power struggles between different units and technical areas rather than constructive communication around how to reach a common goal. Pd *Agreed* to this and described that there is a lack of communication around quality and system-level requirements. In contrast, interviewee Rc *Disagreed* to this cause since the technical roadmaps are reviewed and aligned with company strategy early in the projects.

5.2 Root Cause Analysis

To provide a deeper understanding around the causes of communication gaps, the interviewees were asked to describe the root causes that may be triggering these gaps for each cause. The assumed causes that were categorised as C3 (see Section 4.1) are included as root causes of C3. Figure 1 summarises the full picture of our interpretation of the interview material including the root causes (denoted RC).

Root causes of C1: Complex product & large organisation is the nature of the case company and its products, and its root causes are out of scope for this study.

Root causes of C2: Low understanding of roles of others The complexity of the products (RC2a) requires many skills that are spread over many different roles. The interviewees describe that it is hard to get an understanding of the big picture concerning how they should work and the purpose and responsibility of different roles, both due to the sheer numbers of roles involved (RC2a), as well as, the way

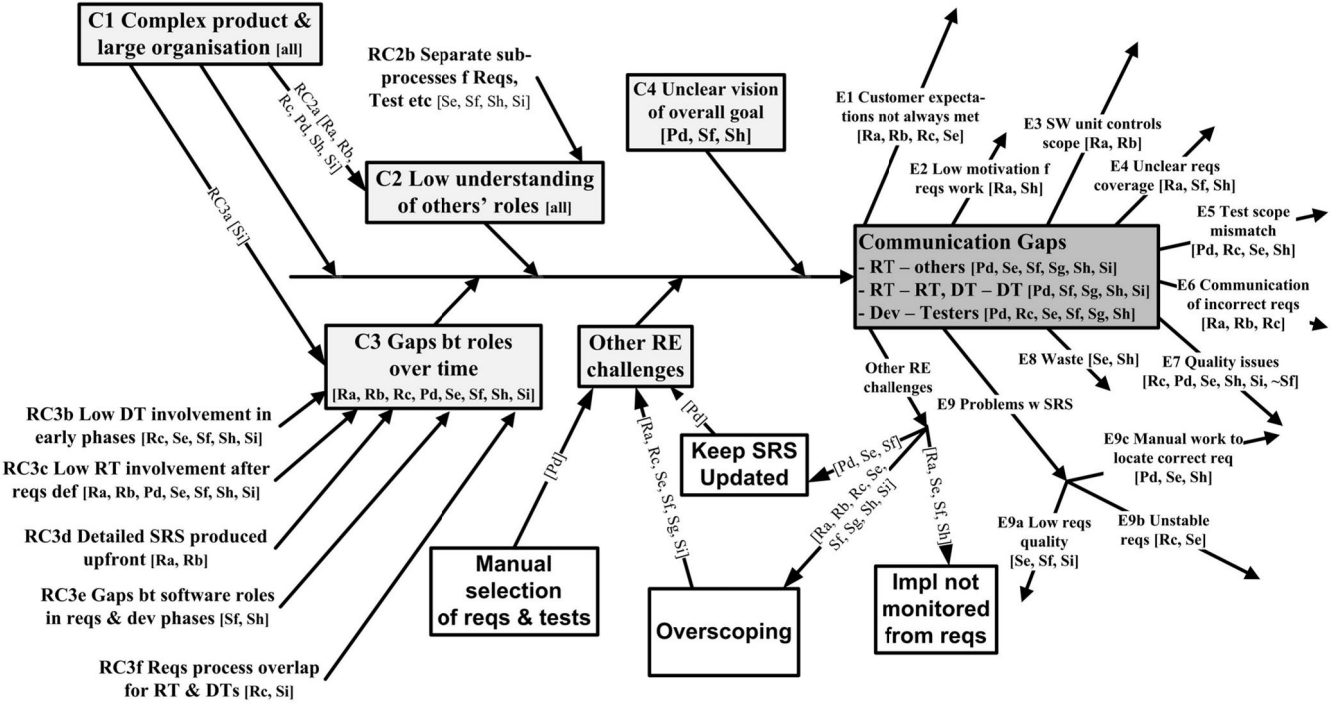


Figure 1. Causes (C), root causes (RC) and effects (E) of communication gaps, interviewee code within brackets.

the process is described in separate sub-processes for each discipline (RC2b), e.g. requirements and test. This affects the communication around requirements, causing gaps when people do not know or understand the roles of others, e.g. the difference in work characteristics between the RT leaders (standardisation & requirements work) and the DTs (design, development & maintenance).

Root causes of C3: Gaps between roles over time The work is distributed over many different people and roles (RC3a), which vary over the life cycle of a project. Our interviewees clearly describe that it is hard to achieve continuity over time especially at the handover points when work is passed on to new roles. The time periods mentioned for such gaps are, from initial scope selection to requirements detailing (RC3b), i.e. MS0-MS2, through the design and planning phase (RC3d), i.e. MS2-MS4, and then in the implementation, testing and later phases (RC3c, RC3e), i.e. MS4-. During all of these phases there is a need for requirements communication between RTs and DTs, but (as our interviewees describe) the level of communication varies between the teams. Ra expressed the situation in this way: ’We deliver requirements, but if you aren't actively checking all the time that they [DT] are implementing according to the requirements, it is quite often the case that it is something different that is being implemented.’ An even more critical point in time (described by Sf and Sh) is at MS4 when the implementation work starts, and the DT and software project responsibility is handed over to new roles without awareness of requirements (RC3e). This handover results in the requirements being more or less ignored after MS4.

Root causes of C4: Unclear vision of overall goal Vision and strategy guidance are expected to be provided by the company management. The root causes for lack of this are out of the scope of the study.

5.3 Effects of Communication Gaps

E1: Customer expectations not met When working with customer-specific requirements, communication is even harder, which Se expressed as ‘Just getting the right specification [from the customer] was impossible. And then when we finally got it, it was outdated and there was a new one.’ In addition, it is not unusual that customers are promised features that are not agreed to by the software unit, which may result in failure to meet them.

E2: Low motivation to contribute to requirements work The communication gaps around requirements between RT and DT were mentioned as leading to decreased motivation among RT leaders to work with requirements. Low understanding of roles leads to some DT testers not seeing any value in participating in requirements work.

E3: Software unit controls what is implemented Due to communication gaps between the Requirements Unit and the Software Unit, the Software Unit (with development resources) control what is finally implemented. In addition, the Software Unit has an internal roadmap that covers more than architectural improvements, and which is not agreed with the Requirements Unit.

E4: Unclear requirements coverage One of the RT leaders (Ra) said that if he does not stay in touch with the DT, he never knows exactly what is implemented. The communication gaps caused by C2 and C3 lead to DTs neither discussing requirements problems with the RTs (e.g. unclarities), nor informing them of changes that affect requirements.

E5: Test scope mismatch The test scope executed by system test is based on the SRS, but since the SRS does not correctly reflect the requirements that are finally implemented (see E4) a lot error reports are created on functionality that is not designed to work according to the SRS. Pd said: 'If you look at the error reports that are submitted, the number of things that are rejected [by DTs] due to being intended to be like this, increases in the later phases because you [system testers] are [physically] further away from requirements and developers.' The communication gaps between RTs and DTs and System test are causing testers to verify invalid requirements for which the changes have not been communicated.

E6: Communication of incorrect requirements When requirements frequently change (which they do in a market-driven context) and also slip through the gaps, it is very hard to communicate correct requirements, both to the customers and internally. Ra said: 'We gave them [customers] information about what we thoughts would be included, which often was completely wrong.'

E7: Quality issues The lack of direct communication between RTs and testers, both system testers and DT testers, lead to weak focus on system aspects (e.g. quality requirements), testing requirements (e.g. test harnesses) and test cost, in early project phases, resulting in quality issues later on. In contrast, Sf stated that gaps between developers and testers are beneficial for software quality, since competition encourages testers to smoke out problems with software produced by the developers.

E8: Wasted effort The communication gaps increase the time it takes to communication changes to all involved parties, and thus increase the amount of work wasted so far on requirements, design and implementation work, which then has to be redone. The gaps caused by roles changing at MS4 leads to waste of effort to transfer knowledge, and missed requirements knowledge and awareness.

E9: Problems with SRS The gaps between RT leaders and, DT testers and developers, result in unclear, ambiguous and non-verifiable SRS requirements (E9a), and subsequent problems when implementing and verifying them. The communication gaps between RT and DT during requirements detailing contribute to unstable requirements (E9b); since the viewpoints of the testers and the developers are not taken into consideration until later project phases. Instead, issues are uncovered when design, implementation and testing start at which point the requirements need modifying. The problem is enhanced when external parties like customers are involved. The communication gaps between RTs and developers and testers result in them being force to locate requirement information (E9c) mainly through other channels. The SRS is one such channel, in which it is hard to locate specific and relevant requirements and sometimes the implemented requirements are not in the SRS (see E4). The DT testers mainly receive requirements by asking the developers.

5.4 Connections to Other Challenges

When analysing the interviews, we found that of the four other challenges covered by the study, all of them had connections to communication gaps, either mentioned as direct causes or consequences of communication gaps, or by resulting in an effect that contributes to another challenge. The full picture of the connections is shown in Figure 1.

Overscoping, or including more requirement than there are resources for, results in increased communication gaps between teams (both DTs and RTs) because they do not have time to communication around requirements. Overscoping also results in friction between the DTs and software project managers, e.g. when failing to deliver according to plan. Gaps in communication between the RTs and stakeholders, as well as, DTs, lead to the RTs specifying a scope missing vital requirements and without reliable cost estimates, all of which leads to overscoping.

Keeping SRS updated partly bridges the gaps in communication between RT and system test. But, when the SRS is not kept updated, this results in error reports on invalid SRS requirements (see E5 in Section 5.3) and increased communication gaps to DTs who claim that the software works as it should. Communication gaps between RTs and DTs in later project phases result in RTs being unaware of implementation changes that affect the requirements, causing a mismatch between SRS and delivered software.

Manual selection of requirements for products contributes to communication gaps for the same reasons as for the challenge *Keep SRS updated* since the product requirements are selected from the requirements in the SRS, which is not in line with the implemented software (see E4 and E9 in Section 5.3)

Implementation not monitored from requirements viewpoint is caused by gaps between roles before and after MS4 (see RC3e in Figure 1). When implementation starts, the responsibility is transferred to roles who have little insight or awareness of requirements, during project phases when RTs have little contact with DTs. Requirement change, but often without RT involvement. The implementation continues, more or less, without being concerned with the requirements.

6 Validation of Results with Practitioners

In phase three of the study (see Section 4.3), the results described in Section 5 were presented to seven practitioners at the case company. They noted their level of agreement in a questionnaire (Bjarnason 2010b) using the following notation:

- *Experienced*: I have experienced this to be valid
- *Agree*: I agree to this, but have no personal experience
- *Partly agree*: I agree to part, but not all, of this
- *Disagree*: I do not agree
- *Don't know*: I have no knowledge of this

A majority of the participants noted *Experienced* or *Agreed* to all, but one, of the causes, root causes, and effects. For *Overlapping requirements process between RT*

& DT (RC3f, see Figure 1), three respondents had *Experienced* this root cause, while three *Partly agreed* and one answered *Don't know*. In addition to the presented results, late test involvement in the projects was mentioned as an additional root cause to C3 *Gaps between roles over time*, resulting in missing requirements from the testers concerning, e.g. test harnesses and other functionality required for verifying the software. Concerning C1 *Complex product & large organisation*, one participant claimed that the way the product portfolio was planned (by business people with little input from the software unit) resulted in a more complex portfolio than necessary since little consideration was given to the cost of implementing and supporting a large number of different configurations.

7 Interpretation and Discussion

In this section, we provide our interpretation and discussion of the results around causes and effects of communication gaps, and compare them to related work. In Section 7.1, we discuss the limitations of this study.

Requirements communication is a challenge for the case company though there are examples of good requirements communication between teams and individuals. The four identified causes correspond to four different factors that contribute to communication gaps, namely *scale* (C1), *common views* (C2), *temporal aspects* (C3) and *decision structures* (C4).

Cause 1: Complex Product and Large Organisation covers the factor of scale. Our responders clearly state that the size of the organisation and the complexity of the products, contribute to communication gaps. A survey study into coordination of large-scale software development (Kraut 1995) found that scale contributes to communication gaps over geographic, organisational and social boundaries, due to dividing the work over many different specialised roles. In addition, organisational boundaries cause communication gaps that hinder the mutual understanding of requirements (Curtis 1988). Our study shows that there is a communication gap upstream towards the Requirements Teams resulting in requirements being received by Development Teams from many different sources, as well as, incomplete requirements specifications, overscoping, and conflicting requirements.

Cause 2: Low understanding of each other's roles covers the factor of *common views*. The domain knowledge and perspectives vary between roles. Without respect and mutual understanding for each other's viewpoints this causes communication gaps, either by not communicating at all (due to lack of understanding that other roles are impacted) or by ineffective communication (e.g. missing tacit requirements due to lack of insight into the customer's domain). Weak understanding of the work of other units negatively affects the communication and cooperation (Sabaliauskaite 2010). Communication around the design between stakeholder and architects leads to shared understanding of the requirements and identification of tacit requirements, as well as, needed requirement changes (Fricker 2010). Similarly, application domain knowledge has been reported as vital in designing a solution that will meet the customer's needs (Curtis 1988).

Cause 3: Gaps between roles over time covers the factor of *temporal aspect*. Our results indicate that requirements communication needs to continue throughout the project life cycle, since requirements are dynamic and change, often until they are implemented. Communication gaps between requirements and development teams during early phases have previously been found to result in requirements that could not be implemented (Berry 2010, Curtis 1988, Sabaliauskaite 2010). Failure to bridge these gaps results in delays, and increases the cost of handling late errors and changes (Berry 2010). Also, there are certain hand-over points (MS2 and MS4 for our case company) when it is crucial that sufficient knowledge of the requirements is transferred to new roles, in order to ensure continuity throughout the project life cycle and avoid development becoming disconnected from requirements. A suggestion for how to avoid some of these gaps is given by Fricker et al. (2010), were communication between stakeholders and architects around design was shown to improve the probability that the requirements are carried on into later phases of the project. A surprising detail of our results indicate that producing a detailed requirement specification upfront may contribute to communication gaps (root cause RC3d), since it then may be assumed that no additional communication of requirements is needed. We found similar conclusions drawn by Curtis et al. (1988), i.e. that the existence of artefacts can contribute to communication gaps since people tend to assume the artefacts in themselves constitute sufficient communication.

Cause 4: Weak vision of overall goal covers the factor of *decision structures*. When there is no clear common goal for the software development it is up to the individual teams and units to make decisions on which requirements to include. For our case company, this, in combination with weak understanding of each other's roles (C2) has led to wide communication gaps between the Requirements and the Software Units, resulting in the Software Unit controlling which requirements are actually implemented (E3). Similar communication gaps are reported by Karlsson et al. (2007) as a challenge for which having a common goal and vision (C4) is a way to resolve, or close, such gaps.

Effects Communication gaps contribute to a number of consequences for the project and for the resulting software. The communication gaps during requirements definition contribute to an instable, unclear and ambiguous SRS (E9). Weak communication with the customers has been found to cause instable requirements (Curtis 1988), while communication between the customer and the development team is seen to mature both the requirements and the design. For our case company (that operates in a consumer market with no direct communication with the end customers) the Requirements Unit represents the (anonymous) customers. The view of what constitutes a *good* requirements has been found to vary between roles (Karlsson 2007), indicating a weak common view (C2).

For the case company, there is a huge gap in requirements communication during the later phases of the projects (after MS4), which results in the software implementation being done without the projects, or teams, being monitored from a requirements perspective. Instead, project management monitors on committed delivery dates and number of error reports, while the developers rely on the design correctly reflecting the requirements, and the testers rely on the SRS being kept updated (which is a challenge). In large-scale market-driven development where

change is constant, this results in unclear requirements coverage (E4); there is no clear and common view of which requirements that are actually supported. Instead, incorrect requirements information is given (E6), both internally and to customers, also mentioned as a consequence of weak communication (Karlsson 2007), and the test scope does not match the implemented requirements (E5). All this results in not always meeting the customers' expectations (E1); either due to lack of desired functionality or quality issues (E7), also reported by Flemming (1978). In addition, effort is wasted (E8), e.g. when testing requirements for which agreed changes have not been communicated, which contributes, together with C2, to low motivation to work with requirements (E2).

7.1 Threats to Validity and Limitations

We discuss the validity threats according to the classification provided by Robson (2002). The main threat to *description validity* is to provide a valid description of what interviewees said and meant. This threat was addressed by recording and transcribing the interviews. The transcripts were sent back to the interviewees to check for misinterpretations and other errors. To ensure open and honest replies the interviewees had full anonymity; the full set of names of the interviewees was only know to the researchers and the company is large enough for the individuals not be identifiable from the information given about them in this paper.

To address the treats to *valid interpretation*, the question on each challenge (of which communication gaps was one) were formulated in an open and indirect way to encourage the interviewee to express her own opinion before mentioning the assumed causes. A possible source of unreliability is related to *observer biases* where the results from the pre-study, as well as, questions asked during the interview, may have been consciously or unconsciously biased by the researcher. This threat was addressed by all the authors discussing the results of the pre-study, the selection of interviewees, and reviewing the interview instrument. Moreover, the practitioner's involvement in the study has played a vital role in focusing on and ensuring that the problems under investigation are authentic problems, that the interpretation of data is based on a deep understanding of the case and its context, and that the outcome of the study is authentic. To mitigate the risk of quotations becoming out of context during the analysis phase (Coffey 1996), the observer triangulation method was used (Robson 2002); one researcher randomly selected two interview recordings and performed an independent transcription and coding. Differences were discussed and conflicts resolved. Data triangulation was also applied by the questionnaire responses from another set of practitioners to further validate the results from the interview study.

The possibility of generalising the results of this case study has been addressed both internally within the study and in respect to *external generalizability*. The *internal generalizability* was addressed by sampling participants from different parts of the company with different roles. As for *external generalizability*, the main threat to validity is no possibility of performing a statistical generalisation due to lack of representative sample and only one company involved in the study. However, the main focus on this study is to increase the understanding of communication around requirements and explore possible causes of gaps in this communication rather than providing a full theory that can be generally applied.

Finally, communication gaps were confirmed as a challenge by all our responders with only minor differences of the importance of this issue and all of the identified causes, and several of the effects, of communication gaps have been reported by other researchers in related studies (see Section 7).

8 Conclusions and Future Work

Communication is one of the key mechanisms in coordinating a project, of which the requirements, or 'a common view of what the software they are developing should do' (Kraut 1995), is a vital part. The organisational theory literature suggests that for an organisation to be successful, an appropriate combination of organisational structure, processes, and communication and coordination mechanisms, is needed (Cataldo 2007). Since software development is as highly collaborative endeavour, many of the problems encountered during software projects can be traced back to social factors (Piri 2008). Despite the fact that several studies have reported the challenging nature of communication in software and requirements engineering (Flemming 1978, Holmstrom 2006, Karlsson 2007, Kotlarsky 2005, Lubars 1993, Piri 2008, Urdangarin 2008) and investigated various aspects of communication (Al-Ani 2008, Lutz 2009, Marczak 2008, 2009, Niinimaki 2010), no consolidated empirical evidence on the causes, root causes effect and relations to other requirements engineering challenges has (to the best of our knowledge) been presented.

In this paper, we address this gap by reporting empirical evidence based on an interview study performed with nine interviewees at a large software development company. To further strengthen the validity of the study we also conducted a questionnaire with a different set of seven practitioners who confirmed the results. The study confirms that communication is a challenging part of requirements engineering and may cause a situation where requirements *slip through the gaps*; are misinterpreted or overlooked, resulting in failure to meet customers' expectations both concerning functionality, as well as, quality.

We have identified four main factors that may cause communication gaps: *scale, common views, temporal aspects, and decision structures*. The size and complexity of the software development, i.e. *scale*, increases the challenge of requirements communication. We found communication gaps between the requirements engineers and a number of stakeholders, resulting in missing requirements, e.g. for quality. Instead, these requirements surface in later phases, thus, incurring increased cost.

Common views and mutual understanding are necessary for communication to be productive. Weak understanding of each other's roles and responsibilities causes gaps in communication. For example, the testers' competences are not utilised when defining and reviewing requirements, or the requirements engineers are not consulted when making implementation choices that affect the requirements.

Temporal aspects come into play when there is a lack of continuity in requirements awareness through the project life cycle. This may cause gaps in the requirements communication. Hand-over points, e.g. defined by the process, where the responsibility is passed on to new roles constitute a risk of missing vital requirements knowledge and awareness. This may result in requirements being

misunderstood and incorrectly implemented, or, making decisions that affect the requirements without considering all relevant aspects. For example, if there is no requirements awareness in the implementation phase, the developers tend to make their own requirement modifications without considering the impact on the customer or on other parts of the development organisation, such as test.

Decision structures also contribute to communication gaps. Weak, or unclear, visions or goals for the software development (due to not being communicated or not being clear enough) contributes to weak communication, primarily, between those defining the requirements and the development unit, since there is no mutual understanding of the goal.

Our study shows that communication gaps can have serious and expensive consequences in terms of wasted effort and quality issues, as well as, not meeting the customers' expectations and even communicating an incorrect picture of what requirements a product fulfils to the customers. In addition, communication gaps can contribute to a number of other RE-related challenges, like overscoping and keeping the SRS updated. This, in turn, contributes to communication gaps, i.e. the software development ends up in a vicious cycle.

The increased understanding of the causes and risk of gaps in requirements communication provided through this study, can be a help in identifying potential communication gaps in existing software development processes and organisations. The goal should be to close such gaps and enable requirements management to efficiently support and guide development projects towards producing quality software that will meet customers' expectations.

Future work includes investigating how aspects such as organisational set-up, software development model (agile or waterfall) and application of different software engineering methods affect the challenges, and their causes both within the case company, and in a broader context.

References

Al-Ani B, Edwards HK (2008) A Comparative Empirical Study of Communication in Distributed and Collocated Development Teams. Proc. IEEE Int. Conference on Global Software Engineering (ICGSE '08). IEEE Computer Society, pp. 35-44. doi: 10.1109/ICGSE.2008.9.

Berry DM, Czarnecki K, Antkiewicz M, AbdElRazik M (2010) Requirements Determination is Unstoppable: An Experience Report. Proc. IEEE Requirements Engineering Conference (RE'10), IEEE Computer Society, Sept 2010, pp. 311-316.

Bjarnason E, Wnuk K, Regnell B (2010a) Overscoping: Reasons and Consequences – A Case Study in Decision Making in Software Product Management. Proc IEEE Int. Workshop on Software Product Management (IWSPM'10), Sept 2010, pp. 30-39.

Bjarnason (2010b) The interview instrument and validation questionnaire for the Before aNd After study (BNA) is available at http://serg.cs.lth.se/research/experiment_packages/bna

Cataldo M, Bass M, Herbsleb JD, Bass L (2007) On Coordination Mechanisms in Global Software Development. Proc IEEE Int Conf. on Global Software Engineering (ICGSE '07). IEEE Press, pp. 71-80, doi: 10.1109/ICGSE.2007.33

Coffey AJ, Atkinson PA (1996) Making Sense of Qualitative Data: Complementary Research Strategies. Sage Publications, Inc, 1996.

Curtis B, Krasner H, Iscoe N (1988) A Field Study of the Software Design Process for Large Systems. Commun. ACM Nov. 1988, pp. 1268-1287, doi:10.1145/50087.50089.

Flemming WR (1978) Requirements Communication. Int. Conf. Automatic Testing (AUTOMTESTCON'78), pp. 228-229, 1978, doi: 10.1109/AUTEST.1978.764370

Fricker SM, Glinz M (2010) Comparison of Requirements Hand-Off, Analysis, and Negotiation: Case Study. Proc IEEE Int. Requirements Engineering Conference, Sept 2010, pp. 167-176, doi: 10.1109/RE.2010.29

Holmstrom H, Conchuir EO, Agerfalk PJ, Fitzgerald B (2006) Global Software Development Challenges: A Case Study on Temporal, Geographical and Socio-Cultural Distance. Proc IEEE Int Conf. on Global Software Engineering (ICGSE '06), pp. 3-11.

Karlsson L, Dahlstedt AG, Natt Och Dag J, Regnell B, Persson A (2007) Requirements Engineering Challenges in Market-Driven Software Development An Interview Study with practitioners. Inf and Soft Techn, vol. 49, Dec. 2007, pp. 588-604.

Kotlarsky J, Oshri I (2005) Social Ties, Knowledge Sharing and Successful Collaboration in Globally Distributed System Development Projects. Eur Journal of Inf Systems, vol. 14, Mar. 2005, pp. 37-48, doi: 10.1057/palgrave.ejis.3000520.

Kraut RE, Streeter L (1995) Coordination in Software Development. Communications of the ACM, vol. 38, Mar 1995, pp. 69-81, doi: 10.1145/203330.203345.

Lubars M, Potts C, Richter C (1993) A Review of the State of the Practice in Requirements Modelling. Proc. IEEE Int. Symposium on Requirements Engineering (RE'93), IEEE Press, Jan. 1993, pp. 2-14, doi: 10.1109/ISRE.1993.324842.

Lutz B (2009) Linguistic Challenges in Global Software Development: Lessons Learned in an Int. SW Development Division. Proc. Int. Conf. on Global Software Engineering, Jul. 2009, pp. 249-253, doi: 10.1109/ICGSE.2009.33

Marczak S, Damian D, Stege U, Schröter A (2008) Information Brokers in Requirement-Dependency Social Networks. Proc. IEEE Int Conference on Requirements Engineering, IEEE Press, Sep. 2008, pp. 53-62, doi: 10.1109/RE.2008.26.

Marczak S, Kwan I, Damian D (2009) Investigating Collaboration Driven by Requirements in Cross-Functional Software Teams. Proc. Int. Workshop on Collaboration and Intercultural Issues on Requirements: Communication, Understanding and Softskills, Aug. 2009, pp.15-22, doi: 10.1109/CIRCUS.2009.2.

Niinimaki T, Piri A, Lassenius C, Paasivaara M (2010) Reflecting the Choice and Usage of Communication Tools in GSD Projects with Media Synchronicity Theory. Proc. IEEE Int. Conf. on Global Software Engineering, Sep. 2010, pp. 3-12.

Penzenstadler B, Schlosser T, Frenzel G (2009) Soft Skills REquired: A practical approach for empowering soft skills in the engineering world. Proc. Int. Workshop of Collaboration and Intercultural Issues on Requirements: Communication, Understanding and Softskills, IEEE Computer Society, Aug. 2009, pp. 31-36, doi: 10.1109/CIRCUS.2009.5.

Piri A (2008) Challenges of Globally Distributed Software Development - Analysis of Problems Related to Social Processes and Group Relations. Proc IEEE Int Conf on Global Software Engineering, Sep. 2008, pp. 264-268, doi: 10.1109/ICGSE.2008.33.

Pohl C, Böckle G, van der Linden FJ (2005) Software Product Line Engineering: Foundations, Principles and Techniques. Springer-Verlag, New York USA, 2005.

Regnell B, Berntsson-Svensson R, Wnuk K (2008) Can We Beat the Complexity of Very Large-Scale Requirements Engineering? Proc. Int. Conf. on Requirements Engineering: Foundation for Software Quality (REFSQ '08), vol. 5025 of LNCS, Springer-Verlag, Berlin, Heidelberg, pp. 123-128. doi: 10.1007/978-3-540-69062-7_11

Robson C (2002) Real World Research. Blackwell.

Sabaliauskaite G, Loconsole A, Engström E, Unterkalmsteiner M, Regnell B, Runeson P, Gorschek T, Feldt R (2010) Challenges in Aligning Requirements Engineering and Verification in a Large-Scale Industrial Context. Proc. Int. Working Conference Requirements Engineering: Foundation for Software Quality (REFSQ 2010), Mar. 2010, LNCS, vol. 6182, pp. 109 – 115, doi: 10.1007/978-3-642-14192-8_14.

Stapel K, Knauss E, Schneider K (2009) Using FLOW to Improve Communication of Requirements in Globally Distributed Software Projects. Proc. Int. Workshop on Collaboration and Intercultural Issues on Requirements: Communication, Understanding and Softskills. IEEE Computer Society, Aug. 2009, pp. 5-14, doi=10.1109/CIRCUS.2009.6.

Strauss A, Corbin J (1990) Basics of Qualitative Research: Grounded Theory Procedures and Techniques. Sage Publications, New York, 1990.

Urdangarin R, Fernandes P, Avritzer A, Paulish D (2008) Experiences with Agile Practices in the Global Studio Project. Proc. IEEE Int. Conf. on Global Software Engineering, Aug. 2008, pp. 77-86, doi: 10.1109/ICGSE.2008.11.

PAPER II

ARE YOU BITING OFF MORE THAN YOU CAN CHEW? A CASE STUDY ON CAUSES AND EFFECTS OF OVERSCOPING IN LARGE-SCALE SOFTWARE ENGINEERING[1]

Scope management is a core part of software release management and often a key factor in releasing successful software products to the market. In a market-driven case, when only a few requirements are known a priori, the risk of overscoping may increase. This paper reports on findings from a case study aimed at understanding overscoping in large-scale, market-driven software development projects, and how agile requirements engineering practices may affect this situation. Based on a hypothesis of which factors that may be involved in an overscoping situation, semi-structured interviews were performed with nine practitioners at a large, market-driven software company. The results from the interviews were validated by six (other) practitioners at the case company via a questionnaire. The results provide a detailed picture of overscoping as a phenomenon including a number of causes, root causes and effects, and indicate that overscoping is mainly caused by operating in a fast-moving, market-driven domain, and how this ever-changing inflow of requirements is managed. Weak awareness of overall goals, in combination with low development involvement in early phases may contribute to 'biting off' more than a project can 'chew'. Furthermore, overscoping may lead to a number of potentially serious and expensive consequences, including quality issues, delays and failure to meet customer expectations. Finally, the study indicates that overscoping occurs also when applying agile RE practices, though the overload is more manageable and perceived to result in less wasted effort when applying a continuous scope prioritisation, in combination with gradual requirements detailing and a close cooperation within cross-functional teams. The results provide an increased understanding of scoping as a complex and continuous activity, including an analysis of the causes, effects, and a discussion on possible impact of agile requirements engineering practices to the issue of overscoping. The results presented in this paper can be used to identify potential factors to address in order to achieve a more realistic project scope.

[1] By E. Bjarnason, K. Wnuk and B. Regnell, Published in Information and Software Technology, 54(10): 1107-1124, 2012.

1 Introduction

Maximising the business value for a product and a set of available resources may sound like a simple task of selecting features according to the highest return of investment. However, in market-driven requirements engineering (MDRE) (Karlsson 2007a, Regnell 2005) software product managers face the challenge of managing continuously shifting market needs (Abramovici 2002) with a large number of new and changing requirements (Gorschek 2005) caused both by a capricious market situation (DeBaud 1999) and by evolving technologies. In this situation, selecting which requirements to include into the next release of a software product (also called *scoping* by Schmid 2002 or *project scoping* by PMI 2000) is a complex and continuous task of assessing and re-assessing how these scoping changes impact the common code base of the software product line (Pohl 2005) on which those products are built (Wnuk 2009). This domain scoping is considered part of the product line scoping (Schmid 2002), which derives value from the opportunities to reuse functionality of the product line. These factors, combined with increased market competition and unpredictable market response to new products, force decision makers to continuously face the task of making and re-evaluating decisions in an ever evolving world (Aurum 2003).

Defining the scope of a product to fit a required schedule is a known risk in project management (Boehm 1989) and in our previous work (Wnuk 2009) we found that the project scope at a large software company changed significantly throughout the entire project life cycle. These changes were partly due to *overscoping*, i.e. setting a scope that requires more resources than are available. Several researchers have focused on *scope creep* where the scope is increased by the developers, and highlighted this as a serious project risk (Carter 2001, Crockford 1980, Iacovou 2004). Others have investigated scoping as a part of release planning (Schmid 2002, Svahnberg 2010, Wnuk 2009). However, no study has yet attempted to investigate the causes and effects of overscoping even though requirements engineering (RE) decision making is an acknowledged challenge (Alenljung 2008, Aurum 2003, Ngo-The 2005). In this study, we have investigated this phenomenon of *overscoping* a project, or biting off more that you can chew, in particular in a market-driven and very-large scale RE (VLSRE) context (Regnell 2008).

Agile development processes claim to address several of the challenges involved in scoping frequently changing requirements. For example, in eXtreme programming (XP) (Beck 1999) and Scrum (Schwaber 2002) the balance between scope and available resources is managed by extreme prioritisation and constant (re)planning of the scope in combination with time boxing of the individual development iterations. However, agile requirements engineering (RE) practices have also been found to pose new challenges, e.g., in achieving consensus on priorities among multiple stakeholders and in creating accurate project plans (cost and timeline) for an entire project (Ramesh 2007).

The main goal of the case study reported on in this paper was to increase the understanding of factors involved in overscoping and thereby highlight this risk and take a step towards addressing and avoiding overscoping of projects. To achieve this, the study was designed to answer the following questions: (RQ1) what causes overscoping?; (RQ2) what are the resulting effects of overscoping?; and (RQ3) how may agile RE practices impact the causes and effects of overscoping? The case

study has been conducted at a large market-driven software development company that has started to shift towards a more agile way of working. The study includes interviews with nine practitioners working with requirements engineering, software development and product testing. The interview results were then validated via a questionnaire with another six practitioners from the case company. The contribution of the presented work includes eight main causes of overscoping complemented by a number of root causes, and nine main effects of overscoping. In addition, the results indicate that three of the agile RE practices adopted by the case company may impact some of these causes and root causes and, thus, may also reduce the effects of overscoping.

Partial results from this study have previously been published as workshop publications in (Bjarnason 2010a) where overscoping was preliminarily investigated and in (Bjarnason 2011a) where preliminary results on the benefits and side effects of agile RE practices were published. For this article, the results are extended with (1) additional causes, root causes and effects of overscoping; (2) additional empirical results on overscoping from 6 (other) practitioners; and (3) details on the impact of agile RE practices specifically on overscoping. These extensions were achieved by further analysis of the full interview material and further validation of the results through a questionnaire.

The remainder of this paper is structured as follows: Section 2 describes related work. Section 3 describes the case company, while the research method is outlined in Section 4. The results are reported in Section 5 for the interviews and in Section 6 for the validation questionnaire. In Section 7, the results are interpreted and related to other work, and limitations and validity threats are discussed. Finally, Section 8 contains conclusions and further work.

2 Related Work

Unrealistic schedules and budgets are among the top ten risks in software engineering (Boehm 1989) and some reasons for overloading projects with scope have been suggested. For example, DeMarco and Lister (2003) mentioned that a failure among stakeholders to concur on project goals (also one of the challenges of agile RE according to Ramesh et al. 2007) can result in an excessive scope burden on a project. Project overload may also result from sales staff agreeing to deliver unrealistically large features without considering scheduling implications (Hall 2002). Furthermore, scope that is extended beyond the formal requirements by the developers, i.e. scope creep, is stated by Iacovou and Dexter (2004) as a factors leading to project failures. Scope creep is also mentioned as having a big impact on risk and risk management in enterprise data warehouse projects (Legodi 2010). In addition, it is listed as one of five core risks during the requirements phase, and is a direct consequence of how requirements are gathered (DeMarco 2003). On the other hand, Gemmer (1997) argues that people's perceptions of risk and their subsequent behaviour is overlooked within risk management and that an increased awareness of causes and effects of risks may lead to an improved discussion and management of these risks. Some methods and tools to mitigate and manage risks related to scoping have been presented (Crockford 1980). For example, Carter et al. (2001) suggested combining evolutionary prototyping and risk-mitigation strategy to mitigate the

negative effects of scope creep. However, the full issue of overscoping is not explicitly named as a risk in the related work, nor empirically investigated for their causes and consequences.

Requirements engineering (RE) is a decision intense part of the software engineering process (Aurum 2003), which can support and increase the probability of success in the development process (Aurum 2005a). However, the efficiency and effectiveness of RE decision making has cognitive limitations (Aurum 2003), due to being a knowledge intensive activity. Furthermore, research into the field of RE decision making is still in its infancy (Alenljung 2008, Ngo-The 2005). A major challenge in this research (according to Alenljung and Persson 2008) lies in understanding the nature of RE decision making and identifying its obstacles and several authors (Alenljung 2008, Aurum 2003, 2005a, Ngo-The 2005) mention the need to: (1) identify decision problems in the RE process; (2) perform empirical studies of RE decision making; and (3) examine how non-technical issues affect or influence decision making. Communication gaps are an example of such non-technical issues which have been reported to negatively affect the decision making and contribute to defining an unrealistic scope (Bjarnason 2011b).

There are two characteristics of MDRE (Regnell 2005) which further aggravates RE decision making, namely a lack of actual customers with which to negotiate requirements (Karlsson 1997, Potts 1995) and a continuous inflow of requirements from multiple channels (Gorschek 2005, Karlsson 2007a). As a result, rather than negotiate with specific customers, the demands and requirements of an anonymous consumer market have to be 'invented' (Potts 1995) through market research. Moreover, the success of the final product is primarily validated by the market with the uncertainty (Regnell 2005) of how well the 'invented' requirements compare to the market needs. Commonly, market research continuously issues more requirements (Regnell 2005) than can be handled with available resources. A state of congestion then occurs in the MDRE process (Karlsson 2007a) and the set of requirements to implement in the next project has to be selected using prioritisation techniques based on market predictions and effort estimates (Carlshamre 2002, Jorgensen 2007, Karlsson 1997).

Scope management is considered as one of the core functions of software release planning and a key activity for achieving economic benefits in product line development (Schmid 2002). Accurate release planning is vital for launching products within the optimal market window. And, this is a critical success factor for products in the MDRE domain (Sawyer 2000). Missing this window might cause both losses in sales and, additional cost for prolonged development and delayed promotion campaigns. However, making reliable release plans based on uncertain estimates (Karlsson 2007a) and frequently with features with dependencies to other features (Carlshamre 2001) is a challenge in itself. In addition, a rapidly changing market situation may force a project to consider new market requirements at a late project stage. Release planning is then a compromise where already committed features may need to be sacrificed at the expense of wasted effort (Wnuk 2009) of work already performed. The area of release planning is well researched and Svahnberg et al. (2010) reported on 24 strategic release planning models presented in academic papers intended for market-driven software development. Furthermore, Wohlin and Aurum (2005) investigated what is important when deciding to include

a software requirement in a project or release. Despite this, the understanding of the challenges related to scope management and their causes and effects is still low.

Scoping in agile development projects mainly involves three of the agile RE practices identified by Ramesh et al. (2007), namely *extreme prioritisation*, *constant planning* and *iterative RE*. High-level requirements are prioritised and the features with the highest market value are developed first. This approach ensures that if the project is delayed launch may still be possible since the most business-critical requirements will already be developed. Ramesh et al. (2007) also identified a number of benefits for companies applying these agile RE practices and challenges and varying impact on project risks. The identified benefits include an ability to adapt to changing prioritisation of requirements, as well as, a clearer understanding of what the customers want, thus reducing the need for major changes. On the other hand, agile RE practices were also found to include challenges in (1) correctly estimating and scheduling for the full project scope (which continuously changes), (2) a tendency to omit quality requirements and architectural issues (with the risk of serious and costly problems over time), and (3) constant reprioritisation of the requirements (with subsequent instability and waste).

3 The Case Company

The case company has around 5,000 employees and develops embedded systems for a global market using a product line approach (Pohl 2005). The projects in focus for this case study are technology investments into an evolving common code base of a product line (a.k.a. platform) on which multiple products are based. There are several consecutive releases of this platform where each release is the basis for one or more products. The products mainly reuse the platform's functionality and qualities, and contain very little product-specific software. A major platform release has a lead time of approximately two years from start to launch, and is focused on functionality growth and quality enhancements for a product portfolio. For such projects typically around 60-80 new features are added, for which approximately 700-1,000 system requirements are produced. These requirements are then implemented by 20-25 different development teams with, in total, around 40-80 developers per team assigned to different projects. The requirements legacy database amounts to a very complex and large set of requirements, at various abstraction levels, in the order of magnitude of 20,000 entities. This makes it an example of the VLSRE (very-large scale RE) context (Regnell 2008). Both the flow of new requirements (added to and removed from the scope of platform projects) and the scoping decisions associated with this flow may change frequently and rapidly. This exposes the project management to a series of unplanned, and often difficult, decisions where previous commitments have to be changed or cancelled.

3.1 Organisational Set-up

Several organisational units within the company are involved in the development. For this case study, the relevant units are: the *requirements unit* that manages the scope and the requirements; the *software unit* that develops the software for the platform; and the *product unit* that develops products based on the platform releases.

In addition, there is a *usability design unit* responsible for designing the user interface. Within each unit there are several groups of specialists divided by technical area. These specialists are responsible for the work in various stages of the development process. For this study, the most essential groups are the *requirements teams (RTs)* (part of the requirements unit) that, for a specific technical area, define the scope, and elicit and specify system requirements, and the *development teams (DTs)* (part of the software unit) that design, develop and maintain software for the (previously) defined requirements. Each RT has a team leader who manages the team. Another role belonging to the requirements unit is the *requirements architect* who is responsible for managing the overall scope, which includes coordinating the RTs. In the DTs there are several roles, namely

- *development team leader* who leads and plans the team's work for the implementation and maintenance phases
- *development team requirements coordinator* who leads the team's work during the requirements management and design phase, and coordinates the requirements with the RTs
- *developer* who designs, develops and maintains the software
- *tester* who verifies the software

The software unit also has a project management team consisting of (among others): *quality managers* who set the target quality levels and follow up on these, and *software project managers* who monitor and coordinate the DTs and interact with the requirements architects. For the product development unit in this study, we focus on the *system testing* task from the viewpoint of the functionality and quality of the platform produced by the software unit.

3.2 Phase-Based Process

The company used a stage-gate model. There were *milestones* (MS) for controlling and monitoring the project progress. In particular, there were four milestones for the requirements management and design (MS1-MS4) and three milestones for the implementation and maintenance (MS5-MS7). For each of these milestones, the project scope was updated and baselined. The milestone criteria were as follows:

- **MS1:** At the beginning of each project, RT roadmap documents were extracted to formulate a set of features for an upcoming platform project. A *feature* in this case is a concept of grouping requirements that constitute a new functional enhancement to the platform. At this stage, features contained a description sufficient for enabling estimation of its market value and implementation effort, both of which were obtained using a cost-value approach (Karlsson 1997). These values were the basis for initial scoping inclusion for each technical area when the features were reviewed, prioritised and approved. The initial scope was decided and baselined per RT, guided by a project directive and based on initial resource estimates given by the primary receiving (main) DT. The scope was then maintained in a document called *feature list* that was updated each week after a meeting of the *change control board (CCB).* The role of the CCB was to decide upon adding or removing features according to changes that occur.
- **MS2:** The features were refined to requirements and specified by the RTs, and assigned to their main DTs, responsible for designing and implementing the feature. The requirements were reviewed with the main DTs and were then

approved. Other (secondary) DTs that were also affected by the features were identified. The DTs make an effort estimate per feature for both main and secondary DT.

- **MS3:** The DTs had refined the system requirements and started designing the system. The set of secondary DTs were refined along with the effort estimates, and the scope was updated and baselined.
- **MS4:** The requirements refinement work and the system design were finished, and implementation plans were produced. The final scope was decided and agreed with the development resources, i.e. the software unit.
- **MS5:** All requirements had been developed and delivered to the platform.
- **MS6:** The software in the platform had been stabilised and prepared for customer testing.
- **MS7:** Customer-reported issues had been handled and the software updated. The software was ready to be released.

According to the company's process guidelines, the majority of the scoping work should have been done by MS2. The requirements were expressed using a domain-specific, natural language, and contained many special terms that required contextual knowledge to be understood. In the early phases, requirements contained a customer-oriented description while later being refined to detailed implementation requirements.

3.3 Agile Development Process

In order to meet the challenges of managing high requirements volatility, the case company was introducing a new development process at the time of this study. The size and complexity of the software development, including the usage of product lines, remained the same irrespective of the process used. The new process has been influenced by ideas and principles from the agile development processes eXtreme programming (XP) (Beck 1999) and Scrum (Schwaber 2002). The phase-based process was replaced by a continuous development model with a toll-gate structure for the software releases of the software product line (to allow for coordination with hardware and product projects, see P1 below). The responsibility for requirements management was transferred from the (previous) requirements unit, partly into the business unit and partly into the software unit. The following agile RE practices were being introduced:

- *one continuous scope & release-planning flow (P1)* The scope for all software releases is continuously planned and managed via one priority-based list (comparable to a product backlog). The business unit gathers and prioritises features from a business perspective. All new high-level requirements are continuously placed into this list and prioritised by the business unit. The software unit estimates the effort and potential delivery date for each feature based on priority and available software resource capacity. Development is planned and executed according to priority order. Planning and resource allocation is handled via one overall plan which contains all the resources of the software unit. The scope of the platform releases are synchronised with the product releases by gradual commitment to different parts of the scope. Critical scope is requested to be committed for specific product releases, while non-

critical features are assigned to product releases when they are implemented and ready to be integrated into the platform.

- *cross-functional development teams (P2)* that include a customer representative assigned by the business unit (comparable to the agile practice of customer proxy) have the full responsibility for defining detailed requirements, implementing and testing a feature (from the common priority-based list). Each new feature is developed by one cross-functional team specifically composed for that feature. The different software engineering disciplines and phases (e.g. requirements, design and test) are performed in an integrated fashion and within the same team. The team has the mandate to decide on changes within the value, effort and time frames assigned for the feature.
- *gradual & iterative detailing of requirements* (P3) The requirements are first defined at the high level (as features in the priority-based list) and then iteratively refined in the development teams into more detailed requirements as the design and implementation work progresses.
- *integrated requirements engineering* (P4) The requirements engineering tasks are integrated with the other development activities. The requirements are detailed, agreed and documented during design and development within the same cross-functional development team through close interaction between the customer representative and other team members, e.g. designers, developers and testers.
- *user stories & acceptance criteria* (P5) are used to formally document the requirements agreed for development. User stories define user goals and the acceptance criteria define detailed requirements to fulfill for a user story to be accepted as fully implemented. The acceptance criteria are to be covered by test cases.

This study mainly focuses on the situation prior to introducing the new agile way of working, i.e. for projects working as described in Section 3.2. The agile RE practices covered in this paper were defined in the company's internal development process at the time of the study. Practices P1 and P2 were being used in the projects, while P3 was partly implemented, and P4 and P5 were in the process of being implemented. Thus, it was not possible to investigate the full impact of the new agile RE practices at the time of this study. Nevertheless, the study investigates how these (new) practices are believed to affect the overscoping situation, i.e. which causes and root causes may be impacted by the agile RE practices and, thus, lead to reducing overscoping and its effects.

4 Research Method

The study was initiated due to a larger transition taking place within the case company and with the aim of understanding the differences between the scoping processes of the phase-based process and the new agile development process. Our previous research into scoping (Wnuk 2009) served as the basis for identifying research questions aimed at seeking a deeper understanding of overscoping as a phenomenon. In order to obtain detailed insight, an explanatory approach (Robson 2002) was taken and the study design was based on the specific company context

and the authors' pre-understanding. (These investigations can then be broadened in future studies.) Existing knowledge from literature was taken into account in interpretation and validation of the results.

A single-company explanatory case study (Robson 2002) was performed using mainly a qualitative research approach complemented by a quantitative method for some of the data gathering. Qualitative research is suitable for investigating a complex phenomenon (such as overscoping) in a real-life context where it exists (Myers 2002) (such as our case company). In this study, practitioners' perceptions of overscoping were studied through interviews where the verbalised thoughts of individuals with a range of different roles at the case company were captured (Myers 2002, Robson 2002).

The case study was performed in three phases (see Figure 1). In the first phase, the industrial experience of one of the authors was used to formulate a hypothesis concerning possible (assumed) causes of overscoping and (assumed) effects which may result from overscoping. This hypothesis was used as a starting point in creating the interview instrument (Bjarnason 2010b) for the interviews, which took place in the second phase of the study. In the third phase, the interview results were presented to (another) six practitioners from the same company and validated by using a questionnaire (see Section 4.3 for more details and Bjarnason 2010b for the validation questionnaire). This was done to reduce the risk of inappropriate (false) certainty of the correctness of the results (Robson 2002).

4.1 Phase 1: Pre-study, Hypothesis Generation

The purpose of the first phase of the study was to formulate a hypothesis on overscoping and prepare for the interviews. The experience of one of the authors (who has worked at the case company, with experience in several areas including coding, design, requirements engineering and process development) was used to identify possible (assumed) causes and effect of overscoping. In addition to these assumptions for the phase-based way of working, this author also identified the agile RE practices being introduced at the case company. These practices were assumed to impact one or more of the issues believed to cause overscoping in the phase-based process. If these assumptions were correct, applying the new practices should then result in reducing (or eliminating) the effects connected to those causes, and thus reduce (or eliminate) overscoping. In order to avoid selecting a set of assumptions biased only by one person, a series of brainstorming sessions on the hypothesis were conducted with the researchers involved in this study (i.e. the authors). The resulting (updated) hypothesis was then used as the main input in creating an interview study instrument (accessible online Bjarnason 2010b).

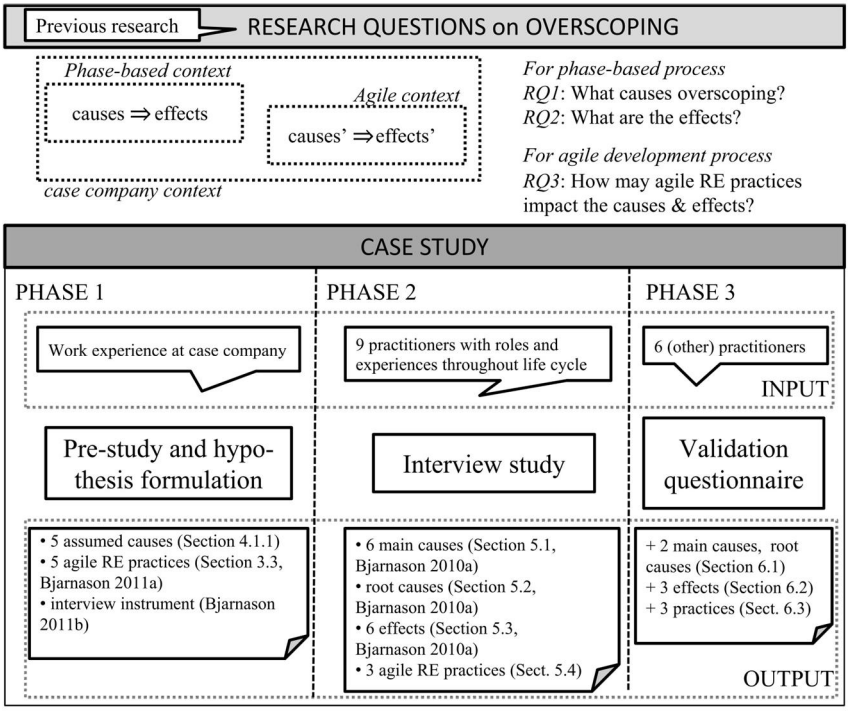


Figure 1. Overview of research method for study.

4.1.1 Formulated hypothesis

The hypothesis formulated for this study is that overscoping is caused by a number of factors, and that by addressing one or more of these factors, e.g. through agile RE practices, the phenomenon of overscoping may be alleviated, or even eliminated. The following five factors were identified as assumed causes for overscoping in phase one:

- *continuous requirements inflow via multiple channels* (C1) was assumed to cause overscoping by the many inflows increasing the difficulty of defining a realistic scope for multiple parallel projects. Requirements continuously arrive from the market, as well as, from internal stakeholders. This inflow was managed by batching those requests into one or two projects per year. It was a challenge to manage the execution of multiple parallel projects, while handling requests for new features and requirements, as well as, requests for changes to the agreed project scope.
- *no overview of software resource availability* (C2) was assumed to cause overscoping due to the challenge of balancing the size of the total scope for several (parallel) development projects against the (same) set of development resources. The resource allocation for the software development unit was handled at the DT level, i.e. there was no total overview of the load and available capacity of all the development resources of the software unit.
- *low DT involvement in early phases* (C3) was assumed to contribute to defining unrealistic and unclear requirements in the early phases, that are later deemed

too costly or even impossible to implement, thus causing overscoping. The development teams were not very involved in the early project phases (MS1-MS4) with providing cost estimates and feedback during requirements writing.

- *requirements not agreed with DT* (C4) was assumed to cause overscoping due to not ensuring that the suggested scope was feasible and understandable. The requirements specification was not always agreed with the development teams at the handover point (MS2). Even if there was a formal review by DTs, we assumed that there was a low level of commitment from DTs. Furthermore, this low level of agreement was assumed to lead to low DT motivation to fulfil the requirements defined by the RTs.
- *detailed requirements specification is produced upfront* (C5) by the requirements teams by MS2 before the design starts was assumed to cause overscoping by limiting the room for negotiating requirements that could enable a more efficient design and realistic development plan. Furthermore, time and cost overhead for managing such changes was also assumed to contribute to overscoping.

4.2 Phase 2: Interview Study at Case Company

In phase two, semi-structured interviews with a high degree of open discussion between the interviewer and the interviewee were held. The hypothesis provided a framework that helped to discuss, explore and enrich the understanding of this complex phenomenon. To avoid imposing this hypothesis on the interviewees, the discussion both on overscoping in general and on the detailed causes and effect was always started with an open ended question. In addition, the interviewees were asked to talk freely about the roles and phases she had experience from at the beginning of the interviews. In order to separate between the situation with the phase-based process and with the new agile RE practices, the impact of the new practices was discussed specifically in a (separate) section at the end of the interviews.

Our aim was to cover the whole process from requirements definition through development (design, implementation and testing) to the resulting end product (quality assurance, product projects), mainly for the phase-based process. This was achieved by selecting people to cover all the relevant organisational units (see Section 3) for interviews and thereby catch a range of perspectives on the phenomenon of overscoping. Nine people in total were selected to be interviewed. Two of the interviewees with identical roles requested to have their interviews together. The roles, organisational belongings, and relevant experience of the interviewed persons within the case company for the phase-based process can be found in Table 1. We have used a coding for the interviewees that also includes their organisational belonging. For example, interviewees belonging to the requirements unit are tagged with a letter R, belonging to product unit with a letter P and belonging to software unit with a letter S.

Table 1. Interviewee roles (f phase-based process), organisational belonging and length of experience for each role within company (see Section 3.1).

Code	Organisational unit	Role (s) within company	Years in role
Ra	Requirements	RT leader	5
Rb	Requirements	RT leader	2
Rc	Requirements	Requirements architect	3
Pd	Product	System test manager	7
Se	Software	Tester	3
Sf	Software	Software project manager	2
		DT leader	2
		Developer	2
Sg	Software	Quality manager	3
Sh	Software	DT requirements coordinator	1
		Developer	2
		DT leader	1
Si	Software	DT requirements coordinator	7

The interviews were scheduled for 90 minutes each with the possibility to reduce time or prolong it. All interviews were recorded and transcribed, and the transcripts sent back to the interviewees for validation. For each interview, the transcript was 7-10 pages long and contained in average 3 900 words. Transcription speed varied from 3-7 times of recorded interview time. The coding and analysis was done in MS Excel. The underlying section structure of interview instrument, i.e. causes, effects and agile RE practices, were numbered and used to categorise the views of the interviewees. For each interview, the transcribed chunks of text were placed within the relevant sections and, if so needed, copied to multiple sections. Relationships between different categories, as well as, the level of agreement on causes, effects and agile RE practices were noted in specific columns. The viewpoints of the two practitioners interviewed together (interviewees Ra and Rb) were separated into different columns in order to allow reporting their individual responses.

4.3 Phase 3: Result Validation Questionnaire

To further strengthen the validity of the results from the interviews a set of six (additional) practitioners at the case company was selected in phase three, see Table 2. To ensure that these six practitioners understood the results correctly and in a uniform way, the interview results were presented to them. During the meeting the participants could ask for clarifications and comment on the results, especially when they disagreed or had other different or additional viewpoints. In order to gather their views on the results in a uniform and non-public way, the participants were asked to fill out a questionnaire (available online at Bjarnason 2010b) stating to which degree they agreed to the results and if additional relevant items had been missed. Due to limited availability of the participants a total of three such sessions were held. Each session was scheduled for 90 minutes with the possibility to extend or decrease the time as needed. The results from the questionnaire can be found in Section 6.

Table 2. Questionnaire respondents: roles and organisational belonging (for phase-based process), and length of experience within company. See Section 3.1 for descriptions of organisational units and roles.

Organisational unit	Role(s)	Years in company
Software	Software project manager, DT leader	4
Software	Tester	7
Software	DT reqs coordinator, DT leader	5
Requirements	Requirements architect	5
Requirements	RT leader	13
Product	System test manager	15

5 Interview Results

The causes and effects of overscoping derived from the interviews performed in phase two of the study (see Figure 1) are outlined in Figure 2 and described in the following sections. Section 5.1 covers the main causes for overscoping, while the root causes are reported in Section 5.2 and the effects in Section 5.3. The findings from the interviews concerning how the agile RE practices may address overscoping are described in Section 5.4. The outcome of the validation questionnaire (phase 3) on these results is reported in Section 6.

5.1 Causes of Overscoping (RQ1)

The viewpoint of each interviewee concerning the causes of overscoping was categorised and matched against the hypothesis regarding the assumed causes of overscoping (C1-C5, see Section 4.1.1). In addition, five of the eight interviewees were found to describe a sixth main cause for overscoping, namely C6 *unclear vision of overall goal.* A lack of (clearly communicated) strategy and overall goals and business directions for software development led to un-clarities concerning the intended direction of both software roadmaps and product strategies, as well as, unclear business priority of project scope. The interviewees described how this cause (C6) led to scope being proposed primarily from a technology aspect, rather than from a business perspective, and without an (agreed) unified priority. Instead, most of the scope of a project was claimed to be critical and non-negotiable.

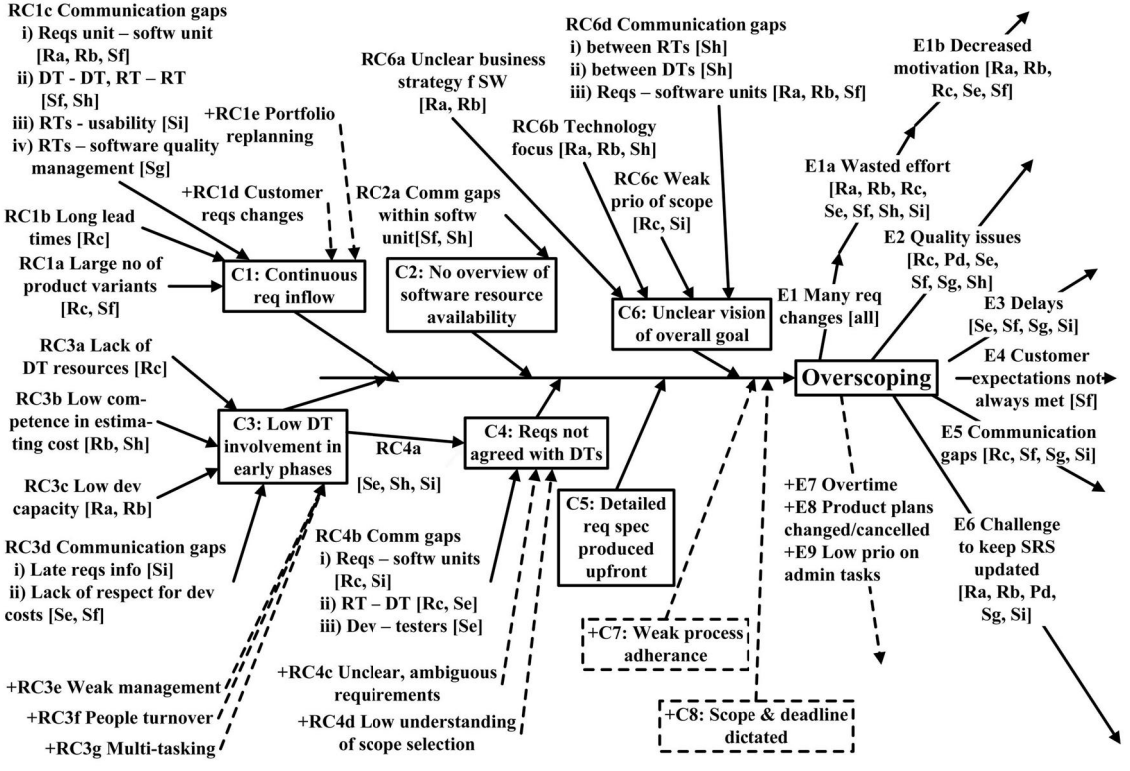


Figure 2. Overview of all found causes (C), root causes (RC) and effects (E) of overscoping. Items derived from questionnaire noted with + and dashed lines. Interviewee code (see Section 4.2) noted within brackets.

The interviewee results around the main causes of overscoping are shown in Table 3. The opinions of the interviewees have been classified in the following way:

- *Experienced:* the cause (both its occurrence and its impact on overscoping) is experienced and was mentioned without prompting.
- *Agreed:* either the cause (occurrence and impact) was not directly mentioned, but derived or agreed after direct question, or when interviewee has no direct experience, but had observed it or heard about it from others.
- *Partly agreed*: partly *Experienced* or partly *Agreed*.
- *Disagreed:* does not agree to the cause, either its occurrence, or that it caused overscoping.
- *Not mentioned*: even though within expected experience for role.
- *NA:* not within the expected experience for the role (according to the process).

All interviewees had *Experienced* or *Agreed* to overscoping as a challenge, and a majority had *Experienced* or *Agreed* to causes 1-3. No interviewees *Disagreed* to any of the causes, though causes 4 and 5 both had less than a majority of *Experienced* or *Agreed* interviewees. Causes 4-5 were *Not mentioned* by all, while cause 6, which was derived from 5 of the interviewees, was *Not mentioned* by the others.

The entries marked *NA* (Not Applicable) indicate that the interviewee in her role was not expected to have experience of that cause. The system test manager (Pd) and the quality assurance manager (Sg) were classified as NA for C2, C3 and C4 since they merely observed the requirements flow from their management-level positions and were not directly involved in the early phases of the projects. In addition, Sg was also classified as NA for C5 due to lack of contact with the SRS. Furthermore, the software tester (Se), who had no insight into project planning, was categorised as *NA* for the causes C1 and C2.

Table 3. For each identified main causes of overscoping, the number of interviewees per response category (see Section 5.1) and organisational unit (Requirements etc, see Section 3.1).

	Over-scoping as a challenge			C1 Continuos requirem inflow			C2 No overview of software resources			C3 Low DT involvem in early phases			C4 Reqs not agreed with DTs			C5 Detailed reqs spec produced upfront			C6 Unclear vision of overall goal		
	Reqs	Software	Product	Reqs	Software	Product	Reqs	Software	Product	Reqs	Software	Product	Reqs	Software	Product	Reqs	Software	Product	Reqs	Software	Product
Experienced	2	5	1	1	3	1	1	2		3	1		1	1			2		3	2	
Agreed	1			2			2				1			1		1	1				
Partly agreed					1			1			2			2		1					
Disagreed																					
Not mentioned													2			1	2			3	1
NA					1			2	1		1	1		1	1			1			

For all assumed causes there were some counts of *Partly agreed*, namely

- *continuous requirements inflow via multiple channels* (C1).
  The quality manager (Sg) mentioned the continuous inflow of requirement changes after setting the scope as causing overscoping, but no root causes prior to this milestone, and is therefore classified as 'Partly agreed'.
- *no overview of software resource availability* (C2).
  One of the DT requirements coordinators (Si) is noted as 'Partly agreed' to this cause, due to believing that a better overview of available resources would not alleviate the overscoping to any greater extent. In contrast, another interviewee (Sf) saw this as a strong cause for overscoping; 'There was no control of what people were working with. There were different DT leaders who just sent out [tasks].'
- *low DT involvement in early phases* (C3).
  Both DT requirements coordinators (Sh, Si) were categorised as 'Partly agree' since the involvement from the rest of the DT including producing cost estimates was low, even though they personally had experienced good cooperation with the RT leaders during MS1-MS2. This lack of involvement was seen by the DT tester (Se) as leading to an unrealistically large scope being specified, 'The full view of requirements would be improved by including input from more roles, and a more realistic scope could be identified earlier on.'
- *requirements not agreed with DT* (C4).
  The DT requirements coordinators (Sh, Si) believed that the requirements were understood and agreed with the DT at MS2, though the DT did not commit to implementing them at that point. One of them (Sh) mentioned that the system requirements specification was primarily agreed with the DT requirements coordinators and not with developers and testers in the DT.
- *detailed requirements specification produced upfront* (C5).
  One of the RT leaders (Rb) had an agile way of working and did not produce a detailed requirements specification upfront, but instead regularly and directly interacted with the DT. This increased insight into the DT enabled a more flexible discussion around scope and detailed requirements, led to overscoping being experienced as a more manageable challenge by Rb. The other RT leader (Ra, interviewed together with Rb) did not mention C5 as causing overscoping, but agreed to Rb's conclusions and was noted as 'Partly agreed'. Ra had the opposite experience, i.e. of producing and relying on a requirements specification, and then not staying in touch with the DT during the later phases of development (after MS2) who then developed software that was usually different from what was specified in the SRS. One of the DT interviewees (Sf) believed that the (wasted) effort of producing and agreeing to detailed requirements upfront (for features that were later descoped) increased the overscoping since it hindered those resources from working on viable features. Another interviewee (Sh) said: 'At this point [MS2] we [DT] approved a lot of things, because we liked what they [RT] wrote here and we really wanted that functionality then we [DT] started to over commit.'

5.2 Root Cause Analysis (RQ1)

To provide a deeper understanding the interviewees were asked to describe what may be triggering overscoping, i.e. the root causes of overscoping. These root causes have been grouped according to the main cause (C1-C6, outlined in Sections 4.1 and 5.1) that they affect. A full picture of the cause-effects relationships for overscoping identified through this study is depicted in Figure 2. The results around root causes from both the interviews and from the questionnaire are also summarised in Table 4.

Table 4. Summary of all identified causes and root causes of overscoping: Number of responses for interviewees (see Section 5 for details) and for questionnaire responses per level of agreement (see Section 6 for details). Additional items from questionnaire responses are marked with +.

	Mentioned as causes / root causes by # of interviewees (9 in total)	Nr of questionnaire responses (6 in total)				
		Experienced	Agree	Partly agree	Disagree	Don't know
Overscoping (as a challenge)	9	6				
C1: Continuous requiremts inflow via multiple channels	8	4	2			
a) Large number of product variants	2	3	1	2		
b) Long lead times	1	4	2			
c) Communication gaps	6	3	1	1		1
i. between reqs & software unit	3	3	2		1	
ii. between RT-RT & DT-DT	2	3	2			1
iii. between RT and usability design	1	2	1	1		2
iv. between RT and software quality managers	1		2			4
+d) Customer requirements changes (many & late)		3				
+e) Product portfolio re-planning		1				
C2: No overview of software resource availability	6	2	3		1	
a) Communication gaps within software unit	2	1	2	1	1	1
C3: Low dev. team involvement in early phases	7	1	2	2	1	
a) Lack of DT resources for pre-development work	1	2	2	2		
b) Low competence in estimating cost	2	2	1	3		
c) Low development capacity	2	1	1	2	1	1
d) Communication gaps	3					
i. Late reqs information to DT	1	2		2	1	1
ii. Lack of respect/understanding of develpmnt costs	2	2	3		1	
+e) Weak leadership incl. ineffective communication		1				
+f) Change of people during the project		1				
+g) Multi-tasking		1				
C4: Reqs not agreed with development teams	5	2	2	2		
a) Low DT involvement in early phases (C3)	3	2	2	1	1	
b) Communication gaps	3	1	2	2	1	
i. between reqs and software units	2	1	2		1	2
ii. between RT and DT	2	1	1	2	1	1
iii. betw. developers and testers	1	1	1	1	2	1
+c) Unclear and ambiguous reqs		3				

+d) Low understanding of why particular scope is selected		1				
C5: Detailed reqs spec.produced upfront	5	1	3	1	1	
C6: Unclear vision of overall goal	5	4	1		1	
a) Unclear business strategy f software development	2	3	2	1		
b) Technology focus when scope set	3	3	2		1	
c) Weak priority of scope	2	3	2	1		
d) Communication gaps		2	3			1
i. between RTs	1	1	3			2
ii. between DTs	1	2	2			2
iii. bt reqs and software units	3	1	3		1	1
Additional (new) causes (from questionnaire responses)						
+C7 Weak process adherence		1				
+C8 Overall scope & deadline dictated by management		1				

5.2.1 Root causes of C1: Continuous requirements inflow via multiple channels

A number of requirement sources besides the regular requirement flow (defined by the RTs) were mentioned as contributing to the continuous inflow. These include: requirements produced by defining many different product variants in the product line (RC1a); and, many late new market requirements and changes incurred by the long project lead times (RC1b) compared to rate of requirements change. Furthermore, communication gaps (RC1c) were mentioned as causing additional requirements inflow through-out the project life-cycle. These consist of communication gaps between the requirements unit and the software unit (RC1ci) which resulted in the software unit preferring to work according to their own software-internal roadmap containing a large amount of requirements not agreed with the requirements unit. Communication gaps between technical areas, both for RTs and for DTs, (RC1cii) led to indirect requirements between DTs being discovered after the initial scope selection at MS2, which greatly increased the amount of implementation required. The impact of these indirect requirements was especially large for DTs responsible for service-layer functionality like display frameworks and communication protocols. Furthermore, communication gaps between usability design and the RTs (RC1ciii) resulted in additional functional requirements appearing in usability design specification, sometimes in conflict with RT requirements. And, finally, due to lack of communication between the software quality managers and the requirements unit (RC1civ), requirements on quality aspects were not defined and prioritised together with the RT requirements, but managed separately in a later phase.

5.2.2 Root causes of C2: No overview of software resource availability

The lack of overview of available software development resources was believed to be a consequence of communication gaps within the software unit and between the DTs (RC2a). The organisational structures and the high scope pressure were seen to result in each DT focusing on their own areas rather than striving for cooperation and good communication with other DTs. One interviewee described that enabling DTs to coordinate their plans had the effect of improving the scoping situation by

increasing the delivery rate and efficiency, 'We tried to solve the overscoping by enabling the DTs to co-plan and deliver incrementally. This resulted in more deliveries and increased efficiency.' (Sf)

5.2.3 Root causes of C3: Low DT involvement in early phases

Several interviewees described that software resources were rarely available in early project phases (RC3a) due to development and maintenance work for previous projects. Rc said: 'by MS2, but it was hard to get [DT] resources. That probably was the problem.' In addition, weak and incorrect cost estimations (RC3b) were mentioned as leading to including too much into the project scope. In contrast, low development capacity of the software unit (RC3c) caused by bad architecture was believed by the two RT leaders to be the main reason for overscoping. Furthermore, gaps in the communication (RC3d) between the requirements unit and the software unit were mentioned as causing low DT involvement. For example, interviewees mentioned that early DT involvement was often postponed due to a lack of understanding within the software unit for the importance of this work. However, the opposite was also mentioned, namely that the DTs received requirements information too late (RC3di) which then resulted in extending the project scope without realistic plans. Similarly, the cost estimates for both development and testing were not always respected (RC3dii). In contrast, close cooperation between the RTs and the DTs were experienced (by Rc) to lead to an early uncovering of problems, thereby enabling definition of more stable requirements that were then successfully implemented.

5.2.4 Root causes of C4: Requirements not agreed with DTs

Low DT involvement in the early phases (C3, RC4a) was seen as leading to weak agreement and commitment to the requirements, by all three interviewees with experience from planning DT work (Se, Sh, Si). The interviewees connected the level of requirements agreement with the level of communication around requirements (RC4b), i.e. RTs and DTs that communicated well also tended to have a mutual understanding and agreement of the requirements. Due to variations in communication between teams, the view on C4 varied between interviewees (see Section 5.1). Even so, one interviewee (Sh) who had experienced good cooperation with the RT mentioned that the different organisational belongings (RC4bi) caused timing issues due to different priorities for different units. In addition, communication gaps between RTs and DTs (RC4bii) including no contact between testers and RT leaders were caused by physical and organisational distances and resulted in weak DT agreement on the requirements. Weak communication on requirements and design between developers and testers (RC4biii) was also mentioned (by Se) as causing weak requirements agreement.

5.2.5 Root causes of C5: Detailed req spec produced upfront

The phase-based process defined that a requirements specification should be produced by MS2, therefore no further root causes have been identified for this cause.

5.2.6 Root causes of C6: Unclear vision of overall goal

The RT leaders (Ra and Rb) described that the lack of clear business strategy (RC6a) and vision that could guide them in defining a roadmap resulted in proposing a project scope from a pure technology standpoint (RC6b). A weak and un-unified business priority (RC6c) of the proposed scope (almost everything was 'critical') was described (by Si) as pushing the DTs to commit to unrealistic project plans. In addition, Rc mentioned that the lack of unified priority hindered the project management from effectively addressing the overscoping. Furthermore, several communication gaps (RC6d) were seen to contribute to this cause. Weak communication both between RTs (RC6di) and between DTs (RC6dii) were described by Rc as resulting in weak scope coordination between functional areas, as well as, conflicts and lack of clarity concerning the overall goal. Finally, both RT leaders described that communication gaps and low common understanding between the requirements unit and the software unit (RC6diii) of the overall goal resulted in the project scope being decided to a large extent by the DTs , and not (as the process stated) by the RTs.

5.3 Effects of Overscoping (RQ2)

The interviews uncovered the following six main effects of overscoping (marked as E1 to E6, see Figure 2).

5.3.1 Many req changes after the project scope is set (E1)

All interviewees had experienced that overscoping caused requirement changes to take place after the project scope was set (at MS2). As the projects proceeded and the overload was uncovered large amounts of features were removed from scope (descoped). The phenomena was so common that the phrases 'overscoping' and 'descoping' have become part of company vocabulary. This descoping of already started features was a waste (E1a) of both RT and DT effort and led to frustration and decreased motivation (E1b) to work with new requirements. As interviewee Sh said: 'There are many things that you as a tester or developer have spent time on that never resulted in anything. And that isn't very fun. There is a lot of overtime that has been wasted.' However, the many requirement changes were experienced by Pd as having only minor impact on the system testing. They merely adjusted the test plans, and rarely wasted any effort due to this effect.

5.3.2 Quality issues (E2)

All interviewed practitioners involved after MS4 (when development started, Rc, Pd, Se, Sf, Sg, Sh) mentioned that software quality was negatively affected by overscoping both due to the high workload and due to the many requirement changes. The software quality manager Sg expressed, 'If you get too much scope, you get quality problems later on and you haven't got the energy to deal with them.' Similarly, interviewee Pd said: 'When you have a lot going on at the same time, everything isn't finished at the same time and you get a product with lower quality.' Furthermore, the lack of respect for development costs (C3dii) in the earlier phases was mentioned by the software tester (Se) to contribute to insufficient testing and subsequent quality issues.

5.3.3 Delays (E3)

The overscoping and subsequent overloading of the DTs was described by several practitioners as resulting in delayed deliveries being the norm rather than the exception. In addition, overscoped DTs were often forced to commit to customer-critical requests and changes which in turn resulted in even more delays and quality issues (E2). One DT interviewee (Sf) stated that 'our team was always loaded to 100% at MS4, which was too much since there were always customer requests later on that we had to handle. That meant that we were forced to deliver functionality with lower quality or late.' The same situation was described by the quality manager (Sg) who said: 'Even if we decided on a scope for MS4, there were extremely many changes underway, so we were never ready by MS5, as we had said, but were delayed.'

5.3.4 Customer expectations are not always met (E4)

Overscoping was mentioned by a few interviewees as resulting in sometimes failing to meet customer expectations. For example, customers sometimes file change requests for features that had previously been removed due to overscoping. In addition, overscoping caused by requiring a large number of products (RC1a) with different display sizes and formats was experienced by interviewee Sf as resulting in releasing products with faulty software, e.g. misplaced icons.

5.3.5 Communication gaps (E5)

Overscoping and overloading an organisation was described as leading to several communication gaps; between the requirements and software units; within the software unit itself, between DTs (Sg, Si) and between DTs and software project managers (Sf); and between the software and the product unit. For example, the many descoped features (E1) and wasted effort (E1a) resulted in distrust between the requirements unit and the software unit, so much so that the software unit defined their own internal roadmap without coordinating this with the requirements unit. Furthermore, invalid error reports filed by the system testers based on an unreliable SRS (caused by E1-E6) caused an increase both in work load and in frustration at the software unit and, consequently friction and widened communication gaps between these units.

5.3.6 Challenge to keep the SRS updated (E6)

The situation caused by overscoping, with a high workload and many late requirement changes (E1), increased the challenge of keeping the SRS updated. The practitioners mentioned that in an overscoping situation the task of updating the SRS was given low priority (partly caused by E1b). Furthermore, the amount of required updates both for changed and descoped requirements was increased (Ra, Rb, Pd, Sg, Si) by producing the requirements upfront (C5) with a low level of DT agreement (C4). The RT leaders (Ra, Rb) had also experienced that many requirement-related changes were made during development without informing the RTs (or the system testers), many of which might have been a result of insufficient DT involvement in the early phases (C3).

5.4 Impact of Agile RE Practices (RQ3)

The general opinion of the interviewees on the situation after introducing the agile RE practices (see Section 3.3) is that even though some overscoping is still experienced, it is a more manageable challenge than with the previous phase-based process. For example, there is less descoping and most of the features worked on by the software unit now continue until completion (Si). Interviewee Sg said: 'We still have overscoping in all projects. But, it is more controlled now and easier to remove things without having done too much work.' Many of the interviewees stated that in theory the agile RE practices address overscoping, but that these practices also incur a number of new challenges. The following practices were mentioned by the interviewees as impacting some of the causes and/or root causes of overscoping.

5.4.1 One continuous scope & release-planning flow (P1)

This practice (which was implemented at the time of the interviews) was seen to address the root cause *weak prioritisation of scope* (RC6c, mentioned by Rc, Pd, Sg, Sh) and the causes *continuous requirements inflow via multiple channels* (C1, mentioned by Se, Sf) and *no overview of software resource availability* (C2, mentioned by Sf, Sg), by enforcing that all scope and development resources are managed through a uniformly prioritised list.

5.4.2 Cross-functional development teams (P2)

This practice (which was implemented at the time of the interviews) was seen to address several communication gaps, and, thus, impact causes C1-C4 by closing the gaps (identified as root causes) between RTs and DTs and between different functional areas. This practice was also believed to impact C5 (*detailed requirements specification produced upfront*) since detailing of requirements is now handled within the development teams together with the customer representative. Interviewee Sf said: 'It is an advantage that they [the team] sit together and can work undisturbed, and there is no us-and-them, but it is us. And when they work with requirements the whole group is involved and handshakes them.'

5.4.3 Gradual & iterative detailing of requirements (P3)

This practice (which was partly implemented at the time of the interviews) was mentioned as directly impacting the cause C5 (*detailed SRS produced upfront*). Furthermore, this practice was also seen by Sf and Sg to reduce both the lead time for each high-level requirement (RC1b) and the amount of *changes after project scope is set* (E1) and, thus also reduce the amount of *wasted effort* (E1a, also mentioned by Ra, Rb).

6 Validation Questionnaire on Interview Results

Overscoping was further investigated through the validation questionnaires (Bjarnason 2010b), see Table 4. Each of the six respondents noted her level of agreement by using the following notation:

- *Experienced:* I have experienced this (item and connection) to be valid
- *Agree:* I agree to this, but have not experienced it personally
- *Partly agree:* I agree to part, but not all, of this
- *Disagree:* I do not agree
- *Don't know*: I have no knowledge of this item or its impact

6.1 Causes and Root Causes (RQ1)

A majority of the questionnaire respondents confirmed (i.e. *Experienced* or *Agreed* to) all main causes as contributing to overscoping, except C3 (low DT involvement) for which there was also one *Disagree* response. Causes C2, C3, C5 and C6 each had one count of *Disagree* from respondents with experience from the requirements unit. Two additional main causes were given by two respondents, namely *weak processes adherence* (+C7) and *dictation of scope and deadlines from management* (+C9). Furthermore, some additional root causes were given for C1, C3 and C4. For C3, two alternative root causes were given, namely turn-over of DT members as the project progressed (RC3f) and assigning the same resources to multiple parallel projects (RC3g). For C4 (*requirements not agreed with DT*) three respondents stated that this was caused by unclear and ambiguous requirements (RC4c), while one respondents suggested that DTs often lacked insight into why certain features and requirements were important, which is related to C6 (*unclear vision of overall goal)*. All responses from the validation questionnaire on causes and root causes can be found in Table 4.

The impact of each main cause on overscoping was gauged by asking the questionnaire respondents to distribute 100 points over all causes according to the extent of their impact (see Table 5) C1 got the highest score in total and all six respondents, thereby indicating that the continuous requirements inflow was a main cause of overscoping. The second highest total score was given to C6 (*unclear vision of overall goal*), which all the participants from the software unit graded with 30-60, while the other participants graded this with 0 or 30. Causes C4, C5, +C6 and +C7 were seen as having a minor or no impact on the overscoping situation.

Table 5. The total number of points reflecting the impact of each cause on overscoping. Each questionnaire respondent distributed 100 points. The columns show the number of points per responder (organisational belonging given in header, see Section 3.1).

	Total impact	Software	Software	Software	Requirements	Requirements	Product
C1: Continuous reqs inflow via multiple channels	275	20	20	15	50	100	70
C2: No overview of software resource availability	60	10	20		20		10
C3: Low DT involvement in early phases	80		10	50	20		
C4: Reqs not agreed with DTs	10	5	5				
C5: Detailed reqs specification produced upfront	15	5	5	5			
C6: Unclear vision of overall goal	140	60	40	30	10		
+C7: Weak process adherence	0						
+C8: Overall scope and deadline dictated from top	20						20

6.2 Effects of Overscoping (RQ2)

In large, the questionnaire respondents had experienced or agreed to all the effects of overscoping identified from the interviews. The respondent from the product unit had no view on E5 or E6, while the requirements architect partly agreed E5. In addition, the respondents mentioned the following effects of overscoping: overtime (+E7); changed and sometimes cancelled product plans (+E8); low prioritisation of administrative tasks (+E9). The full questionnaire response on effects is shown in Table 6.

In addition to stating the level of agreement to the identified effects of overscoping, the respondents were asked to grade their impact. The following notation was used:

- *Critical:* Company or customer level
- *Major*: Project or unit level
- *Medium*: Team level
- *Minor*: Individual level
- *None*: No impact

All the effects identified from the interviews were seen as having an impact. All effects except E5 (communication gaps) were seen as having major or critical impact by a majority of the participants. There were two counts of minor impact: one for E6 (keeping SRS updated) and one for +E7 (overtime).

Table 6. Number of questionnaire responses on the effects of Overscoping per level of agreement (notation described in Section 6) and per impact category (notation described in Section 6.2). Additional items derived from questionnaire marked with +.

	Mentioned by # interviewees (9 in total)	Questionnaire responses (6 in total)									
		Agreement					Impact				
		Experienced	Agree	Partly agree	Disagree	Don't know	Critical	Major	Medium	Minor	None
E1: Many req changes after scope is set	9	5	1				4	2			
a) Wasted effort	7	5	1				3	3			
b) Decreased motivation	5	4	2				3	2	1		
E2: Quality issues	6	6					5	1			
E3: Delays	4	6					5	1			
E4: Customer expectat. not always met	1	4	2				5	1			
E5: Communication gaps	4	2	1	1		1	2	1	3		
E6: Keep SRS updated	5	1	4			1		5		1	
+E7: Overtime		3					1		1	1	
+E8: Changed/cancelled product plans		1					1				
+E9: Administrative tasks not always performed		1					1				

6.3 Impact of Agile RE Practices (RQ3)

The questionnaire respondents mostly agreed to the three identified agile RE practices as impacting the challenge of overscoping, either through their own experience or by believing the practice should work in theory. Furthermore, some additional practices were mentioned as impacting overscoping: (+P4) clearer company vision (i.e. directly addressing C6), (+P5) open source development (limiting C1 by restricting what the customer can reasonably expect when large parts of the software are outside of company control) and (+P6) incremental deliveries (shorter cycles facilitate scope size control for each cycle). Table 7 contains the questionnaire responses on the impact of the agile RE practices on overscoping.

Finally, the respondents had all experienced, agreed or partly agreed that overscoping was still a challenge for the case company. The new agile process and practices are seen to, at least partly, address the situation and provided ways to better manage and control the extent of overscoping and its effects. The practitioners' responses concerning the current situation are shown in Table 8.

Table 7. Number of questionnaire responses on the impact of Agile RE Practices on overscoping per level of agreement (notation described in Section 6). Additional practices identified through questionnaire responses are marked with +.

	Experienced	Agree (in theory)	Partly agree	Disagree	Don't know
P1: One continuous scope & release-planning flow	2	4			
P2: Cross-functional development teams	3	2			1
P3: Gradual & iterative detailing of requirements	2	2	2		
+P4: Company vision	1				
+P5: Open source development	1				
+P6: Incremental deliveries	1				

Table 8. Number of questionnaire responses per agreement category (described in Section 6) on the current situation at the case company with Agile RE practices, as compared to when using phase-based process.

	Experienced	Agree	Partly Agree	Disagree	Don't know
Overscoping is still a challenge	3	1	2		
There is less overscoping now	1	1	3	1	
Overscoping is more manageable now	1	3	1		1

7 Interpretation and Discussion

The results of this study corroborate that overscoping is a complex and serious risk for software project management (Boehm 1989, DeMarco 2003, Legodi 2010) both for phase-based and for agile development processes. In addition, the results show that communication issues have a major impact on overscoping. This complements the work by Sangwan et al. (2006) and Konrad et al. (2008) who mentioned that weak communication can cause project failures in large-scale development and global software engineering. Moreover, our results extend the lists of effects of weak coordination proposed by Sangwan et al. (2006) (long delays, leave teams idle and cause quality issues) by adding overscoping. Further research is needed to fully identify and address the factors involved. The results are discussed and related to other research in further detail, per research question, in Sections 7.1 (RQ1), 7.2 (RQ2) and 7.3 (RQ3). Finally, the limitations of this study and threats to validity of the results are discussed in Section 7.4.

7.1 Causes of Overscoping (RQ1)

Our results indicate that overscoping is caused by a number of causes and root causes. These causes mainly originate from the nature of the MDRE context in which the company operates, but are also due to issues concerning organisational culture and structures, and communication. This was further highlighted by interviewees describing the additional cause C6 (unclear vision of overall goal) and two questionnaire respondents mentioning additional causes connected to lack of respect for the decision- and development process, i.e. C7 and C8. In contrast, practitioners with experience of good cooperation and well-communicating teams described overscoping as a less serious and more manageable challenge. This may explain all the *Disagree* questionnaire responses but one (i.e. C5).

We interpret the results around the six causes of overscoping identified through the interviews (see Section 5.1 and Figure 2) as follows.

7.1.1 Continuous requiremts inflow from multiple channels (C1)

We interpret the homogeneity of the interview and questionnaire results (see Table 3 and Table 5) to mean that a large and uncontrollable inflow of requirements has the potential to cause overscoping when not managed and balanced against the amount of available capacity. This cause was also been identified by Regnell and Brinkkemper (2005) and Karlsson at el. (2007a) as one of the challenges of MDRE. In addition to corroborating this challenge, our work also identifies that this continuous inflow of requirements can cause overscoping. The importance and seriousness of this factor are indicated by this cause scoring the highest total impact factor in the questionnaire (see Table 5). The extent to which this cause affects companies that operate in the bespoke requirements engineering context (Regnell 2005) requires further research.

Our study also reveals that the inflow of requirements can be further increased by scope creep at the software management level through a software-internal roadmap (RC1ci, see Section 5.2). In effect, this hindered resources from being available for managing new customer requirements. Similar results have been reported by Konrad et al. (2008) who found that scope creep can result in problems with meeting customer expectations, i.e. effect E4 (see Section 5.3). Konrad et al. (2008) propose addressing scope creep by increased understanding and traceability of customer requirements, and by creating an effective hierarchical CCB structure. The impact of these methods on overscoping remains to be evaluated.

7.1.2 No overview of software resource availability (C2)

The majority of our responders (six of nine interviewees and five of six questionnaire respondents) had experienced or agreed to the lack of overview of available resources being a cause of overscoping. However, the questionnaire results suggest that the impact of this cause is not as critical as cause C1. This result is surprising, when considering the importance of management of the daily workload including coordination of tasks and activities reported by, e.g. Philips et al. (2002). The contrasting opinions of low development capacity (RC3c, held by RT leaders) and low respect for development costs (RCdii, held by DT roles) is interesting. This difference can be interpreted as a low understanding of each other's viewpoint

around cost and an indication that this viewpoint is dependent on role (related to Jorgensen 2007). If the development capacity really is low is a different issue. Finally, this cause specifically includes the lack of overview, or awareness of the total load on the resources. To the best of our knowledge, this issue has not been empirically investigated. Rather software cost estimation research (Jorgensen 2007) mainly focuses on effort estimation and on optimising resource assignment (Lixin 2008).

7.1.3 Low development team involvement in early phases (C3)

The results indicate that low development involvement in the requirements phase can cause overscoping (mentioned by 6 out of 9 interviewees and 5 out of 6 questionnaire respondents did not disagree to this). This confirms previous work that points out the need of early development involvement in requirements engineering, e.g. required by interdependencies between product management and software development (Nuseibeh 2001). Glinz et al. (2002) also mentioned that lack of communication between project management and development at requirements hand-off may lead to unsatisfactory results. Similarly, Karlsson et al. (2007a) reported that communication gaps between marketing (requirements unit for our case company) and development, can result in insufficient effort estimates (i.e. RC3b) and in committing to unrealistically large features without considering the technical and scheduling implications. Our results corroborate these results in that low involvement and weak communication in early phases may lead to problems later on, including overscoping. These communication issues may also exacerbate the problem of getting accurate and reliable effort estimates (RC3b). Furthermore, the fact that one questionnaire respondent expressed experiencing good communication and cooperation between requirements and development teams may also explain the one *Disagree* response for this cause. On the other hand, a surprising result from the validation questionnaire is that this cause (C3) was seen to influence overscoping less than cause C6 (unclear vision of overall goal) both in total (among all respondents) and by 2 of the 3 software respondents. These results indicate that there may be additional (uncovered) factors that influence the impact this cause has on overscoping.

Finally, several methods have been proposed for addressing cause C3, e.g. negotiation of implementation proposals (Fricker 2007), model connectors for transforming requirements to architecture (Medvidovic 2003), cooperative requirements capturing (Macaulay 1993) and involving customers in the requirements management process (Kabbedijk 2009). Goal-oriented reasoning can also provide constructive guidelines for architects in their design tasks (van Lamsweerde 2003). If and to which degree the mentioned methods can alleviate overscoping by impacting this cause remains a topic for further research.

7.1.4 Requirements not agreed with development team (C4)

The results provide empirical evidence that weak agreement on requirements between requirements and software units can cause overscoping (all 6 questionnaire responders agreed to cause C4 and five interviewees mentioned C4 as a cause of overscoping). A significant root cause for this cause was found to be communication gaps, mainly between the requirements-related roles and the development and

testing roles. This confirms the viewpoint of Hall et al. (2002) that most requirement problems are actually organisational issues. In addition, this confirms the importance of seamless integration of different processes in collaborative work (Ebert 2002). The impact of insufficient communication on software engineering has been reported as a general issue within requirements engineering and product management (Bjarnason 2011b, Fricker 2007, Hall 2002, Kabbedijk 2009, Karlsson 2007a). Surprisingly, C4 scored the lowest impact among all the causes and only two questionnaire responders (both from the software unit) rated this cause as having any (low) impact factor on overscoping. In contrast, cause C6 (*weak vision of overall goal)* was rated as having the largest impact on overscoping.

7.1.5 Detailed requirements specification produced upfront (C5)

Our results indicate that too much detailed documentation produced upfront may cause overscoping (mentioned by five interviewees and experienced, agreed or partly agreed to by five questionnaire respondents, see section 5.1). This complements other studies into documentation in software engineering projects. For example, El Emam and Madhavji (1995) mentioned that in organisations which require more control the pressure to produce much detail is also greater. Lethbridge et al. (2003) reported that, for software engineers, there is often too much documentation for software systems, frequently poorly written and out of date. Furthermore, Sawyer et al. (1999) mention that premature freezing of requirements may cause scope creep and communication problems (both of which are identified as root causes of overscoping in our study) and suggest evolutionary prototyping as a remedy. Other remedies suggested for addressing excessive documentation include reuse of requirements specifications (Faulk 2001), as well as, simply creating less documentation (Aurum 1999). The effectiveness of these methods for the risk of overscoping remains to be investigated. The differing views on this cause between respondents may be explained by their roles and relationship to RE. All the disagreeing questionnaire respondents for this cause worked with requirements related roles. These roles are more likely to consider detailed requirements specifications as positive and good, rather than an issue. However, these roles have less insight into the later phases when development takes place and the effects of overscoping are experienced. Three of the respondents with experience from later development phases had experienced C5 as causing overscoping. Furthermore, Berry et al. (2010) mentioned that when time for elicitation is short, i.e. there is a lack of upfront documentation (or lack of C5), the requirements usually end up as an enhancement or become de-scoped since all of the client's requests cannot be delivered. Considering this, we conclude that both under specifying (as in Berry 2010) and over specifying (as in our study) can cause overscoping and later descoping, and that it remains to be investigated how to strike a good balance.

7.1.6 Unclear vision of overall goal (C6)

Our study identifies that a lack of clearly communicated goals and strategy for software development may cause defining the project scope primarily from a technology perspective, rather than with a business focus, thereby contributing to overscoping. Overall this cause was graded as having the second largest impact on overscoping, despite one questionnaire respondent (an RT leader) disagreeing to this

cause. Our results support the findings from related papers (Aurum 2005a, Cusumano 1995, DeMarco 2003, Khurum 2007, Neumann-Alkier 1997, Rosca 1997) that stress the importance of selecting requirements aligned with the overall business goals and discarding others as early as possible. In addition, failure of stakeholders to concur on project goals was found by DeMarco and Lister (2003) to pose the biggest risk for a project. A method for early requirements triage based on management strategies was proposed by Khurum et al. (2007). Aurum and Wohlin (2005a) have proposed a framework for aligning requirements with business objectives. Rosca et al. (1997) mention that the most demanding characteristic of business is the likelihood of change which cannot be fully controlled. This can be managed when business objectives are clear to the software developers, thus enabling them to manage a system requiring modifications while meeting the business objectives (Cusumano 1995). Finally, Karlsson et al. (2007a) mentioned the lack of common goals and visions as a challenge in achieving good cooperation, quoting their responders: 'If everyone has the same goal and vision, then everyone works in the right direction.'

7.1.7 Weak process adherence(+C7) and Scope & deadline dictated by management (+C8)

These two causes were mentioned in the questionnaires, though none of them were seen as having any major impact on overscoping. Karlsson et al. (2007a) found that weak process adherence may be caused both by high process complexity, as well as, lack of time for process implementation. The latter could be a consequence of overscoping. The direction of causal relationship between overscoping and process adherence remains to be investigated.

7.2 The Effects of Overscoping (RQ2)

The results indicate that overscoping may lead to a number of effects (or consequences), many of which are judged to be serious and potentially very costly for the company. Several of the identified effects may be in line with held beliefs about what overloading a project with too much work may lead to. The aim of this study is to investigate if such beliefs can be supported by empirical evidence or not, and if more surprising phenomena arise in relation to a specific, real-world overscoping situation.

7.2.1 Many changes after the project scope is set (E1)

The results show that overscoping leads to a large number of scope changes (experienced by all responders and impact graded as critical or major by all six questionnaire responders). This confirms evidence provided by Harker (1993) that requirements are not static and, thus, are hard to capture or classify. In addition, requirements volatility is mentioned as one of the challenges in MDRE by Karlsson et al. (2007a) and identified by Ramesh et al. (2007) as one of the 14 assumptions underlying agile software development. Furthermore, origins of requirements volatility have been listed (Harker 1993). Despite this awareness, causes for requirements volatility have not been empirically explored. Our results highlight

overscoping as one possible cause of late requirement changes. Furthermore, our results confirm that it is challenging to manage requirement changes.

7.2.2 Quality issues (E2)

The results indicate this as an important effect of overscoping (experienced and agreed for both interviews and questionnaires, and graded as having critical or major impact). This confirms that the quality of requirements engineering determines the software quality, as reported, e.g. by Aurum and Wohlin (2005b). In addition, our results highlight overscoping as a potential reason for quality issues.

7.2.3 Delays (E3)

This study shows (with a high degree of alignment between interviewees and questionnaire responses) that delays can be an effect of overscoping. Within MDRE, delays in launching products can be very costly and result in loss of market shares (Karlsson 2007a, Regnell 2005, Sawyer 1999, 2000). Therefore, the insight that overscoping may have this effect is important evidence that indicates that overscoping is a (potentially) serious risk.

7.2.4 Customer expectations are not always met (E4)

Our results indicate that overscoping can have the effect of failing to meet customer expectations. This could be explained by an overloaded project having no time or capacity neither to analyse or implement new requirements, nor to validate if market or customer needs could have changed. Furthermore, Karlsson et al. (2007a) reported failure to meet customer needs as one of the risks of developing products based on a technology focus (root cause RC6b). Another crucial part of producing software products that will satisfy the customers, as pointed out by Aurum and Wohlin (2005b), is working with RE throughout the project life cycle (as opposed to upfront requirements detailing, C5). The results of this study highlight the importance of selecting a feasible scope as one factor to consider when attempting to better understand and capture the customers' needs.

7.2.5 Communication gaps (E5)

Our results indicate that overscoping may cause increased communication gaps. (Roughly half of our interviewees and questionnaire respondents mentioned and agreed to this effect.) This may be explained by the tendency to deflect by blaming others when under pressure, rather than cooperate to solve problems together. Furthermore, interviewees described that the many changes resulting from overscoping (E1) were badly communicated to the product unit and resulted in false error reports being filed on changed, but not updated requirements. This in turn, caused irritation among the development teams and further increased the communication gaps. Similarly, Karlsson et al. (2007) reported that constant inflow of requirements (cause C1) caused decision conflicts between marketing and development roles.

7.2.6 Challenge to keep SRS updated (E6)

The majority of the respondents confirmed that overscoping increases the challenge to keep the SRS updated. When the SRS is detailed upfront (C5), the combination of the two (overscoping) effects E1 (many scope changes) and E1b (decreased motivation) lead to an increased need, but a lower motivation to update the SRS. This complements previous work, which reports requirements volatility as a common challenge for software projects (Harker 1993, Hood 2008, Jönsson 2005, Wiegers 2003) and that the view of RE as concerning a static set of requirements is inappropriate (Hall 2002, Harker 1993). In addition, Berry et al. (2010) report that time and resources are never sufficient to keep the documentation updated and that scope creep occurs when programmers code while the documentation keeps changing. Furthermore, our study highlights that the challenge of keeping the SRS updated is increased as an effect of overscoping. Harker and Eason (1993) proposed to address this challenge by defining a minimum critical specification combined with incremental deliveries (i.e. +P6) and thereby gradually providing more value. Further research is needed to investigate if the methods proposed to address the challenge of updating the requirements documentation could also minimise this effect for overscoping.

7.2.7 Overtime (+E7), Changed/cancelled product plans (+E8), Low priority for administrative tasks (+E9)

These effects were mentioned in the validation questionnaires and each got one count of critical impact. Further investigations are needed to validate their relationship to overscoping.

7.3 How Agile RE Practices May Impact Overscoping (RQ3)

Our study identifies that three of the agile RE practices being introduced at the case company may impact several of the causes and root causes of overscoping. In addition, three more practices were suggested by questionnaire respondents as addressing overscoping. The details of how the identified agile RE practices may impact overscoping (mentioned root causes can be seen in Figure 2) are discussed below. We interpret the results as an indication that overscoping is still a challenge for the case company, though more manageable with the (partly implemented) agile RE practices. Further investigations are needed to fully understand the situation in the agile context.

7.3.1 One continuous scope and release planning flow (P1)

This practice is experienced by the responders to directly impact cause C2 (no overview of software resource availability) by enabling transparency and insight into the full project scope and into the current workload of the software unit. The increased visibility of the load and available resource capacity to both business and software unit may bridge several communication gaps identified as root cause of overscoping, i.e. RC1c, RC3d and RC4b. This practice covers the agile RE practices of requirements prioritisation and constant re-planning for the high-level requirements (Ramesh 2007). Our results confirm the findings of Dybå and

Dingsøyr (2008) that managers of agile companies are more satisfied with the way they plan their projects than are plan-based companies. Furthermore, our study also corroborates the findings that agile prioritisation of the scope in combination with a stage-gate model at the feature level can avoid delaying critical features and also provides early feedback on features (Karlström 2005). However, achieving correct high-level cost and schedule estimation has been identified as a challenge also for agile project (Ramesh 2007), which may be one reason why overscoping remains an issue for the case company.

7.3.2 Cross-functional development teams (P2)

Cross-functional development teams are indicated by our results as improving several of the communication gaps identified by our study as important root causes to overscoping (i.e. RC1c, RC2a, RC3d, RC4b, RC6d). This case company practice is equivalent to the agile RE practice of preferring face-to-face requirements communication over written documentation (Beck 2001) in combination with agile prioritisation and constant re-planning at the detailed requirements level (Ramesh 2007). At this detailed requirements level, cost and schedule estimations in an agile fashion (by only allowing additions when simultaneously removing something less prioritised) have been found to be efficient (Karlström 2005, Ramesh 2007) and eliminate the 'requirements cramming' problem (Karlström 2005), which is equivalent to overscoping. Other studies have found that communication within development teams is improved by agile practices, but that communication towards other (dependent) teams remains a challenge (Karlström 2005, Pikkarainen 2008). This challenge is addressed with P2 by including competence covering all the involved functional areas within the same team (thus, impacting root causes RCicii, RC2a, RC4b and RC6dii). Furthermore, the agile RE practice of including a customer representative in the development teams is summarised by Dybå and Dingsøyr (2008) as improving the communication been customer and engineers, while filling this role can be stressful and challenging (Karlström 2005, Ramesh 2007).

7.3.3 Gradual and iterative requirements detailing (P3)

This practice is seen (by our interviewees) to decrease the total lead time for development of a feature (root cause RC1b) by delaying the detailing of requirements until they are actually needed for design and development. This in turn reduces the amount of requirement changes within the (shorter) time frame for the feature development, which in a market with high requirements volatility is a significant improvement. It may also reduce the communication gaps that occur due to the timing aspect of detailing requirements before design and implementation starts, i.e. root causes RC3d, RC4a, RC4b. The case company practice P3 is equivalent to the agile practice of *iterative RE* (Ramesh 2007).

7.4 Threats to Validity and Limitations

As for every study there are limitations that should be discussed and addressed. These threats to validity and steps taken to mitigate them are reported here based on guidelines provided by Robson (2002) for flexible design studies. Another important aspect for the quality of a flexible design research is the investigator (Robson 2002),

and for this study all researchers involved have previous experience in conducting empirical research, both interview studies and surveys.

7.4.1 Description validity

Misinterpretation of the interviewees (Robson 2002) poses the main threat to *description validity*. This threat was addressed in several ways. The interviews were recorded and transcribed. To enhance *reliability* of the transcriptions, the person taking notes during the interviews also transcribed them. In addition, this person has worked for the case company for several years and is well versed in company culture and language. Also, data triangulation was applied to the transcriptions by another researcher performing an independent transcription and coding of two randomly selected interviews. Furthermore, the interviewees checked both the transcriptions and the results of the study for errors and misinterpretations. Finally, data triangulation was applied to the interview results by collecting additional viewpoints from six (other) practitioners through a questionnaire (Robson 2002).

7.4.2 Interpretation validity

For this study, the main threat to valid interpretation has been the risk of imposing the hypothesis (formulated in phase one) onto the interviewees. To address this threat, open interview questions were always posed before asking specific questions based on the hypothesis. Furthermore, spontaneous descriptions of causes (without prompting) have been reported (as *Experienced*) separately from responses to follow-up questions on specific causes (as *Agreed*), see Section 5.1 and Table 3.

For phase three, the threat to valid description was addressed by the researchers jointly designing the questionnaire and the session held in connection to it. To ensure that all questionnaire responders correctly and uniformly understood the interview results, the results were presented to the participants. They could then ask for clarifications before filling out the questionnaire. The fact that questionnaire responders were confronted with a framework of results remains an open threat to interpretation validity. On the other hand, both interviewees and questionnaire respondents were explicitly encouraged to disagree and mention additional causes, effects and practices, which they also did. One of the main limitations of the study is the limited number of respondents. Although representatives from each of the covered units of the case company were involved in both interviews and validation questionnaire, the number of persons is relatively small and more factors may be identified by including additional viewpoints.

7.4.3 Theory validity

The main threat to theory validity for this study is the risk of missing additional or alternative factors. One source of this threat is the limited set of practitioners from which data has been gathered. Another potential source is the risk of *observer biases* limiting the study to the researcher's pre-knowledge of the company. This was a risk mainly in phase one and was addressed by involving the other researchers in discussing and reviewing the study design and the hypothesis which shaped the interview instrument. The fact that an additional main cause (i.e. C6) was identified as a result of the interviews shows that this bias was successfully addressed. However, identification of additional results in phase 3 may indicate that saturation

and the full exploration of the problem under investigation is not yet reached. As the goal of this work is exploratory our aim is not to present or achieve a complete coverage of the problem under investigation.

The involvement of the researcher with work experience from the case company has played a vital role in the study. This has ensured that the investigated problem is authentic and that the results are derived though an interpretation of the data based on a deep understanding of the case and its context. However, the results are limited to the case company and there is a risk that other possible causes of overscoping experienced at other companies were not identified. This also applies to the set of agile RE practices, which are limited to the ones that were currently known and partly implemented at the case company at the time of the study.

Internal generalizability was addressed by sampling interviewees and questionnaire respondents from different parts of the company thereby selecting roles and responsibilities involved throughout the development life cycle. Even so, it was not possible to include representatives from sales and marketing (they were unavailable at the time of the study). However, the requirements team leaders provided some insight into these aspects based on their experience from contacts with customers and with sales and marketing roles.

Considering *external generalizability*, the results should be interpreted with the case company context in mind. External validity is addressed by using *analytical generalisation* which enables drawing conclusions without statistical analysis and, under certain conditions, relating them also to other cases (Robson 2002, Runeson 2012). Within the scope of this paper, analytical generalisation is argued by applying the *making a case* strategy (Robson 2002, p. 107) by analysing related work and reporting similarities, differences and disagreements to our results (see Section 7). This analysis builds a supporting argument towards external validity of our study by seeking data which is not confirming a pre-assumed theory. In addition, follow-up studies in other domains can be conducted to utilise the direct demonstration strategy (Robson 2002) to further address the threat to external validity.

8 Conclusions and Further Work

Decision making is at the heart of requirements engineering (RE) (Aurum 2003) and within market-driven requirements engineering (MDRE) release planning is one of the most important and challenging tasks (Karlsson 2007a, 2007b, Regnell 2005, Sawyer 1999). Decisions concerning what to develop, and when, are inherently related to achieving customer satisfaction. Even though release planning (Karlsson 1997, 2007b, Regnell 2005) is well researched, RE decision making is acknowledged as challenging (Alenljung 2008, Aurum 2003, Ngo-The 2005) and scope creep is ranked as a serious project risk (Carter 2001) (Crockford 1980, Iacovou 2004), other aspects of scope management have been less explored (van de Weerd 2006). Furthermore, techniques for prioritising requirements (Karlsson 1997, 2007b) often focus on planning the scope of a project as a discrete activity, or one in a series of releases (Ngo-The 2005). Our previous work reported that scoping in an MDRE context is a continuous activity that may last throughout the entire project lifecycle (Wnuk 2009). If not successfully managed, and more requirements are

included into the project scope than can be handled with available resources the result is *overscoping*, i.e. the project 'bites off more than it can chew'.

Our study provides a detailed picture of factors involved in overscoping and confirms that scoping is a challenging part of requirements engineering and one of the risks in project management (Boehm 1989, DeMarco 2003, Legodi 2010). Our results indicate that overscoping is mainly caused by the fast-moving market-driven domain in which the case company operates, and how this inflow of requirements is managed. In the early project phases, low involvement from the development-near roles in combination with weak awareness of overall goals may result in an unrealistically large project scope. Our study indicates that overscoping can lead to a number of negative effects, including quality issues, delays and failure to meet customer expectations. Delays and quality problems are expensive, not just considering the cost of fixing the quality issues, but also in loss of market shares and brand value (Regnell 2005). Furthermore, we found indications that a situation of overscoping may cause even more overscoping, i.e. an organisation may end up in a vicious cycle when overscoping ties up development resources which are then not available for participating in early project phases. Furthermore, overscoping leads to increased communication gaps, which in turn are root causes of overscoping.

Companies, such as our case company, that develop embedded software for a business domain with a high market pressure need an organisational set-up and process suited to efficiently managing frequent changes in a cost effective way. Development projects need to respond quickly to changes, while at the same time handling the complexity of developing software in a large-scale setting. Agile processes are claimed to be better adapted to managing change than phase-based ones. As one interviewee stated: 'The waterfall approach is good from a preparation perspective, if you can then stick to what is planned. But, since we live in a world that changes a lot it doesn't work after all.' Our study indicates, that despite introducing agile RE practices, overscoping is still an issue for the case company, although more manageable. We conclude that the improvements may be explained by the agile RE practices of continuous prioritisation of the project scope, in combination with performing cost and schedule estimation, and gradual requirements detailing, in close collaboration within cross-functional teams, thereby closing a number of communication gaps. However, agile RE practices also pose challenges (Ramesh 2007), e.g. communication between teams (Karlström 2005, Pikkarainen 2008), difficulty in cost estimation (Ramesh 2007). This, in combination with a fast-moving, market-driven domain may explain why overscoping remains a challenge also with the agile development process.

The causes and effects unveiled through this study (summarised in Figure 2) can be used as a basis for identifying potential issues to address in order to avoid or alleviate an overscoping situation. For example, the root cause of *low competence in cost estimations* may be addressed by introducing techniques for improving cost estimation, which should lead to more realistic plans. Finally, supported by our findings of potentially serious effects of overscoping, we conclude that this phenomenon can be a major risk of requirements engineering and project management, complementary to the risk of scope creep mentioned by De Marco and Lister (2003).

Future work includes evaluating the agile RE practices when they are fully implemented; how do they affect overscoping and what additional challenges do

they pose over time? Furthermore, it would be interesting to investigate how aspects such as organisational set-up, software development model (agile or waterfall) and application of different software engineering methods affect decision making. In addition, extending the results from this study to include other companies and also other perspectives, such as marketing and sales, may strengthen the generalizability of our findings.

References

Abramovici M, Sieg OC (2002) Status and Development Trends of Product Lifecycle Management Systems. Published at Ruhr-University Bochum, Germany.

Alenljung B, Persson A (2008) Portraying the Practice of Decision-Making in Requirements Engineering: a Case of Large scale bespoke Development. Req. Eng. 13, 2008, 257-279.

Aurum A, Martin E (1999) Managing both Individual and Collective participation in Software Requirements Elicitation Process. 14th Int. Symposium on Computer and Information Sciences, (ISCIS'99), Kusadasi, Turkey, 1999, pp. 124-131.

Aurum A, Wohlin C (2003) The Fundamental Nature of Requirements Engineering Activities as a Decision-Making Process. Inf. Software Technol. 45, 2003, 945-54.

Aurum A, Wohlin C (2005a) Aligning Requirements with Business Objectives: a Framework for Requirements Engineering Decisions. Workshop on Requirements Engineering Decision Support, REDECS'05, 29 August-September 2, 2005, Paris, France.

Aurum A, Wohlin C (2005b) Requirements Engineering: Setting the Context. A. Aurum, C. Wohlin (Eds.), Managing and Engineering Software Requirements, Springer- Verlag, Germany, 2005, pp. 1-15.

Beck K (1999) Extreme Programming Explained. Published by Addison-Wesley, 1999.

Beck et al. (2001) The Agile Manifesto. Published at http://agilemanifesto.org/ (Latest access June 2013.)

Berry DM, Czarnecki K, Antkiewicz M, AbdElRazik M (2010) Requirements Determination is Unstoppable: An Experience Report. Proc. of the 18th Int. IEEE Requirements Engineering Conference, IEEE Computer Society, 2010, pp. 311-316.

Bjarnason E, Wnuk K, Regnell B (2010a) Overscoping: Reasons and Consequences – A Case Study in Decision Making in Software Product Management. Proc. of 4th Int. Workshop on Softw. Product Management, pp. 30-39.

Bjarnason E (2010b) Case study material (interview instrument, questionnaire etc) for the Before aNd After (BNA) study. Published online at http://serg.cs.lth.se/research/experiment_packages/bna/ (latest access June 2013)

Bjarnason E, Wnuk K, Regnell B (2011a) A Case Study on Benefits and Side-Effects of Agile Practices in Large-Scale Requirements Engineering. Proc. of 1st Workshop on Agile Requirements Engineering (AREW '11). ACM, New York, NY, USA, 2011.

Bjarnason E, Wnuk K, Regnell B (2011b) Requirements Are Slipping Through the Gaps - A Case Study on Causes & Effects of Communication Gaps in Large-Scale Software Development. Proc. of 19th IEEE Int. Requirements Engineering Conf., pp.37-46.

Boehm B (1989) Tutorial: Software Risk Management. Published by IEEE Computer Society Press, 1989

Carlshamre P, Sandahl K, Lindvall M, Regnell B, Natt och Dag J (2001) An Industrial Survey of Requriements Interdependencies in Software Product Release Planning. Proc. of 5th IEEE Int. Symposium on Requirements Engineering, pp. 84–91.

Carlshamre P (2002) A Usability Perspective on Requirements Engineering – From Methodology to Product Development. Published Ph.D Thesis, Linköping University Sweden, 2002.

Carter RA, Anton AI, Dagnino A, Williams L (2001) Evolving Beyond Requirements Creep: A Risk-Based Evolutionary Prototyping Model. Proc. of 5th IEEE Int. Symposium on Requirements Engineering, pp. 84-101.

Crockford N (1980) An Introduction to Risk Management (2 ed.) Published by Cambridge, UK: Woodhead-Faulkner. 112 p.

Cusumano MA, Selby RW (1995) Microsoft Secrets. Published by Simon and Schuster, New York, 1995.

DeBaud JM, Schmid K (1999) A Systematic Approach to Derive the Scope of Software Product Lines. Proc. of 21st Int. Conf. on Software Engineering, ACM, Los Angeles USA, 1999, pp. 34-43.

DeMarco T, Lister T (2003) Risk Management during Requirements. IEEE Software 20, 2003, 99-101.

Dybå T, Dingsøyr T (2008) Empirical Studies of Agile Software Development: A Systematic Review. Information Software Technology 50, 2008, 833–859.

Ebert C, De Man J (2002) e-R&D – Effectively Managing Process Diversity. Ann. Softw. Eng. 14, 2002, 73-91.

El Emam K, Madhavji NH (1995) A field study of requirements engineering practices in information systems development. Proc. of 2nd IEEE Int. Symposium on Requirements Engineering, IEEE Computer Society, Washington, DC, USA, 1995, pp. 68-80.

Faulk SR (2001) Product-Line Requirements Specification (PRS): An Approach and Case Study. Proc. of 5th IEEE Int. Symposium on Requirements Engineering, IEEE Computer Society, Washington, DC, USA, 2001, pp. 48-55.

Fricker S, Gorschek T, Myllyperkiö P (2007) Handshaking between Software Projects and Stakeholders Using Implementation Proposals. Proc. of Int. Working Conference on Requirements Engineering: Foundation for Software Quality, Trondheim, Norway, 2007, Vol. 4542 of Lecture Notes in Computer Science, Springer Berlin/Heidelberg, 2007, pp. 144 – 159.

Gemmer A (1997) Risk Management Moving Beyond Process. Computer 30, 1997, pp. 33-43, DOI=10.1109/2.589908 http://dx.doi.org/10.1109/2.589908

Glinz M, Berner S, Joos S (2002) Object-Oriented Modelling with ADORA. Information Systems 27, 2002, pp. 425-444.

Gorschek T, Wohlin C (2005) Requirements Abstraction Model. Requir. Eng. 11, 2006, pp. 79-101.

Hall T, Beecham S, Rainer A (2002) Requirements problems in twelve software companies: an empirical analysis. IEEE Software 149, 2002, pp. 153- 160.

Harker SDP, Eason KD (1993) The Change and Evolution of Requirements as challenge to the Practice of Software Engineering. Proc. of IEEE Int. Symposium on Requirements Engineering, SanDiego, CA, USA, 1993, pp. 266-292.

Hood C, Wiedemann S, Fichtinger S, Pautz U (2008). Chapter on Change Management interface. Requirements Management – The Interface Between Requirements Development and All Other Systems Engineering Processes, Springer-Verlag Berlin Heidelberg, 2008, pp. 175-191.

Iacovou CL, Dexter AS (2004) Turning around runaway information technology projects. In: IEEE Engineering Management Review 3, 2004, pp. 97- 112.

Jorgensen M, Shepperd M (2007) A Systematic Review of Software Development Cost Estimation Studies. IEEE Trans. on Soft. Eng. 33, 2007, 33-53.

Jönsson P, Lindvall M (2005) Impact Analysis. In: Managing and Engineering Software Requirements, A. Aurum, C. Wohlin (Eds.), Springer- Verlag, Germany, 2005, pp. 117-142.

Kabbedijk J, Brinkkemper S, Jansen S, van der Veldt SB (2009) Customer Involvement in Requirements Management: Lessons from Mass Market Software Development. Proc. of 17th IEEE Int. Requirements Engineering Conf., pp.281-286.

Karlsson J, Ryan K (1997) A Cost-Value Approach for Prioritizing Requirements. IEEE Soft. 14, 1997, pp. 67–74.

Karlsson L, Dahlstedt ÅG, Natt Och Dag J, Regnell B, Persson A (2007a) Requirements Engineering Challenges in Market-Driven Software Development An Interview Study with Practitioners. Inf. and Soft. Techn. 49, 2007, 588-604.

Karlsson L, Thelin T, Regnell B, Berander P Wohlin C (2007b) Pair-wise Comparisons versus Planning Game Partitioning--Experiments on Requirements Prioritisation Techniques. Emp. Soft. Eng. 12, 2007, pp. 3-33.

Karlström D, Runeson P (2005) Combining Agile Methods with Stage-Gate Project Management. IEEE Soft. 22, 2005, pp. 43 – 49.

Khurum M, Aslam K, Gorschek T (2007) A Method for Early Requirements Triage and Selection Utilizing Product Strategies. Proc. of 14th IEEE Asia-Pacific Software Engineering Conf., Washington, DC, USA, 2007, pp. 97-104.

Konrad S, Gall M (2008) Requirements Engineering in the Development of Large-Scale Systems. Proc. of 16th IEEE Int. Requirements Engineering Conf. 2008, pp. 217-222.

van Lamsweerde A (2003) From System Goals to Software Architecture. Formal Methods for Software Architectures, M.Bernardo, P. Inverardi, (Eds), Springer, 2003.

Legodi I, Barry ML (2010) The Current Challenges and Status of Risk Management in Enterprise Data Warehouse Projects in South Africa. Proc. of PICMET '10, pp.1-5.

Lethbridge TC, Singer J, Forward A (2003) How Software Engineers Use Documentation: the State of the Practice. IEEE Software 20, 2003, pp. 35- 39.

Lixin Z (2008) A Project Human Resource Allocation Method Based on Software Architecture and Social Network. Proc. of 14th Int. Conf. on Wireless Communications, Networking and Mobile Computing, October 2008, pp.1-6.

Macaulay L (1993) Requirements Capture as a Cooperative Activity. Proc. of 1st IEEE Symposium on Requirements Engineering, USA, 1993, pp. 174-181.

Medvidovic N, Grünbacher P, Egyed A, Boehm B (2003) Bridging Models Across the Software Lifecycle. Journal of Syst. Softw. 68, 2003, pp. 199-215.

Myers MD, Avison D (2002) Qualitative Research in Information System. Sage Publications, USA, 2002.

Neumann-Alkier L (1997) Think Globally, Act Locally – Does it Follow the Rule in Multinational Corporations? Proc. of 5th European Conf. on Information Systems, 1997, pp. 541-552.

Ngo-The A, Ruhe G (2005) Decision support in requirements engineering. Managing and Engineering Software Requirements, A. Aurum, C. Wohlin (Eds.), Springer- Verlag, Germany, 2005, pp. 267-286.

Nuseibeh B (2001) Weaving Together Requirements and Architectures. Computer 34, 2001, pp. 115-117.

Phillips JJ, Bothell TW, Snead GL (2002) The Project Management Scorecard: Measuring the Success of Project Management Solutions. Elsevier, USA, 2002.

Pikkarainen M, Haikara J, Salo O, Abrahamsson P, Still J (2008) The Impact of Agile Practices on Communication in Software Development. Empirical Software Engineering 13, 2008, pp. 303–337.

Pohl C, Böckle G, van der Linden FJ (2005) Software Product Line Engineering: Foundations, Principles and Techniques. Springer-Verlag, New York USA, 2005.

Potts C (1995) Invented Requirements and Imagined Customers: Requirements Engineering for Off-the-shelf Software. Proc. of 2nd IEEE Int. Symposium on Requirements Engineering, March 1995, pp. 128-131, doi: 10.1109/ISRE.1995.512553.

PMI (Project Management Institute) (2000) Chapter 5: Project Scope Management. In: A Guide to the Project Management Body of Knowledge (PMBOK Guide) 2000 Edition. Project Management Institute, Four Campus Boulevard, Newtown Square, PA 19073-3299, USA, 2000.

Ramesh B, Cao L, Baskerville R (2007) Agile Requirements Engineering Practices and Challenges: An Empirical Study. Information Systems Journal 20, 2007, pp. 449-280.

Regnell B, Brinkkemper S (2005) Market–Driven Requirements Engineering for Software Products. Managing and Engineering Software Requirements, A. Aurum, C. Wohlin (Eds.), Springer-Verlag, Germany, 2005, pp. 287-308.

Regnell B, Berntsson Svensson R, Wnuk K (2008) Can We Beat the Complexity of Very Large-Scale Requirements Engineering? Proc. of 14th Int. Conf. on Requirements Engineering: Foundation for Software Quality (REFSQ '08), B. Paech, C. Rolland (Eds.), vol. 5025 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 123-128.

Robson C (2002) Real World Research. Blackwell Publishing, 2002.

Rosca D, GreenSpan S, Feblowitz M, Wild C (1997) A Decision Making Methodology in Support of the Business Rules Lifecycle. Proc. of 3rd IEEE Int. Symposium on Requirements Engineering, Annapolis, MD, USA, January 1997, pp. 236-246.

Runeson P, Rainer A, Höst M, Regnell B (2012) Case Study Research in Software Engineering: Guidelines and Examples. Wiley, 2012.

Sangwan R, Bass M, Mullick N, Paulish DJ, Kazmeier J (2006) Global Software Development Handbook. Auerbach Publications, Boston MA, USA 2006.

Sawyer P, Sommerville I, Kotonya G (1999) Improving Market-Driven RE Processes. Proc. of Int. Conf. on Product-Focused Softw. Process Improvem., 1999, pp. 222–236.

Sawyer P (2000) Packaged Software: Challenges for RE. Proc. of 6th Int. Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'2000).

Schmid K (2002) A Comprehensive Product Line Scoping Approach and Its Validation. Proc. of 24th Int. Conf. on Software Engineering, 2002, pp. 593-603.

Schwaber K, Beedle M (2002) Agile Software Development with SCRUM. Prentice Hall.

Svahnberg M, Gorschek T, Feldt R, Torkar R, Bin Saleem S, Shafique MU (2010) A Systematic Review on Strategic Release Planning Models. Inf. Softw. Technol. 52, 2010, pp. 237-248.

Wiegers KE (2003) Software Requirement (2nd edition) Microsoft Press, Redmond 2003.

Wnuk K, Regnell B, Karlsson L (2009) What Happened to Our Features? Visualization and Understanding of Scope Change Dynamics in a Large-Scale Industrial Setting. Proc. of 17th IEEE Int. Requirements Engineering Conf. 2009, pp. 89-98. doi 10.1109/RE.2009.32

van de Weerd I, Brinkkemper S, Nieuwenhuis R, Versendaal J, Bijlsma L (2006) Towards a Reference Framework for Software Product Management. Proc. of 14th IEEE Int. Conf. on Requirements Engineering, Sep. 2006, pp. 319-322.

Wohlin C, Aurum A (2005) What is Important When Deciding to Include a Software Requirement in a Project or Release? Proc. of 4th Symposium on Empirical Software Engineering, 17-18 Nov. 2005, doi: 10.1109/ISESE.2005.1541833.

PAPER III

CHALLENGES AND PRACTICES IN ALIGNING REQUIREMENTS WITH VERIFICATION AND VALIDATION: A CASE STUDY OF SIX COMPANIES[1]

Weak alignment of requirements engineering (RE) with verification and validation (VV) may lead to problems in delivering the required products in time with the right quality. For example, weak communication of requirements changes to testers may result in lack of verification of new requirements and incorrect verification of old invalid requirements, leading to software quality problems, wasted effort and delays. However, despite the serious implications of weak alignment research and practice both tend to focus on one or the other of RE or VV rather than on the alignment of the two. We have performed a multi-unit case study to gain insight into issues around aligning RE and VV by interviewing 30 practitioners from 6 software developing companies, involving 10 researchers in a flexible research process for case studies. The results describe current industry challenges and practices in aligning RE with VV, ranging from quality of the individual RE and VV activities, through tracing and tools, to change control and sharing a common understanding at strategy, goal and design level. The study identified that human aspects are central, i.e. cooperation and communication, and that requirements engineering practices are a critical basis for alignment. Further, the size of an organisation and its motivation for applying alignment practices, e.g. external enforcement of traceability, are variation factors that play a key role in achieving alignment. Our results provide a strategic roadmap for practitioners in improvement work to address alignment challenges. Furthermore, the study provides a foundation for continued research to improve the alignment of RE with VV.

[1] By E. Bjarnason, P. Runeson, M. Borg, M. Unterkalmsteiner, E. Engström, B. Regnell, G. Sabaliauskaite, A. Loconsole, T. Gorschek, R. Feldt. Published in Journal of Empirical Software Engineering, July 2013.

1 Introduction

Requirements engineering (RE) and verification and validation (VV) both aim to support development of products that will meet customers' expectations regarding functionality and quality. However, to achieve this RE and VV need to be aligned and their '*activities or systems organised so that they match or fit well together*' (MacMillan Dictionary's definition of 'align'). When aligned within a project or an organisation, RE and VV work together like two bookends that support a row of books by buttressing them from either end. RE and VV, when aligned, can effectively support the development activities between the initial definition of requirements and acceptance testing of the final product (Damian 2006).

Weak coordination of requirements with development and testing tasks can lead to inefficient development, delays and problems with the functionality and the quality of the produced software, especially for large-scale development (Kraut 1995). For example, if requirements changes are agreed without involving testers and without updating the requirements specification, the changed functionality is either not verified or incorrectly verified. This weak alignment of RE and work that is divided and distributed among engineers within a company or project poses a risk of producing a product that does not satisfy business and/or client expectations (Gorschek 2007). In particular, weak alignment between RE and VV may lead to a number of problems that affect the later project phases such as non-verifiable requirements, lower product quality, additional cost and effort required for removing defects (Sabaliauskaite 2010). Furthermore, Jones et al. (2009) identified three other alignment related problems found to affect independent testing teams, namely uncertain test coverage, not knowing whether changed software behaviour is intended, and lack of established communication channels to deal with issues and questions.

There is a large body of knowledge for the separate areas of RE and VV, some of which touches on the connection to the other field. However, few studies have focused specifically on the alignment between the two areas (Barmi 2011) though there are some exceptions. Kukkanen et al. (2009) reported on lessons learnt in concurrently improving the requirements and the testing processes based on a case study. Another related study was performed by Uusitalo et al. (2008) who identified a set of practices used in industry for linking requirements and testing. Furthermore, RE alignment in the context of outsourced development has been pointed out as a focus area for future RE research by Cheng and Attlee (2007).

When considering alignment, traceability has often been a focal point (Watkins 1994, Barmi 2011, Paci 2012). However, REVV alignment also covers the coordination between roles and activities of RE and VV. Traceability mainly focuses on the structuring and organisation of different related artefacts. Connecting (or tracing) requirements with the test cases that verify them support engineers in ensuring requirements coverage, performing impact analysis for requirements changes etc. In addition to tracing, alignment also covers the interaction between roles throughout different project phases; from agreeing on high-level business and testing strategies to defining and deploying detailed requirements and test cases.

Our case study investigates the challenges of RE and VV (REVV) alignment, and identifies methods and practices used, or suggested for use, by industry to

address these issues. The results reported in this paper are based on semi-structured interviews of 90 minutes each with 30 practitioners from six different software companies, comprising a wide range of people with experience from different roles relating to RE and VV. This paper extends on preliminary results of identifying the challenges faced by one of the companies included in our study (Sabaliauskaite 2010). In this paper, we report on the practices and challenges of all the included companies based on a full analysis of all the interview data. In addition, the results are herein categorised to support practitioners in defining a strategy for identifying suitable practices for addressing challenges experienced in their own organisations.

The rest of this paper is organised as follows: Section 2 presents related work. The design of the case study is described in Section 3, while the results can be found in Section 4. In Section 5 the results are discussed and, finally the paper is concluded in Section 6.

2 Related Work

The software engineering fields RE and VV have mainly been explored with a focus on one or the other of the two fields (Barmi 2011), though there are some studies investigating the alignment between the two. Through a systematic mapping study into alignment of requirements specification and testing, Barmi et al. (2011) found that most studies in the area were on model-based testing including a range of variants of formal methods for describing requirements with models or languages from which test case are then generated. Barmi et al. also identified traceability and empirical studies into alignment challenges and practices as main areas of research. Only 3 empirical studies into REVV alignment were found. Of these, 2 originate from the same research group and the third one is the initial results of the study reported in this paper. Barmi et al. draw the conclusions that though the areas of model-based engineering and traceability are well understood, practical solutions including evaluations of the research are needed. In the following sections previous work in the field is described and related to this study at a high level. Our findings in relation to previous work are discussed in more depth in Section 5.

The impact of RE on the software development process as a whole (including testing) has been studied by Damian et al. (2005) who found that improved RE and involving more roles in the RE activities had positive effects on testing. In particular, the improved change control process was found to 'bring together not only the functional organisation through horizontal alignment (designers, developers, testers and documenters), but also vertical alignment of organisational responsibility (engineers, teams leads, technical managers and executive management)' (Damian 2005). Furthermore, in another study Damian and Chisan (2006) found that rich interactions between RE and testing can lead to pay-offs in improved test coverage and risk management, and in reduced requirements creep, overscoping and waste, resulting in increased productivity and product quality (Damian 2006). Gorschek and Davis (2007) have proposed a taxonomy for assessing the impact of RE on, not just project, but also on product, company and

society level; to judge RE not just by the quality of the system requirements specification, but also by its wider impact.

Jointly improving the RE and testing processes was investigated by Kukkanen et al. (2009) through a case study on development performed partly in the safety-critical domain with the dual aim of improving customer satisfaction and product quality. They report that integrating requirements and testing processes, including clearly defining RE and testing roles for the integrated process, improves alignment by connecting processes and people from requirements and testing, as well as, applying good practices that support this connection. Furthermore, they report that the most important aspect in achieving alignment is to ensure that 'the right information is communicated to the right persons'. Successful collaboration between requirements and test can be ensured by assigning and connecting roles from both requirements and test as responsible for ensuring that reviews are conducted. Among the practices implemented to support requirements and test alignment were the use of metrics, traceability with tool support, change management process and reviews of requirements, test cases and traces between them (Kukkanen 2009). The risk of overlapping roles and activities between requirements and test, and gaps in the processes was found to be reduced by concurrently improving both processes (Kukkanen 2009). These findings correlate very well with the practices identified through our study.

Alignment practices that improve the link between requirements and test are reported by Uusitalo et al. (2008) based on six interviews, mainly with test roles, from the same number of companies. Their results include a number of practices that increase the communication and interaction between requirements and testing roles, namely early tester participation, traceability policies, consider feature requests from testers, and linking test and requirements people. In addition, four of the companies applied traceability between requirements and test cases, while admitting that traces were rarely maintained and were thus incomplete (Uusitalo 2008). Linking people or artefacts were seen as equally important by the interviewees who were unwilling to select one over the other. Most of the practices reported by Uusitalo et al. were also identified in our study with the exception of the specific practice of linking testers to requirements owners and the practice of including internal testing requirements in the project scope.

The concept of traceability has been discussed, and researched since the very beginning of software engineering, i.e. since the 1960s (Randell 1969). Traceability between requirements and other development artefacts can support impact analysis (Gotel 1994, Watkins 1994, Ramesh 1997, Damian 2005, Uusitalo 2008, Kukkanen 2009), lower testing and maintenance costs (Watkins 1994, Kukkanen 2009), and increased test coverage (Watkins 1994, Uusitalo 2008) and thereby quality in the final products (Watkins 1994, Ramesh 1997). Tracing is also important to software verification due to being an (acknowledged) important aspect in high quality development (Watkins 1994, Ramesh 1997). The challenges connected to traceability have been empirically investigated and reported over the years. The found challenges include volatility of the traced artefacts, informal processes with lack of clear responsibilities for tracing, communication gaps, insufficient time and

resources for maintaining traces in combination with the practice being seen as non-cost efficient, and a lack of training (Cleland-Huang 2003). Several methods for supporting automatic or semi-automatic recovery of traces have been proposed as a way to address the cost of establishing and maintaining traces, e.g. De Lucia 2007, Hayes 2007, Lormans 2008. An alternative approach is proposed by Post et al. (2009) where the number of traces between requirements and test are reduced by linking test cases to user scenarios abstracted from the formal requirements, thus tracing at a higher abstraction level. When evaluating this approach, errors were found both in the formal requirements and in the developed product (Post 2009). However, though the evaluation was performed in an industrial setting the set of 50 requirements was very small. In conclusion, traceability in full-scale industrial projects remains an elusive and costly practice to realise (Gotel 1994, Watkins 1994, Jarke 1998, Ramesh 1998). It is interesting to note that Gotel and Finkelstein (1994) conclude that a particular concern in improving requirements traceability is the need to facilitate informal communication with those responsible for specifying and detailing requirements. Another evaluation of the traceability challenge reported by Ramesh identifies three factors as influencing the implementation of requirements traceability, namely environmental (tools), organisational (external organisational incentive on individual or internal), and development context (process and practices) (Ramesh 1998).

Model-based testing is a large research field within which a wide range of formal models and languages for representing requirements have been suggested (Dias Neto 2007). Defining or modelling the requirements in a formal model or language enables the automatic generation of other development artefacts such as test cases, based on the (modelled) requirements. Similarly to the field of traceability, model-based testing also has issues with practical applicability in industrial development (Nebut 2006, Mohagheghi 2008, Yue 2011). Two exceptions to this is provided by Hasling et al. (2008) and by Nebut et al. (2006) who both report on experiences from applying model-based testing by generating system test cases from UML descriptions of the requirements. The main benefits of model-based testing are in increased test coverage (Nebut 2006, Hasling 2008), enforcing a clear and unambiguous definition of the requirements (Hasling 2008) and increased testing productivity (Grieskamp 2011). However, the formal representation of requirements often results in difficulties both in requiring special competence to produce (Nebut 2006), but also for non-specialist (e.g. business people) in understanding the requirements (Lubars 1993). Transformation of textual requirements into formal models could alleviate some of these issues. However, additional research is required before a practical solution is available for supporting such transformations (Yue 2011). The generation of test cases directly from the requirements implicitly links the two without any need for manually creating (or maintaining) traces. However, depending on the level of the model and the generated test cases the value of the traces might vary. For example, for use cases and system test cases the tracing was reported as being more *natural* than when using state machines (Hasling 2008). Errors in the models are an additional issue to consider when applying model-based testing (Hasling 2008). Scenario-based models where test cases are defined to cover requirements defined as use cases, user stories or user scenarios have been proposed as an alternative to the formal models, e.g. by

Regnell and Runeson (1998), Regnell et al. (2000) and Melnik et al. (2006). The scenarios define the requirements at a high level while the details are defined as test cases; acceptance test cases are used to document the detailed requirements. This is an approach often applied in agile development (Cao 2008). Melnik et al. (2006) found that using executable acceptance test cases as detailed requirements is straight-forward to implement and breeds a testing mentality. Similar positive experiences with defining requirements as scenarios and acceptance test cases are reported from industry by Martin et al. (2008)

3 Case Study Design

The main goal of this case study was to gain a deeper understanding of the issues in REVV alignment and to identify common practices used in industry to address the challenges within the area. To this end, a flexible exploratory case study design (Robson 2002, Runeson 2012) was chosen with semi-structured interviews as the data collection method. In order to manage the size of the study, we followed a case study process suggested by Runeson et al. (2012, chapter 14) which allowed for a structured approach in managing the large amounts of qualitative data in a consistent manner among the many researchers involved. The process consists of the following five interrelated phases (see Figure 1 for an overview, including in- and outputs of the different phases):

1) *Definition* of goals and research questions
2) *Design and planning* including preparations for interviews
3) *Evidence collection* (performing the interviews)
4) *Data analysis* (transcription, coding, abstraction and grouping, interpretation)
5) *Reporting*

Phases 1-4 are presented in more detail in sections 3.1 to 0, while threats to validity are discussed in section 3.5. A more in-depth description with lessons learned from applying the process in this study is presented by Runeson et al (2012, Chapter 14). A description of the six case companies involved in the study can be found in section 3.2.

The ten authors played different roles in the five phases. The senior researchers, Regnell, Gorschek, Runeson and Feldt lead the *goal definition* of the study. They also coached the *design and planning*, which was practically managed by Loconsole, Sabaliauskaite and Engström. *Evidence collection* was distributed over all ten researchers. Loconsole and Sabaliauskaite did the transcription and coding

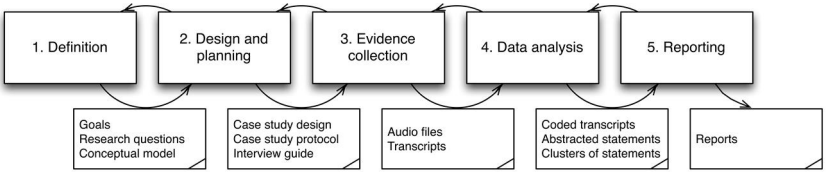


Figure 1. Overview of the research process including in- and output for each phase.

together with Bjarnason, Borg, Engström and Unterkalmsteiner, as well as the preliminary *data analysis* for the evidence from the first company (Sabaliauskaite 2010). Bjarnason, Borg, Engström and Unterkalmsteiner did the major legwork in the intermediate *data analysis*, coached by Regnell, Gorschek and Runeson. Bjarnason and Runeson made the final *data analysis*, *interpretation* and *reporting*, which was then reviewed by the rest of the authors.

3.1 Definition of Research Goal and Questions

This initial phase (see Figure 1) provided the direction and scope for the rest of the case study. A set of goals and research questions were defined based on previous experience, results and knowledge of the participating researchers, and a literature study into the area. The study was performed as part of an industrial excellence research centre, where REVV alignment was one theme. Brainstorming sessions were also held with representatives from companies interested in participating in the study. In these meetings the researchers and the company representatives agreed on a main long-term research goal for the area: to improve development efficiency within existing levels of software quality through REVV alignment, where this case study takes a first step into exploring the current state of the art in industry. Furthermore, a number of aspects to be considered were agreed upon, namely agile processes, open source development, software product line engineering, non-functional requirements, and, volume and volatility of requirements. As the study progressed the goals and focal aspects were refined and research questions formulated and documented by two researchers. Four other researchers reviewed their output. Additional research questions were added after performing two pilot interviews (in the next phase, see Section 3.2). In this paper, the following research questions are addressed in the context of software development:

- RQ1: What are the current challenges, or issues, in achieving REVV alignment?
- RQ2: What are the current practices that support achieving REVV alignment?
- RQ3: Which current challenges are addressed by which current practices?

The main concepts of REVV alignment to be used in this study were identified after discussions and a conceptual model of the scope of the study was defined (see Figure 2). This model was based on a traditional V-model showing the artefacts and processes covered by the study, including the relationships between artefacts of varying abstraction level and between processes and artefacts. The discussions undertaken in defining this conceptual model led to a shared understanding within the group of researchers and reduced researcher variation, thus ensuring greater validity of the data collection and results. The model was utilised both as a guide for the researchers in subsequent phases of the study and during the interviews.

3.2 Design and Planning

In this phase, the detailed research procedures for the case study were designed and preparations were made for data collection. These preparations included designing the interview guide and selecting the cases and interviewees.

The interview guide was based on the research questions and aspects, and the conceptual model produced in the *Definition* phase (see Figures 1 and 2). The guide was constructed and refined several times by three researchers and reviewed by another four. User scenarios related to aligning requirements and testing, and examples of alignment metrics were included in the guide as a basis for discussions with the interviewees. The interview questions were mapped to the research questions to ensure that they were all covered. The guide was updated twice; after two pilot interviews, and after six initial interviews. Through these iterations the general content of the guide remained the same, though the structure and order of the interview questions were modified and improved. The resulting interview guide is published by Runeson et al. (2012, appendix C). Furthermore, a consent information letter was prepared to make each interviewee aware of the conditions of the interviews and their rights to refuse to answer and to withdraw at any time. The consent letter is published by Runeson et al. (2012, Appendix E).

The case selection was performed through a brainstorming session held within the group of researchers where companies and interviewee profiles that would match the research goals were discussed. In order to maximise the variation of companies selected from the industrial collaboration network, with respect to size, type of process, application domain and type of product, a combination of maximum variation selection and convenience selection was applied (Runeson 2012, p. 35, 112). The characteristics of the case companies are briefly summarised in Table 1. It is clear from the summary that they represent: a wide range of domains; size from 50 to 1,000 software developers; bespoke and market driven development; waterfall and iterative processes; using open source components or not, etc. At the time of the interviews a major shift in process model from waterfall to agile was underway at company F. Hence, for some affected factors in Table 1, information is given as to for which model the data is valid.

Our aim was to cover processes and artefacts relevant to REVV alignment for the whole life cycle from requirements definition through development to system testing and maintenance. For this reason, interviewees were selected to represent

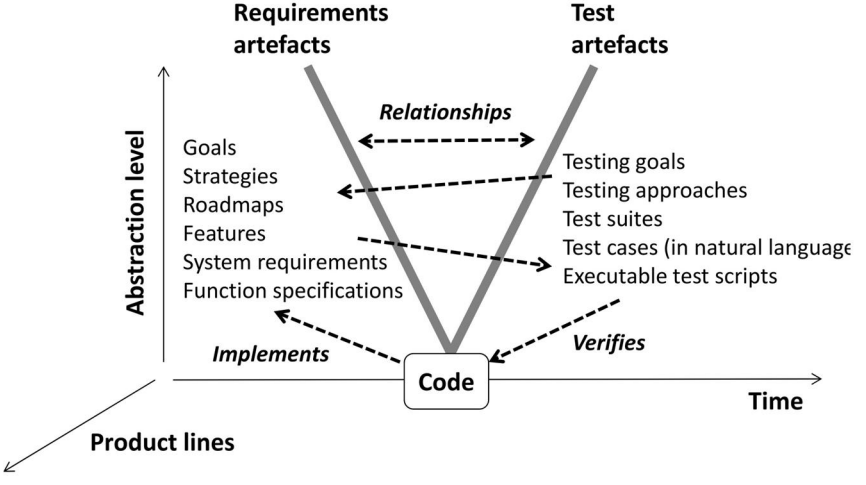


Figure 2. The conceptual model of the area under study, produced in phase 1.

the relevant range of viewpoints from requirements to testing, both at managerial and at engineering level. Initially, the company contact persons helped us find suitable people to interview. This was complemented by s*nowball sampling* (Robson 2002) by asking the interviewees if they could recommend a person or a role in the company whom we could interview in order to get alignment-related information. These suggestions were then matched against our aim to select interviewees in order to obtain a wide coverage of the processes and artefacts of interest. The selected interviewees represent a variety of roles, working with requirements, testing and development; both engineers and managers were interviewed. The number of interviews per company was selected to allow for going in-depth in one company (company F) through a large number of interviews. Additionally, for this large company the aim was to capture a wide view of the situation and thus mitigate the risk of a skewed sampled. For the other companies, three interviews were held per company. An overview of the interviewees, their roles and level of experience is given in Table 2. Note that for company B, the consultants that were interviewed typically take on a multitude of roles within a project even though they can mainly be characterised as software developers they also take part in requirements analysis and specification, design and testing activities.

3.3 Evidence Collection

A semi-structured interview strategy (Robson 2002) was used for the interviews, which were performed over a period of one year starting in May 2009. The interview guide (Runeson 2012, appendix C) acted as a checklist to ensure that all selected topics were covered. Interviews lasted for about 90 minutes. Two or three researchers were present at each interview except for five interviews, which were performed by only one researcher. One of the interviewers led the interview, while the others took notes and asked additional questions for completeness or clarification. After consent was given by the audio recordings were made of each interview. All interviewees consented.

The audio recordings were transcribed word by word and the transcriptions were validated in two steps to eliminate un-clarities and misunderstandings. These steps were: (i) another researcher, primarily one who was present at the interview, reviewed the transcript, and (ii) the transcript was sent to the interviewee with sections for clarification highlighted and the interviewee had a chance to edit the transcript to correct errors or explain what they meant. These modifications were included into the final version of the transcript, which was used for further data analysis.

Table 1. Overview of the companies covered by this case study. At company F a major process change was taking place at the time of the study and data specific to the previous waterfall-based process are marked with 'previous'.

Company	A	B	C	D	E	F
Type of company	Software development, embedded products	Consulting	Software development	Systems engineering, embedded products	Software development, embedded products	Software development, embedded products
#employees in softw. dev. of targeted org.	125-150	135	500	50-100	300-350	1
#employees in typical project	10	Mostly 4-10, but varies greatly	50-80	software developers: 10-20	6-7 per team, 10-15 teams	Previous process: 800-1,000 person years
Distributed	No	Collocated (per project, often on-site at customer)	Yes	Yes	Yes	Yes
Domain / System type	Computer networking equipment	Advisory/technical services, appl. managem.	Rail traffic management	Automotive	Telecom	Telecom
Source of reqts	Market driven	Bespoke	Bespoke	Bespoke	Bespoke, market driven	Bespoke, market driven
Main quality focus	Availability, performance, security	Depends on customer focus	Safety	Safety	Availability, Performance, reliability, security	Performance, stability
Certification	No software related certification	No	ISO9001, ISO14001, OHSAS18001	ISO9001, ISO14001	ISO9001, ISO14001	ISO9001
Process Model	Iterative	Agile in variants	Waterfall	RUP, Scrum	Scrum, eRUP, a sprints is 3 months	Iterative with gate decisions (agile) Previous: Waterfall
Duration of a typical project	6-18 months	No typical project	1-5 years to first delivery, then new software release for 1-10 years	1-5 years to first delivery, then new software releases for 1-10 years	1 year	Previous process 2 years
#requirements in typical project	100 (20-30 pages HTML)	No typical project	600-800 at system level	For software: 20-40 use cases	500-700 user stories	Previous process:14,000
#test cases in a typical project	~1,000 test cases	No typical project	250 at system level		11,000+	Previous process 200,000 at platform level, 7,000 at system level
Product Lines	Yes	No	Yes	Yes	Yes	Yes
Open Source	Yes	Yes incl. contributions	Yes, partly	No	No	Yes (w agile dev model)

Table 2. Overview of interviewees' roles at their companies incl. level of experience in that role; S(enior) = more than 3 years, or J(unior) = up to 3 years. Xn refers to interviewee n at company X. Note: most interviewees have additional previous experience.

Role	**A**	**B**	**C**	**D**	**E**	**F**
Requirements engineer						F1 (S), F6 (S), F7 (S)
Systems architect				D3 (J)	E1 (S)	F4 (S)
Software developer		B1 (J), B2 (S), B3 (S)				F13 (S)
Test engineer	A2 (S)		C1 (S), C2 (J)	D2 (S)	E3 (S)	F9 (S), F10 (S), F11 (J), F12 (S), F14 (S)
Project manager	A1(J)		C3 (S)	D1 (S)		F3 (J), F8 (S)
Product manager	A3(S)				E2 (S)	
Process manager						F2 (J), F5 (S), F15 (J)

The transcripts were divided into chunks of text consisting of a couple of sentences each to enable referencing specific parts of the interviews. Furthermore, an anonymous code was assigned to each interview and the names of the interviewees were removed from the transcripts before data analysis in order to ensure anonymity of the interviewees.

3.4 Data Analysis

Once the data was collected through the interviews and transcribed (see Figure 1), a three-stage analysis process was performed consisting of: coding, abstraction and grouping, and interpretation. These multiple steps were required to enable the researchers to efficiently navigate and consistently interpret the huge amounts of qualitative data collected, comprising more than 300 pages of interview transcripts.

Coding of the transcripts, i.e. the chunks, was performed to enable locating relevant parts of the large amounts of interview data during analysis. A set of codes, or keywords, based on the research and interview questions was produced, initially at a workshop with the participating researchers. This set was then iteratively updated after exploratory coding and further discussions. In the final version, the codes were grouped into multiple categories at different abstraction levels, and a coding guide was developed. To validate that the researchers performed coding in a uniform way, one interview transcript was selected and coded by all researchers. The differences in coding were then discussed at a workshop and the coding guide was subsequently improved. The final set of codes was applied to all the transcripts. The

coding guide and some coding examples are published by Runeson et al. (2012, Appendix D).

Abstraction and grouping of the collected data into statements relevant to the goals and questions for our study was performed in order to obtain a manageable set of data that could more easily be navigated and analysed. The statements can be seen as an index, or common categorisation of sections belonging together, in essence a summary of them as done by Gorschek and Wohlin (2004, 2006), Petterson et al. (2008) and Höst et al. (2010). The statements were each given a unique identifier, title and description. Their relationship to other statements, as derived from the transcripts, was also abstracted. The statements and relationships between them were represented by nodes connected by directional edges. Figure 3 shows an example of the representation designed and used for this study. In particular, the figure shows the abstraction of the interview data around cross-role reviews of requirements, represented by node N4. For example, the statement 'cross-role reviews' was found to contribute to statements related to requirements quality. Each statement is represented by a node. For example, N4 for 'cross-role review', and N1, N196 and N275 for the statements related to requirements quality. The connections between these statements are represented by a 'contributes to' relationship from N4 to each of N1, N196 and N275. These connections are denoted by a directional edge tagged with the type of relationship. For example, the tags 'C' for 'contributes to', 'P' for 'prerequisite for' and 'DC' for 'does not contribute to'. In addition, negation of one or both of the statements can be denoted by applying a post- or prefix 'not' (N) to the connection. The type of relationships used for modelling the connections between statements were discussed, defined and agreed on in a series of work meetings. Traceability to the original statements and the relationships between them was captured and maintained by noting the id of the relevant source chunk, both for nodes and for edges. This is not shown in Figure 3.

The identified statements including relationships to other statements were extracted per transcript by one researcher per interview. To ensure a consistent abstraction among the group of researchers and to enhance completeness and

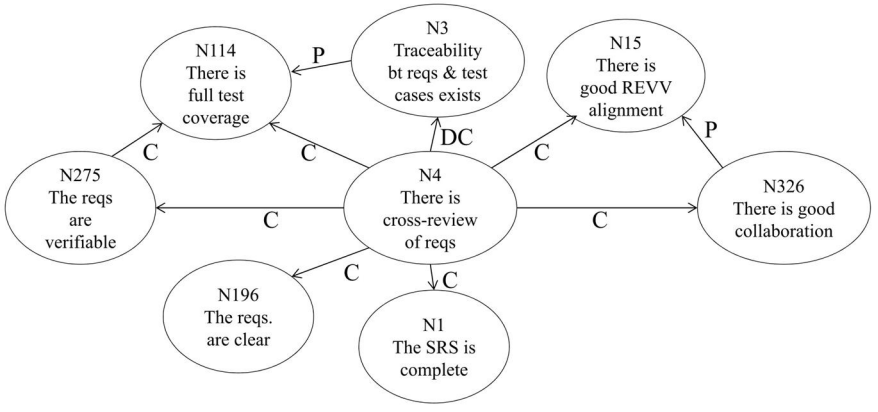


Figure 3. Part of the abstraction representing the interpretation of the interviewee data. The relationships shown denote C - contribute to, P - prerequisite for, and DC – does not contribute to.

correctness, the abstraction for each interview was reviewed by at least one other researcher and agreed after discussing differences of opinion. The nodes and edges identified by each researcher were merged into one common graph consisting of 341 nodes and 552 edges.

Interpretation of the collected evidence involved identifying the parts of the data relevant to a specific research question. The abstracted statements derived in the previous step acted as an index into the interview data and allowed the researchers to identify statements relevant to the research questions of challenges and practices. This interpretation of the interview data was performed by analysing a graphical representation of the abstracted statements including the connections between them. Through the analysis nodes and clusters of nodes related to the research questions were identified. This is similar to explorative coding and, for this paper, the identified codes or clusters represented REVV alignment challenges and practices with one cluster (code) per challenge and per practice. Due to the large amount of data, the analysis and clustering was initially performed on sub-sets of the graphical representation, one for each company. The identified clusters were then iteratively merged into a common set of clusters for the interviews for all companies. For example, for the nodes shown in Figure 3 the statements 'The requirements are clear' (N196) and 'The requirements are verifiable' (N275) were clustered together into the challenge 'Defining clear and verifiable requirements' (challenge Ch3.2, see Section 4.1.3) based on connections (not shown in the example) to other statements reflecting that this leads to weak alignment.

Even with the abstracted representation of the interview transcripts, the interpretation step is a non-trivial task which requires careful and skilful consideration to identify the nodes relevant to specific research questions. For this reason, the clustering that was performed by Bjarnason was reviewed and agreed with Runeson. Furthermore, the remaining un-clustered nodes were reviewed by Engström, and either mapped to existing clusters, suggested for new clusters or judged to be out of scope for the specific research questions. This mapping was then reviewed and agreed with Bjarnason.

Finally, the agreed clusters were used as an index to locate the relevant parts of the interview transcripts (through traces from the nodes and edges of each cluster to the chunks of text). For each identified challenge and practice, and mapping between them, the located parts of the transcriptions were then analysed and interpreted, and reported in this paper in Sections 4.1, 4.2 and 4.3, respectively for challenges, practices, and the mapping.

3.5 Threats to Validity

There are limitations and threats to the validity to all empirical studies, and so also for this case study. As suggested by Runeson et al (2009, 2012), the construct validity, external validity and reliability were analysed in the phases leading up to the *analysis* phase of the case study, see Figure 1. We also report measures taken to improve the validity of the study.

3.5.1 Construct validity

Construct validity refers to how well the chosen research method has captured the concepts under study. There is a risk that academic researchers and industry practitioners may use different terms and have different frames of reference, both between and within these categories of people. In addition, the presence of researchers may threaten the interviewees and lead them to respond according to assumed expectations. The selection of interviewees may also give a limited or unbalanced view of the construct. In order to mitigate these risks, we took the following actions in the design step:

- *Design of the interview guide and reference model.* The interview guide was designed based on the research questions and reviewed for completeness and consistency by other researchers. It was piloted during two interviews and then revised again after another six. The risk that the language and terms used may not be uniformly understood was addressed by producing a conceptual model (see Figure 2), which was shown to the interviewees to explain the terminology. However, due to the semi-structured nature of the guide and the different interviewers involved the absence of interviewee data for a certain concept, challenge or practice cannot be interpreted as the absence of this item either in the interviewees experience or in the company. For similar reasons, the results do not include any ranking or prioritisation as to which challenges and practices are the most frequent or most effective.

- *Prolonged involvement.* The companies were selected so that at least one of the researchers had a long-term relation with them. This relationship helped provide the trust needed for openness and honesty in the interviews. To mitigate the bias of knowing the company too well, all but five interviews (companies D and E) were conducted by more than one interviewer.

- *Selection of interviewees*. To obtain a good representation of different aspects, a range of roles were selected to cover requirement, development and testing, and also engineers as well as managers, as reported in Table 2. The aim was to cover the relevant aspects described in the conceptual model, produced during the *Definition* phase (see Section 3.1 Figures 1 and 2). There is a risk that the results might be biased due to a majority of the interviewees being from Company F. However, the results indicate that this risk was minor, since a majority of the identified items (see Section 4) could be connected to multiple companies.

- *Reactive bias*: The presence of a researcher might limit or influence the outcome either by hiding facts or responding after assumed expectations. To reduce this threat the interviewees were guaranteed anonymity both within the company and externally. In addition, they were not given any rewards for their participation and had the right to withdraw at any time without requiring an explanation, though no interviewees did withdraw. This approach indicated that we were interested in obtaining a true image of their reality and encouraged the interviewees to share this.

3.5.2 Internal validity

Even though the conclusions in this paper are not primarily about causal relations, the identification of challenges and practices somewhat resembles identifying factors in casual relations. In order to mitigate the risk of identifying incorrect factors, we used data source triangulation by interviewing multiple roles at a

company. Furthermore, extensive observer triangulation was applied in the analysis by always including more than one researcher in each step. This strategy also partly addressed the risk of incorrect generalisations when abstracting challenges and practices for the whole set of companies. However, the presented results represent one possible categorisation of the identified challenges and practices. This is partly illustrated by the fact that not all identified practices can be connected to a challenge.

The interviews at one of the case companies were complicated by a major process change that was underway at the time of the study. This change posed a risk of confusing the context for which a statement had been experienced; the previous (old) way of working or the newly introduced agile practices. To mitigate this risk, we ensured that we correctly understood which process the response concerned, i.e. the previous or the current process.

Furthermore, due to the nature of semi-structured interviews in combination with several different interviewers it is likely that different follow-on questions were explored by the various researchers. This risk was partly mitigated by jointly defining the conceptual model and agreeing on a common interview guide that was used for all interviews. However, the fact remains that there are differences in the detailed avenues of questioning which has resulted in only being able to draw conclusions concerning what was actually said at the interviews. So, for example, if the completeness of the requirements specification (Ch3.2) was not explicitly discussed at an interview no conclusions can be drawn concerning if this is a challenge or not for that specific case.

3.5.3 External validity

For a qualitative study like this, external validity can never be assured by sampling logic and statistical generalisation, but by analytical generalisation which enables drawing conclusions and, under certain conditions, relating them also to other cases (Robson 2002, Runeson 2012). This implies that the context of the study must be compared to the context of interest for the findings to be generalised to. To enable this process, we report the characteristics of the companies in as much detail as possible considering confidentiality (see Table 1). The fact that six different companies of varying size and domain are covered by the study, and some results are connected to the variations between them indicates that the results are more general than if only one company had been studied. But, of course, the world consists of more than six kinds of companies, and any application of the results of this study need to be mindfully tailored to other contexts.

3.5.4 Reliability

The reliability of the study relates to whether the same outcome could be expected with another set of researchers. For qualitative data and analysis, which are less procedural than quantitative methods, exact replication is not probable. The analysis lies in interpretation and coding of words, and the set of codes would probably be partly different with a different set of researchers.

To increase the reliability of this study and to reduce the influence by single researchers, several researchers have taken part in the study in different roles. All findings and each step of analysis have been reviewed by and agreed with at least

one other researcher. In addition, a systematic and documented research process has been applied (see Figure 1) and a trace of evidence has been retained for each analysis steps. The traceability back to each source of evidence is documented and kept even in this report to enable external assessment of the chain of evidence, if confidentially agreements would allow.

Finally, the presentation of the findings could vary depending on categorisation of the items partly due to variation in views and experience of individual researchers. For example, a challenge in achieving alignment such as Ch2 *Collaborating successfully* (see Section 4.1.2) could be identified also as a practice at the general level, e.g. *to collaborate successfully* could be defined as an alignment practice. However, we have chosen to report specific practices that may improve collaboration and thereby REVV alignment. For example, P1.1 *Customer communication at all requirements levels and phases* can support improved coordination of requirements between the customer and the development team. To reduce the risk of bias in this aspect, the results and the categorisation of them was first proposed by one researcher and then reviewed by four other researchers leading to modifications and adjustments.

4 Results

Practitioners from all six companies in the study found alignment of RE with VV to be an important, but challenging, factor in developing products. REVV alignment was seen to affect the whole project life cycle, from the contact with the customer and throughout software development. The interviewees stated clearly that good alignment is essential to enable smooth and efficient software development. It was also seen as an important contributing factor in producing software that meets the needs and expectations of the customers. A software developer stated that alignment is 'very important in creating the right system' (B1:27[2]). One interviewee described the customer's view of a product developed with misaligned requirements as: 'There wasn't a bug, but the behaviour of the functionality was interpreted or implemented in such a way that it was hard to do what the customer [originally] intended.' (A3:43) Another interviewee mentioned that alignment between requirements and verification builds customer trust in the end product since good alignment allows the company to 'look into the customer's eyes and explain what have we tested… on which requirements' (D2:10).

In general, the interviewees expressed that weak and unaligned communication of the requirements often cause inconsistencies that affect the verification effort. A common view was that these inconsistencies, caused by requirements that are misunderstood, incorrect or changed, or even un-communicated, leads to additional work in updating and re-executing test cases. Improved alignment, on the other hand, was seen to make 'communication between different levels in the V-model a lot easier' (E3:93). One of the interviewed testers stated: 'Alignment is necessary. Without it we [testers] couldn't do our job at all.' (C1:77)

[2] Reference to source is given by interviewee code, see Table 2.

Below, we present the results concerning the challenges of alignment (Ch1-Ch10) and the practices (P1-P10) used, or suggested, by the case companies to address REVV challenges. Table 3 provides an overview of the challenges found for each company, while Table 4 contains an overview of the practices. Table 6 shows which challenges each practices is seen to address.

4.1 Alignment Challenges

The alignment challenges identified through this study are summarised in Table 3. Some items have been categorised together as one challenge, resulting in 10 main challenges where some consist of several related challenges. For example, Ch3 *Requirements specification quality* consists of three challenges (Ch3.1-Ch3.3) concerning different aspects of requirements quality. Each challenge including sub items is described in the subsections that follow.

4.1.1 Challenge 1: Aligning goals and perspectives within an organisation (Ch1)

The alignment of *goals* throughout the organisation was mentioned by many interviewees as vital in enabling cooperation among different organisational units (see challenge 2 in Section 4.1.2). However, goals were often felt to be missing or unclearly defined, which could result in 'making it difficult to test [the goals]' (B3:17). In several companies problems with differing and unaligned goals were seen to affect the synchronisation between requirements and testing, and cause organisational units to counteract each other in joint development projects. For example, a product manager mentioned that at times, requirement changes needed from a business perspective conflicted with the goals of the development units; 'They [business roles] have their own directives and … schedule target goals' and 'they can look back and see which product was late and which product was good' (A3:74). In other words, misaligned goals may have an impact on both time schedules and product quality.

Many interviewees described how awareness and understanding of different *perspectives* on the problem domain is connected to better communication and cooperation, both towards the customers and external suppliers, and internally between competence areas and units. When there is a lack of aligned perspectives, the customer and the supplier often do not have the same understanding of the requirements. This may result in 'errors in misunderstanding the requirements' (B3:70). Lack of insight into and awareness of different perspectives was also seen to result in decisions (often made by other units) being questioned and requirements changed at a late stage in the development cycle with a subsequent increase in cost and risk. For example, a systems architect described that in a project where there is a 'higher expectations on the product than we [systems architect] scoped into it' (E1:20) a lot of issues and change requests surface in the late project phases. A software developer stated concerning the communication between requirements engineers and developers that 'if both have a common perspective [of technical possibilities], then it would be easier to understand what [requirements] can be set and what cannot be set' (F13:29). Or in other words, with increased common understanding technically infeasible requirements can be avoided at an early stage.

Table 3. Alignment challenges mentioned for each company. Note: a blank cell means that the challenge was not mentioned during the interviews, not that it is not experienced.

	Id	Challenge	Company					
			A	B	C	D	E	F
	Ch1	Aligning goals and perspectives within an organisation	X	X	X		X	X
	Ch2	Cooperating successfully	X		X	X	X	X
Req spec quality	Ch3.1	Defining clear and verifiable requirements			X	X	X	X
	Ch3.2	Defining complete requirements		X		X	X	X
	Ch3.3	Keeping requirements documents updated						X
VV quality	Ch4.1	Full test coverage	X	X	X	X		X
	Ch4.2	Defining a good verification process						X
	Ch4.3	Verifying quality requirements		X		X		X
	Ch5	Maintaining alignment when requirements change	X		X			X
Req's abstract levels	Ch6.1	Defining requirements at abstraction level well matched to test cases				X		X
	Ch6.2	Coordinating requirements at different abstraction levels	X					X
Traceability	Ch7.1	Tracing between requirements and test cases	X	X	X	X		X
	Ch7.2	Tracing between requirements abstraction levels		X	X	X		
	Ch8	Time and resource availability			X		X	X
	Ch9	Managing a large document space			X	X		X
	Ch10	Outsourcing of components or testing				X		X

Weak alignment of goals and perspectives implies a weak coordination at higher organisational levels and that strategies and processes are not synchronised. As stated by a process manager, the involvement of many separate parts of an organisation then leads to 'misunderstandings and misconceptions and the use of different vocabulary' (F2:57). In addition, a test engineer at Company A mentioned that for the higher abstraction levels there were no attempts to synchronise, for example, the testing strategy with the goals of development projects to agree on important areas to focus on (A2:105). Low maturity of the organisation was thought to contribute to this and result in the final product having a low degree of correspondence to the high-level project goals. A test engineer said: 'In the long run, we would like to get to the point where this [product requirements level] is aligned with this [testing activities].' (A2:119)

4.1.2 Challenge 2: Cooperating successfully (Ch2)

All of the companies included in our study described close cooperation between roles and organisational units as vital for good alignment and coordination of both people and artefacts. Weak cooperation is experienced to negatively affect the alignment, in particular at the product level. A product manager stated that 'an "us and them" validation of product level requirements is a big problem' (A3:058-059). Ensuring clear agreement and communication concerning which requirements to support is an important collaboration aspect for the validation. At Company F (F12:063) lack of cooperation in the early phases in validating requirements has been experienced to result in late discovery of failures in meeting important product requirements. The development project then say at a late stage: 'We did not approve these requirements, we can't solve it' (F12:63) with the consequence that the requirements analysis has to be re-done. For Company B (consulting in different organisations) cooperation and communication was even described as being prioritised above formal documentation and processes, expressed as: 'We have succeeded with mapping requirements to tests since our process is more of a discussion' (B3:49). Several interviewees described that alignment at product and system level, in particular, is affected by how well people cooperate (C2:17, E1:44, 48, E2:48, F4:66, F15:46). When testers have a good cooperation and frequently communicate with both requirements-related and development-related roles, this leads to increased alignment (E3:093).

Organisational boundaries were mentioned as further complicating and hindering cooperation between people for two of the companies, namely companies E and F. In these cases, separate organisational units exist for requirements (E2:29, E3:94, F2:119), usability (F10:108) and testing (F3:184). As one interviewee said: 'it is totally different organisations, which results in ... misunderstandings and misconceptions...we use different words' (F02:57). Low awareness of the responsibilities and tasks of different organisational units was also claimed to negatively affect alignment (F2:264). This may result in increased lead times (E1:044, F15:033), need for additional rework (E1:150, E1:152), and conflicts in resource allocation between projects (F10:109, E1:34).

4.1.3 Challenge 3: Good reqs specification quality (Ch3)

'If we don't have good requirements the tests will not be that good.' (D3:14) When the requirement specification is lacking the testers need to guess and make up the missing information since 'the requirements are not enough for writing the software and testing the software' (D3:19). This both increases the effort required for testing and the risk of misinterpretation and missing vital customer requirements. One process manager expressed that the testability of requirements can be improved by involving testers and that 'one main benefit [of alignment] is improving the requirements specifications' (F2:62). A test leader at the same company identified that a well aligned requirements specification (through clear agreement between roles and tracing between artefacts) had positive effects such as 'it was very easy to report when we found defects, and there were not a lot of discussions between testers and developers, because everyone knew what was expected' (F9:11).

There are several aspects to good requirements that were found to relate to alignment. In the study, practitioners mentioned good requirements as being

verifiable, clear, complete, at the right level of abstraction, and up-to-date. Each aspect is addressed below.

- **Defining clear and verifiable requirements (Ch3.1)** was mentioned as a major challenge in enabling good alignment of requirements and testing, both at product and at detailed level. This was mentioned for four of the six companies covered by our study, see Table 3. Unclear and non-verifiable requirements were seen as resulting in increased lead times and additional work in later phases in clarifying and redoing work based on unclear requirements (F2:64, D1:80). One test manager said that 'in the beginning the requirements are very fuzzy. So it takes time. And sometimes they are not happy with our implementation, and we have to do it again and iterate until it's ready.' (F11:27, similar in E3:44.) Failure to address this challenge ultimately results in failure to meet the customer expectations with the final product. A project manager from company D expressed this by saying that non-verifiable requirements is the reason 'why so many companies, developers and teams have problems with developing customer-correct software' (D1:36).

- **Defining complete requirements (Ch3.2)** was claimed to be required for successful alignment by interviewees from four companies, namely companies B, D, E and F. As expressed by a systems architect from Company D, 'the problem for us right now is not [alignment] between requirements and testing, but that the requirements are not correct and complete all the time' (D3:118). Complete requirements support achieving full test coverage to ensure that the full functionality and quality aspects are verified. (F14:31) When testers are required to work with incomplete requirements, additional information is acquired from other sources, which requires additional time and effort to locate (D3:19).

- **Keeping requirements documentation updated (Ch3.3)** Several interviewees from company F described how a high frequency of change leads to the requirements documentation not being kept updated, and consequently the documentation cannot be relied on (F14:44, F5:88). When a test for a requirement then fails, the first reaction is not: 'this is an error', but rather 'is this really a relevant requirement or should we change it' (F5:81). Mentioned consequences of this include additional work to locate and agree to the correct version of requirements and rework (F3:168) when incorrect requirements have been used for testing. Two sources of requirements changes were mentioned, namely requested changes that are formally approved (F14:50), but also changes that occur as the development process progresses (during design, development etc.) that are not raised as formal change requests (F5:82, F5:91, F11:38). When the requirements documentation is not reliable, the projects depend on individuals for correct requirements information. As expressed by one requirements engineer: 'when you lose the people who have been involved, it is tough. And, things then take more time.' (F1:137)

4.1.4 Challenge 4: Validation and verification quality

Several issues with validation and verification were mentioned as alignment challenges that affect the efficiency and effectiveness of the testing effort. One

process manager with long experience as a tester said: 'We can run 100,000 test cases but only 9% of them are relevant.' (F15:152) Testing issues mentioned as affecting alignment were: obtaining full test coverage, having a formally defined verification process and the verification of quality requirements.

- **Full test coverage (Ch4.1)** Several interviewees described full test coverage of the requirements as an important aspect of ensuring that the final product fulfils the requirements and the expectations of the customers. As one software developer said: 'having full test coverage with unit tests gives a better security... check that I have interpreted things correctly with acceptance tests' (B1:117). However, as a project manager from Company C said: 'it is very hard to test everything, to think about all the complexities' (C3:15). *Unclear* (Ch3.2, C1:4) and *non-verifiable requirements* (Ch3.1, A1:55, D1:78, E1:65) were mentioned as contributing to difficulties in achieving full test coverage of requirements for companies A, B, D and E. For certain requirements that are expressed in a verifiable way a project manager mentioned that they cannot be tested due to limitations in the process, competence and test tools and environments (A1:56). To ensure full test coverage of requirements the testers need knowledge of the full set of requirements, which is impeded in the case of *incomplete requirements specifications* (Ch3.3) where features and functionality are not described (D3:16). This can also be the case for *requirements defined at a higher abstraction level* (F2:211, F14:056). *Lack of traceability* between requirements and test cases was stated to making it harder to know when full *test coverage* has been obtained (A1:42). For company C, traceability was stated as time consuming but necessary to ensure and demonstrate full test coverage, which is mandatory when producing safety-critical software (C1:6, C1:31). Furthermore, obtaining sufficient coverage of the requirements requires analysis of both the requirement and the connected test cases (C1:52, D3:84, F14:212). As one requirements engineer said, 'a test case may cover part of a requirement, but not test the whole requirement' (F7:52). Late requirements changes was mentioned as a factor contributing to the challenge of full test coverage (C1:54, F7:51) due to the need to update the affected test cases, which is hampered by failure to keep the requirements specification updated after changes (Ch3.5, A2:72, F15:152).

- **Having a verification process (Ch4.2)** was mentioned as directly connected to good alignment between requirements and test. At company F, the on-going shift towards a more agile development process had resulted in the verification unit operating without a formal process (F15:21). Instead each department and project 'tries to work their own way... that turns out to not be so efficient' (F15:23), especially so in this large organisation where many different units and roles are involved from the initial requirements definition to the final verification and launch. Furthermore, one interviewee who was responsible for defining the new verification process (F15) said that 'the hardest thing [with defining a process] is that there are so many managers ... [that don't] know what happens one level down'. In other words, a verification process that supports requirements-test alignment needs to be agreed with the whole organisation and at all levels.

- **Verifying quality requirements (Ch4.3)** was mentioned as a challenge for companies B, D and F. Company B has verification of quality in focus with continuous monitoring of quality levels in combination with frequent releases; 'it is easy to prioritise performance optimisation in the next production release' (B1:52). However, they do not work proactively with quality requirements. Even though they have (undocumented) high-level quality goals the testers are not asked to use them (B1:57, B2:98); 'when it's not a broken-down [quality] requirement, then it's not a focus for us [test and development]' (B3:47). Company F does define formal quality requirements, but these are often not fully agreed with development (F12:61). Instead, when the specified quality levels are not reached, the requirements, rather than the implementation, are changed to match the current behaviour, thus resigning from improving quality levels in the software. As one test engineer said: 'We currently have 22 requirements, and they always fail, but we can't fix it' (F12:61). Furthermore, defining verifiable quality requirements and test cases was mentioned as challenging, especially for usability requirements (D3:84, F10:119). Verification is then faced with the challenge of subjectively judging if a requirement is passed or failed (F2:46, F10:119). At company F, the new agile practices of detailing requirements at the development level together with testers was believed to, at least partly, address this challenge (F12:65). Furthermore, additional complication is that some quality requirements can only be verified through analysis and not through functional tests (D3:84).

4.1.5 Challenge 5: Maintaining alignment when requirements change (Ch5)

Most of the companies of our study face the challenge of maintaining alignment between requirements and tests as requirements change. This entails ensuring that both artefacts and tracing between them are updated in a consistent manner. Company B noted that the impact of changes is specifically challenging for test since test code is more sensitive to changes than requirements specifications. 'That's clearly a challenge, because [the test code is] rigid, as you are exemplifying things in more detail. If you change something fundamental, there are many tests and requirements that need to be modified' (B3:72).

Loss of traces from test cases to requirements over time was also mentioned to cause problems. When test cases for which traces have been outdated or lost are questioned, then 'we have no validity to refer to ... so we have to investigate' (A2:53). In company A, the connection between requirements and test cases are set up for each project (A2:71): 'This is a document that dies with the project'; a practice found very inefficient. Other companies had varying ambitions of a continuous maintenance of alignment and traces between the artefacts. A key for maintaining alignment when requirements change is that the requirements are actively used. When this is not the case there is a need for obtaining requirements information from other sources. This imposes a risk that 'a requirement may have changed, but the software developers are not aware of it' (D3:97).

Interviewees implicitly connected the traceability challenge to tools, although admitting that 'a tool does not solve everything... Somebody has to be responsible for maintaining it and to check all the links ... if the requirements change' (C3:053).

With or without feasible tools, tracing also requires personal assistance. One test engineer said, 'I go and talk to him and he points me towards somebody' (A2:195).

Furthermore, the *frequency of changes* greatly affects the extent of this challenge and is an issue when trying to establish a base-lined version of the requirements. Company C has good tool support and traceability links, but require defined versions to relate changes to. In addition, they have a product line, which implies that the changes must also be coordinated between the platform (product line) and the applications (products) (C3:019, C3:039).

4.1.6 Challenge 6: Requirements abstraction levels (Ch6)

REVV alignment was described to be affected by the abstraction levels of the requirements for companies A, D and F. This includes the relationship to the abstraction levels of the test artefacts and ensuring consistency between requirements at different abstraction levels.

- **Defining requirements at abstraction levels well-matched to test cases (Ch6.1)** supports defining test cases in line with the requirements and with a good coverage of them. This was mentioned for companies D and F. A specific case of this at company D is when the testers 'don't want to test the complete electronics and software system, but only one piece of the software' (D3:56). Since the requirements are specified at a higher abstraction level than the individual components, the requirements for this level then need to be identified elsewhere. Sources for information mentioned by the interviewees include the design specification, asking people or making up the missing requirements (D3:14). This is also an issue when retesting only parts of a system which are described by a high-level requirement to which many other test cases are also traced (D3:56). Furthermore, synchronising the abstraction levels between requirements and test artefacts was mentioned to enhance coverage (F14:31).
- **Coordinating requirements at different abstraction levels (Ch6.2)** when breaking down the high-level requirements (such as goals and product concepts) into detailed requirements at system or component level was mentioned as a challenge by several companies. A product manager described that failure to coordinate the detailed requirements with the overall concepts could result in that 'the intention that we wanted to fulfil is not solved even though all the requirements are delivered' (A3:39). On the other hand, interviewees also described that the high-level requirements were often vague at the beginning when 'it is very difficult to see the whole picture' (F12:144) and that some features are 'too complex to get everything right from the beginning' (A3:177).

4.1.7 Challenge 7: Tracing between artefacts (Ch7)

This challenge covers the difficulties involved in tracing requirements to test cases, and vice versa, as well as, tracing between requirements at different abstraction levels. Specific tracing practices identified through our study are described in Sections 4.2.6 and 4.2.7.

- **Tracing between requirements and test cases (Ch7.1). The** most basic kind of traceability, referred to as 'conceptual mapping' in Company A (A2:102), is

having a line of thought (not necessarily documented) from the requirements through to the defining and assessing of the test cases. This cannot be taken for granted. Lack of this basic level of tracing is largely due to weak awareness of the role requirements in the development process. As a requirements process engineer in Company F says, 'One challenge is to get people to understand why requirements are important; to actually work with requirements, and not just go off and develop and do test cases which people usually like doing' (F5:13).

Tracing by using matrices to map between requirements and test cases is a major cost issue. A test architect at company F states, that 'we don't want to do that one to one mapping all the way because that takes a lot of time and resources' (F10:258). Companies with customer or market demands on traceability, e.g. for safety critical systems (companies C and D), have full traceability in place though 'there is a lot of administration in that, but it has to be done' (C1:06). However, for the other case companies in our study (B3:18, D3:45; E2:83; F01:57), it is a challenge to implement and maintain this support even though tracing is generally seen as supporting alignment. Company A says 'in reality we don't have the connections' (A2:102) and for Company F 'in most cases there is no connection between test cases and requirements' (F1:157). Furthermore, introducing traceability may be costly due to large legacies (F1:57) and maintaining traceability is costly. However, there is also a cost for lack of traceability. This was stated by a test engineer in Company F who commented on degrading traceability practices with 'it was harder to find a requirement. And if you can't find a requirement, sometimes we end up in a phase where we start guessing' (F12:112).

Company E has previously had a tradition of 'high requirements on the traceability on the products backwards to the requirements' (E2:83). However, this company foresees problems with the traceability when transitioning towards agile working practices, and using user stories instead of traditional requirements. A similar situation is described for Company F, where they attempt to solve this issue by making the test cases and requirements one; 'in the new [agile] way of working we will have the test cases as the requirements' (F12:109).

Finally, traceability for quality (a.k.a. non-functional) requirements creates certain challenges, 'for instance, for reliability requirement you might ... verify it using analysis' (D3:84) rather than testing. Consequently, there is no single test case to trace such a quality requirement to, instead verification outcome is provided through an analysis report. In addition, tracing between requirements and test cases is more difficult 'the higher you get' (B3:20). If the requirements are at a high abstraction level, it is a challenge to define and trace test cases to cover the requirements.

- **Tracing between requirements abstraction levels (Ch7.2)** Another dimension of traceability is vertical tracing between requirements at different abstraction levels. Company C operates with a detailed requirements specification, which for some parts consists of sub-system requirements specifications (C1:31). In this case, there are no special means for vertical traceability, but pointers in the text. It is similar in Company D, where a system architect states that 'sometimes it's not done on each individual requirement but only on maybe a heading level

or something like that' (D3:45). Company F use a high-end requirements management tool, which according to the requirements engineer 'can trace the requirement from top level to the lowest implementation level' (F7:50).

Company E has requirements specifications for different target groups, and hence different content; one market oriented, one product oriented, and one with technical details (E1:104). The interviewee describes tracing as a 'synch activity' without specifying in more detail. Similarly, Company F has 'roadmaps' for the long term development strategy, and there is a loosely coupled 'connection between the roadmaps and the requirements' to balance the project scope against strategy and capacity (F11:50).

4.1.8 Challenge 8: Time and resource availability (Ch8)

In addition to the time consuming task of defining and maintaining traces (Ch7) further issues related to time and resources were brought forward in companies C, E and F. Without sufficient resources for validation and verification the amount of testing that can be performed is not sufficient for the demands on functionality and quality levels expected of the products. The challenge of planning for *enough test resources* is related to the alignment between the proposed requirements and the time and resources required to sufficiently test them. A requirements engineer states that 'I would not imagine that those who are writing the requirements in anyway are considering the test implications or the test effort required to verify them' (F6:181). A test manager confirms this view (F14:188). It is not only a matter of the amount of resources, but also in which time frame they are available (E1:18). Furthermore, availability of all the necessary *competences and skills* within a team was also mentioned as an important aspect of ensuring alignment. A software developer phrased it: 'If we have this kind of people, we can set up a team that can do that, and then the requirements would be produced properly and hopefully 100% achievable' (F13:149). In addition, experienced individuals were stated to contribute to strengthening the alignment between requirements and testing, by being 'very good at knowing what needs to be tested and what has a lower priority' (C2:91), thereby increasing the test efficiency. In contrast, inexperienced testing teams were mentioned for Company C as contributing to weaker alignment towards the overall set of requirements including goals and strategies since they 'verify only the customer requirements, but sometimes we have hazards in the system which require the product to be tested in a better way' (C2:32-33).

4.1.9 Challenge 9: Managing a large document space (Ch9)

The main challenge regarding the information management problems lies in the sheer numbers. A test engineer at Company F estimates that they have accumulated 50,000 requirements in their database. In addition, they have 'probably hundreds of thousands of test cases' (F2:34, F12:74). Another test engineer at the same company points out that this leads to information being *redundant* (F11:125), which consequently may lead to inconsistencies. A test engineer at Company D identifies the *constant change* of information as a challenge; they have difficulties to work against the same baseline (D2:16).

Another test engineer at Company F sees information management as a tool issue. He states that 'the requirements tool we have at the moment is not easy to

work with....Even, if they find the requirements they are not sure they found the right version' (F9:81). In contrast, a test engineer at company C is satisfied with the ability to find information in the same tool (C2). A main difference is that at Company F, 20 times as many requirements are handled than at Company C.

The investment into introducing explicit links between a huge legacy of requirements and test cases is also put forward as a major challenge for companies A and F. In addition, connecting and integrating different tools was also mentioned as challenging due to separate responsibilities and competences for the two areas of requirements and testing (F5:95, 120).

4.1.10 Challenge 10: Outsourcing or offshoring of components or testing (Ch10)

Outsourcing and offshoring of component development and testing create challenges both in agreeing to which detailed requirements to implement and test, and in tracing between artefacts produced by different parties. Company D stresses that the *timing* of the outsourcing plays a role in the difficulties in tracing component requirement specifications to the internal requirements at the higher level; 'I think that's because these outsourcing deals often have to take place really early in the development.' (D3:92). Company F also mentions the timing aspect for acquisition of hardware components; 'it is a rather formal structured process, with well-defined deliverables that are slotted in time' (F6:21).

When testing is outsourced, the *specification* of what to test is central and related to the type of testing. The set-up may vary depending on competence or cultural differences etc. For example, company F experienced that cultural aspects influence the required level of detail in the specification; 'we [in Europe] might have three steps in our test cases, while the same test case with the same result, but produced in China, has eight steps at a more detailed level' (F15:179). A specification of what to test may be at a high level and based on a requirements specification from which the in-sourced party derives tests and executes. An alternative approach is when a detailed test specification is requested to be executed by the in-sourced party (F6:251-255).

4.2 Practices for Improved Alignment

This study has identified 27 different alignment practices, grouped into 10 categories. Most of the practices are applied at the case companies, though some are suggestions made by the interviewees. These categories and the practices are presented below and discussed and summarised in Section 5. In Section 4.3 they are mapped to the challenges that they are seen to address.

Table 4. Alignment practices and categories, and case companies for which they were mentioned. Experienced practices are marked with X, while suggested practices are denoted with S. Note: a blank cell means that the practice was not mentioned during the interviews. It does not mean that it is not applied at the company.

			Company					
Cat.	**Id**	**Description**	**A**	**B**	**C**	**D**	**E**	**F**
Requirements Engineering	P1.1	Customer communication at all requirements levels and phases		X	X	X	X	X
	P1.2	Development involved in detailing requirements	X	X				X
	P1.3	Cross-role requirements reviews	X		X	X	X	X
	P1.4	Requirements review responsibilities defined					X	X
	P1.5	Subsystem expert involved in requirements definition				X		X
	P1.6	Documentation of requirement decision rationales					S	S
Validation	P2.1	Test cases reviewed against requirements						X
	P2.2	Acceptance test cases defined by customer		X				
	P2.3	Product manager reviews prototypes	X				X	
	P2.4	Management base launch decision on test report						X
	P2.5	User / Customer testing		X		X	X	X
Verification	P3.1	Early verification start					X	X
	P3.2	Independent testing			X	X	X	
	P3.3	Testers re-use customer feedback from previous projects				X	X	X
	P3.4	Training off-shore testers			X			
Change	P4.1	Process for requirements changes involving VV	X		X	X	X	X
	P4.2	Product-line requirements practices	X		X			
	P5	Process enforcement			X			S
Tracing	P6.1	Document-level traces	X					
	P6.2	Requirements-test case traces						X
	P6.3	Test cases as requirements	X					X
	P6.4	Same abstraction levels for requirements and test spec			X	X		
	P7	Traceability responsibility role			X	X	X	
Tools	P8.1	Tool support for requirements and testing	X		X	X	X	X
	P8.2	Tool support for requirements-test case tracing	X		X	X	X	X
	P9	Alignment metrics, e.g. test coverage			X	X	X	X
	P10	Job rotation				S		S

4.2.1 Requirements engineering practices

Requirements engineering practices are at the core of aligning requirements and testing. This category of practices includes customer communication and involving development-near roles in the requirements process. The interviewees described close cooperation and team work as a way to improve RE practices (F12:146) and thereby the coordination with developers and testers and avoid a situation where product managers say '"redo it" when they see the final product' (F12:143).

- **Customer communication at all levels and in all phases of development (P1.1)** was mentioned as an alignment practice for all but one of the case companies. The communication may take the form of customer-supplier co-location; interaction with the customer based on executable software used for demonstrations or customer validation; or agreed acceptance criteria between customer and supplier. For the smaller companies, and especially those with bespoke requirements (companies B and C), this interaction is directly with a physical customer. In larger companies (companies E and F), and especially within market driven development, a *customer proxy* may be used instead of the real customer, since there is no assigned customer at the time of development or there is a large organisational distance to the customer. Company F assigns a person in each development team 'responsible for the feature scope. That person is to be available all through development and to the validation of that feature' (F2:109). Furthermore, early discussions about product roadmaps from a four to five year perspective are held with customers and key suppliers (F6:29) as an initial phase of the requirements process.

- **Involving developers and testers in detailing requirements (P1.2)** is another practice, especially mentioned by companies A and F. A product manager has established this as a deliberate strategy by conveying the vision of the product to the engineers rather than detailed requirements: 'I'm trying to be more conceptual in my ordering, trying to say what's important and the main behaviour.' (A3:51) The responsibility for detailing the specification then shifts to the development organisation. However, if there is a weak awareness of the customer or market perspectives, this may be a risky practice as 'some people will not [understand this] either because they [don't] have the background or understanding of how customers or end-users or system integrators think' (A3:47). Testers may be involved to ensure the testability of the requirements, or even specify requirements in the form of test cases. Company F was in the process of transferring from a requirements-driven organisation to a design-driven one. Splitting up the (previous) centralised requirements department resulted in 'requirements are vaguer now. So it's more up to the developers and the testers to make their own requirements.' (F12:17) Close cooperation around requirements when working in an agile fashion was mentioned as vital by a product manager from Company E: 'Working agile requires that they [requirements, development, and test] are really involved [in requirements work] and not only review.' (E2:083)

- **Cross-role requirements reviews (P1.3)** across requirements engineers and testers is another practice applied to ensure that requirements are understood and testable (A2:65, C3:69, F2:38, F7:7). The practical procedures for the reviews,

however, tend to vary. Company A has an early review of requirements by testers while companies C and D review the requirements while creating the test cases. Different interviewees from companies E and F mentioned one or the other of these approaches; the process seems to prescribe cross-role reviews but process compliance varies. A test engineer said '[the requirements are] usually reviewed by the testers. It is what the process says.' (F11:107) Most interviewees mention testers' reviews of requirements as a good practice that enhances both the communication and the quality of the requirements, thereby resulting in better alignment of the testing effort. Furthermore, this practice was described as enabling early identification of problems with the test specification avoiding (more expensive) problems later on (C2:62). A systems architect from Company F described that close collaboration between requirements and testing around quality requirements had resulted in 'one area where we have the best alignment' (F4:101).

- **Defining a requirements review responsible (P1.4)** was mentioned as a practice that ensures that requirement reviews are performed (E2:18, F2:114). In addition, for Company F this role was also mentioned as reviewing the quality of the requirements specification (F2:114) and thereby directly addressing the alignment challenge of low quality of the requirements specification (Ch3).

- **Involving domain experts in the requirements definition (P1.5)** was mentioned as a practice to achieve better synchronisation between the requirements and the system capabilities, and thereby support defining more realistic requirements. The expert 'will know if we understand [the requirement] correctly or not' (D3:38), said a system architect. Similar to the previous RE practices, this practice was also mentioned as supporting alignment by enhancing the quality of the requirements (Ch3) which are the basis for software testing.

- **Documentation of requirement decision rationales (P1.6)**, and not just the current requirement version, was suggested as a practice that might facilitate alignment by interviewees from both of the larger companies in our study, namely E and F. 'Softly communicating how we [requirements roles] were thinking' (E3:90) could enhance the synchronisation between project phases by better supporting hand-over between the different roles (F4:39). In addition, the information could support testers in analysing customer defect reports filed a long time after development was completed, and in identifying potential improvements (E3:90). However, the information needs to be easily available and connected to the relevant requirements and test cases for it to be practically useful to the testers (F1:120).

4.2.2 Validation practices

Practices for validating the system under development and ensuring that it is in-line with customer expectations and that the right product is built (IEEE610) include test case reviews, automatic testing of acceptance test cases, and review of prototypes.

- **Test cases are reviewed against requirements (P2.1)** at company F (F14:62). In their new (agile) development processes, the attributes of ISO9126

(ISO9126) are used as a checklist to ensure that not only functional requirements are addressed by the test cases, but also other quality attributes (F14:76).

- **Acceptance test cases defined by customer (P2.2),** or by the business unit, **is practiced at c**ompany B**.** The communication with the customer proxy in terms of acceptance criteria for (previously agreed) user scenarios acts as a 'validation that we [software developers] have interpreted the requirements correctly' (B1:117). This practice in combination with full unit test coverage of the code (B1:117) was experienced to address the challenge of achieving full test coverage of the requirements (Ch4, see Section 4.1.4).

- **Reviewing prototypes (P2.3)** and GUI mock-ups was mentioned as an alignment practice applied at company A. With this practice, the product manager in the role as customer proxy validates that the developed product is in-line with the original product intents (A3:153,163). Company partners that develop tailor-made systems using their components may also be involved in these reviews.

- **Management base launch decisions on test reports (P2.4)** was mentioned as an important improvement in the agile way of working recently introduced at Company F. Actively involving management in project decisions and, specifically in deciding if product quality is sufficient for the intended customers was seen as ensuring and strengthening the coordination between customer and business requirements, and testing; 'Management ... have been moved down and [made to] sit at a level where they see what really happens' (F15:89).

- **User/customer testing (P2.5)** is a practice emphasised by company B that apply agile development practices. At regular intervals, executable code is delivered, thus allowing the customer to test and validate the product and its progress (B3: 32, B3:99). This practice is also applied at company E, but with an organisational unit functioning as the user proxy (E3:22). For this practice to be effective the customer testing needs to be performed early on. This is illustrated by an example from company F, namely 'before the product is launched the customer gets to test it more thoroughly. And they submit a lot of feedback. Most are defects, but there are a number of changes coming out of that. That's very late in the process ... a few weeks […] before the product is supposed to be launched' (F1:12). If the feedback came earlier, it could be addressed, but not at this late stage.

4.2.3 Verification practices

Verification ensures that a developed system is built according to the specifications (IEEE610). Practices to verify that system properties are aligned to system requirements include starting verification early to allow time for feedback and change, using independent test teams, re-use of customer feedback obtained from previous projects, and training testers at outsourced or off-shored locations.

- **Early verification (P3.1)** is put forward as an important practice especially when specialised hardware development is involved, as for an embedded

product. Verification is then initially performed on prototype hardware (F15:114). Since quality requirements mostly relate to complete system characteristics, early verification of these requirements is harder, but also more important. Company E states: 'If we have performance issues or latency issues or database issues then we usually end up in weeks of debugging and checking and tuning.' (E3:28)

- **Independent test teams (P3.2)** are considered a good practice to reduce bias in interpreting requirements by ensuring that testers are not influenced by the developers' interpretation of requirements. However, this practice also increases the risk of mis-alignment when the requirements are insufficiently communicated since there is a narrower communication channel for requirements-related information. This practice was emphasised especially for companies with safety requirements in the transportation domain (companies C and D); 'due to the fact that this is a fail-safe system, we need to have independency between testers and designers and implementers' (C3:24, similar in C2:39, D2:80), 'otherwise they [test team] might be misled by the development team' (D1:41). Similarly, company F emphasises alternative perspectives taken by an independent team. As a software developer said: 'You must get another point of view of the software from someone who does not know the technical things about the in-depth of the code, and try to get an overview of how it works.' (F13:32)

- **Testers re-use customer feedback from previous projects (P3.3)** when planning the verification effort for later projects (F14:94), thereby increasing the test coverage. In addition to having knowledge of the market through customer feedback, verification organisations often analyse and test competitor products. With a stronger connection and coordination between the verification and business/requirements units, this information could be utilised in defining more accurate roadmaps and product plans.

- **Training off-shore/outsourced testers (P3.4)** in the company's work practices and tools increases the competence and motivation of the outsourced testers in the methods and techniques used by the outsourcing company. This was mentioned by a project manager from Company C as improving the quality of verification activities and the coordination of these activities with requirement (C3:49, 64).

4.2.4 Change management practices

Practices to manage the (inevitable) changes in software development may mitigate the challenge of maintaining alignment (Ch5, see Section 4.1.5). We identified practices related to the involvement of testing roles in the change management process and also practices connected to product lines as a means to support REVV alignment.

- **Involving testing roles in change management (P4.1)**, in the decision making and in the communication of changes, is a practice mentioned by all companies, but one, as supporting alignment through increased communication and coordination of these changes with the test organisation. '[Testers] had to show

their impacts when we [product management] were deleting, adding or changing requirements' (E2:73) and 'any change in requirement ... means involving developer, tester, project manager, requirements engineer; sitting together when the change is agreed, so everybody is aware and should be able to update accordingly' (F8:25). In companies with formalised waterfall processes, a change control board (CCB) is a common practice for making decisions about changes**.** Company D has weekly meetings of the 'change control board with the customer and we also have more internal change control boards' (D1:106). The transitioning to agile practices affected the change management process at companies E and F. At company F the change control board (CCB) was removed, thus enhancing local control at the expense of control of the whole development chain. As expressed by a process manager in company F: 'I think it will be easy for developers to change it [the requirements] into what they want it to be.' (F12:135) At company E centralised decisions were retained at the CCB (E2:73), resulting in a communication challenge; 'sometimes they [test] don't even know that we [product management] have deleted requirements until they receive them [as deleted from the updated specification]' (E2:73).

- **Product-line requirements practices (P4.2) are applied in** order to reduce the impact of a requirements change. By sharing a common product line (a.k.a. platform), these companies separate between the requirements for the commonality and variability of their products. In order to reduce the impact of larger requirements changes and the risks these entail for current projects, company A 'develop it [the new functionality] separately, and then put that into a platform' (A3:65). Company C use product lines to leverage on invested test effort in many products. When changing the platform version 'we need to do the impact analysis for how things will be affected. And then we do the regression test on a basic functionality to see that no new faults have been introduced.' (C3:55)

4.2.5 Process enforcement practices (P5)

External requirements and regulations on certain practices affect the motivation and incentive for enforcing processes and practices that support alignment. This is especially clear in company C, which develops safety critical systems. 'Since it is safety-critical systems, we have to show that we have covered all the requirements, that we have tested them.' (C1:6) It is admitted that traceability is costly, but, non-negotiable in their case. 'There is a lot of administration in that, in creating this matrix, but it has to be done. Since it is safety-critical systems, it is a reason for all the documentation involved.' (C1:06) They also have an external assessor to validate that the processes are in place and are adhered to. An alternative enforcement practice was proposed by one interviewee from company F (which does not operate in a safety-critical domain) who suggested that alignment could be achieved by enforcing traceability through integrating process enforcement in the development tools (F14:161) though this had not been applied.

4.2.6 Tracing between artefacts

The tracing practices between requirements and test artefacts vary over a large range of options from simple mappings between documents to extensive traces between detailed requirements and test cases.

- **Document-level traces (P6.1)** where links are retained between related documents is the simplest tracing practice. This is applied at company A**:** 'we have some mapping there, between the project test plan and the project requirement specification. But this is a fragile link.' (A2:69)
- **Requirement - test case traces (P6.2)** is the most commonly mentioned tracing practice where individual test cases are traced to individual requirements. This practice influences how test cases are specified: 'It is about keeping the test case a bit less complex and that tends to lead to keep them to single requirements rather than to several requirements.' (F6:123)
- **Using test cases as requirements (P6.3)** where detailed requirements are documented as test cases is another option where the tracing become implicit at the detailed level when requirements and test cases are represented by the same entity. This practice was being introduced at company F. 'At a certain level you write requirements, but then if you go into even more detail, what you are writing is probably very equivalent to a test case.' (F5:113) While this resolves the need for creating and maintaining traces at that level, these test-case requirements need to be aligned to requirements and testing information at higher abstraction levels. 'There will be teams responsible for mapping these test cases with the high-level requirements.' (F10:150) Company A has this practice in place, though not pre-planned but due to test cases being better maintained over time than requirements. 'They know that this test case was created for this requirement some time ago […and] implicitly […] the database of test cases becomes a requirements specification.' (A2:51)
- **Same abstraction levels used for requirements and test specifications (P6.4)** is an alignment practice related to the structure of information. First, the requirements information is structured according to suitable categories. The requirements are then detailed and documented within each category, and the same categorisation used for the test specifications. Company C has 'different levels of requirements specifications and test specifications, top level, sub-system, module level, and down to code' (C3:67), and company D presents similar on the test processes and artefacts (D3:53). It is worth noting that both company C and D develop safety-critical systems. At company F, a project leader described 'the correlation between the different test [levels]' and different requirement levels; at the most detailed level 'test cases that specify how the code should work' and at the next level 'scenario test cases' (F8:16).

4.2.7 Practice of traceability responsible role (P7)

For large projects, and for safety-critical projects, the task of creating and maintaining the traces may be assigned to certain roles. In company E, one of the interviewees is responsible for consolidating the information from several projects to the main product level. 'This is what I do, but since the product is so big, the

actual checking in the system is done by the technical coordinator for every project.' (E3:54) In one of the companies with safety-critical projects this role also exists; 'a safety engineer […] worked with the verification matrix and put in all the information […] from the sub products tests in the tool and also we can have the verification matrix on our level' (C2:104.)

4.2.8 Tool support

Tool support is a popular topic on which everyone has an opinion when discussing alignment. The tool practices used for requirements and test management vary between companies, as does the tool support for tracing between these artefacts. See Table 5 for an overview.

- **Tool support for requirements and test management (P8.1)** varies hugely among the companies in this study, as summarised in 5. Company A uses a test management tool, while requirements are stored as text. Companies D and E use a requirements management tool for requirements and a test management tool for testing. This was the previous practice at company F too. Company C uses a requirements management tool for both requirements and test, while Company F aims to start using a test management tool for both requirements and testing. Most of the companies use commercial tools, though company A has an in-house tool, which they describe as 'a version handling system for test cases' (A2:208).

- **Tool support for requirements-test case tracing (P8.2)** is vital for supporting traceability between the requirements and test cases stored in the tools used for requirements and test management. Depending on the tool usage, tracing needs to be supported either within a tool, or two tools need to be integrated to allow tracing between them. For some companies, *only manual tracing is supported*. For example, at company D a systems architect describes that it is possible to 'trace requirements between different tools such as [requirements] modules and Word documents' (D3:45). However, a software manager at the same company mentions problems in connecting the different tools and says 'the tools are not connected. It's a manual step, so that's not good, but it works' (D1:111). *Tracing within tools* is practiced at company C where requirements and test cases are both stored in a commercial requirements management tool: 'when we have created all the test cases for a certain release, then we can automatically make this matrix show the links between [system] requirements and test cases' (C1:8). Company F has used the *between-tools practice* 'The requirements are synchronised over to where the test cases are stored.' (F5:19) However, there are issues related to this practice. Many-to-many relationships are difficult to handle with the existing tool support (F2:167). Furthermore, relationships at the same level of detail are easier to handle than across different abstraction levels. One requirements engineer asks for 'a tool that connects everything; your requirement with design documents with test cases with your code maybe even your planning document,' (F5:17). In a large, complex system and its development organisation, there is a need for 'mapping towards all kinds of directions – per function group, per test cases, and from the requirement level' (F11:139).

Table 5. Tool usage for requirements and test cases, and for tracing between them. For company F the tool set-up prior to the major process change are also given (marked with 'previous').

	Requirements tool	Tracing tool	Testing tool
Requirements	C, D, E, F (previous)		F
Traces	C	D, E, F (previous)	F
Test cases	C		A, D, E, F (current and previous)

Many interviewees had complaints about their tools, and the integration between them. Merely having tool support in place is not sufficient, but it must be efficient and useable. For example, company E have tools for supporting traceability between requirements and test state of connected test cases but 'we don't do it because the tool we have is simply not efficient enough' (E3:57) to handle the test state for the huge amount of verified variants. Similarly, at company E the integration solution (involving a special module for integrating different tools) is no longer in use and they have reverted to manual tracing practices: 'In some way we are doing it, but I think we are doing it manually in Excel sheets' (E2:49).

Finally, companies moving from waterfall processes towards agile practices tend to find their tool suite too heavy weight for the new situation (E3:89). Users of these tools not only include engineers, but also management, which implies different demands. A test manager states: 'Things are easy to do if you have a lot of hands on experience with the tools but what you really need is something that the [higher level] managers can use' (F10:249).

4.2.9 Alignment metrics (P9)

Measurements can be used to gain control of the alignment between requirements and testing. The most commonly mentioned metrics concern test case coverage of requirements. For example, company C 'measure[s] how many requirements are already covered with test cases and how many are not' (C1:64). These metrics are derived from the combined requirements and test management tool. Companies E and F have a similar approach, although with two different tools. They both point out that, in addition to the metrics, it is a matter of judgment to assess full requirements coverage. 'If you have one requirement, that requirement may need 16 test cases to be fully compliant. But you implement only 14 out of those. And we don't have any system to see that these 2 are missing.' (E3:81) And, 'just because there are 10 test cases, we don't know if [the requirement] is fully covered' (F11:34). Furthermore, there is a versioning issue to be taken into account when assessing the requirements coverage for verification. 'It is hard to say if it [coverage] should be on the latest software [version] before delivery or ...?' (F10:224) The reverse relationship of requirements coverage of all test cases is not always in place or measured. 'Sometimes we have test cases testing functionality not specified in the requirements database.' (F11:133) Other alignment metrics were mentioned, for example, missing links between requirements and tests, number of requirements at different levels (F5:112), test costs for changed

requirements (F14:205), and requirements review status (F14:205). Not all of these practices were practiced at the studied companies even though some mentioned that such measures would be useful (F14:219).

4.2.10 Job rotation practice (P10)

Job rotation was suggested in interviews at companies D and F as a way to improve alignment by extending contact networks and experiences across departments and roles, and thereby supporting spreading and sharing perspectives within an organisation. In general, the interviews revealed that alignment is very dependent on individuals, their experience, competence and their ability to communicate and align with others. The practice of job rotation was mentioned as a proposal for the future and not currently implemented at any of the included companies.

4.3 Practices that Address the Challenges

This section provides an overview of the relationships between the alignment challenges and practices identified in this study (and reported in Sections 0 and 4.2). The mapping is intended as an initial guide for practitioners in identifying practices to consider in addressing the most pressing alignment challenges in their organisations. The connections have been derived through analysis of the parts of the interview transcripts connected to each challenge and practice, summarised in Table 6 and elaborated next. The mapping clearly shows that there are many-to-many relations between challenges and practices. There is no single practice that solves each challenge. Consequently, the mapping is aimed at a strategic level of improvement processes within a company, rather than a lower level of practical implementation. After having assessed the challenges and practices of REVV alignment within a company, the provided mapping can support strategic decisions concerning which areas to improve. Thereafter, relevant practices can be tailored for use within the specific context. Below we discuss our findings, challenge by challenge.

The practices observed to address the challenge of having **common goals within an organisation (Ch1)** mainly concern increasing the synchronisation and communication between different units and roles. This can be achieved through involving customers and development-near roles in the requirements process (P1.1, P1.2, P1.3, P1.5); documenting requirement decision rationale (P1.6); validating requirements through test case reviews (P2.1) and product managers reviewing prototypes (P2.3); and involving testing roles in change management (P4.1). Goal alignment is also increased by the practice of basing launch decisions made by management on test reports (P2.4) produced by testers. Furthermore, tracing between artefacts (P6.1-6.4) provides a technical basis for supporting efficient communication of requirements. Job rotation (P10) is mentioned as a long-term practice for sharing goals and synchronising perspectives across the organisation. In the mid-term perspective, customer feedback received by testers for previous projects (P3.3) can be re-used as input when defining roadmaps and products plans thereby further coordinating the testers with the requirements engineers responsible for the future requirements.

Table 6. Mapping of practices to the challenges they are found to address. An S represents a suggested, but not implemented practice. Note: a blank cell indicates that no connection was mentioned during the interviews.

	P1 RE practices	P2 Validation practices	P3 Verification practices	P4 Change management	P5 Process enforcement	P6 Tracing between artefacts	P7 Traceability responsibility role	P8 Tool practices	P9 Alignment metrics	P10 Job rotation
Ch1 Aligning goals and perspectives within organisation	P1.1-1.3, 1.5-1.6	P2.1, 2.3-2.4	P3.3	P4.1		P6.1-6.4				P10 (S)
Ch2 Cooperating successfully	P1.2-1.3, 1.5-1.6	P2.1, 2.3, 2.4	P3.1	P4.1						P10 (S)
Ch3 Requirements specification quality	P1.1-1.5	P2.1, 2.5		P4.1	P5	P6.2-6.3			P9	
Ch4 VV quality	P1.1-1.5	P2.1-2.3, 2.5	P3.1-3.3		P5	P6.1-6.4			P9	
Ch5 Maintaining alignment when requirements change		P2.2, P2.5		P4.1-4.2	P5	P6.1-6.4	P7		P9	
Ch6 Requirements abstraction levels	P1.1, 1.6					P6.4				
Ch7 Traceability		P2.1			P5	P6.1-6.4	P7	P8.1-8.2	P9	
Ch8 Time and resource availability				P4.1	P5					
Ch9 Managing a large document space						P6.1-6.4	P7	P8.1-8.2	P9	
Ch10 Outsourcing of components or testing	P1.1-1.5	P2.1-2.3	P3.4			P6.4				

The challenge of **cooperating successfully (Ch2)** is closely related to the first challenge (Ch1) as being a means to foster common goals. Practices to achieve close cooperation across roles and organisational borders hence include cross-functional involvement (P1.2, P1.5, P2.4) and reviews (P1.3, P2.1, P2.3), feedback through early and continuous test activities (P3.1), as well as, joint decisions about changes in change control boards (P4.1) and documenting requirement decision rationales (P1.6). The former are practices are embraced in agile processes, while the latter practices of change control boards and documentation of rationales were removed for the studied cases when agile processes were introduced. Job rotation (P10), with its general contribution to building networks, is expected to facilitate closer cooperation across organisational units and between roles.

The challenge of achieving good **requirements specification quality (Ch3)** is primarily addressed by the practices for requirements engineering (P1.1-1.5), validation (P2.1, P2.5) and managing requirement changes (P4.1). Some of the traceability practices (P6.2, P6.3) also address the quality of requirements in terms of being well structured and defined at the right level of detail. Furthermore, awareness of the importance of alignment and full requirements coverage may induce and enable organisations in producing better requirements specifications. This awareness can be encouraged with the use of alignment metrics (P9) or enforced (P5) through regulations for safety-critical software and/or by integrating process adherence in development tools.

The challenge of achieving good **validation and verification quality (Ch4)** is addressed by practices to ensure clear and agreed requirements, such as cross-functional reviews (P1.3, P2.1), involving development roles in detailing requirements (P1.2) and customers in defining acceptance criteria (P2.2). Validation is supported by product managers reviewing prototypes (P2.3) and by user/customer testing (P2.5). Verification is improved by early verification activities (P3.1) and through independent testing (P3.2) where testers are not influenced by other engineers' interpretation of the requirements. Complete and up-to-date requirements information is a prerequisite for full test coverage, which can be addressed by requirements engineering practices (P1.1-1.5), testers re-using customer feedback (P3.3) (rather than incorrect requirements specification) and indirectly by traceability practices (P6.1-6.4). The external enforcement (P5) of the full test coverage and alignment metrics (P9) are practices that provide incentives for full test coverage including quality requirements.

Maintaining alignment when requirements change (Ch5) is a challenge that clearly connects to change and traceability practices (P4.1-4.2, P6.1-6.4 and P7). However, also the validation practices of having acceptance tests based on user scenarios (P2.2) and user/customer testing (P2.5) address this challenge by providing feedback on incorrectly updated requirements, test cases and/or software. Furthermore, having alignment metrics in place (P9) and external regulations on documentation and traceability (P5) is an incentive to maintain alignment as requirements change.

The challenge of managing **requirements abstraction levels (Ch6)** is addressed by the requirements practice of including the customer in requirements work throughout a project (P1.1) and the tracing practices of matching abstractions levels for requirements and test artefacts (P6.4). Both of these practices exercise the different requirements levels and thereby support identifying mismatches. This

challenge is also supported by documentation of requirement decision rationales (P1.6) by providing additional requirements information to the roles at the different abstraction level.

Traceability (Ch7) in itself is identified as a challenge in the study, and interviewees identified practices on the information items to be traced (P6.1-6.4), as well as, tools (P8.1-8.2) to enable tracing. In addition, the practice of reviewing test cases against requirements (P2.1) may also support identifying sufficient and/or missing traces. Furthermore, requirements coverage metrics (P9) are proposed as a means to monitor and support traceability. However, as noticed by companies E and F, simple coverage metrics are not sufficient to ensure ample alignment. Process enforcement practices (P5) and assigning specific roles responsible for traceability (P7) are identified as key practices in creating and maintaining traces between artefacts.

The practical aspects of the challenge on **availability of time and resources (Ch8)** are mainly a matter of project management practices, and hence not directly linked to the alignment practices. However, the practice of involving testing roles in the change management process (P4.1) may partly mitigate this challenge by supporting an increased awareness of the verification cost and impact of changes. Furthermore, in companies for which alignment practices are externally enforced (P5) there is an awareness of the importance of alignment of software development, but also an increased willingness to take the cost of alignment including tracing.

The **large document space (Ch9)** is a challenge that can be partly addressed with good tool support (P8.1-8.2) and tracing (P6.1-6.4, P7) practices. The study specifically identifies that a tool that fits a medium-sized project may be very hard to use in a large one. One way of getting a synthesised view of the level of alignment between large sets of information is to characterise it, using quantitative alignment measurements (P9). It does not solve the large-scale problem, but may help assess the current status and direct management attention to problem areas.

Outsourcing (Ch10) is a challenge that is related to timing, which is a project management issue, and to communication of the requirements, which are to be developed or tested by an external team. The primary practice to apply to outsourcing is customer communication (P1.1). Frequent and good communication can ensure a common perspective and direction, in particular in the early project phases. In addition, other practices for improved cooperation (P1.2-P1.5, P2.1-P2.3) are even more important when working in different organisational units, times zones, and cultural contexts. Furthermore, in an outsourcing situation the requirements specification is a key channel of communication, often also in contractual form. Thus, having requirements and tests specified at the same level of abstraction (P6.4), feasible for the purpose, is a practice to facilitate the outsourcing. Finally, training the outsourced or off-shored team (P3.4) in company practices and tools also addresses this challenge.

In summary, the interviewees brought forward practices, which address some of the identified challenges in aligning requirements and testing. The practices are no quick-fix solutions, but the mapping should be seen as a guideline to recommend areas for long-term improvement, based on empirical observations of industry practice.

5 Discussion

Alignment between requirements and test ranges not only the life-cycle of a software development project, but also company goals and strategy, and affects a variety of issues, from human communication to tools and their usage. Practices differ largely between companies of varying size and maturity, domain and product type, etc. One-size alignment practices clearly do not fit all.

A wide collection of alignment challenges and practices have been identified based on the large amount of experiences represented by our 30 interviewees from six different companies, covering multiple roles, domains and situations. Through analysing this data and deriving results from it, the following general observations have been made by the researchers:

1) the human and organisational sides of software development are at the core of industrial alignment practices
2) the requirements are the frame of reference for the software to be built, and hence the quality of the requirements is critical for alignment with testing activities
3) the large difference in size (factor 20) between the companies, in combination with variations in domain and product type, affects the characteristics of the alignment challenges and applicable practices
4) the incentives for investing in good alignment practices vary between domains

Organisational and human issues are related to several of the identified challenges (Ch1, Ch2, Ch8, and Ch10). Successful cooperation and collaboration (Ch2) is a human issue. Having common goals and perspectives for a development project is initially a matter of clear communication of company strategies and goals, and ultimately dependant on human-to-human communication (Ch1). Failures to meet customer requirements and expectations are often related to misunderstanding and misconception; a human failure although technical limitations, tools, equipment, specifications and so on, also play a role. It does not mean that the human factor should be blamed in every case and for each failure. However, this factor should be taken into account when shaping the work conditions for software engineers. These issues become even more pressing when outsourcing testing. Jones et al. (2009) found that failure to align outsourced testing activities with in-house development resulted in wasted effort, mainly due to weak communication of requirements and changes of them**.**

Several of the identified alignment practices involve the human and organisational side of software engineering. Examples include communication practices with customers, cross-role and cross-functional meetings in requirements elicitation and reviews, communication of changes, as well as, a proposed job rotation practice to improve human-to-human communication. This confirms previous research that alignment can be improved by increasing the interaction between testers and requirements engineers. For example, including testers early on and, in particular, when defining the requirements, can lead to improved requirements quality (Uusitalo 2008). However, Uusitalo also found that cross collaboration can be hard to realise due to unavailability of requirements owners

and testers on account of other assignments and distributed development (Uusitalo 2008). In general, processes and roles that support and enforce the necessary communication paths may enhance alignment. For example, Paci et al. (2012) report on a process for handling requirements changes through clearly defined communication interfaces. This process relies on roles propagating change information within their area, rather than relying on more general communication and competence (Paci 2012). This also confirms the findings of Uusitalo et al. that increased cross communication reduces the amount of assumptions made by testers on requirements interpretation, and results in an increased reliability of test results and subsequent products (Uusitalo 2008). Similarly, Fogelström et al. (2007) found that involving testers as reviewers through test-case driven inspections of requirements increases the interaction with requirements-related roles and can improve the overall quality of the requirements, thereby supporting alignment. Furthermore, even technical practices, such as tool support for requirements and test management, clearly have a human side concerning degree of usability and usefulness for different groups of stakeholders in an organisation.

Defining requirements of good quality (Ch3) is central to enabling good alignment and coordination with other development activities, including validation and verification. This challenge is not primarily related to the style of requirements, whether scenario based, plain textual, or formal. But, rather the quality characteristics of the requirements are important, i.e. being verifiable, clear, complete, at a suitable level of abstraction and up-to-date. This relates to results from an empirical study by Ferguson et al. (2006) that found that unclear requirements have a higher risk of resulting in test failures. A similar reverse relationship is reported by Graham (2002), that clearer and verifiable requirements enable testers to define test cases that match the intended requirements. In addition, Uusitalo et al. (2008) found that poor quality of requirements was a hindrance to maintaining traces from test cases. Sikora et al. (2012) found that requirements reviews is the dominant practice applied to address quality assurance of the requirements for embedded systems and that industry need additional and improved techniques for achieving good requirements quality. Furthermore, requirements quality is related to involving, not only requirements engineers in the requirements engineering, but also VV roles in early stages. This can be achieved by involving non-RE roles in reviews and in detailing requirements. This also contributes to cross-organisational communication and learning, and supports producing requirements that are both useful and used. Uusitalo et al. (2008) found that testers have a different viewpoint that makes them well suited to identifying deficiencies in the requirements including un-verifiability and omissions. Martin et al. (2008) take this approach one step further by suggesting that the requirements themselves be specified as acceptance test cases, which are then used to verify the behaviour of the software. This approach was evaluated through an experiment by Ricca et al. (2009) who found that this helped to clarify and increase the joint understanding of requirements with substantially the same amount of effort. Furthermore, our findings that RE practices play a vital role in supporting REVV alignment confirm previous conclusions that the requirements process is an important enabler for testing activities and that RE improvements can support alignment with testing (Uusitalo 2008).

Company size varies largely between the six companies in this study. Similarly, the challenges and practices also vary between the companies. While smaller project groups of 5-10 persons can handle alignment through a combination of informal and formal project meetings. Large-scale projects require more powerful process and tool support to ensure coordination and navigation of the (larger) information space between different phases and hierarchies in the organisation. This was illustrated by different views on the same state-of-the-art requirements management tool. The tool supported alignment well in one medium-sized project (company C), but was frequently mentioned by the interviewees for the largest company (company F) as a huge alignment challenge.

In some cases (e.g. company F), agile practices are introduced to manage large projects by creating several smaller, self-governing and less dependent units. Our study shows that this supports control and alignment at the local level, but, at the expense of global control and alignment (company E). The size-related alignment challenges then re-appear in a new shape, at another level in the organisation. For example, granting development teams mandate to define and change detailed requirements increases speed and efficiency at the team level, but increases the challenge of communicating and coordinating these changes wider within a large organisation.

The incentives for applying alignment practices, specifically tracing between requirements and test artefacts, vary across the studied companies. Applying alignment practices seems to be connected to the incentives for enforcing certain practices, such as tracing and extensive documentation. The companies reporting the most rigid and continuously maintained alignment practices are those working in domains where customers or regulatory bodies require such practices. Both of these companies (C and D) have enforced alignment practices in their development including tracing between requirements and tests. Interestingly these are also the companies in our study which apply a traditional and rigorous development model. It is our interpretation that the companies with the most agile, and least rigorous, development processes (A and B) are also the companies which rely heavily on people-based alignment and tracing, rather than on investing in more structured practices. These are also the two companies that do not have tracing between artefacts in place, even partially. While for the remaining companies (E and F) which apply agile-inspired processes, but with structured elements (e.g. eRUP), traceability is in place partly or locally. Our interpretation of the relationship between the included companies concerning incentives and degree of rigour in applying structured alignment practices is illustrated in Figure 4 together with the relative size of their software development. The observed connection between degree of rigour and incentives for alignment are similar to other findings concerning safety-critical development. Namely, that alignment is enabled by more rigorous practices such as concurrently designed processes (Kukkanen 2009) or model-based testing (Nebut 2006, Hasling 2008). Furthermore, companies in safety-critical domains have been found to apply more rigorous processes and testing practices (Runeson 2003). In contrast, neglect of quality requirements, including safety aspects has been found to one of the challenges of agile RE (Cao 2008).

Interestingly, several alignment challenges (e.g. tracing, communicating requirements changes) were experienced also for the companies developing safety-critical software (C and D) despite having tracing in place and applying practices to mitigate alignment challenges (e.g. frequent customer communication, tracing responsible role, change management process involving testers etc.) This might be explained by a greater awareness of the issues at hand, but also that the increased demands posed by the higher levels of quality demands requires additional alignment practice beyond those needed for non-critical software.

When the documentation and tracing practices are directly enforced from outside the organisation, they cannot be negotiated and the cost has to be taken (Watkins 1994). In organisations without these external requirements the business case for investing in these practices needs to be defined, which does not seem to be the case for the studied organisations. Despite the existence of frustration and rework due to bad alignment, the corresponding costs are seldom quantified at any level. Improving alignment involves short term investments in tools, work to recapture traces between large legacies of artefacts, and/or in changed working practices. The returns on these investments are gained mainly in the longer term. This makes it hard to put focus and priority on alignment practices in a short-sighted financial and management culture. Finally, requirements volatility increases the importance and cost to achieve REVV alignment. This need to manage a rate of requirements changes often drives the introduction of agile practices. These practices are strong in team cooperation, but weak in documentation and traceability between artefacts. The companies (C and D) with lower requirements volatility and where development is mainly plan-driven and bespoken, have the most elaborated documentation and traceability practices. In both cases, the

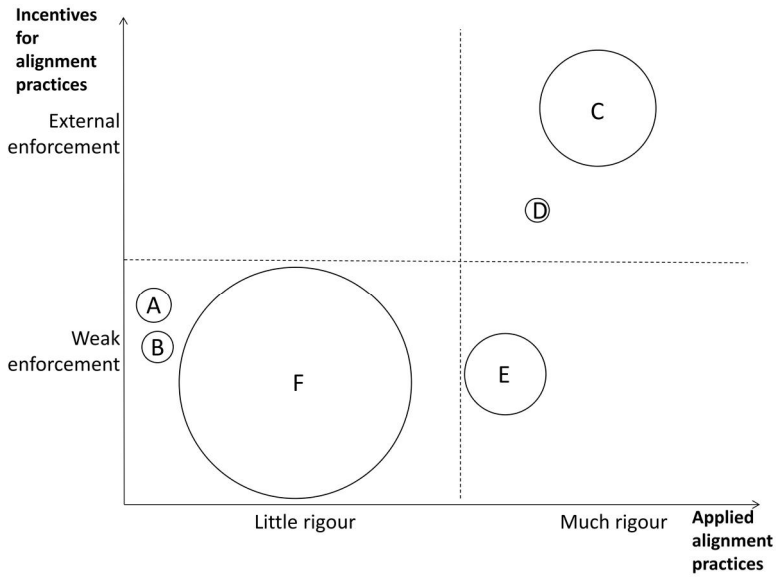


Figure 4. Rough overview of the relationship between the variation factors size, rigour in applying alignment practices and incentive for alignment practices for the studied companies. Number of people in software development is reflected by the relative size of the circle.

practices are enforced by regulatory requirements. However, in our study, it is not possible to distinguish between the effects of different rates of change and the effects of operating in a safety-critical domain with regulations on documentation and traceability.

In summary, challenges and practices for REVV alignment span the whole development life cycle. Alignment involves the human and organisational side of software engineering and requires the requirements to be of good quality. In addition, the incentives for alignment greatly vary between companies of different size and application domain. Future research and practice should consider these variations in identifying suitable practices for REVV alignment, tailored to different domains and organisations.

6 Conclusions

Successful and efficient software development, in particular on the large scale, requires coordination of the people, activities and artefacts involved (Kraut 1995, Damian 2005, 2006). This includes alignment of the areas of requirements and test (Damian 2006, Uusitalo 2008, Kukkanen 2009, Sabaliauskaite 2010). Methods and techniques for linking artefacts abound including tracing and use of model-based engineering. However, companies experience challenges in achieving alignment including full traceability. These challenges are faced also by companies with strong incentives for investing in REVV alignment such as for safety critical software where documentation and tracing is regulated. This indicates that the underlying issues lie elsewhere and require aligning of not only the artefacts, but also of other factors. In order to gain a deeper understanding of the industrial challenges and practices for aligning RE with VV, we launched a case study covering six companies of varying size, domain, and history. This paper reports the outcome of that study and provides a description of the industrial state of practice in six companies. We provide categorised lists of (RQ1) industrial alignment challenges and (RQ2) industrial practices for improving alignment, and (RQ3) a mapping between challenges and practices. Our results, based on 30 interviews with different roles in the six companies, add extensive empirical input to the existing scarce knowledge of industrial practice in this field (Uusitalo 2008, Sabaliauskaite 2010). In addition, this paper presents new insights into factors that explain needs and define solutions for overcoming REVV alignment challenges.

We conclude with four high-level observations on the alignment between requirements and testing. Firstly, as in many other branches of software engineering, the *human side* is central, and communication and coordination between people is vital, so also between requirements engineers and testers, as one interviewee said: ‘start talking to each other!’ (F7:88) Further, the *quality and accuracy of the requirements* is a crucial starting point for testing the produced software in-line with the defined and agreed requirements. Additionally, the *size of the development organisation* and its projects is a key variation factor for both challenges and practices of alignment. Tools and practices may not be scalable, but rather need to be selected and tailored to suit the specific company, size and domain. Finally, alignment practices such as good requirements documentation and

tracing seem to be applied for safety-critical development through external enforcement. In contrast, for non-safety critical cases only internal *motivation* exists for the alignment practices even though these companies report facing large challenges caused by misalignment such as incorrectly implemented requirements, delays and wasted effort. For these cases, support for assessing the cost and benefits of REVV alignment could provide a means for organisations to increase the awareness of the importance of alignment and also tailor their processes to a certain level of alignment, suitable and cost effective for their specific situation and domain.

In summary, our study reports on the current practice in several industrial domains. Practical means are provided for recognising challenges and problems in this field and matching them with potential improvement practices. Furthermore, the identified challenges pose a wide assortment of issues for researchers to address in order to improve REVV alignment practice, and ultimately the software engineering practices.

Acknowledgment

We want to thank Börge Haugset for acting as interviewer in three of the interviews. We would also like to thank all the participating companies and anonymous interviewees for their contribution to this project. The research was funded by EASE Industrial Excellence Center for Embedded Applications Software Engineering (http://ease.cs.lth.se).

References

Barmi ZA, Ebrahimi AH, Feldt R (2011) Alignment of Requirements Specification and Testing: A Systematic Mapping Study. Proc 4th Int. Conf. On Softw. Testing, Verification and Validation Workshops (ICSTW):476-485.

Cao L, Ramesh B (2008) Agile Requirements Engineering Practices: An Empirical Study. IEEE Software Jan/Feb 2008.

Cheng BH, Atlee JM (2007) Research Directions in Requirements Engineering. Proc. Future of Software Engineering (FOSE):285-303.

Cleland-Huang J, Chang CK, Christensen M (2003) Event-Based Traceability for Managing Evolutionary Change. IEEE Transactions on Software, 29(9).

Damian D, Chisan J, Vaidyanathasamy L, Pal Y (2005) Requirements Engineering and Downstream Software Development: Findings from a Case Study. Empirical Software Engineering, vol 10:255-283.

Damian D, Chisan J (2006) An Empirical Study of the Complex Relationship between Requirements Engineering Processes and Other Processes that Lead to Payoffs in Productivity, Quality, and Risk Management. IEEE Transactions on Software Engineering, 32(7):33 - 453.

De Lucia A, Fasano F, Oliveto R, Tortora G (2007) Recovering Traceability Links in Software Artifact Management Systems using Information Retrieval Methods. ACM Transactions on Softw. Engineering and Methodology, 16(4):Article 13.

Dias Neto AC, Arilo C, Subramanyan R, Vieira M, Travassos GH (2007) A Survey on Model-Based Testing Approaches: A Systematic Review. Proc of 1st ACM Int Workshop on Empirical Assessm. of Softw. Engineering Languages and Technologies, pp. 31-36.

Ferguson RW, Lami G (2006) An Empirical Study on the Relationship between Defective Requirements and Test Failures. Proc of 30th Annual IEEE/NASA Software Engineering Workshop SEW-30 (SEW'06).

Fogelström N, Gorschek T (2007) Test-case Driven versus Checklist-based Inspections of Software Requirements – An Experimental Evaluation. Proc. of 10th Workshop on Requirements Engineering (WER'07).

Gorschek T, Wohlin C (2004) Packaging Software Process Improvement Issues - A Method and a Case Study. Softw. Pract & Experience 34:1311-1344

Gorschek T, Wohlin C (2006) Requirements Abstraction Model. Requir Eng J 11:79-101

Gorschek T, Davis AM (2007) Requirements Engineering: In Search of the Dependent Variables. Information and Software Technology, 50(1–2):67-75

Gotel O, Finkelstein A (1994) An Analysis of the Requirements Traceability Problem. Proc. First Int Conf. Requirements Eng., pp. 94-101.

Graham D (2002) Requirements and Testing: Seven Missing-Link Myths. IEEE Software, vol 19:15-17.

Grieskamp W, Kicillof N, Stobie K, Braberman V (2011) Model-Based Quality Assurance of Protocol Documentation: Tools and Methodology. Softw. Test Verification Reliability. 21(1):55–71

Hasling B, Goetz H, Beetz K (2008) Model Based Testing of System Requirements using UML Use Case Models. Proc of 2008 Int. Conf. on Software Testing, Verification, and Validation.

Hayes JH, Dekhtyar A, Sundaram SK, Holbrook EA, Vadlamudi S, April A (2007) REquirements TRacing On target (RETRO): Improving Software Maintenance Through Traceability Recovery. Innovations in Systems and Software Engineering, 3(3):193-202.

Höst M, Feldt R, Lüders F (2010) Support for Different Stakeholders in Software Engineering Master Thesis Projects. IEEE Transactions on Education, 52(2):288-296. doi:10.1109/TE.2009.2016106

IEEE. IEEE standard glossary of software engineering terminology. Technical Report 610.12-1990, IEEE, New York, NY, USA, 1990.

ISO/IEC 9126-1:2001(E), International standard software engineering product quality part 1: Quality model. Technical report, ISO/IEC, 2001.

Jarke M (1998) Requirements Traceability. Comm. ACM, vol. 41, no. 12, pp. 32-36

Jones JA, Grechanik M, van der Hoek A (2009) Enabling and Enhancing Collaborations between Software Development Organizations and Independent Test Agencies. Cooperative and Human Aspects of Softw. Engineering (CHASE'09), May 17, 2009, Vancouver, Canada.

Kraut RE, Streeter L (1995) Coordination in Software Development. Communications of the ACM, 38(3):69-81.

Kukkanen J, Vakevainen K, Kauppinen M, Uusitalo E (2009) Applying a Systematic Approach to Link Requirements and Testing: A Case Study. Proc of Asia-Pacific Software Engineering Conference (APSEC '09):482 – 488.

Lormans M, van Deursen A, Gross H (2008) An Industrial Case Study in Reconstructing Requirements Views. Empirical Software Engineering, online first, September 03, 2008.

Lubars M, Potts C, Richter C (1993) A Review of the State of the Practice in Requirements Modelling. Proceedings of 1st IEEE Int. Symposium on Requirements Engineering, pp. 2–14.

Martin R, Melnik G (2008) Tests and Requirements, Requirements and Tests a Möbius Strip. IEEE Software, 25(1):54-59.

Melnik G, Maurer F, Chiasson M (2006) Executable Acceptance Tests for Communicating Business Requirements: Customer Perspective. Proc. of Agile Conference, Minneapolis, USA, pp. 12-46.

Mohagheghi P, Dehlen V (2008) Where Is the Proof? - A Review of Experiences from Applying MDE in Industry. LNCS, Model Driven Architecture – Foundations and Applications, vol 5095:432-443.

Nebut C, Fleurey F, Traon YL, Jézéquel J (2006) Automatic Test Generation: A Use Case Driven Approach. IEEE Trans. on Softw. Engineering, 32(3):140-155.

Paci F, Massacci F, Bouquet F, Debricon S (2012) Managing Evolution by Orchestrating Requirements and Testing Engineering Processes. Proc. of IEEE 5th Int. Conf. On, pp.834-841.

Pettersson F, Ivarsson M, Gorschek T (2008) A Practitioner's Guide to Light Weight Software Process Assessment and Improvement Planning. J. of Systems and Software 81(6):972-995

Post H, Sinz C, Merz F, Gorges T, Kropf T (2009) Linking Functional Requirements and Software Verification. Proc. of 17th IEEE Int. Requirements Engineering Conf., pp. 295-302.

Ramesh B, Stubbs C, Powers T, Edwards M (1997) Requirements traceability: Theory and practice. Annals of Software Engineering, 3(1):397-415.

Ramesh B (1998) Factors Influencing Requirements Traceability Practice. Communications of the ACM CACM Homepage archive, 41(12):37-44.

Randell B (1969) Towards a methodology of computing system design. Naur, P., and Randell, B (Eds.) NATO Working Conference on Software Engineering 1968, Report on a Conference Sponsored by NATO Scientific Committee, Germany, pp. 204-208.

Regnell B, Runeson P (1998) Combining Scenario-based Requirements with Static Verification and Dynamic Testing. Proc. 4th Int. Working Conf. Requirements Engineering: Foundation for Software Quality, pp.195–206.

Regnell B, Runeson P, Wohlin C (2000) Towards Integration of Use Case Modelling and Usage-Based Testing. J. of Systems and Softw. 50(2):117–130.

Ricca F, Torchiano M, Di Penta M, Ceccato M, Tonella P (2009) Using Acceptance Tests as a Support for Clarifying Requirements: A Series of Experiments. Information and Software Technology 51, pp. 270–283

Robson C (2002) Real World Research: A Resource for Social Scientists and Practitioner Researchers, 2nd edition. Blackwell Publishing.

Runeson P, Andersson C, Höst M (2003) Test Processes in Software Product Evolution - A Qualitative Survey on the State of Practice. Journal of Software Maintenance and Evolution: Research and Practice 15(1):41–59.

Runeson P, Höst M (2009) Guidelines for Conducting and Reporting Case Study Research in Software Engineering. Empir Softw Eng 14(2):131–164

Runeson P, Höst M, Rainer A, Regnell B (2012). Case Study Research in Software Engineering – Guidelines and Examples. Wiley.

Sabaliauskaite G, Loconsole A, Engström E, Unterkalmsteiner M, Regnell B, Runeson P, Gorschek T, Feldt R (2010) Challenges in Aligning Requirements Engineering and Verification in a Large-Scale Industrial Context. Proceedings of REFSQ 2010

Sikora E, Tenbergen B, Pohl K (2012) Industry Needs and Research Directions in Requirements Engineering for Embedded Systems. Requirements Engineering, 17(1):57–78.

Uusitalo EJ, Komassi M, Kauppinen M, Davis AM (2008) Linking Requirements and Testing in Practice. Proc. of 16th Int. Requirements Engineering Conf., pp. 295-302.

Watkins R, Neal M (1994) Why and How of Requirements Tracing. IEEE Software 11(4):104-106.

Wohlin C, Runeson P, Höst M, Ohlsson MC, Regnell B, Wesslen A (2000) Experimentation in Software Engineering: An Introduction (Int. Series in Softw. Eng.). Springer, Heidelberg.

Yue T, Briand L, Labiche Y (2011) A Systematic Review of Transformation Approaches Between User Requirements and Analysis Models. Requirements Engin. 16(2):75-99

PAPER IV

DISTANCES BETWEEN REQUIREMENTS ENGINEERING AND LATER SOFTWARE DEVELOPMENT ACTIVITIES: A SYSTEMATIC MAP[1]

The main role of requirements engineering (RE) is to guide development projects towards implementing products that will appeal to customers. To effectively achieve this RE needs to be coordinated with and clearly communicated to the later software development activities. Communication gaps between RE and other development activities reduce coordination and alignment, and can lead to project delays and failure to meet customer needs. The main hypothesis is that coordination is enhanced by proximity to RE roles and artefacts, and that distances to later activities increase the effort needed to align requirements with other development work. Thirteen RE-related distances have been identified through a systematic map of existing research. Reported distances are mapped according to research type, RE activity and later software development activities. The results provide an overview of RE distances and can be used a basis for defining a theoretical framework.

[1] By E. Bjarnason, published in Proc. 19th Int. Working Conf. Requirements Engineering: Foundation for Software Quality (REFSQ'13), pp. 292-307. 2013.

1 Introduction

Effective requirements engineering (RE) greatly depends upon successful coordination (Curtis 1988, Ebert 2006) and communication of requirements with the downstream development activities (Bjarnason 2011, Marczak 2011), e.g. design, implementation, and testing. Merely producing a perfect requirements specification is not sufficient. Rather it is vital to ensure that the requirements are clearly understood and agreed with implementation-near roles, and that sufficient requirements information is available for later development activities (Damian 2006, Marczak 2011). Communication gaps between people may contribute to project delays, software quality issues and even failure to meet customer expectations (Bjarnason 2011).

Within global software development (GSD), project teams and members are globally distributed. These geographical distances between people have been found to negatively affect the communication and thereby also the coordination and success of the distributed development. In addition to geographical distance, socio-cultural and temporal distances have been found to be in play within GSD (Agerfalk 2005). Agerfalk et al. (2005) have defined a theoretical framework of these different types of distances and how they affect communication, coordination and control. However, coordination and communication is also a challenge within non-distributed development, in particular for large development organizations and projects (Bjarnason 2011, Curtis 1988).

Our main hypothesis is that distance plays an important role in development, whether distributed or not. In particular, the distances between RE and later software development activities may impact project effectiveness and efficiency. The systematic mapping study reported in this paper provides an overview of existing knowledge of RE-related distances within software engineering research.

Work related to the targeted area is described in Section 2. Section 3 outlines the research method while Section 4 presents the results, which are then discussed in Section 5. Finally, the paper is concluded in Section 6.

2 Software Development and RE

‘Requirements are the basic building blocks gluing together [the] different ... activities needed to define, develop, implement, build, operate, service, and phase out a product and its related variants.’ (Ebert 2006) However, in general most people focus mainly on one area of expertise: RE, project management, architecture, implementation, testing etc. Both in practice and in research, there is generally weak insight and knowledge into how to leverage software development by improving on the interaction and coordination of RE with later activities within software development.

In contrast, concurrent engineering (Lawson 1994) is an approach to product development where several engineering activities are carried out concurrently (at the same time by the same project team) with extensive feedback and iteration. The developers are to consider all aspects of the development cycle from requirements to cost and quality. Reported gains for this approach include increased efficiency, productivity and quality, and reduced waste and shortened lead times (Lawson

1994). A concurrent approach is applied within agile software development by integrating the activities for requirements, architecture, implementation and testing, and the claimed gains are similar to those for concurrent engineering, including increased responsiveness to change.

Damian et al. (2006) found that improved RE practices within a more traditional plan-based development project may have an effect also on later software development activities. Effective RE can thereby support increased development effectiveness and augment the efficiency and productivity of the other development activities, and lead to improvements for a wide range of software development aspects, e.g. project planning, managing feature creep, testing, defects, rework, and product quality (Damian 2006). This indicates that RE can play a vital role for the total development effort, if RE is effective and well-coordinated with later development activities.

Requirements and design are interdependent activities. While design (either by architecture or directly during implementation) aims to realize the requirements, architectural and technical limitations, and new technical possibilities may affect the requirements and, thus, require requirements changes. For these reasons, it has been suggested that RE should be intertwined and performed in parallel with design (Nuseibeh 2001, Swartout 1982). Nuseibeh et al. (2001) have designed a method that does this while still separating between problem and solution structure. This method is receptive to handling change in an efficient way, allows early exploration of the problem space, and enables engineers to identify requirements and match them to available components and products Nuseibeh (2001). Similarly, Fricker et al. (2010) found that aligning requirements and architecture through a negotiation process between product management and architecture led to identifying missed requirements, and to a shared requirements understanding that mitigated problems related to missed requirements and requirements dependencies.

Coordination and alignment of RE and testing. We have reported on the situation of alignment between RE and testing in industry (Bjarnason 2013b). Two of the main challenges were found to be RE quality and the softer aspects of development, i.e. communication and collaboration (Bjarnason 2013b). Furthermore, a number of industrial practices for supporting alignment have been reported both by Bjarnason et al. (2013) and by Uusitalo et al.(2008) These practices include traceability between requirements and test cases, and increased communication between roles (Bjarnason 2013b, Uusitalo. 2008), e.g. by involving testers early in the project and in requirement reviews, and by establishing communication between testers and requirement owners (Uusitalo 2008). Similarly, Marczak et al. (2011) found that in requirements-driven collaboration, close communication between requirements and testing depends on key roles which when absent cause disruptions within the development team.

3 Research Method

The systematic map reported in this paper was performed based on guidelines for systematic mapping (Petersen 2008) and insights for systematic literature reviews

(Brereton 2007). The steps taken in designing and performing the study are described below. The study protocol and full list of papers included in the study can be found on-line (Bjarnason 2013a).

3.1 Research Questions

With the aim of locating research into RE distances within/between RE and later software development activities, the following research questions were formulated:
RQ1: Which RE-related distances are reported in peer-reviewed literature?
RQ2: To which extent is 'distance' used in GSD versus non-GSD papers?
RQ3: For which activities within RE has the concept of distance been researched?
RQ4: Towards which later development activities are RE distances investigated?

3.2 Search Strategy

The defined scope covers RE research and its intersection with later development activities. Papers focusing on non-RE topics were excluded, while general software development papers were included. Based on scope and research questions, search keywords were defined. The initial keywords were searched in well-known databases, e.g. IEEE Xplore, SciVerse. Based on search results, the keyword, scope and research questions were refined and search strings reformulated. The set of databases was expanded and re-searched for relevant papers.

3.3 Data Sources

Searches into the following databases are included in this mapping study:

1. IEEE Xplore (http://ieeexplore.ieee.org) covers computer science, electrical engineering, and electronic subject areas. Full-text and bibliographic access to almost 3 million of IEEE's publication including transactions, journals, magazines and conference proceedings published are provided.
2. Elsevier's SciVerse (http://sciencedirect.com) covers papers from more than 2,500 computer science and engineering journal.
3. ACM Digital Library (http://dl.acm.org) provides access to ACM journals, proceedings and transaction including ACM computing literature.
4. Inspec and Compendex provide access to huge amounts of scientific literature in many subjects including information technology, and are accessible via Engineering village's unified search interface (http://www.engineeringvillage2.org).

3.4 Data Retrieval

Search strings were constructed by combining the defined scope (software engineering OR software development OR requirements engineering) with the term 'distance'. The searches were limited to peer-reviewed material written in English. Material on 'distance learning' was excluded in the search to avoid a large number of irrelevant hits. The searches were limited to title, abstract and keywords.

3.5 Screening Process

The final searches yielded 2,427 papers (see Table 1). A title scan resulted in 161 relevant papers. The full references, abstract and search source of these papers were then stored in MS Excel (available on-line, see Bjarnason 2013a). Duplicates were removed; 148 unique papers. These papers were then included or excluded based on the abstracts. The inclusion/exclusion decisions for both title and abstract were cautious, i.e. when in doubt the paper was included. When an abstract contained insufficient information, the introduction was reviewed. In total 53 papers were included in the final set.

Table 1. Number of papers in each step of the screening process.

Source	Initial selection	Title review	Abstract review
SciVerse	51	7	2
IEEE Xplore	79	4	1
ACM Digital Library	1,951	52	33
Inspec	346	11	0
Compendex		74	17
TOTAL	**2,427**	**148**	**53**

3.6 Data Extraction, Classification and Synthesis

During data extraction and mapping, a classification scheme was developed according to guidelines provided by Petersen et al. (2008). A set of keyword were identified through exploratory coding of the abstracts, and then clustered into the categories of the map. In a few cases, the abstract was insufficient and parts of the full text were reviewed to ensure a correct understanding. Two sets of categories were identified. One related to context and focus of the research (main development activity, specific RE activity, and organisational distribution) and the other related to distance type.

The initial set of keywords for distance types was refined through analysing parts of the full paper text. In some cases, *forwards snowballing* was applied to locate additional papers, which were consulted to ensure a correct understanding of the used terms. The coding of all included papers was then revised to match the final set of codes. The final coding of the included papers is available on-line (Bjarnason 2013a).

Finally, a synthesis was performed on the included papers for each distance type to identify how the term is defined and applied, and if any causal relationships are reported for that term. In some cases, additional papers were located through forwards snowballing. For example, in GSD papers distances would typically be mentioned with a reference to previous work. In addition, for distances with only a few located papers supplementary searches on the specific distance type names were performed to identify additional papers. Parts of the full text was analysed for the synthesis, in particular introduction and conclusions sections, and all mentions of the term ‘distance’.

4 Results

4.1 Demographics of Retrieved Literature (and RQ2)

The search and selection resulted in 53 individual peer-reviewed papers. The majority of these (42) were within GSD. The distribution of papers over time, split into GSD / non-GSD context, is shown in Figure 1. The maximum was in 2009 with 11 papers. It is worth noting that within GSD a framework for categorizing GSD challenges based on three types of distances was published in 2005 (Agerfalk 2005) and that the following 4 years (2006-2009) have the largest number of papers found in this study.

The research type for each paper was classified according to the scheme suggested by Wieringa et al. (2006) The following categories were considered in this study:

1. *Evaluation research* investigates a problem or technique in practice and provides new knowledge of causal or logical relationships.
2. *Solution proposals* present a solution without a full-blown validation.
3. *Validation research* presents a solution proposal validated outside of industrial practice, e.g. experiments, prototyping, theoretical proof etc.
4. *Philosophical papers* sketch new theories or frameworks.
5. *Experience papers* describe the author's personal experience and may contain anecdotal evidence.

The distribution of the included papers according to research type and distribution context (GSD or non-GSD) is shown in Figure 2. The numbers indicate that, for the GSD context, more empirical evaluations and theoretical frameworks on the concept of distance have been researched than for the non-GSD context. For general development (non-GSD), the majority of included papers are in the form of validation research, indicating that more evaluation research is required into distances in the general software development context to establish foundations for more mature knowledge and for establishing theories based on empirical evidence.

4.2 Type of Distances (RQ1)

This study identifies thirteen distances. Eight of these, are distances between people, e.g. between roles, teams and organizations, while four address distances between artefacts. One distance concerns distance between an artefact (e.g. formal

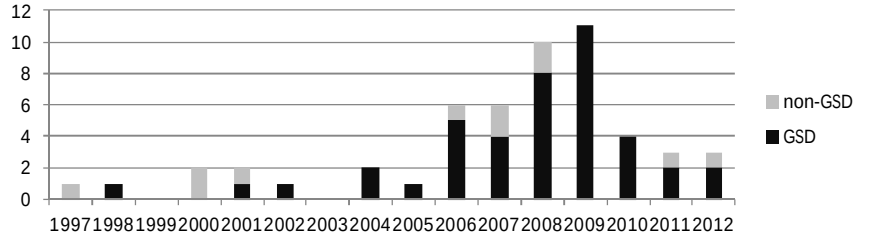


Figure 1. Number of papers per year, categorised according to GSD or non-GSD context.

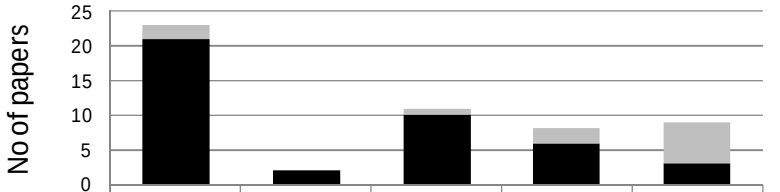
model) and reality. Unsurprisingly (since the majority of included papers address GSD), the most commonly referred distances are the ones defined within GSD, i.e. geographical, socio-cultural and temporal distances. Table 2 shows an overview of the number of papers for each distance. (The distances are described in Section 4.4.)

Table 2. The number of papers per distance type and software development activity. The bar indicates relative amount. Papers covering several categories are counted for each category.

			General	RE	Impl	Tools	Prj mgmt	Design	Process
		TOTAL	*29*	*17*	*8*	*7*	*6*	*1*	*1*
PEOPLE	Geographical	*41*	27	7	6	6	3		
	Socio-cultural	*25*	17	3	4	1	2		
	Temporal	*15*	11	1	2	1	2		
	Power	*3*	3						
	Opinions	*2*		2		1			
	Psychological	*2*			2				
	Organisational	*1*		1					
	Cognitive	*1*		1					
ARTEFACTS	Similarity	*3*		3			3		
	Semantic	*2*	1	2					
	Syntactic	*1*		1					
	Impact	*1*		1				1	
OTHER	Adherence	*2*	1	1	1				1

4.3 RE Activities (RQ3) and Later Software Development Activities (RQ4)

Distances were found in papers related to RE, project management, design, implementation, tools and processes. More than half of the papers (29 of 53) cover software development in general, while a third of the papers (17 of 53) cover RE, and a fourth (8 of 53) cover implementation. The numbers indicate that RE is acknowledged as an important activity for which distances are relevant to investigate. However, more research is needed to fully explore the field. In particular, research is needed on how RE distances relate to testing for which no papers were found, which is surprising considering that testing verifies that the requirements are fulfilled in the final product. A map of papers per distance type and development activity for which they were mentioned is shown in Table 2.



	Evaluation	Experience	Philosophical	Solution Proposal	Validation
non-GSD (11)	2		1	2	6
GSD (42)	21	2	10	6	3

Figure 2. The number of papers per research type and GSD vs. non-GSD context.

Of the 17 RE-specific papers, 7 address negotiation and 4 cover RE in general, while for handling changes, elicitation, specification, validation and traceability only the odd papers was found for each RE activity. 7 of the RE-specific papers purely address RE, while the others also cover software development in general (3), project management (3), tools (3) and implementation (1). Table 3 shows a map of RE-specific papers per development activity and RE activity.

Table 3. The number of RE-specific papers per RE activity and later activities. The bar indicates relative amount. Papers covering several categories are counted for each category.

		Negotiation	General RE	Specificaton	Analysis	Validation	Changes	Elicitation	Traceability
TOTAL		*7*	*4*	*3*	*3*	*2*	*1*	*1*	*1*
Pure RE	*7*	**4**	**1**	**1**	**1**	**1**		**1**	**1**
General	*3*			**2**	**2**	**1**			
Project managmt	*3*		**3**						
Tools	*3*	**3**							
Design	*1*						**1**		
Implementation	*0*								

4.4 RE Distances in Context

The systematic map identifies 13 RE distances between people, artefacts, and other entities. This section describes each distance based on included papers.

4.4.1 Distance between people

Geographical distance denotes 'a directional measure of the effort required for one actor to visit another at the latter's home site [or home work place]' (Agerfalk 2005). Even a geographical distance of 25 metres, i.e. within the same office building, has been found to reduce communication between engineers (Allen 1977). For off-shored projects where RE is geographically separated from other software development activities Dibbern et al. (2008) found that this distance can be a significant cost driver. In particular, in cases where client-specific knowledge was crucial face-to-face collaboration was required for adequate knowledge transfer of domain knowledge and for requirements analysis and specification (Dibbern 2008). Tools for enhancing distributed group communication have been suggested for collaborative RE activities such as requirement negotiation and requirements traceability towards goals and design artefacts (Herlea 1998). Calefato et al. (2007) found that computer-based communication provided better support for elicitation than for negotiation, and suggest that the general preference for face-to-face communication might be explained by this weakness of computer-based negotiations. In contrast, Damian et al. (2001) found that when using technology for negotiating requirements the group's overall performance was not decreased compared to when negotiating face-to-face, and could even be more effective in integrating multiple stakeholders' needs. Similarly, Wolf et al. (2008) found no significant delays for geographical distance in a case study. This was believed to be due to practices applied to bridge these distances (collaborative tools, and processes and practices adapted to distributed software teams), but may also be explained by

the fact that the delays were quantified as opposed to qualitatively measured as for most other studies (Wolf 2008).

Temporal distance denotes 'a directional measure of the dislocation in time experienced by two actors wishing to interact' (Agerfalk 2005) due to different time zone, work shifts etc. In general, short temporal distances allow for timely synchronization between team members, while long temporal distances reduced the opportunities for synchronous communication and introduce delayed feedback (Agerfalk 2005). Time zones and work shift schedules may work together to decrease temporal distance by adjusted office hours or utilized for working around the clock by passing on tasks between teams in different time zones (Agerfalk 2005). Yousuf et al. (2008) suggest that when temporal distance is present certain requirements validation techniques which do not rely on synchronous communication are more suitable than others.

Socio-cultural distance denotes 'a directional measure of an actor's understanding of another actor's values and normative practices' (Agerfalk 2005) and includes organisational and national culture, language, individual motivations, work ethics, and politics. In general, communication is improved by low socio-cultural distance thereby reducing risk, while long socio-cultural distances increase the risk of misunderstandings and may make coordination harder (Agerfalk 2005). However, long distances also have a potential for increased learning and access to a richer skill set, and be stimulating for innovation (Agerfalk 2005).

In the context of RE for GSD, Dibbern et al. (2008) found that cultural distance can be a significant cost driver for a company with off-shored projects. Increased costs may be incurred for transfer of knowledge of domain, requirements etc., and additional specification effort to ensure accurate requirements. Yousuf et al. (2008) mention socio-cultural distance as potentially influencing requirements validation though without specifically analysing how. Real-time machine translation has been proposed for requirements negotiation among stakeholders separated by language barriers, and found to not disrupt real-time interaction in text-based chat (Calefato 2007).

Opinion distance denotes a measure of the difference of opinion on a certain aspect of an item between two actors. This distance has been investigated between decision makers and stakeholders in requirements negotiations with the aim of supporting group decision by measuring the differences in linguistic opinions of alternatives based on multiple criteria (Chakraborty 2007). Chakraborty and Chakraborty (2007) propose using a fuzzy distance measure to measure the distance between fuzzy clusters of the opinions in order to improve 'accuracy' of the decision by identifying dissimilar opinions. Similarly, Zhu and Hipel (2012) propose a method for dealing with multi-stage information, i.e. when information about alternatives evolves over time.

Organisational distance denotes a measure of one organisational unit's understanding of another unit's goals and perspectives, e.g. concerning priority of customer requirements relative cost of code design and quality. The organisational distance between people involved in RE was categorised in a study on pairing on RE tasks as *internal* or *external* depending on if they are part of the development

team or not (Yu 2011). The study suggests that sharing RE tasks is more effective when there is a shorter organisational distance due to less delay in the (shorter) communication paths (Yu 2011).

Psychological distance denotes a measure of the perceived psychological (subjective) effort of an actor to communicate with another actor (Prikladnicki 2012). This distance has been researched for software development in general, though not specifically for RE. Prikladnicki (2012) has defined a measurement for the perceived distance between people. This measurement relates to the social dimension of psychological distance that addresses the distance of a stimulus (social object or event) from the perceiver's self (Liberman 2007), e.g. my best friend or a person from another culture. The measurement was evaluated in a project with development distributed between Brazil and India. The study shows that the psychological distance does not necessarily correspond to the geographical distance, but to a high degree depends upon trust and communication though the impact of these factors varied per country and per role (Prikladnicki 2012). For example, a project engineer in Brazil perceived the lowest distance while a project manager (also in Brazil) perceived the highest psychological distance.

Power distance denotes a measure of the degree to which unequal distribution of power is accepted within a society (Hofstede 1993). This distance has been researched for software development in general, though not specifically for RE. This distance is one of the dimensions of socio-cultural distance and has been found to affect relationships within distributed development and thereby also the success of distribution (Winkler 2008). Winkler et al. (2008) found that difference in power distances may negatively affect communication. For example, in a culture with a large power distance saying no or voicing criticism is avoided, detailed specifications are preferred and instructions are preferred from superiors rather than from peers. All of these factors pose a risk of complicating collaboration with team members used to shorter power distances and more open communication (Winkler 2008). Wende and Philip (1010) found communication via instant messaging improved communication and, thus, enabled bridging power distances.

Cognitive distance denotes a measure of the difference between two actors' cognition, e.g. what they each know and are aware of. Yu and Sharp (2011) observed this distance in a case study on pairing on RE tasks and identified that when one person fills many roles communication is immediate since the cognitive distance between the roles is zero, which is beneficial for communication and coordination.

4.4.2 Distance between artefacts

Similarity distance denotes a measure of the similarity between an entity and another entity of the same type, e.g. project. This distance has been suggested as supporting the coordination between RE and project management, in particular for cost estimation of requirements. In analogy-based software effort estimation, the concept of similarity distance is used to identify completed projects with similar characteristics by measuring the Euclidian distance between project features (Sheppard 1997), e.g. number of requirements, number of interfaces, project model

etc. This approach has been validated using industrial data sets and the results confirm that this approach outperforms the usage of algorithmic models for effort estimation (Sheppard 1997).

Several different approaches and variations have been proposed for measuring similarity distance. Chiu and Huang (2007) propose adjusting the estimations to take into account the re-use effect of the project identified as the most similar. Azzeh et al. (2008) propose an approach that supports handling uncertainties and imprecision in project attributes by the use of fuzzy C-means clustering and fuzzy logic. With this approach, each attribute is represented with several fuzzy sets instead of by a single value. Furthermore, this approach clusters together the most similar projects and their values are represented in the same fuzzy set. The similarity between two projects is then measured by the similarity distance between the two sets to which they mostly belong (Azzeh 2008).

Impact distance denotes a measure of the number of steps with which a change in one entity impacts another entity, e.g. through dependencies. This distance has been proposed by Briand et al. for addressing the issue of impact analysis, e.g. for requirements changes, in a UML modelling context. A measurement of the distance between a changed element and an impacted element is defined as the number of impact analysis rules, or steps, required to identify that the impacted element is affected by the change (Briand 2006). Initial empirical evaluations indicate that impacted elements at distance one lead to code changes, while those with a greater distance, in most cases, do not. However, further evaluations are required to determine at which maximum distance code changes for impacted elements should be considered (Briand 2006).

Semantic distance denotes a directional measure of the amount of functionality of a specification that distinguishes it from another related specification. Semantic distance between requirements specifications and other artefacts may be used for supporting software re-use, e.g. to identify library components with a short semantic distance to the requirements. Jilani et al. (2001) pose a theoretical case that the use of semantic distance is applicable for decisions on black-box re-use and define a number of metrics for semantic distances. These include metrics for functional deficit that reflect how much functionality needs to be added to one specification in order to satisfy another, and metrics for functional excess that measure the amount of functional features of one specification that are irrelevant to another one (Jilani 2001).

Syntactic distance denotes a measure of dissimilarity of the design structure of two artefacts (Jilani 2001). Syntactic distance between specifications has been suggested by Jilani et al. (2001) for supporting decisions on white-box reuse (where a component is modified). While providing theoretical arguments for applicability of this type of distance Jilani et al. (2001) also argues that it is unrealistic to define a measure for syntactic distances since this requires a uniform representation of specifications irrespective of abstraction level and a canonical scheme that supports the definition of a unique representation of specifications. Instead, semantic distances (for which measurements are defined) are suggested to be used as an approximation of syntactic distances (Jilani 2001).

4.4.3 Distance between other entities

Adherence distance denotes the size of the difference between a formal or theoretical model of a process or a phenomena and the actual enactment of it. Within software development this distance has been suggested for gauging the degree of adherence for models. For example, Huo et al. (2008) consider the distance between a formal process model and the actual work practices observed in a project, though no measurement of this distance is defined. Furthermore, a measure of the distance between a theoretical distribution and actual estimates is defined and evaluated by Thelin and Runeson (2000) in the context of assessing the accuracy of remaining faults in an inspected software artefact, which could be applied to validation of requirements specifications.

4.5 Limitations

Reliability of the results due to the risk of researcher bias in the inclusion process and the classification process remains an open issue since only one researcher was involved. However, for inclusions/exclusion a generous policy was used, and independent validation of both inclusion and classification is possible since the full set of papers, including the ones excluded through abstract review, is available on-line. Furthermore, there is a risk of incorrect classification when only performed on an abstract. This was addressed by reviewing the full text when the abstract was unclear. However, replication of the study may result in a slightly different set of papers, both in the initial search and in the inclusion/exclusion step.

Conclusion validity concerning the completeness of the results (e.g. number of distances) is one of the main limitations of this study. The search string was limited to 'distance' and did not include synonyms such as gap, proximity etc. This risk of missing relevant papers was partly addressed by broad searches for other aspects. For example, papers were collected from multiple sources incl. IEE and ACM, and wide search terms (software development, software engineering) were used for the scope aspect of the search. Furthermore, no limitation was set on publication year or type of publication (journal, conference etc.). These measures resulted in the study starting with a large set of papers (more than 2,000). However, extending the search to include synonyms would produce an even larger set of papers, and may uncover additional types of distances and applications of these. The main intention of this study was to act as a starting point and further research is planned to further explore the area.

5 Discussion

RE is a communication intense activity and the identified distances between people (see overview in Figure 3) may have an impact on the efficiency and effectiveness of communication and collaboration (Agerfalk 2005, Allen 1977, Winkler 2008, Wolf 2008, Yu 2011) and can be a significant cost driver (Dibbern 2008). Within GSD, cases where communication is equally strong, or even improved, compared to co-located development have been reported (Damian 2001, Wolf 2008). For

example, computer-based group meetings were found to be more effective for requirements negotiation than face-to-face meetings (Damian 2001). Similarly, development environments with computer-based support for collaborative work in combination with best practices were found to contribute to reducing communication delays (Wolf 2008).

These contradicting results might be explained by the effect the applied practices have on the division between formal and informal communication. When (previously) informal information is re-routed to more formal communication channels the communication flow may be improved, resulting in reaching a wider audience. This correlates well with findings by Agerfeldt et al.(2005). Distance tends to affect informal communication in particular and leads to reduced trust, difficulty in conveying vision and strategy and lack of awareness (Agerfalk 2005). Cases where formal communication including documentation is weak and the informal channels are important (e.g. for agile development) are likely to be very vulnerable to distances between people.

Some of the distances are subjective (e.g. geographical) while others are objective and based on perception (Prikladnicki 2012), values and normative practices. The perceived (objective) distance can vary over team members and over time and research has shown that quantifying this distance can support management and be beneficial for GSD practices (Prikladnicki 2012). All the objective people distances, i.e. organisational, power, opinions, cognitive and psychological, seem to be covered by the socio-cultural distance (see Figure 3). More research into these distances specifically for RE and for collocated development could potentially explain issues reported for RE communication and collaboration (Bjarnason 2011, Curtis 1988, Marczak 2011). For example, several distances may be at play in co-located cross-functional teams with a product owner from a different organisational unit and with an RE background; short geographical, but long organisational and cognitive distances between the product owner and other team members. Awareness of distance and their impact could support management in optimising organisations (Yu 2011), training efforts, and selected methods (Yousuf 2008) and tools (Calefato 2007, Damian 2001, Herlea 1998, Wende 2010, Wolf 2008).

Temporal distance affects the possibility of synchronous communication, and within GSD asynchronous communication is common (Agerfalk 2005, Yousuf 2008). In addition, subjective distances caused by differences in culture, language etc. may make people reluctant to communicate directly, thus resulting in preferring to communicate via e-mail or through issue management systems. In general, the asynchronous communication that these distances may incur induce delays and increase lead times of RE and the entire development effort (Yousuf 2008). This may affect communication intense activities such as RE, both in general and in

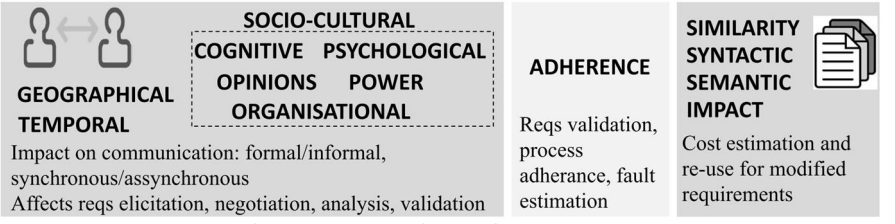


Figure 3. Overview of interpretation of identified RE distances including relevant RE areas.

particular for elicitation and negotiation.

Artefacts play an important role in communicating requirements to stakeholders and within a development project. The identified distances between artefacts have primarily been researched for cost estimation and re-use based on changes to, or different versions of, RE artefacts (Briand 2006, Jilani 2001, Shepperd 1997). These distances may be used to characterise coverage and consistency between artefacts of different activities, e.g. as a measure of the alignment between RE and later development activities. For this reason, RE distances to artefacts of later development activities are an important area to research.

Adherence distance between an artefact and the actual enactment of it has been suggested for process improvement (Huo 2008) and for estimating remaining fault content (Thelin 2000). Additional interesting applications could be adherence between a requirement specification and the final product, as well as, the actual customer needs. Both of which are key factors for successful RE.

Finally, most of the identified distances are reported to be better the shorter they are, but there are some interesting exceptions. Within GSD, long socio-cultural distance may potentially increase learning by providing access to a richer skill set, and be stimulating for innovation (Agerfalk 2005). Furthermore, organisational distance between testers and developers has been reported to improve alignment between testing and requirements by avoiding testing against developers' interpretation of the requirements (Bjarnason 2011). Identifying and understanding additional cases where long distances result in positive effects can support defining a comprehensive theory of the impact of RE distances on software development.

6 Conclusions

Coordination and alignment of requirements with later activities is vital for enabling continuous development of successful products. Within global software engineering distances are reported as increasing risk and cost. Distances between RE and other development activities, e.g. in decision making and requirements communication, may hinder effective and efficient development of customer requirements.

In this systematic mapping study 13 RE-related distances were identified. Distances were mainly found between people (roles, teams etc.) and between artefacts (requirements and design specifications etc). Distance between people has primarily been researched within the context of GSD (geographic, socio-cultural and temporal), while distance between artefacts was found exclusively in non-GSD research.

GSD research on *distance between people* is fairly mature, though more empirical research is needed to understand the impact of these distances for non-distributed development, e.g. for large-scale development. Furthermore, no theory was found in the reviewed papers that could explain the contradicting findings of several studies concerning geographical distance. Further investigations are required to gain a deeper insight into relationship between different distances and the impact they have on division between formal and informal communication. Findings from other fields like psychology and cognitive science are relevant to

consider when investigating these people-related distances in relation to RE activities.

Distance between artefacts has been suggested in the context of requirements change and traceability and is an interesting area for future RE research. Distance between RE artefacts and artefacts of later development activities, e.g. design and testing, could potentially be used to measure coverage and consistency between RE and other artefacts such as design and test specifications, and source code.

The systematic map reveals that RE distances in relationship to later development activities (e.g. design, implementation and testing) is largely un-researched. If distance is indeed an important factor in the coordination and communication of RE, research is much needed to address this gap. Examples of RE activities where distance may play an important role include elicitation, negotiation, specification, managing requirements changes and requirements traceability.

This study is a first step towards exploring and defining a theory for the role of RE distances in software development. Future work includes constructing a theoretical framework for RE distances in relationship to testing based on previous research and on empirical data.

Further empirical research into how RE distances, and combinations of these, affect later development activities may support constructing a theory that explains what mechanisms are at play in development projects, between people, artefacts and activities. Increased knowledge of such factors might enable optimization of RE methods and practices for eliciting, negotiating and communicating requirements. Furthermore, through researching new methods and practices for bridging or decreasing distances the effectiveness of RE in software development may be improved, ultimately resulting in more efficient development of better products.

References

Agerfalk PJ, Fitzgerald B, Holmstrom Olsson H, Lings B, Lundell B, Ó Conchúir E (2005) A Framework for Considering Opportunities and Threats in Distributed Software Development. Proc. Of Int. Works. on Distr. Softw. Eng., DiSD 2005, pp. 47-61.

Allen T (1977) Managing the flow of technology. Cambridge, MA, MIT Press

Azzeh M, Neagu D, Cowling P (2008) Software project similarity measurement based on fuzzy C-means. Proc. of Int. Conf. on the Softw. process, ICSP'08, pp. 123-134.

Bjarnason E (2013a) Study material for RE distance study incl. list of all papers, http://serg.cs.lth.se/research/experiment_packages/distmap/, latest access 2013-01-28.

Bjarnason E, Runeson P, Borg M et al. (2013b) Challenges and Practices in Aligning Requirements with Verification and Validation: A Case Study of Six Companies. Journal of Empirical Software Engineering, nn(nn):x-x, 2013.

Bjarnason E, Wnuk K, Regnell B (2011) Requirements are Slipping Through the Gaps – A Case Study on Cause & Effects of Communication Gaps in Large-Scale Software Development. Proc. of 19th IEEE Int Requirements Engineering Conf., pp. 37-46.

Brereton P, Kitchenham BA, Budgen D et al. (2007) Lessons from applying the systematic literature review process within the software engineering domain. Journal of Systems and Softw., 80(4), Pages 571-583.

Briand LC, Labiche Y, O'Sullivan L, Sówka MM (2006) Automated impact analysis of UML models. Journal of Systems and Softw. 79(3), pp. 339-352.

Calefato F, Damian D, Lanubile F (2007) An Empirical Investigation on Text-Based Communication in Distributed Requirements Workshops. Proc. Of Int. Conf. on Global Softw. Engineering (ICGSE '07), pp. 3-11.

Chakraborty C, Chakraborty D (2007) A fuzzy clustering methodology for linguistic opinions in group decision making. J. of Applied Soft Computing, 7(3), pp. 858-869.

Chiu NH, Huang SJ (2007) The adjusted analogy-based software effort estimation based on similarity distances. Journal of Systems and Software, 80(4), pp. 628-640.

Curtis B, Krasner H, Iscoe N (1988) A Field Study of the Software Design Process for Large Systems. Commun. ACM, vol. Nov. 1988, pp. 1268-1287.

Damian D (2001) An empirical study of requirements engineering in distributed software projects: Is distance negotiation more effective? Proc. Of 8th Asia Pacific Softw. Engineering Conf. APSEC'2001, pp. 149-152.

Damian D, Chisan J (2006) An Empirical Study of the Complex Relationships between Requirements Engineering Processes and Other Processes that Lead to Payoffs in Productivity, Quality, and Risk Management. IEEE Trans on Softw. Eng. 43(7), pp 433-453.

Dibbern J, Winkler J, Heinz A (2008) Explaining variations in client extra costs between software projects offshored to India. MIS Quarterly: Management Information Systems, vol. 32, n:o 2, pp. 333-366.

Ebert C (2006) Understanding the product life cycle: four key requirements engineering techniques. IEEE Software, vol.23, no.3, pp.19-25.

Fricker S, Glinz M (2010) Comparison of Requirements Hand-Off, Analysis, and Negotiation: Case Study. Proc. of 18th International Requirements Engineering Conferance, pp. 167-176.

Herlea D, Greenberg S (1998) Using a groupware space for distributed requirements engineering. Proc. of Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, WET ICE, pp. 57-62.

Hofstede G (1993) Cultural constraints in management theories. Academy of Management Executive, 7(1), pp. 81 – 94.

Huo M, Zhang H, Jeffery R (2008) Detection of consistent patterns from process enactment data. Proc. of Int. Conf. on Software Process, ICSP'08, pp. 173-185.

Jilani LL, Desharnais J, Mili A (2001) Defining and Applying Measures of Distance Between Specifications. Journ. IEEE Transactions on Softw. Eng, 27(8), pp. 673-703.

Lawson M, Karandikar HM (1994) A Survey of Concurrent Engineering. Concurrent Engineering 1994 2:1.

Liberman N, Trope Y, Stephan E (2007) Psychological Distance, book chapter in 'Social psychology: handbook of basic principles' (2nd ed), Kruglanski, Higgins (Eds), pp. 353-381, Guilford Press.

Marczak S, Damian D. (2011) How Interaction between Roles Shapes the Communication Structure in Requirements-Driven Collaboration. Proc of 19th IEEE Int Req. Eng. Conf., pp. 47-56

Nuseibeh B (2001) Weaving together requirements and architectures. Computer , vol.34, no.3, pp.115-119.

Petersen K, Feldt R, Mujtaba S et al. (2008) Systematic Mapping Studies in Software Engineering. 12th Int. Conf. on Evaluation and Assessm. in Software Eng., pp.71-80

Prikladnicki R (2012) Propinquity in Global Software Engineering: Examining Perceived Distance in Globally Distributed Project Teams. Journal of Softw. Evolution and Process, 24(2), pp. 119-137.

Shepperd M, Schofield C (1997) Estimating Software Project Effort Using Analogies. IEEE Trans. on Software Eng, Vol. 23 Issue 11, pp. 736-743.

Swartout W, Balzer R (1982) On the Inevitable Intertwining of Specification and Implementation. Comm. ACM, vol. 25, no. 7, pp. 438-440.

Thelin T, Runeson P (2000) Robust Estimations of Fault Content with Capture–Recapture and Detection Profile Estimators. Journal of Systems and Softw. 52(2–3), pp. 139-148

Uusitalo EJ, Komssi M, Kauppinen M et al. (2008) Linking Requirements and Testing in Practice. 16th IEEE Int Requirements Engineering Conf, NJ, USA, pp. 265-270

Wende E, Philip T (2010) Instant Messenger in Offshore Outsourced Software Development Projects: Experiences from a Case Study. Proc. of 44th Annual Hawaii Int. Conf. on System Sciences.

Wieringa R, Maiden N, Mead N, Rolland C (2006) Requirements Engineering Paper Classification and Evaluation Criteria: a Proposal and a Discussion. Journal of Requir. Eng. 11(1), pp. 102-107

Winkler JK, Dibbern J, Heinzl A (2008) The Impact of Cultural Differences in Offshore Outsourcing-Case Study Results from German-Indian Application Development Projects. Inf. Systems Frontiers, v 10, n 2, pp. 243-258

Wolf T, Nguyen T, Damian D (2008) Does Distance Still Matter? Journal of Impr. and Practice of Softw. Process, 13(6), pp. 493-510

Yousuf F, Zaman Z, Ikram N (2008) Requirements Validation Techniques in GSD: A Survey. Proc. 12th IEEE Int. Multitopic Conf. INMIC'08, pp. 553-557

Yu Y, Sharp H (2011) Analyzing Requirements in a Case Study of Pairing. Proc. of 1st Int. Workshop on Agile RE, Lancaster, UK.

Zhu J, Hipel KW (2012) Multiple Stages Grey Target Decision Making Method with Incomplete Weight Based on Multi-Granularity Linguistic Label. Journal of Information Sciences, vol. 212, pp. 15-32

PAPER V

VARIATIONS ON THE EVIDENCE-BASED TIMELINE RETROSPECTIVE METHOD A COMPARISON OF TWO CASES[1]

Even though project retrospectives can be a powerful tool for process improvement and obtaining new learning and insights, pure experience-based reflections pose a risk of leading to incorrect conclusions. Our method, evidence-based timeline retrospectives (EBTR), mitigates this risk by providing a pre-generated timeline that visualises project history based on evidence rather than relying on subjective opinions and biased memories. Within the scope of a comparative study of two cases a set of variation points has been evaluated. The variation points enable configuring the EBTR method to different contexts and retrospective goals. The results indicate that by selecting certain variations the EBTR method can be configured to support either wide assessments (e.g. the overall impact of a new process) or assessments of a specific process area. For example, through using open or semi-structured discussions, or by varying the applied timeline technique.

[1] By E. Bjarnason, A. Hess, J. Doerr, B. Regnell. Published in proc. of 39th Euromicro Conf. Series on Software Engineering and Advanced Applications, 2013, pp. 37-44.

1 Introduction

Software process engineering is considered fundamental in software engineering (Emami 2010, Dikici 2012). Thus, the identification of weaknesses and improvement opportunities of software engineering processes is an important but challenging activity (Emami 2010). Several approaches have been proposed that support software process improvements, e.g., based on simulations (Cabral Silva Filho 2010), application lifecycle management solutions (Lacheiner 2011), or prioritization techniques (Birkholzer 2011).

In general, retrospective analysis can be an effective tool for assessing software processes by identifying problems and best practices. Retrospective meetings can support process improvements both directly by identifying weaknesses and improvement strategies, and indirectly through team members gaining new insights and learning concerning best practices (Collier 1996, Derby 2006, Drury 2011, Bjarnason 2012). However, retrospectives based solely on participants' experiences of events pose a risk of drawing incorrect conclusions (Jørgensen 2000) and may become a forum for emotional venting rather than constructive discussions (Collier 1996, Drury 2011).

An evidence-based retrospective method was designed to combat this by injecting the retrospective with a pre-generated timeline of visualised project history based on evidence gathered from available systems (Bjarnason 2012a). This evidence-based timeline retrospective (EBTR) method was previously evaluated for one case (denoted case 1 in this paper) (Bjarnason 2012b).

In order to further evaluate and explore the EBTR method it was applied to a second case (denoted case 2 in this paper) and the outcome compared. The aim of this comparative study was to empirically observe the effect of varying the EBTR method over a set of variation points (VPs). In this paper, we report on the influence that each VP may have on (RQ1) new insights and learning; (RQ2) timeline support for meeting; and (RQ3) topics discussed at the retrospective meeting.

The remainder of this paper is structured as follows: Section 2 presents the EBTR method, and Section 3 describes the two cases. Section 4 presents the research method and the evaluated EBTR variants are described in Section 5. The results are presented in Section 6 and discussed in Section 7, and we conclude in Section 8.

2 Evidence-Based Timeline Retrospectives

Evidence-based timeline retrospectives (EBTRs) inject pre-constructed timelines into retrospective meetings. Project history is visualised in evidence-based timelines (EBTs) by displaying time-stamped evidence of project events from various systems. EBTs can prompt memory and support reflection of past events. At a retrospective meeting multiple roles share their experiences, reflect on events and good practices, and identify improvements. Kerth (2001) describes a method where a timeline is produced at the meeting by the participants. Our method enhances on this by providing pre-prepared EBTs, which saves meeting time and provides

objective information. In addition, it includes a phase for planning and one for validation to ensure final agreement.

The method was initially designed for assessing RE in a project context though generic enough to be customised for different retrospective goals. The generic method is described in this section, while the EBTR variants for the two cases are described in Section 5.

The EBTR method consists of four phases: (1) planning, (2) EBT construction, (3) EBTR meeting with the project team, and finally, (4) validation of the outcome. Each phase is described in the following sections.

2.1 Phase 1: EBTR Planning

The definition of goals in this phase enables focusing the EBTR on strategic improvement areas. The main vehicles for achieving these goals are the EBTs (see phase 2) and a set of focus questions (see Bjarnason 2013). The focus questions are defined in the planning phase and used at the EBTR meeting (see phase 3) to focus discussions on issues relevant to the EBTR goals.

The EBTs are constructed based on aspects, evidence, and visualisation that are all defined during the planning phase. The aspects to visualise in the EBT are defined based on the goals. The type and source of evidence to collect and suitable visualisations are identified. The projects to include in the assessment are also selected in this planning phase.

2.2 Phase 2: EBT Construction

The EBTs are constructed by collecting evidence from various systems, e.g. scope and prioritisation systems, requirements databases, planning tools, defect management systems, etc. The project history is visualised by displaying this evidence along a timeline for each aspect.

2.3 Phase 3: EBTR Meeting

The EBTR meeting is intended to facilitate group reflection in-line with EBTR goals and was designed according to guidelines for project retrospectives (Kerth 2001) and focus groups (Robson 2002). The focus questions (from phase 1) and EBTs (from phase 2) are used to stimulate a discussion.

The meeting participants represent key roles throughout the project life cycle, similar to Collier's retrospective method (Collier 1996); ideally 4-8 project members and 1 moderator.

The meeting room is prepared by posting the EBTs on the wall. In addition, a whiteboard or flipchart, and pens and post-its are needed for capturing information. Seating the participants around the EBTs encourages interaction with the EBTs and with each other.

The EBTR goal and EBTs are presented at the meeting. The moderator then leads a discussion based on the focus questions (see phase 1). A set of prompting questions suggested by Kerth (2001) is available for reinvigorating or redirecting discussions. The participants add clarifications, corrections and additional information to the EBTs, thus, producing updated and jointly agreed EBTs.

The final part of the meeting consists of jointly summarising the findings and lessons learned by using a set of sum-up questions that are based on the concluding part of Kerth's timeline exercise: things that worked well; what was learnt; what needs improving; what is still puzzling; and what needs to be discussed further.

2.4 Phase 4: EBTR Validation

In this phase the meeting outcome and conclusions are validated by the retrospective participants reviewing the notes and updated EBTs. Additional validation can be obtained through additional meetings to agree on an action plan for addressing identified problems and improvements.

3 The Two Cases

3.1 Case 1: Product Development Company

The EBTR method was initially designed for and applied at a company in the telecommunication domain. The company has around 4,000 employees and develops software using an agile development process. All new functionality is defined as features that are prioritised in a product backlog and developed in order of priority. Each feature is developed in a separate feature project that integrates software into software release projects. A feature project life cycle has a lead time of 9 weeks to 2 years and includes handovers between different units and teams; from request through design, development in cross-functional teams, system integration and system testing, and finally customer acceptance. Typically around 200-250 features are integrated into a main software release project.

A feature project involves several roles including product manager, project sponsor, project manager, project architect, developer and tester. The product manager acts as a customer proxy and is responsible for scope decisions. The project sponsor is responsible for ensuring resources. The feature architect is responsible for adhering to architectural strategy and guidelines. The developers and testers iteratively detail requirements in collaboration with the product manager, and develop and verify software accordingly. Finally, the feature project interacts with system-level roles for architecture, integration, and testing.

3.2 Case 2: Research Project

The EBTR method was applied to a German research project called IBIS (Fetzer 2013), which involved two research partners and two company partners, SMEs (small and medium sized enterprises) with ~20 employees. The project aimed at designing a method that enables developers without specific knowledge in usability engineering to systematically design software products that are intuitive to use, creative and innovative. The resulting IBIS method was designed by integrating image schemas (Hurtienne 2008) (recurring cognitive structures and patterns) into a task-oriented requirements engineering process (Adam 2009); and it was defined to be easy to integrate into the company partners' software engineering (SE) processes. The usefulness and applicability of the IBIS method was evaluated

throughout the research project through comparison of industrial projects conducted at each company's site some using the method.

Different roles were involved in the IBIS project: researchers that developed and evaluated the IBIS method; project managers for each company partner and software engineering roles defined by the IBIS method and typically included in SE projects at the company's sites comprising product managers being responsible to elicit / specify requirements and evaluate intermediate / final product versions with the customers / end users, developer being responsible to design interactions and corresponding UIs, to implement and test the software products.

4 Research Method

The main aim of this comparative study was to explore and evaluate variations of the EBTR method by comparing two cases. A number of variation points (VPs) were identified and an EBTR variant was applied to each case (see Figure 1). The outcome for the two cases has been analysed to identify differences potentially caused by the VPs. This comparative study was performed in three main steps: preparations, data collection and data analysis.

4.1 Preparations

The study was prepared at a number of meetings where the researchers discussed the EBTR method, and designed and planned this study. Previous experience of applying the method (for case 1) was shared and potential VPs were discussed and agreed. For example, the same focus questions were to be used for both cases, but for case 2 the retrospective discussions were to be more structured and limited to these questions. Furthermore, the EBTR meetings were to be longer for case 2 (4 hours vs. 75 minutes) due mainly to availability. The full set of variations points is described in Section 5. Characteristics for comparing the selected projects were also discussed and agreed, and the reported set is shown in Table 1.

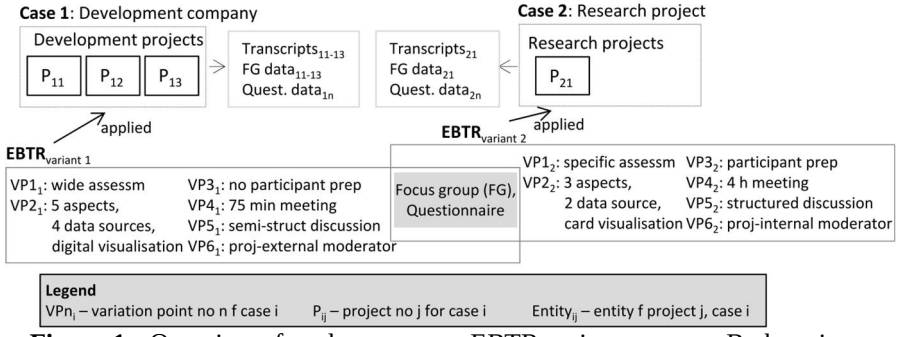


Figure 1. Overview of study setup: one EBTR variant per case. Both variants evaluated through transcription, focus group and questionnaire.

A separate researcher managed each case and was the contact point between this study and the investigated project(s). This researcher planned and performed the EBTR(s) for their case including constructing the EBTs.

4.2 Data Collection

The same data collection protocol was used for both cases. Apart from jointly updated EBTs, extensive notes were taken at the EBTR meetings. Transcriptions were sent to the participants for validation. Furthermore, the participants' EBTR experiences were gauged by a questionnaire with scale Not at all, Somewhat, Fairly much and Very much and a focus group with evaluation questions. Identical questionnaire and evaluation questions were used in both cases (available on-line, Bjarnason 2013).

4.3 Analysis

The final set of VPs and their potential impact were identified at a workshop by the involved researchers. At this workshop the EBTR variants for each case were presented and the collected data compared. Differences and similarities were discussed and classified as VPs or as effects of a VP.

In order to understand the impact of the variations on EBTR meeting discussions, a topic analysis was performed at the workshop on the notes of one meeting for each case. The researcher responsible for the case analysed the notes and identified the discussed topics. These topics were then matched to the focus question topics used at the EBTR meetings and the findings compared between the two cases.

Similarly, the focus group and questionnaire data were compiled and analysed by comparing the results from the two cases. The observed differences were then compared to the VPs and potential connections identified.

5 Two Variations of the EBTR Method

The EBTR variant for case 1 was applied to three development projects (P11-P13), while the EBTR variant for case 2 was applied to one research project (P21). Project characteristics are shown in Table 1. The variants differ in the following VPs:

(VP1) EBTR goal
(VP2) EBT content and visualisation
(VP3) EBTR meeting participant preparations
(VP4) EBTR meeting length
(VP5) discussion structure at EBTR meeting
(VP6) EBTR meeting moderator

Table 1. Each EBTR variant is described below. The relevant VPs are given within parenthesis. Characteristics of the included projects.

Project id	Lead time (months)	Project size: developers of total	N:o of roles in project	N:o of EBTR particip.
Case 1				
P_{11}	28	1 of 4	6	4
P_{12}	13	1-2 of 13	8	9
P_{13}	14	4-5 of 13	9	6
Case 2				
P_{21}	7	4 of 11	4	5

5.1 EBTR Variant for Case 1

5.1.1 Phase 1: EBTR Planning

For case 1, the EBTRs were planned in close collaboration with company representatives and EBTR goals, aspects and evidence were defined and agreed. The main goal ($VP1_1$) was a general assessment of the RE aspects of the company's new development model and what impact it has on project lead time. A secondary goal was to encourage project members to reflect and learn about good requirements practices, and thereby enable improvements in future projects. In line with these goals, evidence was selected to cover a number of aspects ($VP2_1$), namely people, project state, decisions, artefacts and planning.

5.1.2 Phase 2: EBTR Construction

Evidence for each aspect ($VP2_1$) was selected to represent both high-level events (e.g. project phase) and low-level events (e.g. customer meeting, filing of issue report). The evidence was extracted from available systems for scope, release and project planning, defect management, requirements and test management. The extracted data was visualised in one MS Visio timeline per aspect (see Figure 2) and printed on A3 papers (4 in width).

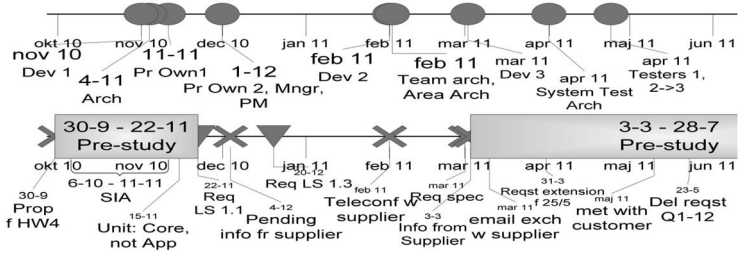


Figure 2. An extract from the EBTs used for case 1.

5.1.3 Phase 3: EBTR Meeting

Participants were invited to the meeting without prior information about the EBTR method or any other preparations ($VP3_1$). The meetings were booked for 75 minutes ($VP4_1$). Two of the meetings ran over with approx. 15 minutes. A semi-structured discussion format was used ($VP5_1$). The EBTs and the focus questions were used to encourage open discussions. Spontaneous follow-up questions were used to explore mentioned issues and topics. Despite attempting to include everyone there were junior participants who said nothing or very little, in particular in the largest meeting (P_{12}) and in one with two very strong leadership roles (P_{11}). The moderator was well acquainted with the case (e.g. development process and terminology) but had no prior relationship to the projects ($VP6_1$).

5.1.4 Phase 4: EBTR Validation

Notes of the discussions were sent out to the participants a few days after the meeting together with updated EBTs. Evidence added at the EBTR meeting was marked in the timelines with a separate colour (VP2).

5.2 EBTR Variant for Case 2

5.2.1 Phase 1: EBTR Planning

The main goal for case 2 ($VP1_2$) was to assess the usefulness and applicability of the IBIS method compared to currently applied methods; to identify problems, ideas for improvements and good practices. Secondary goals were to learn from the experience of working in a research project compared to industrial projects and to identify new ideas for future projects. Three aspects were selected: ($VP2_2$), namely:

(i) performed activities
(ii) important events
(iii) delivered artefacts (including planned and actual dates)

EBT visualisation was decided to be done with flip chart paper and coloured cards.

5.2.2 Phase 2: EBT Construction

Evidence for each aspect ($VP2_2$) was collected by the two project managers at each company. This evidence was extracted from data collected throughout the project for comparing the IBIS method with existing engineering methods. This activity also acted as preparation for the EBTR meeting ($VP3_2$).The moderator constructed the EBT with the evidence by noting it on cards using a colour scheme to separate between activities, artefacts and events. The cards were arranged along a timeline drawn on two flip chart papers, see Figure 3. Thus, data for all aspects was represented in one EBT though visually separated by colour.

Prior to the EBTR meeting the EBTR method was briefly presented to most participants at a project meeting ($VP3_2$).

5.2.3 Phase 3: EBTR Meeting

The EBTR meeting was organised by the moderator who was also actively involved throughout the IBIS project ($VP6_2$). The meeting was opened by a brief introduction and by everyone sharing their expectations ($VP3_2$). The EBT and the visualisation scheme were then presented, and the topics covered by the focus questions discussed topic by topic. The discussion was structured as follows ($VP5_2$):

(i) the topic was presented
(ii) the participants reflected individually and noted issues on post-its (10-15 min)
(iii) each participant presented their issues and added them to the EBT
(iv) the presented issues were discussed

The meeting was concluded by a discussion on the sum-up questions (see Section 2.3) with the same discussion structure as for the EBTR meeting. The meeting took 4 hours ($VP4_2$).

5.2.4 Phase 4: EBTR Validation

After the EBTR meeting, the outcome of the meeting was consolidated and reviewed by the participants. The final outcome has been published in a project report (Fetzer 2013).

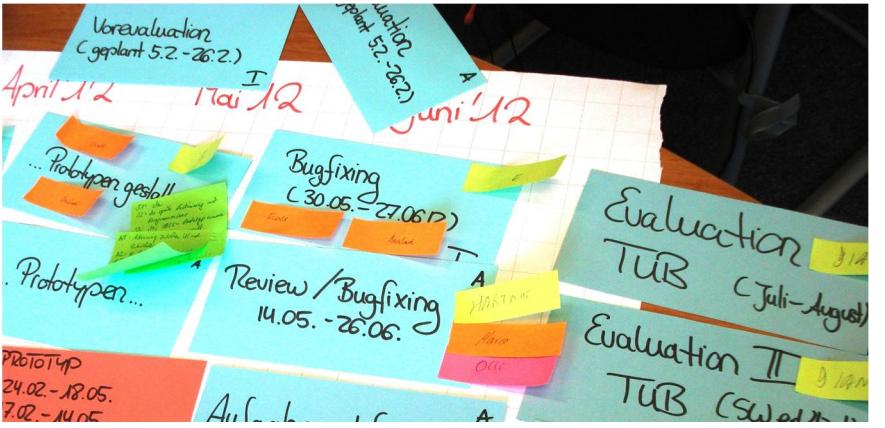


Figure 3. An extract from the EBTs used for case 2.

6 Results

The results of this comparative study are here presented according to the three facets of the research questions RQ1-RQ3 (see Section 1). For each facet, the results are presented per case based on data gathered through focus groups, questionnaire and topic analysis (see Section 4). The results are related to variation points in Section 7.

All retrospective participants were present at the focus groups where they shared experiences of the retrospective including improvements. The 20 questionnaire respondents represent all roles present at EBTR meetings. For case 1, this was product manager, project manager, line manager, architect, developer and tester. Their experience in current roles varies from 3 months to 10 years (4 years for the majority) and in total ranges from 5 to 27 years (evenly distributed over respondents). For case 2, the following roles were represented: project manager, product manager, developer and company CEO. Their experience in current roles varies from 1 to 15 years, and in total 1 to 16 years.

6.1 New Insights and Learning (RQ1)

For case 1, several participants stated at the focus group that they had gained and learnt from the EBTR meeting. One project sponsor said that he now realised that the new company strategy would have had an impact on this project's scoping decisions. One tester gained new insight into the overall process, in particular the early requirements phases and said: 'For me, it is very positive to see the entire picture.' A project manager said that this kind of retrospective could improve and motivate people when starting a new project.

For case 2, the participants stated that they consider EBTR meetings as very useful for reflecting on aspects that went well or could be improved. In this particular case most of the discussed issues were not completely new to the participants due to intensive evaluation activities and frequent discussions at project meetings throughout the IBIS project. However, the EBTR method did support the participants in summarising their experiences. Thus, the participants considered the outcome of the EBTR meeting a very good project result; one that has been delivered to the customer financing the research project.

Comparing the questionnaire responses for new insights and learning from the two cases reveal some interesting differences, see Figure 4. While the participants for case 1 experienced that they gained somewhat to fairly much new insight and learning concerning the big, overall picture (questionnaire 4a), the degree to which participants for case 2 experienced this was not at all to somewhat. For good practices (questionnaire 4d), there is also a higher grading for case 1 than for case 2, while needed improvements (questionnaire 4e) are almost identical. These differences could be explained by the fact that in case 1 the projects are part of a very large organisation while for case 2 the particular development projects are

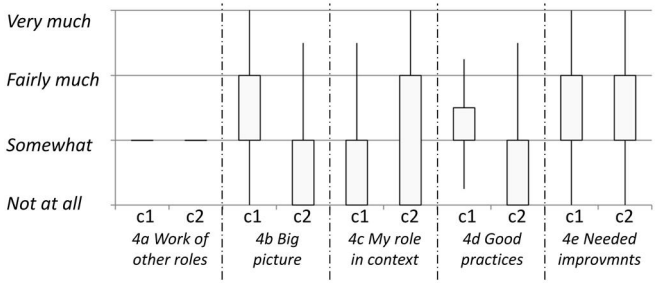


Figure 4. Questionnaire responses for new insights and learning per case (c1 and c2). Boxplots with 25/75 percentiles shown.

more stand-alone and with less 'big picture' to relate to. However, the higher ratings (for 4b and 4d) could be an indication of an effect of a VP, e.g. that the EBTs were more detailed ($VP2_1$) or that the moderator's lack of prior knowledge of the project ($VP6_1$) led to explicitly mentioning more contextual factors and practices as opposed to assuming them to be common knowledge.

6.2 EBT Support for Meeting (RQ2)

For case 1, several participants expressed that compared to experience-based retrospectives the EBTs supported reflection of the entire life cycle. One participant said: 'It would have been harder to discuss the project without the prepared timeline. The graphical presentation makes you think.' A product manager, and some developers and testers appreciated seeing a compilation of the big picture including the phases in which they are not actively involved. Similarly, one participant said that the method supported extending individual perspectives. Furthermore, several participants from different projects said that EBTs support memory recall and that preparing them before the meeting was preferable. One participant said: 'It helps us to remember what happened. It would've been difficult to start talking based on nothing. It's a long time since we did this.'

For case 2, the participants also said that the EBT enabled seeing the big picture and identifying relationships between events. One participant was impressed by being able to see all project activities at a glance and easily become aware of the spent effort and achieved outcome of the project. Thus, the visualisation of project history supported memory recall of certain events and reflection of relevant issues as prompted by the focus questions. Furthermore, the EBT supported the participants in identifying (previously unnoticed) relationships between issues and their consequences through the whole project life cycle. For example, some late scope changes were identified as being caused by a lack of communication at the start of the project. This previously unidentified connection enabled improving the IBIS method to avoid such problems in future.

The participants expressed that at the end of the meeting the EBT was crowded with cards and post-its, making it hard to work with. They proposed preparing the EBT with just flip-chart paper, and use cards/post-it notes during the meeting.

The questionnaire data concerning EBT support for the meeting (see Figure 5) indicates that the EBT variant used for case 1 provided better support for the

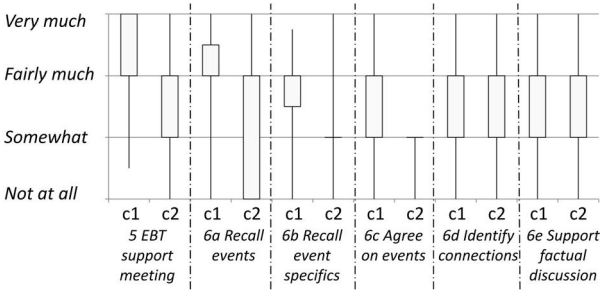


Figure 5. Questionnaire responses for EBT support per case (c1 and c2). Boxplots with 25/75 percentiles shown.

meeting, for memory recall and for agreeing on past events than the one used in case 2. However, the EBTs were perceived to provide the same degree of support in both cases for identifying connections between events and supporting a factual discussion.

6.3 Topic Analysis (RQ3)

Comparison of topics discussed at EBTR meetings for each case (see Section 4.3) reveals that for case 2, all discussed topics could be matched to focus topics. While more topics in total and outside of the focus topics were covered for case 1, see Table 2.

Table 2. Number of focus and non-focus topics discussed at EBTR meetings per case and focus topic area.

	Focus topic area			Non-focus topic area	Sum
	Scope	Communication	Planning		
Case 1	4	4	4	10	22
Case 2	8	2	5	0	15

6.4 Limitations

Limitations are presented here according to Runeson et al (2012).

Construct validity regards how well the research method correlates to the targets research questions. A combination of focus group and questionnaire was used to mitigate the risk of misinterpreting the participants' experience of the EBTR method. Variation points (VPs) were iteratively defined rather than planned from the start. In combination, with varying multiple VPs it is not possible to ensure which variation point causes which effect. However, potential dependencies between VPs have been considered.

Reliability concerning the independence of data and analysis from specific researchers, the risk of researcher bias was addressed with triangulation of meeting notes and cross-analysis of data among the authors. The results were reviewed by researchers not involved in the data collection.

Internal validity concerns whether causal conclusions are warranted or if there are overlooked phenomena. The difference in evidence collection for the EBTs poses a risk. Two EBTR participants collected evidence in case 2, and may have introduced a bias in the EBT, thus limiting the retrospective discussions. Furthermore, intensive evaluation for case 2 throughout the project may have an influence on the results. Since several issues had already been discussed in project meetings some insights were not new.

External validity concerns the ability to generalise and transfer findings to other cases. Our aim is not to draw statistically valid conclusions outside the two cases, rather to understand and describe variability aspects in relation to their contexts. Results transferability needs to be assessed by comparing our cases with other cases. To support this we have characterised the cases and the projects.

7 Discussion

The outcome of applying the EBTR variants to the two cases is compared in this section and the potential effect of each VP (see Section 5) is discussed. A summary of our interpretation of the results is also shown in Figure 6.

7.1 VP1: Retrospective Goals

The width, or focus, of the EBTR goal varied between the two cases, which influenced several other variation points and seems to have affected the outcome. Case 1 had a wide EBTR goal of assessing the RE aspects of their agile development process. While for case 2, the goal was to assess the IBIS project regarding communication, workload between different roles and lessons learnt regarding the IBIS method. The observed differences in amount and focus of discussed topics (see Section 6.3) and extent of new insight into the larger context (see questionnaire 4a, Section 6.1) correspond to the width of the EBTR goal. However, due to the influence that the EBTR goal had on the design of other VPs, we believe that VP1 only has an indirect effect on these factors.

VP1 affected the design of the EBTs (VP2) and the selection of discussion structure (VP5). The aspects and types of evidence for the EBTs were selected in line with the EBTR goal, i.e. for the wider goal of case 1 more aspects and evidence types were selected, while for the more focused goal of case 2 less aspects and evidence types were selected. In addition, the discussion structure was selected to match the goal width, with a more structured discussion for the more focused goal of case 2.

7.2 VP2: EBT Content and Visualisation

The EBTs used in the two cases varied in the amount of aspects and evidence that were included, and in the applied visualisation technique. For case 1, five different aspects were used and evidence extracted from four different systems by the moderator. While for case 2, three aspects were selected and all evidence was selected from two systems by two participants. This resulted in a larger set of data

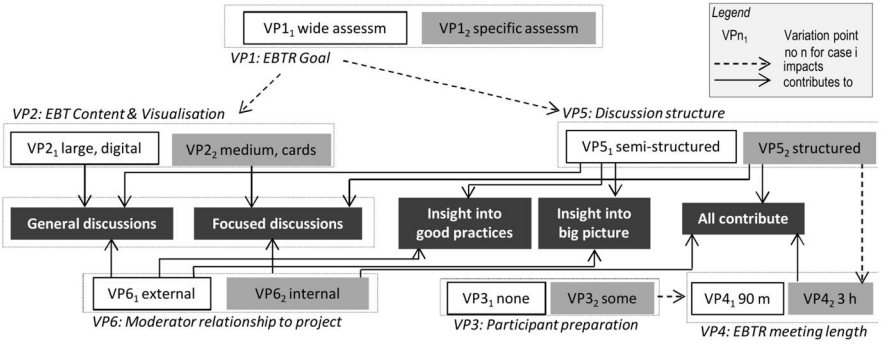


Figure 6. Summary of identified connections between variation points (VP) and effects (black boxes).

for case 1 than for case 2. For case 1, the large amounts of data were managed by visualising the evidence in several EBTs using a digitalised format. For case 2, the time stamped data was visualised in one EBT using physical cards. In both cases, the moderators were responsible for visualising the collected evidence in the EBTs.

The range and amount of evidence in the EBTs correspond well to the amount and range of topics discussed during the EBTR meetings (see Section 6.3). Thus indicating that larger and wider sets of data visualised in a clear digitalized way (as for case 1) can lead to discussing a broader range of topics. In contrast, selecting a more focused and limited set of evidence, and visualising this in a simpler way (as for case 2) can result in focusing the EBTR discussions on more specific topics (see Section 6.3).

Furthermore, for case 1 the significantly higher degree of participant insight into the bigger picture (see questionnaire 4b in Section 6.1) may also be partly explained by the detailed EBTs used for this case. It is possible that they provide a richer picture of a wider range of events, not limited to current insight.

7.3 VP3: EBTR Meeting Participant Preparations

The degree of participant preparation was different for the two cases. In case 1, the participants were consciously not prepared due to a goal to design the EBTR method so as to require minimum development resource effort. Instead, the EBTR method was introduced at the beginning of EBTR meeting. For case 2, the EBTR method was introduced to most of the participants at a project meeting. Thereafter the participants agreed to apply the method. Furthermore, for case 2 two participants were also prepared by being involved in the evidence collection.

Participant preparation (i.e. case 2) was expected to enhance the EBTR meeting by strengthening the degree of new insights and the amount of support provided by the EBTs to the meeting. However, the degree of new insights gained from EBTR (questionnaire #4, see Section 6.1) is either similar for both cases, or higher for case 1. Furthermore, the degree of EBT support for EBTR meeting was seen as much higher for case 1 rather than case 2 (questionnaire 5, see Section 6.2).

This lack of observable effect may be explained by the large difference in meeting time between the cases. Even if the length of the EBTR meetings could have been reduced (or avoided running over time) by a short preparation, the total meeting time is most likely similar. The decision to prepare participants beforehand or not needs to be made case by case depending on the specific situation. For example, for case 1 a 10-minute presentation of the EBT visualisation could have enabled a quicker start of the actual discussions at the EBTR meeting.

7.4 VP4: EBTR Meeting Length

There was a large variation in the length of the EBTR meetings. For case 1, the meetings were booked for 75 minutes; 2 of 3 meetings ran out of time. For case 2, the meeting was planned for 3 hours, but took approx. 4 hours.

There are no directly observable effects of the differences in meeting length. A longer meeting time could be expected to result in a higher degree of new insights and learning. But, this is not the case. Rather, the results indicate more new insight of the bigger picture and for good practices for case 1, and the same degree of

insight for improvements (see questionnaire 4, see Section 6.1). This could partly be explained by the participants in case 2 having a high degree of pre-insight into good practices due to continuous assessments throughout the project.

A positive effect of a longer EBTR meeting is that it allows for more time for discussions and could (if managed correctly) facilitate all participants having a fair opportunity to share and discuss their views. More meeting time also supports selecting a more structure discussion format (this was one of the reasons for not selecting this for case 1).

7.5 VP5: Discussion Structure at EBTR Meeting

Two variations in discussion structure have been evaluated and found to have an effect on the outcome of the EBTR method. For case 1, a semi-structured discussion was moderated, based on the focus questions but not limited to those topics (see Section 0). For case 2, the discussions were more strictly structured according to the focus questions and the participants were given time to individually reflect on each topic before sharing and discussing their views (see Section 5.2.3).

The topic analysis shows that for case 1, a larger number and wider range of issues and topics were discussed at the analysed EBTR meeting. In contrast, the majority of the topics discussed for case 2 can be connected to the focus questions (see Section 6.3). Thus, selecting a structured discussion format may lead to more focused discussions and thus more specific findings. A semi-structured format may support exploring a wider area and be suitable for investigating causes and connections between topics.

Furthermore, the higher rating of new insight into the bigger picture and good practices for case 1 (questionnaire 4b and 4d, see Section 6.1), could potentially be partly attributed to the wider discussions resulting from the open discussion format.

Finally, the structured format used in case 2 encouraged all participants to equally share and discuss their views.

7.6 VP6: EBTR Meeting Moderator

Two different variations concerning the moderator's relationship to the project were evaluated. In both cases, the moderators had good knowledge and insight into the general domain of the projects and previous experience of moderating group meetings. However, for case 1 the moderator had no previous relationship with the projects to which the EBTR was applied. For case 2, the moderator was an active project member.

The more focused set of topics discussed in case 2 (see topic analysis, Section 6.3) may be partly supported by the moderator's existing relationship with the project. This enabled the moderator to support the discussions in identifying potential improvements, though this factor was rated at the same level for case 1 and for case 2 (see questionnaire 4e, Section 6.1). Furthermore, it was easier for the moderator for case 2 to ensure that everyone was included in sharing and discussing their views, which was harder in case 1 since the moderator did not know the name of all the participants.

In contrast, the wider set of topics discussed in case 1 (see Section VI.C) and the higher degree of new insights and good practices among the participants (questionnaire 4b and 4d, see Section VI.A) may be partly supported by the moderator having no previous relationship to the project. The specific project was, thus new to the moderator. This may have led to the moderator asking and the participants sharing relevant information, which would otherwise have been assumed to be common knowledge and not mentioned.

8 Conclusions and Future Work

Project retrospectives can be an effective way for an organisation to assess and continuously improve their development processes. By project members meeting to reflect on project history after project completion, new insights can be gained into good practices, problems and needed improvements. Team reflections can be further supported by injecting facts (evidence) into the retrospective meeting in the form of a pre-constructed visual timeline. A previous evaluation of our evidence-based retrospective (EBTR) method showed that this supports memory recall and factual discussions, and thereby enhancing group reflections around project events.

However, for this to be an effective process improvement tool rather than merely a group bonding activity, the project retrospectives need to be targeted towards strategic goals. Furthermore, the retrospective meetings need to cover topics relevant to those goals.

This comparative study has identified and evaluated a six variation points of the EBTR method and their potential effect. The specificity of the EBTR goal is found to impact the retrospective outcome. By setting the variation points, the method can be customised either towards assessing a specific process area or topic, or towards a broader assessment of a process and its influence on surrounding processes and roles.

Future work includes further evaluations of the EBTR method for other cases and other combinations of VPs. In particular, evaluation of different timeline visualisation techniques is an interesting avenue to explore.

References

Adam S, Doerr J, Eisenbarth M (2009) Lessons Learned from Best Practice-Oriented Process Improvement in Requirements Engineering – A Glance Into Current Industrial RE Application, REET09.

Birkholzer T, Dickmann C, Vaupel J (2011) A Framework for Systematic Evaluation of Process Improvement Priorities. 37th Euromicro Conference on Softw. Eng. and Adv. Appl. (SEAA), pp. 294-301.

Bjarnason E, Regnell B (2012a) Evidence-Based Timelines for Agile Project Retrospectives – A Method Proposal. Proc. Agile Processes in Software Engineering and Extreme Programming (XP 2012), May 2012, pp. 177-184.

Bjarnason E, Berntsson Svensson R, Regnell B (2012b) Evidence-Based Timelines for Project Retrospectives—A Method for Assessing Requirements Engineering in Context. IEEE 2nd Int. Workshop on Empirical Requirements Engineering (EmpiRE), pp. 17-24.

Bjarnason E (2013) Research study material on evidence-based timeline retrospective method on-line at (last accessed 2013-03-20): http://serg.cs.lth.se/research/experiment_packages/ebtretro/

Cabral Silva Filho R, Cavalcanti da Rocha AR (2010) Towards an Approach to Support Software Process Simulation in Small and Medium Enterprises. 36th Euromicro Conference on Softw. Eng. and Adv. Appl. (SEAA), pp.297-305.

Collier B, DeMarco T, Fearey P (1996) A Defined Process for Project Postmortem Review. IEEE Software, vol. 13, issue 4, pp. 65-72.

Derby E, Larsen D (2006) Agile Retrospectives: Making Good Teams Great! Pragmatic Bookshelf.

Dikici A, Turetken O, Demirors O (2012) A Case Study on Measuring Process Quality: Lessons Learned. 38th Euromicro SEAA'12, pp.294-297.

Drury M, Conboy K, Power K (2011) Decision Making in Agile Development: A Focus Group Study of Decisions and Obstacles. Agile Conference, pp. 39-47.

Emami MS, Binti IthninN, Ibrahim O (2010) Software Process Engineering: Strengths, Weaknesses, Opportunities and Threats. 6th Int. Conference on Networked Computing (INC), pp.1-5.

Fetzer K, Hess A, Lange K et al. (2013) Weber Schlussbericht Gestaltung intuitiver Benutzung mit Image Schemata (Final report) (www.ibis-projekt.de)

Hurtienne J, Weber K, Blessing L (2008) Prior Experience and Intuitive Use: Image Schemas in User Centred Design. Langdon, Clarkson, Robinson (Eds.), Designing Inclusive Futures. Springer.

Jørgensen M, Sjøberg D (2000) The Importance of NOT Learning from Experience. Proc. Of European Softw. Process Improvement, EuroSPI'2000, pp. 2.2-2.8

Kerth N (2001) Project Retrospectives. A Handbook for Team Reviews. Dorset House.

Lacheiner H, Ramler R (2011) Application Lifecycle Management as Infrastructure for Software Process Improvement and Evolution: Experience and Insights from Industry. 37th Euromicro Conf. on Softw. Eng. and Adv. Appl. pp. 286-293.

Robson C (2002) Real World Research. 2nd ed. Blackwell Publishing.

Runeson P, Höst M, Rainer A, Regnell B (2012) Case Study Research in Software Engineering: Guidelines and Examples, Wiley.

PAPER VI

GAP FINDER: ASSESSING AND IMPROVING THE INTEGRATION OF REQUIREMENTS AND TESTING[1]

A closer integration of requirements engineering and testing (RET) can strengthen the coordination and alignment within a software project and enable the discovery of issues and misunderstandings earlier, rather than later. This integration of various activities can be achieved by applying alignment practices. However, each organisation and project is different and there is no one-fits-all set of practices. Rather, for each organisation and project the processes need to be tailored and improved to match the current targets and challenges.

We propose a process improvement method called Gap Finder that can support project teams in addressing weak RET alignment. The method can detect problematic gaps between people and between artefacts, and then identify practices that can decrease or bridge these gaps. For example, cognitive gaps in domain knowledge between requirements and testing roles may be bridged by user testing or by cross-reviews of test cases and requirements. The Gap Finder method is based on a theoretical framework of the impact of RET alignment practices on different types of distance. A formative evaluation of this method was performed through a case study in which Gap Finder was applied to an on-going development project. A qualitative and mixed-method approach was taken in the evaluation including ethnographically-informed observations.

The results show that Gap Finder can be used to detect gaps causing misalignment and to identity suitable practices for mitigating these. This demonstrates the feasibility of the approach to consider and measure distances as an underlying factor of alignment. Furthermore, the visualisation of these distances was found to enable a constructive group discussion around gaps and support the project team in identifying new improvement areas.

The insights on the impact of gaps on RET alignment reported in this paper can provide practitioners with an increased awareness of these factors and their potential impact on the development process. Furthermore, Gap Finder provides a stepping stone for further research into RE distances and RET alignment.

[1] By Elizabeth Bjarnason, Helen Sharp and Björn Regnell. To be submitted.

1 Introduction

Repeatedly developing software that meets the demands of the market and of the customer concerning both functionality and quality requires a well-functioning development organisation. Coordination of the different roles and activities of the organisation can enhance their alignment towards common goals. Coordination is supported by suitable software processes and practice, but also relies heavily on softer aspects of interaction and communication between individuals and teams. Software artefacts may also support this communication in which case the structure and quality of those artefacts also affect the coordination within a project.

Software testing requires a clear understanding of the expected behaviour in order to validate that we are 'building the right product' (Boehm 1981) and to verify that we are 'building the product right' (Boehm 1981). This understanding can be provided by requirements engineering (RE) activities and a clear communication of the requirements (Damian 2005, 2006, Bjarnason 2011). However, when the requirements are unclear and ambiguous this can lead to an increased frequency of test failures (Ferguson 2006). Furthermore, weak alignment and coordination of the RE activities and roles with those of software testing may lead to serious implications both for development projects and for the resulting software products. Examples of this include increased development lead time, delayed deliveries, and problems with software functionality and quality (Damian 2005, 2006, Uusitalo 2008, Bjarnason 2013b).

Defining a process that will support the necessary coordination and alignment between RE and testing is non-trivial. Apart from the fact that each organisation and project is different and has different targets, how well a process is applied depends on how individual engineers function together. While there is a plethora of methods, frameworks and practices for improving on software processes (including CMMI, SPICE etc.) methods and techniques for assessing and improving the softer aspects of software development are scarce.

We propose a method called *Gap Finder* for assessing and improving the alignment between requirements and testing by measuring underlying factors in the form of distances between people and between artefacts. The method is designed to pinpoint gaps that negatively affect alignment and propose relevant practices for mitigating these gaps. Alignment can then be improved by applying these practices.

The presented contribution of the Gap Finder adds to our previous empirical work into challenges and practices for RET[2] (RE and testing) alignment (Bjarnason 2013b). Furthermore, the proposed method is based on a previously reported framework of RE distances (Bjarnason 2013a). The results of our previous research indicate that a closer integration of requirements and testing along one or more dimensions may support increased coordination and alignment. For example, that shorter geographical and cognitive distance between the involved roles may decrease communication gaps and thus avoid testing based on an incorrect understanding of requirements.

In this paper, we report on a formative evaluation of the initial version of the Gap Finder with the aim of assessing how the method supports project teams in

[2] In this paper the term *RET alignment* is used to mean the same as the term *REVV (RE and Verification & Validation) alignment* used in our previously published work on this topic.

improving the alignment between requirements and testing. The study addressed the following four research questions, for the area of integration of requirements and testing:

RQ1 How relevant is the set of distances included in Gap Finder to this area?
RQ2 How relevant are the practices identified by the Gap Finder method?
RQ3 How does the Gap Finder approach of assessing and visualising distances stimulate reflections of this area within a project team?
RQ4 What improvements can be made to the Gap Finder method?

The method was applied to an on-going development project and evaluated through observations, interviews and a survey. Ethnographically-informed observations (Robinson 2007) were undertaken in the evaluation study with a focus on understanding how well the method captures relevant issues and identifies suitable practices.

The rest of this paper is structured as follows: Section 2 describes the research underpinning the Gap Finder method, while Section 3 describes related work. In Section 4, the Gap Finder method is presented. The case in which the method was evaluated is presented in Section 5, while the research method used in the evaluation study is described in Section Table 4. The results of the evaluation are reported in Section 7 and discussed in Section 8 including potential improvements. Finally, we conclude by summarising and describing future work in Section 9.

2 Background and Underpinning Research

The Gap Finder method stems from our previous research on RET alignment and RE distances, and rests on the findings of that work, see Figure 1. Challenges and practices of RET alignment were identified through an interview study (Bjarnason 2013b). While working with that study the concept of various kinds of distance between requirements and testing affecting RET alignment was conceived. This concept was explored through a systematic mapping study (Bjarnason 2013a) of the use of the term *distance* between RE and later software development activities. The RET interview material (from a previous study) was then reanalysed using this framework of RE distances to identify relationships between individual RE distances and RET practices. The outcome of this re-analysis is a theoretical

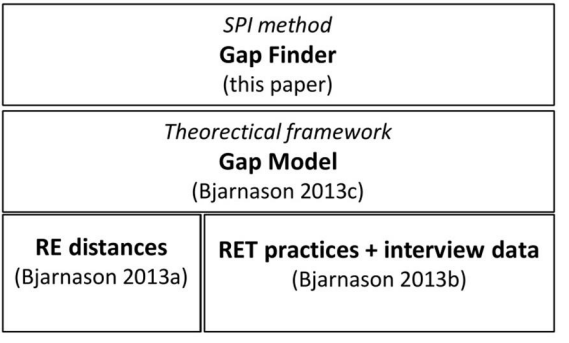


Figure 1. Overview of the building blocks underpinning the Gap Finder method.

framework called the *Gap Model* (Bjarnason 2013c, Chapter 1). Gap Finder was designed as a SPI (software process improvement) method for improved RET alignment by utilising the knowledge of the Gap Model concerning RE distances and which RET practices that are affected by them. Each of the underpinning building blocks including the design of Gap Finder are described below.

2.1 Practices for RET Alignment

In a previous case study into RET alignment a set of RET alignment practices used, or suggested for use, by industry were identified (Bjarnason 2013b). These results were based on 30 semi-structured interviews of 90 minutes each with practitioners from six different software companies, comprising a wide range of people with experience from various roles relating to RE and testing.

Four main factors were identified as affecting RET alignment. Firstly, softer aspects such as communication and coordination within a development organisation is one of the major challenges in achieving good alignment with testing since requirements are a crucial starting point for testing. Secondly, the quality of the requirements engineering effort is critical to the alignment of the testing activities aimed at validating and verifying the requirements. Thirdly, size is a key variation factor of alignment and improvement practices need to be selected and tailored to suit the specific company including its size and domain. Fourthly and finally, the motivation required for applying alignment practices varies. For safety-critical development this motivation is driven by external enforcement, while internal motivation is required for non-safety critical development. Internal motivation in turn relies on insight into the balance between cost and benefits of applying RET alignment practices.

Ten categories of alignment practices were identified, namely (P1) *RE practices*, (P2) *validation practices*, (P3) *verification practices*, (P4) *change management practices*, (P5) *process enforcement*, (P6) *tracing practices*, (P7) *traceability responsible role*, (P8) *tools*, (P9) *alignment metrics*, and (P10) *job rotation*. In total, the results from this RET alignment study include 27 individual practices ranging from high-level practices such as P5 *Process enforcement* to specific practices such as P2.4 *Management base launch decision on test repor*t.

2.2 Requirements Engineering (RE) Distances

Different types of distance between RE and later development activities were investigated through a systematic mapping study (Bjarnason 2013a). The full map contains 53 peer-reviewed papers and 13 different RE distances were found. These were categorised as being between people, between artefacts, or between artefacts and reality. Eight of the distances were between people: *geographical*, *temporal*, *socio-cultural*, *cognitive*, *psychological*, *opinion*, *power* and *organisational*. Four distances were found to be between artefacts, namely *semantic*, *similarity*, *syntactic* and *impact*. While finally, one distance was found to concern *adherence* between an artefact and reality.

In general, long distance between people has been found to have a negative effect on communication and collaboration within projects. However, there are unexplained contradictory findings that indicate decreased delays in communication

over geographical distance in certain contexts (Wolf 2008). These may be explained by the effect that different practices have on the characteristics of the communication channel that is used, i.e. formal or informal, synchronous or asynchronous.

Although there is less research into distances between artefacts it is an interesting area for future RE research. It has been suggested as applicable in the context of requirements change and traceability. For example, that the distance between the previous and a changed version of an artefact can be used to assess the impact of the change (Jilani 2001, Briand 2006). Distance between RE artefacts and artefacts of later development activities, e.g. design and testing, could potentially be used to measure coverage and consistency between requirements specifications and other artefacts such as design and test specifications, and source code.

2.3 The Gap Model: A Framework of Distances

The Gap Model is a theoretical framework of distances relevant to RET alignment that provides a knowledge base of relationships between RET practices and RE distances. The model was constructed by analysing empirical data against the two theoretical frameworks briefly described in the previous sections (Sections 2.1 and 2.2), the one for RE distances (Bjarnason 2013a) and the one for challenges and practices of RET alignment (Bjarnason 2013b). The new framework was derived from the empirical data gathered in the case study on RET alignment. This data was re-analysed to identify which distances were perceived to be affected by each RET alignment practice. Details on the design and content of the Gap Model are reported by Bjarnason (2013c, Chapter 1, Section 4).

Currently eight RE distances are included in the Gap Model: (D1) *geographical*, (D2) *organisational*, (D3) *psychological* and (D4) *cognitive* distances between people; (D5) *adherence* distances to artefacts, (D6) *semantic* and (D7) *navigational* distances between artefacts, and (D8) *Temporal* distance between activities. An overview of these distances is shown in Table 1.

The Gap Model contains seven categories of RET practices, namely RE practices (P1), Validation practices (P2), Verification practices (P3), Change management practices (P4), Tracing practices (P6[3]), Tool practices (P8) and Development process (P11). In total the model contains 32 individual RET practices, see (Bjarnason 2013c, Chapter 1, Table 2) for the full list of practices.

The impact of each RET practice on one or more RE distances is included in the model and categorised as either decreasing or increasing the distance, or bridging it, i.e. the practice reduces the negative effect of it without directly affecting the distance. For example, applying the practice of *Cross-role requirements reviews* (an RE practice) can bridge an organisational distance between requirements engineers and testers without changing the organisational structure and thus the organisational distance. This cross-role review practice can also decrease adherence distance between an agreed and documented set of requirements by identifying inconsistencies at the review and by updating the requirements documentation accordingly. An overview of the Gap Model and included practice-distance connections can be found in (Bjarnason 2013c, Chapter 1, Table 3).

[3] Same numbering of practices as in the RET alignment study, see Section 2.1.

Table 1. The RE distances included in the Gap Model and in the Gap Finder.

	Type of distance	*Between*
PEOPLE	**D1 Geographical distance** Physical distance of desks	Roles related to requirements and testing
	D2 Organisational distance Distance between organisational units	
	D3 Psychological distance Perceived effort to communicate	
	D4 Cognitive distance Difference in knowledge	
ARTE-FACTS	**D5 Adherence distance** Difference betw. documented content and perception of agreement or reality	Artefact and reality
	D6 Semantic distance Difference in meaning	Artefacts
	D7 Navigational distance Effort to navigate between	
ACTIVI-TIES	**D8 Temporal distance** Time between activities, e.g. specifying and using a requirements specification	Activities

2.4 Design of the Gap Finder Method

The Gap Finder was designed to assess and improve on RE distances related to the alignment of requirements and testing based on the Gap Model (see Section 2.3 and Bjarnason 2013c, Chapter 1). Around this theoretical core, a measurement instrument was designed for assessing the RE distances for a development project (see further details below). A process for preparing, analysing and presenting the outcome of these measurements has also been designed and includes the following steps: preparations, measuring, gap analysis and gap workshop. The outcome of this design, i.e. the Gap Finder method, is presented in Section 4.

2.4.1 Design of the Measurement Instrument

A measurement instrument for estimating the RE distances covered by the Gap Model was created. The entities between which the distances were to be measured were specified, and various aspects to measure for each distance were identified. For example, the following aspects of semantic distance were define, namely similarity, abstraction level and coverage between two related artefacts. Furthermore, measurements and relevant scales were also designed for each identified aspect of a distance. The design was guided by existing empirical data and related research findings.

In this initial Gap Finder version the measurement instrument consists of surveys and physical measurements. For practical reasons (timing and variations in targeted participants), the survey questions were split into three separate surveys, namely profile, communication and artefact survey. Templates for these surveys were created and are available on-line, see (Bjarnason 2013d).

2.4.2 Design of the Analysis Step

An explorative approach was taken in the design of the analysis step (the gap analysis, see Section 4.1.3). Based on a set of obtained measurements various calculations of total distance were investigated. For example, average, minimum, maximum, sum of pair-wise distances between the data points, and Cartesian difference for multi-dimensional data points. Similarly, different ways of visualising distances were explored including radar diagrams, plotting of data points, and various graph representations.

When a gap has been recognised in the data, this can be compared against the information in the Gap Model. Thereby RET practices found to address this distance type can be derived based on the knowledge obtained from previous studied (Bjarnason 2013b). This information is represented in the Gap Model.

2.4.3 Design of Discussion with Development Team

One of the main aims of discussing the Gap Finder outcome with the development team at the gap workshop (see Section 4.1.4) is to obtain a consensus and commitment to an agreed set of improvement practices. This is a vital factor in effectively implementing these changes. In addition, the session is intended to facilitate group reflection on the outcome of the assessment and thereby provide validation of the Gap Finder output. Therefore, this discussion session was designed as a focus group (Robson 2002) involving all team members of the assessed project. The intention of this design choice was to support an open discussion of the obtained distance measurements and the identified improvement practices.

3 Related Work

3.1 Aligning Requirements and Testing

There is a limited amount of research into the alignment of requirements and testing, rather most research tends to focus on one area (Barmi 2011). Of the research published in the area of RET alignment Barmi et al. (2011) found that most studies were on model-based testing including a range of variants of formal methods for describing requirements with models or languages from which test case are then generated. Barmi et al. also identified traceability and empirical studies into alignment challenges and practices as main areas of research. Only 3 empirical studies specifically focusing on RET alignment were found. Of these, 2 originate from the same research group (namely Kukkanen 2009 and Uusitalo 2008) and the third one is from our previous RET alignment study (Sabaliaskaute 2010, Bjarnason 2013b). Barmi et al. draw the conclusion that although the areas of model-based engineering and traceability are well understood, there is a need for practical approaches and methods for implementing these.

Related empirical studies of RET alignment consist of a case study into jointly improving the RE and testing processes by Kukkanen et al. (2009) and an interview study investigating alignment practices in industry by Uusitalo et al. (2008). Kukkanen et al. (2009) found that alignment can be improved by integrating the

requirements and testing processes, including clearly defining RE and testing roles for the integrated process. The most important aspect in achieving alignment was found to be ensuring that 'the right information is communicated to the right persons'. This aspect was supported by connecting the processes and the people from both areas in combination with applying good practices that support this connection. The practices implemented to support RET alignment were: the use of metrics, traceability with tool support, change management process and reviews of requirements, test cases and traces between them (Kukkanen 2009). Similar and additional alignment practices are reported by Uusitalo et al. (2008) based on six interviews with mainly test roles from six different companies. Their results include a number of practices that increase the communication and interaction between requirements and testing roles, namely early tester participation, traceability policies, considering feature requests from testers, and linking test and requirements people (Uusitalo 2008). Linking people or artefacts were seen as equally important by the interviewees who were unwilling to select one over the other. Most of the practices reported by Uusitalo et al. were also identified in our RET alignment study (Bjarnason 2013b) with the exception of the specific practice of linking testers to requirements owners and the practice of including internal testing requirements in the project scope.

Traceability is a long-standing topic that has been research since the beginning of software engineering, i.e. the 1960s (Randell 1969). However, despite traceability being an (acknowledged) important aspect in high quality development (Watkins 1994, Ramesh 1997) and therefore important to software verification, the implementation of this practice still remains elusive and a challenge for most companies (Gotel 1994, Watkins 1994, Jarke 1998, Ramesh 1998). Traceability between requirement artefacts and other development artefacts has a number of benefits and can support impact analysis (Gotel 1994, Watkins 1994, Ramesh 1997, Damian 2005, Uusitalo 2008, Kukkanen 2009), lower the cost of testing and maintenance (Watkins 1994, Kukkanen 2009), and increase the test coverage (Watkins 1994, Uusitalo 2008) and thereby the quality of products (Watkins 1994, Ramesh 1997). However, a wide range of challenges connected to traceability have also been reported, e.g. by Cleland-Huang (2003). These challenges include artefact volatility, informal processes, lack of clear responsibilities for tracing, communication gaps, insufficient time and resources for maintaining traces, low insight into cost-benefit of tracing, and a lack of training (Cleland-Huang 2003). It has been suggested that the cost of establishing and maintaining traces can be reduced by automatic or semi-automatic recovery of traces (De Lucia 2007, Hayes 2007, Lormans 2008) or by tracing at a higher abstraction level such as at the user scenario level thereby reducing the number of traces (Post 2009). In the context of our work, it is interesting to note that the traceability gurus Gotel and Finkelstein (1994) express that a particular concern in improving requirements traceability is the need to facilitate informal communication with those responsible for specifying and detailing requirements.

Model-based testing is a large research field within which a wide range of formal models and languages for representing requirements have been suggested (Dias Neto 2007). Similarly to the field of traceability, model-based testing also has issues with practical applicability in industrial development (Nebut 2006, Mohagheghi 2008, Yue 2011). Two exceptions to this are provided by Hasling et al.

(2008) and by Nebut et al. (2006) who both report on experiences from applying model-based testing by generating system test cases from UML descriptions of the requirements. The main benefits of model-based testing are in increased test coverage (Nebut 2006, Hasling 2008), enforcing a clear and unambiguous definition of the requirements (Hasling 2008) and increased testing productivity (Grieskamp 2011). However, the formal representation of requirements often results in difficulties both in requiring special competence to produce (Nebut 2006), but also for non-specialist people, e.g. business roles, in understanding these models of requirements (Lubars 1993). In addition, the risk of errors in the models needs to be considered when applying this approach (Hasling 2008). An alternative to formal models is scenario-based models, which has been proposed by Regnell and Runeson (1998), Regnell et al. (2000) and Melnik et al. (2006). Test cases are then defined to cover requirements defined at a high level of abstraction as use cases, user stories or user scenarios. The details are then defined as test cases and used to document the detailed requirements. This is an approach often applied in agile development (Ramesh 2010). Melnik et al. (2006) found that using executable acceptance test cases as detailed requirements is straight-forward to implement and breeds a testing mentality. Similar positive experiences with defining requirements as scenarios and acceptance test cases are reported by Martin et al. (2008)

3.2 Software Process Improvement

The field of software process improvement (SPI) was established in the 1980s by the pioneers Watts Humphrey (1989, 1997) and Victor Basili (1988). The capability maturity model CMM (now CMMI, Chrissis 2007) was developed by Humphrey (1989) in the 1980s and is now a widely used framework for process improvement. SPI is rooted in the perspective that 'the software process is the set of tools, methods, and practices we use to produce a software product' (Humphrey, 1989, p.3) The process is then an important instrument in developing and maintaining quality software products in an efficient, reliable and repeatable way. In addition, SPI is based on quality management and organisational learning, and emphasises the concept of goal-oriented measurements (GQM, Basili 1992), as identified by Dybå based on synthesis from a literature review (Dybå 2000). Furthermore, CMMI emphasises that the process is the mechanism that integrates and synchronises the people, the work procedures and the tools involved (Chrissis 2007).

There is a wide range of SPI frameworks, which in general share the same main steps of first evaluating (or assessing) the current process, and then identifying, implementing and evaluating suitable process improvements. These frameworks may be categorised into two main approaches: inductive and prescriptive (Briand 1995). Inductive (or bottom-up) frameworks, such as QIP (Basili 1985) and Lean Six Sigma (George 2002), take their stance in the organisational situation and context of the organisation when identifying potential improvements. In contrast, prescriptive (or top-down) frameworks, such as CMMI (Chrissis 2007) and SPICE, i.e. ISO/IEC 15504 (ISO/IEC 2004-2011), mainly base their improvement suggestions on a wide set of best practices. The degree to which organisation-specific goals and underlying conditions are catered for thus varies depending on approach. Furthermore, rather than merely applying a prescribed practice based on previous experience an inductive framework would consider and analyse

underlying factors of observed issues and suggest improvements that address identified root causes. Thus, prescriptive methods do not consider the potential effect of contextual factors, but rather treat all software organisations as equal independent of size, average project lead time, targeted product levels, domain etc.

Within SPI in general, processes are prioritized rather than people (in contrast to the approach taken in agile development). Although people are acknowledge as important it is primarily from the perspective of having the competence required for their roles and the activities they are to perform as prescribed by the software process (Humphrey 1989) rather than how the software process can be tailored to match the individuals working within a software development project or organisation. There is a parallel framework to CMMI for personal software processes (PSP, Humphrey 1997) that focuses on practices for the individual including some collaboration practices, e.g. code review practice, that are reported to decrease defect rates with no cost to developer productivity (O'Beirne 1997).

Finally, techniques such as retrospective reflection, information flow analysis, process modelling and process simulation are also used for assessing and identifying software process improvements.

Retrospective reflection is used to consider and analyse past events and experiences, to identify problems and potential improvements (Collier 1996, Derby 2006, Drury 2011). Iteration retrospectives are a common practice within agile development and they are strongly connected to the concept of self-governing teams. The retrospectives are then the forum for discussing and agreeing on process improvements within the project team (Drury 2011). In traditional development, project retrospectives (also called lessons learnt or project post-mortems) are more common and range from semi-structured meetings were project members discuss the past project to more structured meetings either with or without prepared input concerning project events.

Analysis of information flows and identification of bottlenecks are both based on the idea that software development relies on the transformation of information, and thus aims at ensuring an effective and efficient flow of this information through the project. For example, the resolution time for modification requests has been found to be reduced when the communication patterns between engineers are well-matched to technical dependencies between their work (Cataldo 2008). A similar approach in planning and managing information flows is applied in the FLOW Mapping method (Stapel 2011). The method suggests an improved flow by capturing the information needs of a project and developing a communication strategy based on these needs.

Process modelling (Yu 1994) and *process simulation* (Kellner 1999) are two connected areas, although modelling can be used for SPI without simulating that model. A model of an existing or an improved process can facilitate group communication and understanding of that process, and thereby support improving and managing this process. Furthermore, modelling can enable the implementation of process guidance and steering in the tool environment, thereby enforcing the process prescribed by the model.

3.3 RE and RET Alignment in the Context of SPI

Several studies show that RE is a challenging, but important, area to address with SPI frameworks. This is most likely due to the serious implications that unclear and changing requirements can have on later development activities. Through a systematic literature review Lavallée and Robillard (2012) found that SPI leads to an improved quality of documentation although at an increased cost, but also report on mixed findings concerning the impact on requirements issues in general. For example, there are studies that report that SPI had no impact on requirements problems (Chen and Huang 2009), but also studies that found that SPI can improve the quality of the requirements and then also improve the overall quality of the software product (Kandt 2009). However, Harter et al. (2012) found that there was a significant decrease in the rate of severe defects at higher CMM levels, but that this effect was not present for projects with a high degree of requirements ambiguity. They conclude that 'investments in requirements clarification and process improvement [á la CMM] are not substitutes for each other, but instead tend to be complementary' (Harter 2012). This strong (negative) impact of weak RE correlates well with findings reported by Hall et al. (2002) that requirements issues cause the most software process issues, including changes to user requirements and delivering erroneous software to customers. Furthermore, softer issues including lack of skill and staff circulation were found to cause more requirements-related problems in development than technical issues (Hall 2002).

The current version of CMMI (Chrissis 2007) does include process areas for RE, validation and verification, and practitioners are reported to agree that documentation of requirements is a key practice of CMMI (Lavallée 2012). Requirements engineering is covered in CMMI by the two process areas requirements development (at maturity level 3) and requirements management (at maturity level 2), and verification and validation are covered by corresponding CCMI process areas by those names (both at maturity level 3). Furthermore, the CMMI framework describes intended connections between these process areas and includes alignment practices such as traceability and cross-review of requirements against project plans and other work products, e.g. design and test artefacts, managing requirements changes. These practices have been identified as supporting RET alignment (Bjarnason 2012b, Uusitalo 2008) and applying them as part of a CMMI effort would thus be expected to lead to an improvement. However, we are not aware of any empirical research studies on the impact of CCMI (level 2-3, and higher) on RET alignment.

However, there are some studies that report on the correlation between the RE and the testing processes. Damian and Chisan (2006) found that simultaneously improving these processes can lead to pay-offs in improved test coverage and risk management, and in reduced requirements creep, overscoping and waste, resulting in increased productivity and product quality (Damian 2006). Similarly, Kukkanen et al. (2009) report that improving the requirements and testing processes by integrating these two processes for one case in the safety-critical domain led to improvements in customer satisfaction and in product quality. The implemented improvements included clearly defining RE and testing roles for the integrated process, improved alignment and coordination by connecting processes and people from requirements and testing, and implementation of good practices that support

this connection. Furthermore, the risk of overlaps and gaps in the processes for RE and testing, e.g. in roles and activities, was found to be reduced when concurrently improving both processes (Kukkanen 2009).

3.4 Related SPI Methods

There are some inductive SPI frameworks that similarly to Gap Finder also include project team reflections and some element of theory. These include an *iterative improvement process* by Salo and Abrahamsson (2007), a framework proposed by Pettersson et al. (2008) called *iFLAP* and a method called *the team radar instrument* proposed by Brede Moe et al. (2009) and improved by Angermo Ringstad et al. (2011). In addition, Unterkalmsteiner et al. (2013) present a framework called *REST-bench* specifically aimed at assessing and improving RET alignment. However, we are not aware of any method or framework that applies the Gap Finder approach of deriving improvement suggestions from empirical-based theory in the analysis step of the assessment. Rather, the related SPI frameworks derive specific improvement suggestions through elicitation with the practitioners, either in an assessment step or as part of a project team meeting.

The *iterative improvement process* proposed by Salo and Abrahamsson (2007) consists of a number of short steps which pivot around a retrospective meeting with the development team. These steps including the retrospective meeting can be integrated into agile iteration sprints and repeated throughout a project, thereby achieving continuous process improvement in a structured manner. Process improvements are derived based on obstacles and issues elicited from the software development team. The found issues are grouped and discussed, and an improvement plan for the upcoming iteration is agreed at a retrospective meeting with the team. The method has been evaluated for five industrial cases by applying the method to 3-4 subsequent iterations of development projects. The method was found to yield improvements both in practice and in increased project-team satisfaction. The study also indicates that up to a third of agreed improvement actions were not implemented and the authors draw the conclusions that the SPI activities need to be more systematic. Furthermore, organisational support was found to be required for a third of the actual performed SPI activities of the project teams. The authors' believe that this support is vital for the team's motivation to participate in the SPI activities.

The *iFLAP* SPI improvement framework presented by Pettersson et al. (2008) produces improvement plans through eliciting improvement issues from practitioners and by supporting the organisation in identifying an appropriate improvement plan. In the improvement planning various factors are considered including organisational needs, restrictions, cost and risk. The planning also entails prioritising and identifying dependencies between the improvements, and actively ensuring there is sufficient agreement to the improvement effort. Two case studies of applying iFLAP at Volvo Technology with the aim of improving their RE practices are presented as validation of the method including a number of lessons learnt from these studies.

Brede Moe et al. (2009) propose an instrument called *the team radar* for improving on agile software development projects by qualitatively assessing factors found to influence team work and then visually present and discuss these with the

project team to identify improvements. The five assessed factors were derived from empirically-based theory on team-work challenges and are: shared leadership, team orientation, redundancy, learning and autonomy. These factors are assessed through semi-structured interviews with team members and a subsequent discussion of these with the involved researchers who then rate each of the five factors between 0 and 10. These ratings are presented to the development team using a radar diagram and jointly reflected on with the aim of improving team work by strengthen the underlying factors. The instrument was found useful to practitioners in identifying improvements and the five factors were confirmed as relevant to team work in agile development. Furthermore, the instrument provided a common vocabulary for the practitioners and the researchers to discuss the topic, i.e. team work.

The *team radar instrument* (Brede Moe 2009) was further improved by Angermo Ringstad et al. (2011) through strengthening the diagnosis step and by applying action planning. The diagnosis phase of the method was expanded to (in addition to interviews) also include observations of the assessed team's daily work. The rating of the underlying factors for team work was based on a structured analysis of all the gathered data, i.e. interview transcripts and field notes from the observations. An action plan to address found issues was then specified at a meeting where the ratings were presented to the team who were invited to discuss the presented picture and areas to improve on. The defined actions were based on the underlying theoretical framework of factors affecting team work in agile development. This improved version of the team radar was found to support project teams by illuminating issues not previously discussed within the team. This was contributed to providing a view of the situation by highlighting underlying factors and causes, rather than merely pointing out experienced problems.

The REST-bench (Unterkalmsteiner 2013) assesses RET alignment by modelling the information flow between requirements and testing for a specific project using an artefact map. By eliciting information individually from requirements and testing roles inconsistencies and incongruences can be uncovered. These are then discussed and resolved at a common workshop where the flow is also analysed and improvements identified. When applying the method on a one-year project at Ericsson AB, a number of misunderstandings were uncovered and subsequently resolved at the workshop, which also resulted in identifying bottlenecks and sub optimisations in the RET interaction.

4 The Gap Finder Method

Gap Finder enables assessing a development project by measuring a set of *RE distances* and identifying relevant *RET improvement practices* by consulting the theoretical framework of the Gap Model. These practices can bridge or decrease troublesome distances and can thus support improved alignment between requirements and testing (RET). The distance measurements obtained using the Gap Finder provide an *iRE profile* (integrated RE profile*)* of the current level of RET integration for a project. This profile and the identified improvement practices are presented to the assessed project team at a *gap workshop*. This workshop has the dual purpose of validating the output of the Gap Finder and agreeing with the team on which improvement practices to implement.

Before applying Gap Finder the generic parts of the method need to be tailored to the specific case. In particular, this applies to the measurement instrument which needs to be specialised to the specific roles and artefacts involved in the requirements and testing activities. This requires knowledge of the current process for the specific case. *The user guid*e that is part of the Gap Finder method provides guidelines for how to tailor and apply the method to a case, and is available on-line (Bjarnason 2013d).

The theoretical framework of the *Gap Model* (see Section 2.3) on which the Gap Finder relies acts as a knowledge base. The Gap Model contains relationships between distances and RET alignment practices. This framework is used in the analysis of the measured distances, called *gap analysis*, to identify relevant improvement practices. These practices are identified by comparing the distances found in the obtained iRE profile with the Gap Model and extracting RET practices known to bridge or decrease troublesome distances. The main steps for applying Gap Finder are described in Section 4.1 while the measurements are presented in Section 4.2. The iRE profile is outlined in Section 4.3 and the gap analysis is described in Section 4.3.

4.1 The Four Main Steps of the Method

Applying Gap Finder to a specific case involves four main steps: (I) *preparations*, (II) *measuring*, (III) *gap analysis* and (IV) *gap workshop*. After preparing and tailoring the method for the specific case (step I) the distances can be measured (step II). These measurements are then analysed to identify gaps and potential improvement practices (step III). The outcome of this gap analysis is presented at a gap workshop (step IV) and a set of practices are agreed upon. These practices are then implemented (after step IV) and the project is re-assessed by iterating from step II. An overview of the steps involved in applying the Gap Finder is shown in Figure 2.

4.1.1 Step I: Preparations

For successful application of Gap Finder, the scope, extent and timeframe of the assessment needs to be prepared and planned in agreement with the host organisation in which the assessment is to take place. In addition, the Gap Finder measurement instrument needs to be tailored and adapted to the processes of the assessed project. Both of these activities require insight into the processes and practices of the organisation. The method may (in the future) be applied by someone with this knowledge, e.g. a process engineer. Otherwise initial investigations are needed to obtain this knowledge. In particular, knowledge of roles and artefacts involved in the requirements and testing processes is needed.

The tailoring entails adapting the measurement instrument (see Section 4.2) by configuring it for the exact roles and artefacts applicable to the specific case. For example, if developers are involved in detailing requirements (as for the evaluated case) their role needs to be included in the assessment as part of the set of roles involved in requirements activities. This entails tailoring the measurement instrument to include their technical skills as developers in the measurement of cognitive distance. In this case, the measurement instrument needs to be extended with an additional measure to cover this technical skill (design and development)

and a survey question added for this. Furthermore, planning for applying the method is also affected by this since the people fulfilling this role then need to be included in the assessment, and agreement for this needs to be obtained from their manager.

Furthermore, as part of the tailoring the survey questions need to be adapted to refer to case-specific terminology. This will reduce misunderstandings and support a more consistent understanding of the questions by the survey participants. For example, the survey question that mentions 'the system' can be clarified by adding the name of the system the assessed team are developing.

Tailoring needs to be performed each time Gap Finder is to be applied for a new case. Furthermore, as practices, roles and terminology change for a case the measurement instrument may need to be updated to reflect this.

The output of the preparation step is a measurement instrument adapted to the specific case, and an agreement concerning the project and time period for which to perform the assessment.

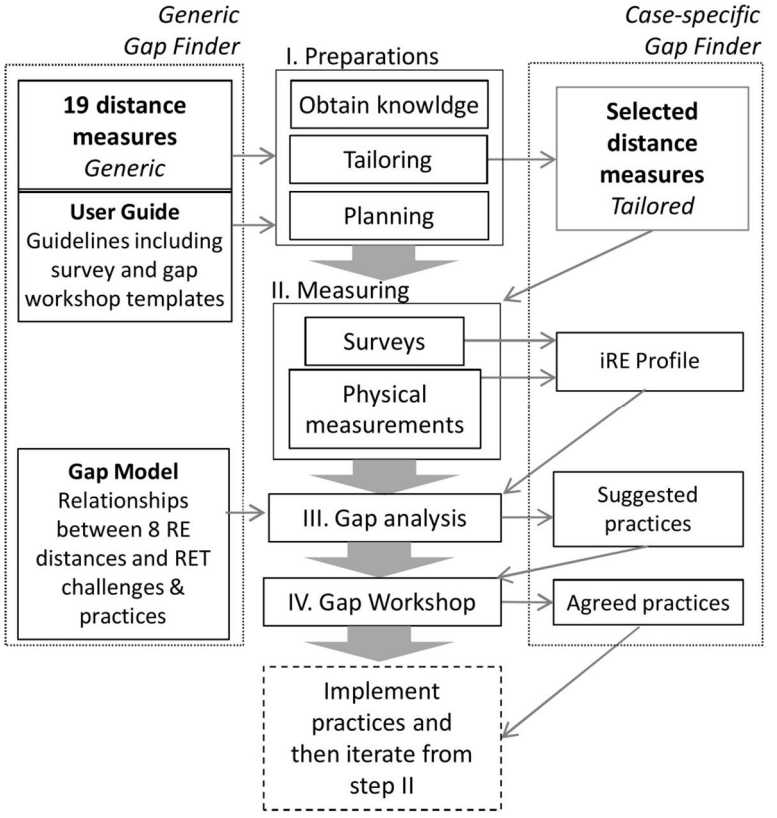


Figure 2. An overview of the Gap Finder method (generic and case-specific parts) and the four steps of method application.

4.1.2 Step II: Measuring Distances

Gap Finder's measurement instrument consists of three surveys: profile, communication and artefact survey. The profile and communication surveys contain questions concerning the project members, while the artefact survey investigates distances for specific requirements.

The surveys are administered to the roles involved in the requirements and testing activities. The first time Gap Finder is applied to an organisation, it is recommended to use interviews for the surveys. This will allow the participant to ask for clarifications, which can enable a more uniform understanding of the questions and of the scales used to answer them. In addition, the interviewer can ask follow-up questions and thereby obtain a richer picture of potential issues and reasons for them. This is particularly important when the interviewer is not intimately acquainted with the project.

4.1.3 Step III: Gap Analysis

When the results of the distant measurements have been collated into the iRE profile this can be analysed to identify gaps. Where the project displays potentially troublesome gaps the Gap Model is consulted. The model provides information on practices that can address these types of distance. Through analysis and comparison of the distances found in the iRE profile against the information in the Gap Model a set of improvement practices are identified. This analysis is further supported by any additional knowledge about the specific case, e.g. contextual factors such as project size, development model, specific practices applied.

The output of the gap analysis consists of a set of improvement practices that may address the gaps identified in the iRE profile.

4.1.4 Step IV: Gap Workshop

The iRE profile and the improvement suggestions are presented to the assessed project team at a gap workshop. For each distance type, the relevant parts of the iRE profile including the gaps are shown and improvement practices presented. The project members are encouraged to share their observations of potential issues caused by the identified gaps and if and how the suggested practices may address them. This allows for a validation of the gaps and practices identified through applying Gap Finder. Furthermore, it includes the project members in the decisions regarding which improvements to implement thereby increasing the probability of successfully implementing the new practices.

4.1.5 After Step IV: Implement Practices and Iterate from Step II

After having implemented the agreed practices, the situation is re-assessed by iterating from step II. The distances are re-measured (step II) and another gap analysis (step III) is performed. In this gap analysis, the original and the new iRE profiles are compared to assess if the previous gaps have been reduced and/or that the effects of them have been minimised by the implemented practices. Additional or different improvement practices may be uncovered through analysis of the new iRE profile. These are then reviewed and discussed with the project team at another

gap workshop (step IV). At this session a decision is made as to whether or not the SPI effort is completed, and if not the Gap Finder is re-iterated again from step II.

4.2 The Gap Finder Measurement Instruments

The Gap Finder measurement instrument used for assessing a project contains eighteen measurements (see Table 2) that cover the eight RE distances of the Gap Model (see Table 1 and Section 2.3). These measurements are applied to artefacts and people involved in the requirements and testing activities. While some distances are straight forward to assess, others are estimated through surveys with self-rating questions. For example, geographical distance (D1) is assessed by measuring the physical distance to walk between desks, while psychological distance (D3) is measured through a survey question asking each team member to rate the distance towards each other member of the team.

A majority of the distances are complex and contain several aspects. For these distances there is one measurement per aspect and, thus, several measurements per distance. For example, for cognitive distance (D4) five aspects are measured: one aspect of prioritisation of quality aspects for the system, and three aspects of different types of knowledge specifically domain, technical skill, organisation and process.

Most of the survey questions have Likert-type scales with five options for the respondent to choose between. For example, for psychological distance (D3, M3.1) the respondents were asked to rate how hard it was to communicate with colleague n by noting 1-5 for *Not hard (1), Some effort required* (2), *Medium effort* (3), *Much effort* (4), *Extremely hard* (5). Similarly, for the knowledge aspects of cognitive distance (M4.1-M4.3) the respondents were asked to grade their own competence using Benner's (1982) five levels of experience, i.e. *Novice* (1), *Advanced beginner* (2), *Competent* (3), *Proficient* (4) and *Expert* (5). The cognitive distance between two people was then measured by calculating the difference between their levels of competence. For the artefact survey, the aspects abstraction (M5.2.3, M6.3) and coverage (M5.1.2, M5.2.2, M6.2) are directional, i.e. the abstraction level of artefact A may be higher or lower than artefact B. For these questions the following scale was used: *Much more*, *Somewhat more*, *The same*, *Somewhat less*, *Much less*, and *Can't say*.

The aspect of priority for cognitive distance (M4.4) was assessed with a survey question on the relative priority of the quality characteristics specified in ISO/IEC 9126-1. The respondent was asked to distribute 30 *resources* over the six quality characteristics. The distance between two people was then assessed by calculating the Cartesian distance between their responses.

The distance for the measured aspects can be calculated in various ways either individually per measurement or combined to a total distance for the whole project. For example, the average value for one aspect of distance between each pair of team members can be considered, or the distance between the minimum and the maximum value. The total distance for a distance type for which multiple aspects are measured can be obtained by calculating the Cartesian distance between the multi-dimensional data points for each participant.

Table 2. Overview of measurements (M1-M8) per distance (D1-D8, see Table 1).

Measurement		*Distance*	*Aspect*	*Survey*
M1	Physical distance between desks	D1	Physical	Profile survey
M2	Length of path in line organisational tree between two people	D2	Home unit in line organisation	
M3.1	Perceived effort to communicate with another person	D3	Uni-directional	Comm survey
M3.2	Perceived effort to communicate between two people		Bi-directional	
M4.1	Difference between people's knowledge of system domain	D4	Domain knowledge	Profile survey
M4.2	Differences in competence within technical areas affecting requirements and testing alignment		Technical skill	
M4.3	Differences in knowledge of project and organisation including processes		Process and organisation	
M4.4	Differences in prioritisation around product		Priorities	
M5.1.1	Difference between product actual and agreed product behaviour	D5.1: Delivered vs agreed reqs	Similarity	Artefct survey
M5.1.2	Difference in coverage between actual and agreed product behaviour		Coverage	
M5.2.1	Difference in meaning between documented vs agreed requirements	D5.2: Agreed vs documntd reqs	Similarity	
M5.2.2	Degree of coverage between documented vs agreed requirements		Coverage	
M5.2.3	Difference in abstraction level between documented vs agreed requirements		Abstraction	
M6.1	Difference in meaning between requirements and testing artefacts	D6: Reqs vs test cases	Similarity	
M6.2	Degree of coverage between requirements and testing artefacts		Coverage	
M6.3	Difference in abstraction level between requirements and testing artefacts		Abstraction	
M7.1	Number of clicks to navigate from a requirement to the test cases which verifies it	D7	Req to Test cases	
M7.2	Number of clicks to navigate from a test case to the requirement(s) that is verifies		Test case to Reqs	
M8	Length of time between specifying a requirement and defining a test case for verifying it	D8	Reqs – Test case definition	

4.3 The iRE Profile

A project's integrated RE profile for testing, or *iRE profile*, provides a view of the project's current level of RET integration. The iRE profile is produced by collating the measurements for each distance. For example, the cognitive and psychological distances between the roles responsible for requirements and testing are included in the iRE profile.

The range and average value for each type of distance can be presented as part of the project's iRE profile. For measurements with the same scale, or scales that can be normalised, the various aspects and distances can be visualised together in a radar diagram, see example in Figure 3. In order to avoid the limitations of this type of visualisation, the ordering of the axes needs to be considered and kept consistent, in particularly when comparing diagrams over time.

The iRE profile is used as input to the gap analysis (step III) and to the gap workshop (step IV). When analysing the iRE profile individual distances between project members and roles may need to be considered to identify distances that need addressing. Similarly upon re-assessing a project, the two versions of the iRE profile can be compared to assess the effect of the implemented practices.

4.4 Gap Analysis: Identifying Improvement Practices

The set of distances within an iRE profile can be compared to the existing knowledge of distances found in the Gap Model (see Section 2.3), thereby identifying improvement practices that may address gaps within a project. For example, if a large organisational distance is seen in the iRE profile the Gap Model, based on empirical knowledge, suggests 14 different practices for mitigating this gap. This large set can be whittled down to a more manageable number of practices by a combination of matching the sets proposed by Gap Model for each identified gap and considering the suitability including cost of each practice for the assessed development organisation. The aim is to identify a small set of practices that can address all the identified gaps and that are a good match for the organisation at hand.

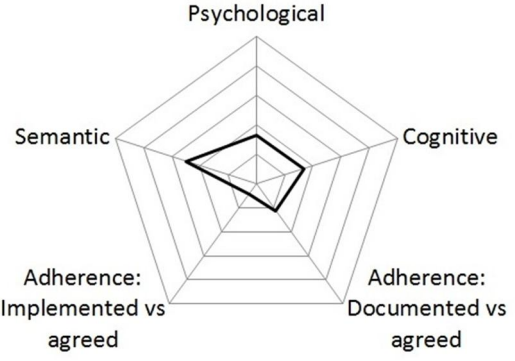


Figure 3. A radar diagram visualising part of an iRE profile of the assessed project. The average distances are shown.

5 Case Description

A development project within The Open University's IT unit provided the case for this evaluation. The Open University is the largest academic institution in the UK with more than 240,000 students studying from all over the world. The IT unit is responsible for the day-to-day management of the university's information systems and in-house development of some systems. The studied project is part of a programme developing a system for student administration and curriculum management to meet the new requirements posed by evolving curriculum needs, changed fees and funding regulations, and subsequent changes to internal business processes. Prior to and in parallel with these changes the IT department itself has also undergone a number of changes. The management structure was overhauled and externally recruited staff appointed at all levels. Replacing the phase-based process for software development with agile work practices has had a large effect on the requirements and testing activities. Test engineers have been recruited for function and system testing. An overview of the case is provided in Table 3.

The Scrum development method is applied at team and intra-team levels. Each development team consists of a product owner, a requirements analyst, a tester, a number of developers and a scrum master. In addition, there is also a project manager responsible for the project to which the team delivers. The product owner represents the business and is responsible for the scope including signing off on acceptance of project deliveries. The requirements analyst is responsible for eliciting and defining the requirements in close collaboration with the product owner and the development team. The scrum master, project manager, developers and testers all take an active part in discussing, and thereby defining, the detailed requirements. Finally, the tester within the team is responsible for verifying that the software produced by the team corresponds to the agreed requirements. The team members of the studied development team are characterised in Table 4.

The project scope is described in definition documents and in agile epics by senior requirements analysts (not necessarily the requirements analysts of the development team) and allocated to one of the planned four releases of the system. For each release the epics are detailed into user stories and acceptance criteria by the requirements analyst for the intended development team and placed in that team's backlog. Development is performed in 2-week sprints (iterations) and prior to each sprint the user stories in the backlog are prioritised by the product owner and requirements analyst. The user stories with the highest priority are then presented to the development team who estimate them and a set of stories are agreed on for that sprint according to priority and team capacity.

Development of a user story is initiated by a discussion between the developer, requirements analyst and tester where requirements and technical details are discussed and agreed. The requirements analyst will take any uncertainties or questions regarding the user requirements back to the product owner to ask for clarifications. Similar requirements clarifications are made throughout the sprint. The tester develops test scripts to verify the agreed requirements. These scripts are executed and any issues found are reported. Completed user stories, i.e. developed and successfully tested, are demonstrated to the product owner at the end of the sprint. A retrospective meeting is then also held where the development team reflect on the past sprint and on ways to improve team work practices.

Table 3. Characteristics of the studied case and Company A (on which Gap Model is based).

	Open University	Company A of Paper III
Type of case	Academic education provider	Software development of embedded products
# people in software development unit	Approx. 150 for IT development (300 for whole IT unit)	125-150
# people in project	Approx. 20	10
Distributed	No	No
Domain / system type	IT: Educational programme management including student services	Computer networking equipment
Source of requirements	In-house	Market driven
Main quality focus	Maintainability	Availability, performance, security
Certification	No	Not for software
Process model	Scrum	Iterative
Duration of project	2-3 years	6-18 months
# requirements in project	Approx. 800 user stories	100 (10-30 pages of html)
# test cases in project	Approx. 1,300 test cases	Approx. 1,000 test cases
Product lines	No	Yes
Open source	No	Yes

Table 4. Roles and length of experience for the members of the studied development team. The number of people included for each survey is also given (see Section Table 4).

Roles	Length of experience in team role (months)	Total length of work experience (years)	Surveys		
			Profile	Communication	Artefact
Product owner	10	26	1	1	1
Requirements analyst	0	28	1	1	1
Tester	3	26	1	1	1
4 developers	8, 9, 9, 0	7, 6, 22, 10	2	4	0
Scrum master	10	26	1	1	0
Project manager	3	25	1	1	0

The epics, user stories and acceptance test cases are stored in a central requirements repository with traceability links. The test scripts are stored in another repository and linked to the relevant user story. These test scripts can then be viewed from the requirements repository.

Once the development team has delivered accepted functionality the system is tested as a whole both from a user perspective and from a system integration perspective. This testing is performed by team-external testers and by representatives from the business unit. Any issues found in this testing is initially analysed by the tester in the development team before further decisions and actions are taken to either reject or agree to address the issue. The team tester and the team-external testers are assigned from the same department.

6 Research Method

The evaluation of Gap Finder was performed through a case study (Robson 2002, Runeson 2012) where the Gap Finder method was applied to a development project. The aim of the study was to perform a formative evaluation of the method, i.e. to seek feedback that could guide further design and improvement of the Gap Finder and thereby ensure that the method is usable and useful (Rogers 2011).

A combination of empirical research methods was applied in the evaluation of the Gap Finder as outlined in Figure 4. Apart from the methods included in Gap Finder (i.e. surveys and focus group, see Section 4), observations and interviews were also performed. An *ethnographically-informed approach* (Robinson 2007) was taken in the observations to ensure that relevant data was collected with the Gap Finder and that it was understood in-line with the team members' perception of the situation.

The study design, data collection and analysis was mainly performed by Bjarnason, and reviewed and validated by Sharp. In addition, Sharp provided support in the contact with the case organisation and participated in one initial interview and in the gap workshop where the outcome of the Gap Finder was presented to the development team.

6.1 Preparations

Some preparations were needed to apply and evaluate Gap Finder in a live development project, namely a) design of a research method for the evaluation, b) tailoring Gap Finder (and its measurements) for the case and c) planning for applying the method. All of these activities required d) obtaining insight into the development organisation and, in particular, into the roles, artefacts and practices of the development team. Each preparational activity is described below.

6.1.1 Obtaining Knowledge of Case

The knowledge of the case required for applying and evaluating Gap Finder was obtained through document studies, a semi-structured interview with two managers from the case organisation, demonstrations, and observations of the development team. One of the authors had an existing relationship with the case organisation and

therefore also some initial documentation and contacts. These documents were studied and discussed between Bjarnason and Sharp and an interview instrument was designed (available on-line Bjarnason 2013d) to clarify and obtain additional knowledge about the roles, artefacts and activities used for requirements and testing.

Two managers within the IT development unit were interviewed in order to provide a picture of the development process and how the project was organised. At the managers' suggestion they were interviewed at the same time using an open semi-structured interview format. The managers shared their view of current challenges and good practices and supplied a number of pointers to information and people including access to various development artefacts, e.g. requirements, backlogs, test cases etc.

Insight into development artefacts and how they are used by the requirements and testing roles was also obtained through demonstrations and document studies. The artefacts used for function- and system-testing were demonstrated to the researchers by two different test engineers. Furthermore, by exploring the requirements and testing artefacts in the application management system used to store these, the researchers gained an understanding of the amount and extent of available artefacts and information about them.

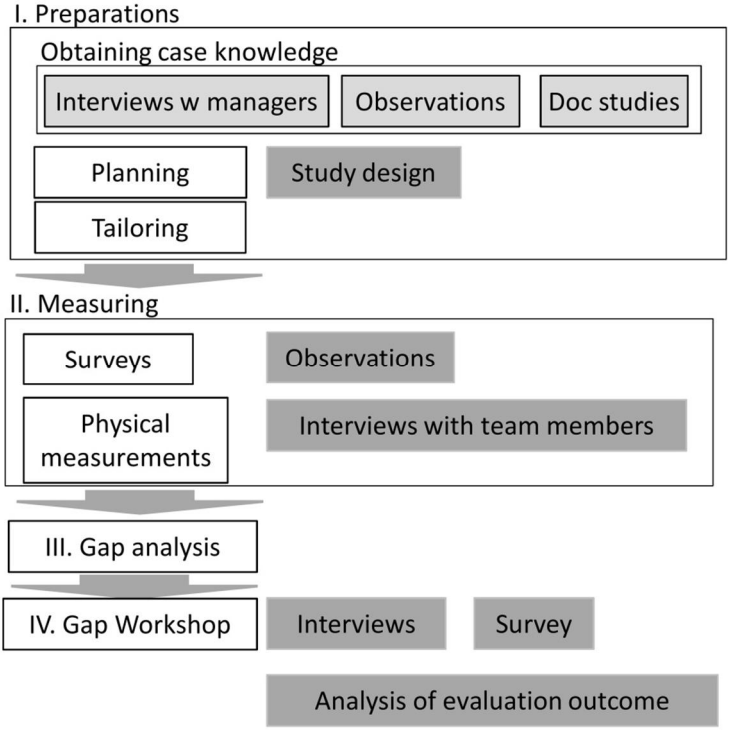


Figure 4. Overview of the Gap Finder evaluation study including the applied Gap Finder steps. Activities specific to evaluation study are marked with grey. The light grey activities (for *Obtaining case knowledge*) are the activities performed for that optional step in this evaluation.

Finally, one development team (the same as was later assessed) was observed for a consecutive period of three days approximately a month before the Gap Finder measurements were taken. These initial observations took place at the end of one sprint including the sprint review and planning for the next sprint. The purpose of these observations was to establish contact and gain familiarity with this team, and to secure an understanding of their day-to-day work. These observations enabled tailoring the Gap Finder measurements and fine-tuning the research method for the evaluation.

6.1.2 Research Study Design

In addition to merely applying Gap Finder, the researchers decided to add additional data collection in parallel to the method application. For this reason, semi-structured interviews were held in connection with the surveys and the team was observed for the time period during which Gap Finder was applied. This additional data allowed the researchers to evaluate the Gap Finder measurements and the outcome of the gap analysis by applying triangulation.

An explorative research approach was taken in this case study meaning that the initial study design evolved and was adapted over time as new insights were gained. Similarly, even though the different parts of the case study are here described as separate and sequential activities an iterative approach was applied throughout the study, meaning that the study design was continuously re-visited. For example, when it became apparent that the applicability of certain practices had not been commented on during the gap workshop a survey of the suggested practices was designed to complement that data.

6.1.3 Planning for Applying Gap Finder

The researchers decided to apply Gap Finder at the sprint iteration level for one development team. This allowed an evaluation of the Gap Finder method covering a full set of development activities including requirements detailing, design, development and testing within a feasible time frame and with a clearly delimited set of requirements. The team that had been initially observed (see Section 6.1.1) agreed to participate also in this part of the study to which their managers consented.

The original plan was to apply Gap Finder twice for two different sprint iterations, with a new practice implemented for the second iteration. However, due to changes in the project which reduced the amount of development of new requirements this was not possible. Instead, the plan was adapted to apply the method iteratively during two consecutive sprints.

6.1.4 Tailoring of Gap Finder

Before applying the Gap Finder its measurement instrument needed to be tailored to the specific case (see Section 4.1.1). The information obtained about the case organisation and project (see Section 5) was utilised for this. In particularly this included the knowledge gained of which roles, and artefacts that were involved in the requirements and test activities, and how the requirements and test cases were managed in the requirements repository. Based on this insight the researchers

decided to exclude the measurements for navigational (D7) and temporal distance (D8) from the evaluation study. Since tracing was applied between requirements and test cases the navigational distance would always have resulted in the value 1, which would not allow for detecting any gaps for this dimension. Temporal distance was excluded from the evaluation due to the practical difficulties in measuring this for the case organisation. Since an agile development approach was applied the on-going discussions within the team is the main source of requirements information rather than the artefacts (which is what the generically defined Gap Finder measurement for this distance assumes). The measurement instrument used for the evaluation (with some terms replaced for confidentiality and anonymity reasons) can be found on-line (Bjarnason 2013d).

Since the requirements were defined through team discussions in this agile project, all roles represented in the development team were involved in requirements activities. However, the product owner, requirements analyst, tester and developers were the primary roles involved in detailing the requirements, while the scrum master and project manager were primarily involved in requirements discussions at a more general level. The measurement instrument was customised accordingly. Namely, the set of measurements were adapted to cover all of the primary roles and their corresponding technical skills of scope management, requirements engineering, testing, design and development. In addition, all roles were asked to participate in the profile and communication surveys, while the artefact survey was limited to the product owner, requirements analyst and tester.

Similarly, the measurement instrument was designed to cover the specific artefacts and activities used for requirements and testing in the case organisation. In addition, generic terminology was replaced with specific terms used within the organisation, e.g. ‘documented requirements’ was replaced with ‘user stories’.

6.2 Measuring Distances

The RE distances within the development team and between their requirements and testing artefacts were measured by applying the Gap Finder. In parallel, additional data relevant to these distances, e.g. experienced issues and strategies applied to mitigate these, was gathered through interviews and observations.

6.2.1 Obtaining the iRE Profile

The iRE profile for the assessed project was obtained through applying the measurement instrument of the Gap Finder including the communication, profile and artefact surveys. The communication survey was taken by all team members, while the profile survey covered each role within the team. Thus, all team members except two of the four developers where included in the profile survey. The two surveyed developers were included due to being available and actively involved in the current sprint.

At the end of the second sprint that was studied distances related to the requirements and test artefacts for the delivered functionality were measured by administering the artefact survey to the product owner, requirements analyst and the tester.

The communication and profile survey were administered during the first sprint while the artefact survey was performed after the following sprint was completed. For each survey, the targeted respondents were free to choose whether or not to participate.

Information on where each team member was located was obtained through the profile survey and through observations for the team member located in the team area. The physical distance between the desks in the team area were measured with a tape measure, while the distance to desks in the other buildings was estimated based on a map.

6.2.2 Interviews

The profile and artefact surveys were administered as semi-structured interviews around the survey questions. For each question, the interviewer ensured that the interviewee understood the question and the scale correctly. In addition, follow-up questions were asked to clarify the interviewee's responses and gather additional information concerning specific events including factors contributing or resulting from each distance.

During the interviews the answers to the survey questions were noted by the interviewer on a copy of the survey in full view of the interviewee. The interviews were audio recorded and transcribed. The transcriptions were used both in the gap analysis and for the analysis of the evaluation outcome.

The communication survey was not combined with interviews but each team member was asked to fill it in individually and return it to the researcher. This difference in approach was due mainly to the sensitive nature of the question on ease of communication with individual team members, but also due to simpler questions. Furthermore, it was administered when 2 whole weeks of observations had been performed by which time a good insight into communication within the team had already been obtained.

6.2.3 Ethnographically-Informed Observations

An ethnographically-informed approach was applied in the observations of the development team. The purpose of these observations was to gain a rich insight into the day-to-day work practices of the team members and their interactions with each other. The ethnographical approach entailed seeking to understand the team's work practices apart from the researcher's assumptions about software development (Robinson 2007). The observations were as unobtrusive as possible and questions were only asked to seek clarification of used terminology or actions, and never to participate in team discussions. The distances and practices of the Gap Model provided a 'protocol' that supported the observer in taking particular note of activities and interactions potentially related to these. Extensive field notes were made during the observations including interactions in the team area, status and information shared during meetings, and individual activities.

6.3 Gap Analysis: Finding Gaps and Improvement Practices

Gaps were identified by analysing the obtained measurements, and generic RET practices for addressing these were extracted from the Gap Model. A qualitative approach was taken in analysing the quantitative measurement data. This was partly due to the fact that since these were the first measurement values obtained there was nothing to compare these to and thereby quantitatively identify troublesome distances. However, even when such reference data become available in the future the analysis needs to respect contextual factors that may influence if a distance is 'good' or 'bad'. For example, for a case with extensive requirements documentation the distance between artefacts is likely more critical than for a case relying heavily on face-to-face communication of requirements.

The gaps were identified by analysing the measured data from a number of different perspectives. This was done by calculating both the total distance for all distance types and the individual distance per measured aspect. In addition, for each distance type and aspect the average and range of obtained values were calculated and analysed. For the measurements using Likert-type scales the median values were calculated and found to be very close to the mean values, which were then chosen in order to present uniform types of values. Since the further analysis of these values was qualitative, we judge that this choice did not affect the following Gap Finder steps. The distance between pairs of people was also calculated to identify potentially large gaps between specific roles and individuals.

The improvement practices were extracted from the Gap Model by querying it for the identified gaps. This set of generic practices were adapted to more specific ones by considering additional data and insight into the case, thereby tailoring the practices for the specific case. For example, the generic practice of co-location was tailored to the specific practice of providing a guest desk for product owner since this person was seated the furthest away and incurring the largest geographical distance within the team.

The initial set of obtained measurements and identified gaps were for distances between people since these were the ones measured during the first sprint included in the study. However, the measurements of artefact-related distances (obtained after the end of the second sprint) did not reveal any additional gaps and subsequently the Gap Model did not need to be queried again. Thus, the initial set of obtained practices remained intact.

6.4 Gap Workshop: Present and Agreed on Improvements

A gap workshop was held at the beginning of the second iteration to present the iRE profile obtained so far and the identified improvement practices to the development team. The main intention of the workshop was to gauge the practitioners' views on the suitability of the suggested practices. Furthermore, the relevance and validity of the presented distances and gaps was also assessed at the workshop. The whole team was invited to the workshop and six of nine team members attended. The content and questions of the workshop was later covered with the three absent team

members by individual semi-structured interviews. Furthermore, the suitability of the suggested practices was also assessed through a survey after the second iteration was completed. The survey was sent by e-mail to one of the managers within the IT development unit, the scrum master and the tester from the assessed (the template is available on-line, see Bjarnason 2013d).

The workshop was opened with an introduction to the Gap Finder method and the concept of RE distances within software development. An overview of the full set of measured distances was then given before presenting and discussing the findings for each distance type. First the obtained measurements were presented and an open question asked if and how this may have an impact on their work. The practices suggested by the Gap Finder method were then presented, and the team asked to comment on if the practice may address the distance and improve on alignment, and if they thought it was a suitable practice to adapt.

For each distance, the average, minimum and maximum values were shown. For measurements with identical scale, or a scale that can be normalised, the distance values were shown using radar diagrams. An example of this is given in Figure 3 where the normalised values for psychological, cognitive, adherence and semantic distance are all shown in the same radar diagram. Furthermore, the multi-value measurement for the priority aspect of cognitive distance was presented by showing the individual values for each factor and for each (anonymous) team member and high-lighting where gaps had been identified.

After having discussed each distance type, the participants were asked to reflect individually on issues related to the presented distances and practices for addressing these. These reflections were written on post-it notes and then shared and discussed within the group.

At the end of the gap workshop the participants were asked if and in which way the Gap Finder method was useful including how they had experienced the workshop.

The gap workshop was audio recorded (after agreement was obtained from the participants), transcribed and summarised. This summary was then distributed to all the team members who were asked to provide feedback if anything was incorrectly described or if they had additional reflections.

6.5 Analysis of Evaluation Outcome

After completing the application of the Gap Finder the complete set of gathered data from the method application and from the evaluation activities was analysed. The researchers' experience of applying the Gap Finder was also considered. For each of the research questions relating to how the Gap Finder supports different aspects (RQ1-RQ3) the relevant data from the different sources was analysed together. This was also done for each measured distance and suggested practice. Data from the measurements, the interviews and the observations including the feedback gathered at the gap workshop was thus compared and triangulated.

7 Results

The outcome of applying Gap Finder consists of the iRE profile (the set of distances measured for the studied project), the set of practices identified through the gap analysis and the shared reflections at the gap workshop. This outcome is presented below alongside the additional data captured for these aspects through the observations and interviews. In addition, limitations and threats to validity for these results also discussed.

7.1 The iRE Profile for the Assessed Project

An iRE profile for the assessed project was constructed from the distance measurements and used as the basis for the gap analysis. Furthermore, measurements for each of the distances within the iRE profile were presented to the development project at the gap workshop. An overview of the derived profile is shown in Table 5. The obtained values for each type of distance are presented below together with qualitative data from the observations and interviews performed as part of the evaluation.

7.1.1 Geographical Distance (M1)

The core team members (scrum master, developers and testers) were co-located in one common team area. However, the other team members (product owner, requirements analyst and project manager) were located elsewhere. The project manager had a desk in the same office as the team while the requirements analyst was located on a different floor in the same building. In addition, the product owner was located in a separate building approximately 300 metres away. The average distance between each pair of team members was 77 metres, see Table 5, while the total distance between each pair was 2,760 metres.

The team was aware of the negative impact of these distances and frequently commented on the lack of proximity to the product owner and the requirements analyst. During the interviews several people commented on the negative impact of geographical distance as causing time delays in obtaining information. As expressed by one team member: ‘the conversation slows down’. For example, quick questions concerning requirements may be postponed and then forgotten, or posed to a team member closer at hand. This then results in moving on with potentially incomplete or incorrect information about the requirements. One interviewee said: ‘Even being 2 desks away can have a negative impact. It makes a big difference! It [co-location] makes it easy to quickly check details you are unsure about.’

Co-location (i.e. short geographical distance) was perceived by the team as enabling them to manage requirements changes in a light-weight manner by relying on frequent face-to-face communication rather than on extensive documentation of requirements. As the product owner stated: ‘we get what we expect due to the constant communication.’ The requirements analyst also stated that the geographical distance to the team reduced this communication and attempted to mitigate this, partly through documentation. Information concerning requirements is also frequently picked up by the tester from on-going discussions in the team area.

Table 5. An overview of the iRE profile derived for the assessed project. All values except for geographical and organisational distance are normalised within the range of 0 to 1.

Measured distance				Min	Aver.	Max
M1	D1 Geographical (metres)			1.8	76.7	322
M2	D2 Organisational (steps in organisational path)			0	2.6	7
M3.1	D3 Psychological	Person to person (uni-directional)		0.20	0.35	1
M3.2		Between two people (bi-directional)		0.20	0.35	0.60
M4	D4 Cognitive		*In total:*	*0.15*	*0.29*	*0.46*
M4.1		Domain knowledge		0.00	0.32	0.80
M4.2		Technical skill	*In total:*	*0.17*	*0.32*	*0.50*
M4.2.1			Scope management	0.00	0.36	0.80
M4.2.2			Requirements engineering	0.00	0.27	0.60
M4.2.3			Testing	0.00	0.23	0.40
M4.2.4			Design and development	0.00	0.23	0.60
M4.3		Process-, organisational knowledge	*In total:*	*0.06*	*0.32*	*0.56*
M4.3.1			Local	0.00	0.38	0.80
M4.3.2			Non-local	0.00	0.25	0.52
M4.4		Priorities		0.03	0.08	0.14
M5.1	D5 Adherence	Artefact vs agreed requirements	*In total:*	*0.00*	*0.24*	*1.00*
M5.1.1			Similarity	0.00	0.00	0.00
M5.1.2			Coverage	0.00	0.00	0.00
M5.1.3			Abstraction level	0.50	0.67	1.00
M5.2		Implemented behaviour vs agreed reqmts	*In total:*	*0.00*	*0.08*	*0.25*
M5.2.1			Similarity	0.00	0.17	0.25
M5.2.2			Coverage	0.00	0.00	0.00
M6	D6 Semantic[4]		*In total:*	*0.3*	*0.4*	*0.5*
M6.1		Similarity		Roughly the		
M6.2		Coverage		Somewhat more		
M6.3		Abstraction level		Somewhat more		

Furthermore, the geographical distance sometimes led to a lack of coordination. This was expressed in interviews with the product owner and the requirements analyst, and observed when meetings were cancelled, delayed or moved to another meeting room with short notice. Information concerning these changes was automatically shared between the co-located team members but did not always reach the team members outside of the team area.

[4] One data point only for the measurements of semantic distance.

7.1.2 Organisational Distance (M2)

All team members except the product owner belonged to the IT unit. The scrum master and the developers were organised into one department, while the requirements analyst, tester and project manager each reported to other managers within the IT unit. Since the role of the product owner is to represent the users (in this case the business owners) this role needs to be filled by someone with insight and knowledge of this. In this case, the product owner was from outside of the IT unit at an organisational distance of in total 7 steps up and down the organisational tree, see Table 5. In total there was an organisational distance between each pair of team members of 92 steps.

Two team members described that people from non-local organisational units can disagree due to different priorities and perspective, e.g. on how and which requirements to implement. The product owner mentioned this for the business unit versus the IT unit, while a developer described a similar situation between the development team and other functions within the IT unit. They both stated that these long organisational distances between units make it infeasible to use the line-organisational path as a communication channel for decision-making and for resolving disagreements concerning requirements. When the common manager to which two distant organisational units escalate issues is at a very high level in the organisation this manager is then often too far removed from the context and day-to-day work of the issue at hand to make an informed decision. Escalating decisions in such cases is experienced by several of the interviewees as causing long delays and miscommunication of information.

When there is long organisational distance between roles, the team members found that communicating informally or via a project organisation was a more direct communication channel and therefore more efficient. For example, the product owner had established direct communication channels by attending various project meetings held by the IT department, including meetings concerning project steering, scope and issue management. Similarly the scrum master described that conflicts with other IT development roles were avoided as far as possible by direct communication and by pro-actively seeking alternative solutions. However, both product owner and scrum master mentioned cases where these more direct communication channels failed to achieve an agreement, e.g. with the IT unit on important user requirements, or with other departments within the IT unit concerning design issues. When this occurs, the issue can either be escalated via the organisational channels with subsequent long delays, or left unresolved.

Furthermore, the organisational distance was also experienced to cause practical issues with coordinating meeting schedules. The product owner who frequently attends various meetings at the IT department expressed that these often conflict with other meetings held within the business unit.

7.1.3 Psychological Distance (M3)

The psychological distance between team members was on average short; between *Not hard* and *Some effort required* (on average 1.7 of 5, see normalised value in Table 5). However, there was a wide range of measurement results (see maximum values in Table 5) which indicates that there is some psychological distance

between certain members of the team. There were two counts of *Extremely hard* to communicate given by one practitioner, and three counts of *Much effort* by two other team members, see Figure 5. These values are for the uni-directional distance, i.e. one person's perception of communicating with another. There is a difference when considering the bi-directional distance, i.e. the total effort to communicate between two people. The maximum for bi-directional distance (see Table 5) is lower, namely at the level of *Medium effort*. The psychological distance between two people as perceived by one person does not seem to be necessarily reciprocated by the other, i.e. that one person might find it hard to communicate with another does not necessarily mean that that person finds it equally hard to communicate with the other.

The observations revealed that the communication within the team is good and that there is a strong awareness of the importance of sharing information. For example, information sharing practices were emphasised by several team members at a sprint retrospective. In addition, application of these practices was observed when new team members were integrated in the team. For example, a developer new to the team was brought up to speed by frequently pairing with the more experienced developers. Furthermore, information concerning context and motivation for specific requirements and work practices were spontaneously shared during discussions with new team members.

Occasional occurrences of communication difficulties were observed. On a couple of occasions team members were observed to indicate reluctance to continue a discussion with a neighbour and instead focused their attention on their screen thereby withdrawing from the conversation. Furthermore, during an interview one team member shared an impression that discussions, in this case about requirements, were sometimes very polite rather than being open and frank. Another team member said: 'it is often easier to speak to people who agree with you most of the time. Otherwise you can spend a lot of time discussing.' In a previous interview, this person described that there are different mindsets within the team concerning to which degree developers should be concerned with requirements detailing. This indicates that some of the ratings for psychological distance can be due to cognitive distance and difficulties in reaching a common view on requirements.

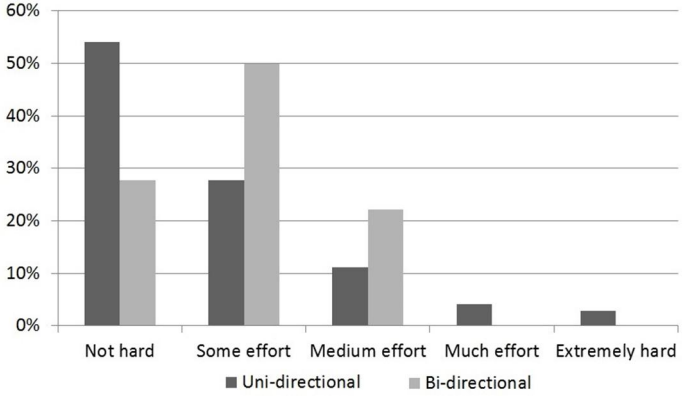


Figure 5. Percentage of occurences for uni-directional vs. bi-directional psychological distance for each pair of team members.

7.1.4 Cognitive Distance (M4)

Within the team the cognitive distance between team members with different roles and length of experience was found to vary greatly. The multiple measures used to assess the cognitive distance were combined to a normalised average value of 0.29 with a maximum of 0.46, i.e. slightly below half the largest possible distance (see Table 5). Long distances were identified for some of the measured aspects. There were large differences concerning technical skills in scope management (M4.2.1, 0.80 of 1) between the product owner and the tester. In addition, long distances were measured between long-standing and new team members for knowledge of the local processes and organisation (M4.3), and the domain (M4.1, 0.80 of 1).

As a whole the team was found to possess near to the maximum amount of knowledge for the three aspects of knowledge that were assessed, see Figure 6. Within the team there is E*xpert* knowledge for the domain (M4.1), local organisation and process (M4.3.1), and for 3 of the 4 technical areas of expertise (M4.4). Furthermore, for wider organisation and process (M4.3.2) and for testing (M4.2.4) there is *Proficient* knowledge. Furthermore, the team members in average have a high level of knowledge, around *Competent*, for all measured knowledge aspects (M4.1-4.3). One team member said: 'I think we have a good mix of people who have been here a long time and new people.' Another one said: 'It is a good team! We're well covered.' This distribution can be a great asset for the team if the knowledge is utilised and shared in an efficient way.

The cognitive distance for the aspect of prioritisation of system quality factors (M4.4) was low in average (below 0.1 of 1) for individual quality factors, although this distance varied for the different quality characteristics, see Figure 7. A larger distance was found when considering the total prioritisation for all quality aspects, in average 0.35 of 1 with a maximum distance of 0.6. In particular, there are longer distances for this aspect between the product owner and other roles. For example, the requirements analyst prioritised functionality lower and maintainability higher than the product owner, while the tester prioritised usability much lower than all non-development roles including the product owner. This is surprising considering that the tester is responsible for verifying and validating the produced software, which in this case is an information system aimed at non-technical users.

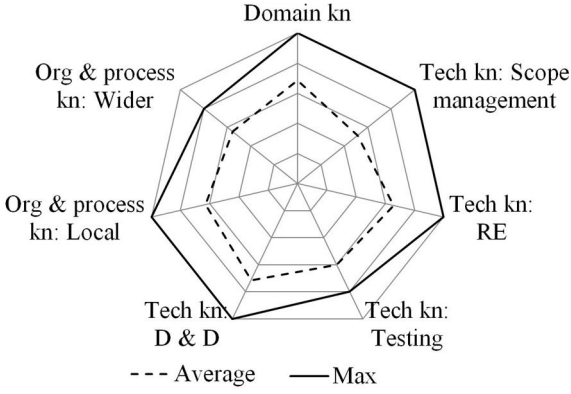


Figure 6. Total and average amount of knowledge of domain, organisational and process, and technical skill within the team.

M4.1 Domain knowledge The tester said that as his domain knowledge increased it was possible to be more proactive, rather than merely reactive. He experienced that the subsequent shorter cognitive distance towards the product owner and requirements analyst enabled a faster response in the testing work and, being quicker to identify issues. This correlates well with the requirements analyst comments on the importance of testers thinking outside of the box and not just testing according to the agreed requirements, although as the interviewee pointed out this requires the tester to have good domain knowledge. Furthermore, one of the developers pointed out that there was a distance in domain knowledge between very experienced requirements analyst and newer developers and tester. On several occasions this gap had resulted in failure to capture incorrect software behaviour (e.g. through testing) due to requirements analysts not communicate what he/she considered to be tacit requirements to the development team. These tacit requirements had thus not being developed or tested, and had not been discovered until further down the line during user acceptance testing.

M4.2 Technical skill The product owner expressed that his/her previous experience of design and of testing supported the communication of requirements with the development and testing roles. Similarly, the requirements analyst mentioned having a good understanding of the requirements information needed by the tester based on his/her own previous experience of testing. Furthermore, this enabled both the product owner and the requirements analyst to perform some user-related testing on the produced software.

M4.3 Organisational & Process knowledge One of the newer team members expressed that he had little insight and knowledge of other teams and areas since there was limited interaction with them. However, this knowledge was increasing as time progressed through getting more involved in work at a wider-project level and through more interactions with other teams and functions outside of his/her own team. The scrum master indicated that the current synchronisation between development teams could be improved by increasing the frequency of the

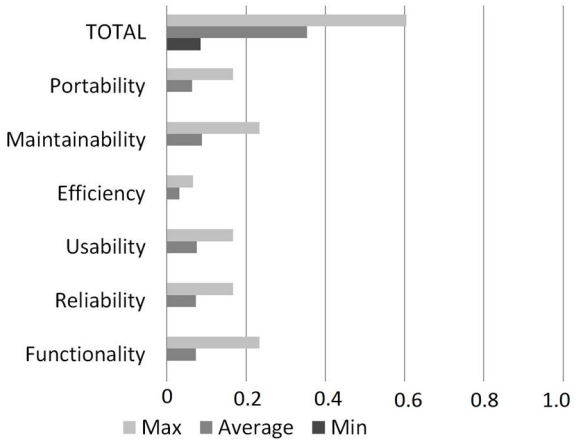


Figure 7. Range of (normalised) cognitive distance for priorities between ISO quality characteristics between team members.

interaction and strengthened by not merely provide status updates but also a more general sharing of information. Furthermore, on several occasions team members were observed to raise questions concerning the system test team. Namely, what the process was for receiving identified issues from system testing, what the focus of the system testing was to be, and in particular, if issues regarding requirements could also be expected to be raised or if the system testing would be limited to the requirements agreed to by the team.

M4.4 Priorities Although there was agreement on the high importance of the quality characteristics maintainability, it was motivated in varying ways by different roles. The product owner, requirements analyst and tester all expressed that this characteristic enabled the team to quickly respond to changing business requirements and bug reports. In contrast, the interviewed developers highlighted that maintainability was required due to the long life-time expectancy of the system in combination with most of the development roles being short term contractors.

There was a difference in view point concerning the quality characteristics between developers and the other roles. One developer indicated that reliability, usability and efficiency were less prioritised since these characteristics were mostly out of the control of this development team. Rather, these characteristics rely on software of lower architectural layers for which other teams are responsible.

Additional aspect of cognitive distance (not covered by the iRE profile) During the interviews the tester and the requirements analyst separately described that the testers which perform the user- and system-level testing have very little knowledge (cognition) of the requirements implemented by the team. In one case a conscious decision had been made for this testing to be performed without any communication with the development team concerning what requirements the software supported, but rather test from a general user perspective. This resulted in a large part of the issues reported from this testing being rejected by the development team since the software worked as designed, thus indicating misaligned views of the requirements. In another case, information was shared through job rotation of a tester previously on the development team circulating to the system test team. In this case, the amount of found and rejected issues was less.

7.1.5 Adherence Distance: Delivered vs. Agreed Requirements (M5.1)

The adherence distance between the agreed requirements and the behaviour of the delivered software was found to be short, with a normalised average of 0.08. The product owner and the requirements analyst both stated that the delivered behaviour was *Almost the same* as the agreed requirements, while the tester judged that *Exactly the same* behaviour had been delivered as had been agreed. This indicates that the tester has a somewhat different understanding of the agreed requirements compared to the product owner and the requirements analyst. The requirements analyst described that the requirement details had evolved after input from and discussions with the developers. Even if these development-initiated modifications were acceptable for the short term, the requirements analyst expected modifications to be required in the future to support the full set of user requirements. Furthermore,

the product owner said that at times the delivered behaviour was more and/or better than what had been agreed.

Additional aspect of adherence distance One developer expressed that being provided with very detailed by the requirements analyst restricts the creativity of the developers. Similarly, the requirements analyst described that testing ought to go beyond the exact details of the requirement that have been agreed in order to validate them by testing from a wider perspective. This wider view in testing might be encouraged by agreeing to requirements at a higher level thereby necessitating active consideration of the details by the developers and the testers.

The short distance between the agreed requirements and the resulting software behaviour was stated by the developer to result in a closed mind-set concerning requirements among the developers. Instead of seeing the given requirement merely as the starting point and then considering a wider picture and context, other team members would work solely within the given set of requirements. This team member had previously experienced more creativity when working in a small company with phase-based development model, but with a longer distance in abstraction level between the requirements agreed with the requirements analyst and the resulting software behaviour.

7.1.6 Adherence Distance: Agreed vs. Documented Requirements (M5.2)

For the adherence distance between the documented requirements and the requirements perceived to have been agreed upon the team members judged there to be no distance for the aspects of meaning (M5.2.1, Exactly the same) or coverage (M5.2.2, The same), merely a distance for the aspect of abstraction level (M5.2.3). This distance in abstraction level is to be expected since documenting all details is infeasible. One of the interviewees also pointed out that the level of detail of the documented requirements was now significantly lower than with the previous phase-based process. This increased in adherence distance for abstraction level is due to the agile development model prescribing frequent face-to-face communication rather than relying on requirements communication through artefacts. The normalised total average adherence distance between requirements artefact and agreement of 0.24 (M5.2) is solely due to this distance in abstraction levels. When interviewed two of the survey participants admitted that the requirements artefact did include requirements that had previously been agreed, but that were now removed from the project scope. Thus, the requirements artefact had not been updated to reflect this change. Subsequently there was a distance in meaning and coverage between the requirements artefacts and the current set of agreed requirements (M5.2.1 and M5.2.2) even though all three survey participants responded to the direct survey question that there was no distance. The requirements analyst explained this by saying that team members are assumed to know about this change of project scope and that a basic understanding of the domain is required of the reader rather than providing extensive details in the requirements artefact.

7.1.7 Semantic Distance (M6)

Some semantic distance was found between the requirements artefacts and the test artefacts. The meaning of the two artefacts (M6.1) was judged by the tester to be *Roughly the same*, thus indicating some differences. For the assessed artefacts this distance was mainly due to requirements information that was not yet in scope for the project, but still included in the documentation.[5]

The tester judged that the test cases covered *Somewhat more* than what was specified in the requirements artefacts (M6.2). In addition, the abstraction level of the test cases (M6.3) was stated to be *Somewhat more* than for the requirements, which is to be expected. Furthermore, the tester described that the level of detail in the test cases was dependent on time availability and was usually more than for the specific set considered in the survey.

7.2 Identified Improvement Practices

Ten practices were suggested as potential improvements for the project based on the output of the gap analysis of its iRE profile. These improvement practices (IP1-IP10) address the gaps found in the iRE profile and may improve the RET alignment by mitigating the found distances.

The improvement practices were presented to and discussed with the development team at a gap workshop. For two members who could not attend the workshop their feedback was gathered through interviews instead. In addition, a survey on the suitability of the improvement practices was performed among three team members and one manager within the organisation. The received feedback for each of the ten suggested practice is compiled and presented below together with a description of each practice, and three additional practices mentioned at the workshop by the team are reported in Section 7.2.11.

7.2.1 Guest Desk for Product Owner (IP1)[6]

Practice. Provide a guest desk in the team area for non-colocated team members, in particular the product owner.

Addressed distance: Geographical, cognitive. The practice can bridge the distance by bringing the product owner physically closer to the team more often and for longer periods of time. This increased co-location can in turn bridge cognitive distance, in particular for domain knowledge, between the product owner and the development team

[5] This survey question was posed after having identified the difference between agreed requirements and documented requirements related to the reduction in project scope through the interview questions around adherence distance.

[6] Re-locating the requirements analyst to the team area would also address geographical distance. However, since this practice was already in the process of being implemented it was not included in the suggestions presented to the team although it was identified as a suitable practice.

Team response. This practice was received very positively by all roles within the team and is one of the practices the team is now aiming to adopt. The scrum master said: '*Having the product owner in our office even 1 hour per day would help a lot with communication.*' The product owner expressed that having a guest desk would enable working in between meetings rather than just waiting or spending time on walking back and forth. One team member believed that a guest desk would make the product owner feel more welcome and encourage spending more time with the team, thus making this important role more available to the team. In addition, another team member believed that this could lead to an increased awareness for the product owner and enable this role to have more insight into the development team and the issues they face. However, office space is limited so reserving a desk for visitors is a challenge. In the meantime, an agreement has been made that the product owner spends more time in the team area and borrows temporarily available desks.

7.2.2 Requirements Communication at all Levels & Throughout Project Life-cycle (IP2)

Practice. Establish additional and strengthen existing communication paths from the team to roles and functions currently with insufficient requirements information, e.g. towards the system test team, between tester and product owner.

Addressed distances: Organisational and cognitive. The practice can bridge organisational distance by creating short-cuts in situations where increased requirements communication may avoid later coordination problems, e.g. between product owner and tester, or between requirements analyst and team-external testers. Requirements validation can also be improved by this increased communication between requirements and testing roles. In addition, the practice can affect cognitive distance by bridging it in the short term by bringing together roles with different knowledge and perspectives, and decreasing it in the long term by sharing this knowledge with others.

Team response. A majority of the team members were positive to this practice and the team aim to implement it. One of the survey respondents said that an increase in requirements communication would reduce the amount of unpleasant surprises that surface later on, e.g. issue reports, and also contribute to a better understanding of different viewpoints and therefor also fewer disagreements around requirements. Similarly, another survey respondent suggested that by involving the user interaction team in requirements discussion they would gain '*a better appreciation of why we ask for particular things and why we think they are important*', thus contribute to decreasing the amount of disagreements to roles to which the team has an organisational distance.

The developers commented that it may be possible to have more frequent demonstrations and communication around requirements of on-going work throughout the sprints. However, they believed that the product owner's limited availability and physical presence in the team area would restrict the frequency of this practice.

One participant reflected that the communication between the tester and the product owner could potentially be increased. For example, a previous tester had been very interactive with the product owner, e.g. showing mock-ups and discussing requirements details.

Several participants described that as a team they have developed a strategy for avoiding potential conflicts with other functions and units by applying a similar approach to the suggested practices, i.e. through direct and frequent communication with people from these other organisational units.

One survey respondent commented on the fact that with the previous document- and phase-based process requirements were primarily communicated through extensive requirements documentation, which thus was believed to bridge distance between the development teams and the system-level testing. However, in the previous development process the testing was also primarily performed by the requirements analysts, i.e. the distance between requirements and testing roles would have been very short.

7.2.3 Test Cases Reviewed Against Requirements (IP3)

Practice. Let someone other than the tester, e.g. the requirements analyst, look at the test cases and consider if they cover and correspond to the requirements in an adequate way.

Addressed distances: Semantic, cognitive, and organisational. This practice primarily reduces semantic distance between artefacts and cognitive distance between the roles responsible for those artefacts, and may also bridge organisational distance between these roles. The semantic distance is reduced by identifying and remedying the causes of the gaps through the review and subsequent updates. The cognitive distance between the reviewers is decreased by the information shared during the review. Potential organisational distance between the involved roles (which for this case is not an issue) can be bridged by the communication channel set up by the review practice.

Team response. There was a mixed response to this practice. Two of the survey respondents stated that this was a practice to adopt. Another respondent was less definite and said that the practice might be useful. In contrast, the tester expressed no opinion on this practice. One team member said that having more people look at the test cases would likely result in improving them, while another said that the practice would increase the sharing of knowledge concerning test cases. Furthermore, one survey respondent believed that this practice would support the requirements analysts in writing clearer and better acceptance criteria.

7.2.4 Let People Have a Say in Seating Arrangements (IP4)

Practice. Let personal preferences regarding ease of communication be one factor when considering team seating.

Addressed distance: Psychological. This practice can bridge distance by allowing people some control over their communication with others.

Team response. The scrum master indicated that psychological distance could be one factor among many to consider when deciding on how to locate different team members. However, in general the response on this practice was that it is hard to accommodate since office space is limited and opinions vary. This practice triggered another team member to initiate a separate discussion on whether seating people next to each other between whom there is a long psychological distance might decrease the distance, or alternatively decrease the communication.

7.2.5 Product Owner Testing (IP5)

Practice. The product owner (or the requirements analyst) performs user testing with the intention of validating that the implemented behaviour and performance aligns with overall system intentions and user expectations.

Addressed distance: Cognitive (domain aspect) and adherence. Differences in domain knowledge between the one performing the product owner testing and the tester in the development team can be bridged by utilising their additional knowledge of the user requirements and the domain. The requirements validation supported by this practice can thus decrease the adherence distance between the agreed requirements and the implemented software behaviour by identifying and addressing mis-alignment within the development team.

Team response. This practice was already applied by the product owner and the requirements analyst, who both indicated that they would consider applying this practice more often. Similarly, the scrum master stated that it is a practice that the team will applied more often in the future. Applying this practice had been experienced to uncover issues related to missed or misunderstood requirements details. As one survey respondents said this practice could 'help to highlight problems with misunderstandings and wrong assumptions earlier in the process, and help the [requirements] analyst feel closer to the working software.' Other team members agreed to the benefits of this practice in uncovering user-related issues. However, technical limitations were also mentioned in that the product owner cannot access the software for the sprint until after it has been delivered, thus only allowing the product owner to apply this practice after the team has delivered. The requirements analyst however was observed to perform this testing during a sprint, thereby identifying a number of missed requirements.

7.2.6 Continuous Competence Development (IP6)

Practice. Increase team member's technical knowledge through personal study, training courses etc. within specific areas, e.g. testing.

Addressed distance: Cognitive (Technical skill). Decrease[7] cognitive distance by increasing the skill level of individual members.

[7] May also increase the distance when individuals gain knowledge beyond that of other team members.

Team response. There was agreement in principle to this practice. Although competence development is encouraged in general within the organisation, personal development is up to the individual. One survey participant expressed that increased competence in teams and individuals would increase their ability to adapt and deal with challenging situations. However, one team member commented on that the majority of the team members were short-term contractors and indicated that competence development was mainly considered for permanent staff.

7.2.7 Job Rotation (IP7)

Practice. Rotate team members to different roles and responsibilities, e.g. team tester to system test team, requirements analyst to testing.

Addressed distance: Cognitive (technical skill, organisational & process knowledge). This practice primarily addresses the organisational & process knowledge aspect of the cognitive distance by increasing a person's knowledge of a new role. In addition, this person can gain technical skills by performing another job, thus also decreasing the distance for the technical skill aspect.

Team response. The team response to this practice was that it would be challenging to apply even if gains had been observed when it was applied in an ad hoc fashion, e.g. when a team tester had been transferred to the system test team. One survey respondent stated that the practice would incur additional costs in the form of temporary productivity drop and increased training needs. In particular, several practitioners mentioned that it would be hard to handle the loss of competence caused by rotating a team member. Furthermore, as indicated by another team member, rotating to a different role may not be in-line with personal preferences. As expressed by one survey respondent, the practice would cause 'improved general knowledge of different areas, but at a cost of less specific knowledge.' However, an interviewee described that it was not uncommon for people to be moved between teams and roles as need arose. In particular, since most of the requirements analysts also have system testing experience it had been discussed within the organisation to have the analysts take on testing roles.

7.2.8 Consider Quality Upfront in Requirements Elicitation (IP8)

Practice. Consider quality characteristics already in the requirements elicitation and identify important quality aspects in the early requirements discussions. These are then detailed in the same way as other requirements.

Addressed distance: Cognitive (priority of quality aspects). Cognitive distance concerning the priority of quality aspects can be decreased or at least bridged by discussing and sharing different perspectives on their relative important.

Team response. Several team members stated that this is a practice that they plan to adopt since it will help identify quality requirements early on. Thus the practice can reduce the amount of issues discovered in later activities, e.g. systems integration and production, and thereby avoid costly and time consuming maintenance. As one survey participant said: the practice 'might catch particular

issues earlier when they are easier to address.' However, some team members were doubtful whether quality aspects could be elicited upfront since there might then not be sufficient awareness of the customers' expectations for these aspects.

7.2.9 Agree on Quality Priority for Project (IP9)

Practice. Discuss which quality aspects are more or less important for the project and establish a common view of relevant quality characteristics within the team.

Addressed distance: Cognitive (priority of quality aspects). Agreeing to a set of quality aspects for the project can decrease the priority aspect of cognitive distance between team members by having shared and aligned their various viewpoints.

Team response. This practice was stated by three of the survey participants as one to adopt. One of them said that the practice 'could help us come to a more common understanding of where we should be focusing our efforts'.

7.2.10 Use Checklist of Quality Characteristics for Testing (IP10)

Practice. Use a set of quality characteristics as a check list during test planning and test design.

Addressed distance: Adherence. This practice can decrease adherence distance between agreed requirements and software behaviour by identifying potential quality issues already through testing within the development team.

Team response. The tester and two more survey participants clearly expressed that this was a practice to adopt. Together with the practice of defining quality requirements upfront (IP8) this practice was believed to enable the team to uncover quality issues early on, thus avoiding costly maintenance by aligning requirements with the testing performed within the team.

7.2.11 Additional Practices Suggested by the Team

In addition to reflecting on the presented practices, the participants of the gap workshop suggested the following practices for mitigating the found gaps:

- The scrum master suggested that misunderstandings of requirements caused, e.g. by organisational distance could be decreased by *improving the acceptance criteria so that they become more like acceptance test cases (+IP11).* Thus, decreasing the semantic distance between the requirements artefacts and the test cases.
- One developer suggested that geographical and psychological distance could be further shortened by *re-organising the team area (+IP12)* thereby further improving on communication including requirements clarification and detailing. For example, removing dividing screens, placing desks facing each other rather than back-to-back would further facilitate visual contact and awareness. Furthermore, this team member mentioned implementing additional communication practices, e.g. always face the person you are talking to, listen when others are talking.

- One developer suggested that cognitive distance concerning technical skill could be decreased by *ensuring that all team members can and know how to access each other's artefacts (+IP13).* For example, no other team members currently have any knowledge of the test cases and cannot access or view them. Similarly, there is a lack of access to other artefacts produced by previous team members, e.g. requirements documents for previous sprints.

7.3 Feedback on Gap Finder

Throughout the study feedback was gathered from the team members concerning their experience of the Gap Finder method, both concerning its approach and output, and the time and effort required of them to participate in the assessment. The main feedback data was gathered at the gap workshop, but also as part of the interviews and through the observations.

At the gap workshop, the team members expressed that they found the approach of the Gap Finder method useful in discussing issues and in identifying new areas for improvement, and that the suggested practices were appropriate. One workshop participant stated that the Gap Finder approach unearthed new perspectives, e.g. concerning the psychological distance, which had then enabled team reflection on previously un-discussed issues. In addition, even though several of the suggested practices were not completely new to the team (e.g. guest desk, product owner testing), presenting and motivating them in light of the concept of distance provided additional motivation for deciding to implement them.

Furthermore, the gap workshop (including the follow-on interviews) revealed that the product owner (who usually does not attend the team retrospectives) was in fact more positive towards adopting some of the practices than the rest of the team thought. In particular, this was the case for IP1 *Guest desk for product owner*. In addition, some of the suggested practices were already applied by the product owner although the team was not aware of this, e.g. IP5 *Product owner testing*.

The development team did not find the assessment particularly costly from their perspective. Even though they had a high work load they found time for taking the surveys, which could each be done in 10-15 minutes. When asked, the scrum master also expressed that the team had not perceived any undue cost of participating in the evaluation study. The most time consuming part from their perspective was the gap workshop, which took just over 60 minutes. However, the researchers' experience from the gap workshop was that presenting and reflecting on all the distance types in a satisfactory way required longer time than was available and that this required the participants to take in too much data at once.

The survey questions were well understood by the participants and required no major clarifications after the measurement scales and the question on priority of quality characteristics (part of the profile survey) had been explained. Minor clarifications were asked for. In particular for the questions on technical skills, for which some participants indicated a difficulty in separating between technical knowledge for an area associated with a role (e.g. RE for the requirements analyst) and knowledge of the processes and practices for that role in the case context.

7.4 Validity and Limitations

In this section, we discuss the limitations of the results including threats to validity according to guidelines provided by Runeson et al. (2012). Steps taken to mitigate these limitations and threats are also mentioned.

7.4.1 Construct Validity

There are two main risks to construct validity in this study, namely the appropriateness of the underlying theoretical framework in relationship to the specific case to which the method was applied and the precision of the distance measures. The risk concerning the underlying framework, i.e. the Gap Model, springs from that fact that the Gap Finder method is based on empirical data from Company A in the RET alignment study (Bjarnason 2013b). Although both studies focus on RET alignment there is a risk that due to differences in case characteristics the empirical knowledge gained from Company A is not applicable to the organisation in this study. In order to allow assessing this risk by comparing the cases, the case used in this study is reported using the same kind of character attributes as the ones that were reported in the previous study (see Section 5), e.g. size of development organisation, project size and length, number of requirements etc. Furthermore, this risk to construct validity was partly mitigated in the design of the gap model by taking these case characteristics into account when selecting one of the six companies from the previous study as a starting point for the theory construction. Similarly to the case in this evaluation, Company A is of about the same size, applies an iterative agile development model for which the development is not distributed, and which does not develop safety critical software. The characteristics which differ are mainly a difference in requirements source (market driven for Company A vs. bespoke for this case) and the use of product lines and open source software for Company A. Based on the insight gained from the previous RET study, no specific impact on RET alignment is known for these factors although the use of product line practices was suggested as supporting alignment.

The risk concerning distance measure precision concerns how well the measurement instrument assesses the distance it is intended to measure. To mitigate this risk, the measurement instrument was designed in an iterative fashion based on empirical knowledge from previous studies, in combination with insight into the assessed case. Despite this the construct validity of the measures requires further research to assess and improve on their precision. However, the main aim of this study was to perform a qualitative evaluation of the Gap method and approach, for which we judge that slight imprecisions of the measurements have a limited impact.

7.4.2 Internal Validity

The main threat to internal validity is the risk of incorrectly gauging the impact of certain factors or missing other impacting factors, and of misinterpreting survey questions. This is particularly relevant in this study where we aim to investigate the causal relationships between RE distances and RET practices in a real live context where there are multiple uncontrollable factors. This risk has been partly mitigated by deciding to study one development team during a specific period of

development. Distances for a defined set of requirements and test cases, and group of people were investigated, thus enabling studying the specifics of how these people relate to each other and how they work with the specific requirements at hand. However, it remains an open risk that study participants and/or researchers have incorrectly identified factors involved in causal relationships, e.g. concerning the effect of an RE distance, and in particular that other relevant factors may have been missed.

Furthermore, there is a potential risk of participants misunderstandings the survey questions used to assess the distances. This risk was partly mitigated by obtaining knowledge and insight of the case, e.g. through document studies, observations and interviews, before tailoring the questions to the specific case. The questions were thus adapted to the terminology used by the studied organisation including the specific roles and artefacts used. In addition, the survey questions were reviewed and discussed with the researcher with more familiarity with the organisation (i.e. Sharp). Furthermore, triangulation was applied to the obtained distance measures by administering the surveys as part of an interview where misunderstandings could then be discovered and resolved, and by comparing with data gathered through observations.

7.4.3 External Validity

The question of external validity concerns the extent to which the results of this study are applicable and of interest beyond that of the studied case, for which analytical generalisation needs to considered. This study is based on the Gap Model which in turn is based on empirical data from a previous study of another case, and no conflicting findings have been found in this study indicating that the two cases are comparable when considering RET alignment. For this reason the results are of interest to cases displaying the characteristics common to the two case on which both of these results are based, i.e. small and medium sized companies (150-200 people) and projects (10-20 people), with an agile and iterative development model, and for which there are no safety-critical aspects of the software development. However, even within this set of cases generalizability needs to be considered on a case-by-case basis by comparing the specifics of a case alongside the full set of characteristics reported for this case (see Section 5).

For cases that apply a non-agile development model with a strong focus on artefacts as the primary channel for requirements communication the iRE profile is most likely very different from the one obtained for this case. Subsequently the suggested practices might not have the same impact as for an agile project. However, results from previous studies show that even for a document-based process the degree of collaboration and thus distance between roles and individuals has a large impact on the collaboration between RE and later development activities, and thus on the project outcome. For this reason, the people distances are most likely relevant also for a traditional process model, although the iRE profile and the set of improvement practices would likely be different. Further research is required to explore and extend the Gap Finder to also cover projects with a phase-based and document-based process.

The general applicability of the Gap Finder for different cases and contexts requires extending the underlying theoretical framework, i.e. the Gap Model.

Initially knowledge available in the Gap Model is limited to the case context on which it is based, currently one company. Over time further knowledge will be obtained and the Gap Model extended. This includes knowledge of what comprises a troublesome gap in an iRE profile, as well as more fine-grained rules concerning contextual factors that impact how a practice affects a distance.

7.4.4 Reliability

There is a risk that researcher biases have influenced the application and evaluation of the Gap Finder method and, thus the reliability of the results of this study. This risk was mitigated by including the perspectives of multiple researchers on design aspects at several points throughout the study and by applying triangulation to the collected data. For example, the research design and the tailored survey instruments were iteratively reviewed and refined by the researchers. Triangulation of the obtained distance measures was done by also collecting data on each distance through observations and interviews. Finally, the outcome of the Gap Finder was presented to and discussed with the development team in order to validate the found distances and suitability of the suggested practices.

8 Findings and Discussions

Through studying the application and outcome of the Gap Finder a number of new insights were gained concerning how Gap Finder can support software process improvement for RET alignment and how this support can be further improved. The development team expressed that the approach was very helpful by providing them with new perspectives and improvement suggestions, some of which have now been implemented. Furthermore, the study provided experience of applying the method and further insight into the impact of RE distances on RET alignment. Based on the collected empirical data, the four research questions can now be answered concerning how Gap Finder can address RET alignment and how the method can be improved (Section 8.4). Based on the findings (see Section 7), the relevance of the distances and practices is discussed in Sections 8.1-8.2, and how Gap Finder supports team reflection in Section 8.3. The main improvements of the method are concluded and discussed in Section 8.4.

8.1 Relevance of Included Distances (RQ1)

All of the six types of RE distances included in the applied version of the Gap Finder were found to be relevant to RET alignment, while no answer can be given for navigational (D7) and temporal (D8) distance that were not included in the evaluation. The development team had experienced the impact that the people-related distances can have on RET alignment primarily on requirements communication. However, for most of the distance types the team had previously not considered the distance as such, merely observed its effects. In particular this was the case for organisational distance, psychological distance and the priority aspect of cognitive distance. Furthermore, the artefact-related distance types, e.g.

adherence, were found to be indicators of weak or strong alignment and selected project model.

The measured aspects were found to be relevant to RET alignment with the exception of local organisational & process knowledge (M4.3.1) for which no clear impact was identified. Two new aspects of distance were identified as potentially relevant to RET alignment. Table 6 outlines the found impact on RET alignment including detected gaps and the team's awareness of each distance and its impact.

Table 6. Summary of relevance of RE distances to RET alignment (RQ1). Detected gaps and the team's previous awareness of distance and impact for the studied case. + denotes additional (new) aspects of distance.

RE distance and impact on RET alignment *Impact on RET alignment*	*Gap (G), awareness of distance (D) and impact (I)*
D1 Geographical	**GDI**
Delays and misunderstandings in requirements communication and coordination with the distant team member	
D2 Organisational	**GI**
Difficulties and delays in decision making concerning disagreements on which requirements to support	
D3 Psychological	**GI**
Conflicts and difficulties in agreeing, e.g. when discussing requirements details	
D4 Cognitive	**GD (domain) I (all)**
Knowledge of - *Domain*: missed communication of tacit requirements leading to identifying missing functionality at a late stage - *Technical skill*: a) for *testing* and *development skills* a short distance supports good requirements communication towards developers and testers, and facilitates user-level testing, b) for scope management and requirements engineering: general impact on communication - *Organisational and process*: a) for *role of others* this distance can cause misalignment of system-level testing relative requirements delivered by team, b) for *own role* no direct impact was found + *Agreed requirements:* Gaps concerning this (new) aspects between system testers and development team was suggested to lead to system-level testing of other non-agreed requirements with subsequent increase in potentially unnecessary issue reports and management of these	
Priorities for product Missing quality requirements with subsequent misalignment of user expectations vs. quality level in delivered software, may surface in system-level testing	
D5 Adherence	**I**
Delivered vs. agreed requirements A long distance is a sign of misalignment in the development flow including missing or misunderstood communication of requirements and that the testing effort has failed to catch discrepancies between agreed and delivered requirements + Abstraction: a long distance may motivate developers and testers roles in validating requirements by providing more freedom and responsibility to detail them, which requires domain knowledge and insight into user expectations	

Agreed vs. documented requirements - Similarity and coverage: a long distance can indicate either misalignment caused by missing or misunderstood requirements communication, or that the documentation is not updated - Abstraction: the distance characterises the development model and in particular the weight given to requirements documentation, i.e. degree of documentation-based communication	
D6 Semantic	**I**
- Similarity and coverage: a long distance can indicate that requirements are not fully updated or a misalignment in requirements communication towards testing - Abstraction: a characteristics of the development model, i.e. degree of requirements detail and documentation upfront or concurrently w implementation	

8.1.1 RE Distance Impact on Requirements Communication

All of the people-related distances were seen by the team members as having an impact on the communication of requirements within the team and towards other roles, e.g. in system- and user-testing. Long people-related distances were seen as increasing the risk of misinterpreting and missing requirements. For example, cognitive gaps concerning domain knowledge had on several occasions caused lack of communication of requirements which were tacit to requirements analysts with long experience, but not to developers and testers who had joined more recently. Similarly, a cognitive gap in the aspect of *priorities for the product* (M4.4) between individual team members and roles was also seen to have contributed to missed quality requirements. Other negative effects of people-related distance include delays and inefficiency in decision making when there are disagreements concerning which requirements to implement. In particular, these delays were experienced in relation to the organisational distance between the product owner and the rest of the development team and whenever possible the team tries to resolve issues internally rather than escalate them to their managers. In addition, psychological distance was believed to explain some difficulties in communicating and agreeing with individual team members. Furthermore, long geographical distance between the product owner and the developers and testers were experienced to cause delays in clarifying requirements and impeded coordination within the team. This physical distance has now been shortened and the team describes an increased frequency of communication with the product owner, which they believe will contribute to reducing the amount of misunderstandings and misalignment towards the users' needs and expectations. Finally, a short distance in cognitive distance was described as supporting a smoother communication and agreement on requirements details. In particular, a short distance from the requirements analyst and the product owner towards the tester concerning knowledge of testing was described as beneficial since this testing knowledge supports, e.g. the requirements analyst in adapting the requirements information to the testers needs.

8.1.2 RE Distance as Indicators

While many of the RE distances were described by the team members as contributing to alignment the artefact-related distances were mainly found to be

indicators, e.g. of weak or strong alignment. For example, an adherence distance between delivered vs. agreed requirements indicates that the testing effort has failed to catch the discrepancy between what has been agreed and what is delivered. This measure thus indicates how well the requirements are aligned with the testing, i.e. the degree of RET alignment for a project.

Similarly for the aspect of abstraction for adherence and semantic distance (M5.2.3, M6.3), a distance was measured that was not judged to be a gap, but rather a characteristic, in this case of the applied development model. As one of the respondents pointed out, the distance in abstraction level between agreed and documented requirements can be expected to be longer for an agile development project than for a project applying a traditional document-based process. Subsequently the impact on RET alignment for this adherence distance (M5.2) will vary depending on how much weight is given to the requirements artefacts. If the requirements artefact is the main communication channel for requirements (as in a traditional process) a short distance is required to achieve good RET alignment. However, a longer adherence distance may be manageable while retaining good RET alignment if this distance is bridged by alternative (non-document based) practices for communicating requirements, e.g. by involving developers and testers in the requirements detailing.

A third kind of indicator was detected concerning adherence distance for the aspects of similarity and coverage, namely as an indicator of un-updated artefacts. During the interviews held in connection with the artefact survey two of the interviewees mentioned that some parts of the documented requirements were no longer applicable due to changes to the project scope and that these then contributed to some distance in similarity and coverage both for adherence distance between agreed and documented requirements (M5.2.1 and M5.2.2) and for semantic distance between documented requirements and test cases (M6.1 and M6.2). However, for this case the documented requirements merely support the primary communication channel, i.e. face-to-face communication, and the ultimate set of requirements is what is found in the software accepted and signed off by the product owner. In contrast, for a traditional development model where the requirements artefact is the main source of requirements information incorrectness in the requirements artefacts, indicated by adherence distance, is more likely to lead to the requirements being miscommunicated to the developers and testers, thus also affecting the implemented software.

8.1.3 Suggestions for Additional Aspects of Distance

In addition to the measured distances, two other aspects were identified through this study, namely the knowledge of current requirements as an aspect of cognitive distance (D4) and the abstraction level of agreed requirements vs. delivered software as an aspect of adherence distance (D5.1). The additional aspect of cognitive distance concerns knowledge of what functionality and behaviour the delivered software is intended to support. In particular, when there is a long distance for this aspect between the development team and testers performing system-level testing this has a negative effect on RET alignment and can result in unnecessary issue reports and subsequent management of these. This aspect is also relevant to consider between the roles within the team. Good RET alignment can be

expected when there is a short distance in knowledge of the agreed requirements between all members of the team, e.g. between requirements analyst and developers, between requirements analyst and tester.

Distance in *knowledge of organisation and process* (M4.3) concerning the roles of others (M4.3.2) may have an impact on the alignment between the development team and the testers performing the system-level testing. Together with organisational distance, these distances affect the requirements communication between these teams. The team were concerned that there might be misalignments towards the system testing concerning the understanding of which requirements that had been agreed and were implemented.

The other new aspect of distance mentioned by the requirements analyst and one of the developers is the distance in abstraction level between agreed and delivered requirements (+M5.1.3). A longer distance for this aspect could encourage testers and developers to take more responsibility for the validity of requirements by providing them with more freedom to detail the requirements. RET alignment could thus be supported by validating and verifying the requirements from a wider perspective. However, this requires good domain knowledge and insight user expectations and business strategies on the part of those detailing the requirements.

8.2 Relevance of Suggested Practices (RQ2)

The development team described that eight of the ten suggested practices could directly address and improve on RET alignment for their project, however the cost-benefit balance varies between practices and thus also the feasibility of implementing them. The practices for which no direct impact on RET alignment was identified are IP4 concerning team seating and IP6 concerning competence development. Both of these practices were believed to have a general impact on communication vs. ability within the team, and thereby indirectly improve on RET alignment. The development team have indicated that they plan to implement four of the practices, i.e. IP1, IP2, IP5 and IP8, while three of them might be implemented, i.e. IP3, IP9 and IP10, and three are judged as infeasible to implement, i.e. IP4, IP6 and IP7. The relevance and suitability of each practice is discussed below, and an overview is given in Table 7.

Table 7. Overview of findings on relevance of suggested practices including addressed distance, impact on RET alignment, and if team plans to implement the practice.

Practice	Addressed distance(s)	Impact on RET alignment	Planned for implementation
IP1 Guest desk	Geographical, cognitive	Improved requirements communication and validation	Yes, new
IP2 Requirements communication, all levels & throughout	Organisational, cognitive	Improved requirements communication and validation, decrease reqs conflicts and no of defects	Yes, new + strengthen existing practice
IP3 Test cases reviewed against requirements	Semantic, cognitive, organisational	Ensure that reqs and test case artefacts are in synch, agreement of requirements	Maybe, if test department agree
IP4 Individual say in team seating	Psychological	None found. General impact on communication	No, space limitations
IP5 Product owner testing	Cognitive: domain, adherence	Improved verification of requirements	Yes, strengthen existing practice
IP6 Competence development	Cognitive: technical skill	None found. General impact on abilities.	No, infeasible in practice
IP7 Job rotation	Cognitive	Improve alignment with system-test team by increasing their knowledge of agreed requirements	No, infeasible to apply systematically
IP8 Consider quality in elicitation	Cognitive	Early alignment on quality requirements can reduce the amount of late issues	Yes, new
IP9 Agee on quality priority for project	Cognitive	Increase agreement within team and organisation on quality requirements	Maybe, if rest of project agree
IP10 Quality checklist for testing	Cognitive, Adherence	Increase amount of quality issues caught by testing early on	Maybe, if test department agree

8.2.1 Guest Desk for Product Owner (IP1)

This practice was immediately picked up by the team members who were aware of the negative effects that the geographical distance was having on the requirements communication and were eager to improve on this. Even though this practice had been considered by the product owner it had not previously been discussed within the team. The practice can increase the frequency of direct communication between the customer representative (i.e. the product owner) and the development team, and thereby enable a quicker resolution of queries and problems related to requirements. In addition, the increased presence of the product owner in the team area can extend this role's awareness of current project issues and contribute to a closer

collaboration with the development team. Furthermore, this increased awareness and closer collaboration can improve the validation of requirements.

8.2.2 Requirements Communication at all Levels & Throughout Project Life-Cycle (IP2)

Although the importance of frequent requirements communication was already emphasised in the team, they stated that strengthening and extending this practice would further improve on the alignment between requirements and testing. Strengthening the communication within the team, in particular between the product owner and the tester, would bridge difference in domain knowledge and further ensure that the right requirements were implemented and tested in the right way. Furthermore, establishing new communication paths between the team and roles outside of the team were mentioned as decreasing disagreements and the number of reported system-level issues. In particular, cognitive and organisational distance between the team and the system testers could be bridged and to some extent decreased by establishing direct communication between the development team and these roles.

8.2.3 Test Cases Reviewed Against Requirements (IP3)

The team response to implementing this practice was mixed even though the practitioners confirmed that it could have an impact on RET alignment. The decision on implementing this practice lies with the testing department who is responsible for the testing processes and practices, in general. However, a review of the test cases can improve on their quality and align them further with both the documented and the agreed requirements, thus decreasing semantic distance. Furthermore, the practice may lead to decreasing cognitive distance. For example, if the review is performed by a requirements analyst this would further increase the understanding of what is required for writing good test cases and lead to the analyst defining clearer acceptance criteria.

8.2.4 Let People Have a Say in Seating Arrangements (IP4)

There are many factors at play for this practice and how to apply it in order to achieve an optimal set of communication paths for requirements is unclear. The team members indicated that the practice could have a general impact rather than directly affect RET alignment, and it was not seen as feasible to implement for the assessed project. For practical reasons, it is hard to accommodate personal preferences although they are considered whenever possible.

There are multiple aspects to consider in influencing the frequency of communication between different individuals. Apart from psychological distance, factors such as cognitive distance and the importance of frequent communication between different roles are likely to have an impact. However, it is unclear how to balance and optimise these different factors. For example, would requirements communication be best increased by placing the team member with the most domain knowledge in a central location, or by placing the requirements analyst there? What if the psychological distance for these people is long, would placing them in a central location shorten this distance or would it decrease the

communication? Further research is required before guidelines and recommendations can be made.

8.2.5 Product Owner Testing (IP5)

This practice was agreed as strengthening the alignment through validating that the developers and testers have correctly understood and fulfilled the customer requirements. Although the practice was already applied periodically it was agreed as a good practice to apply more frequently. When this kind of testing performed within a sprint, e.g. by the requirements analyst, issues caused by misunderstood or missing requirements are caught before they go further down the line, e.g. to system testing.

8.2.6 Continuous Competence Development (IP6)

No direct impact on RET alignment in this case was found for this practice. Competence can have a general impact on the capacity and ability of the team and of individual team members, and may thereby also affect RET alignment. Furthermore, even though the case organisation encourages competence development personal development is up to the individual and not a practice that saw seen as feasible to adopt in a systematic way.

This practice can be expected to be more suitable for organisations with a clearly defined strategy and plan for learning and competence development for its employees. For example, a competence program with defined categories and levels of competence enables an open and objective discussion on which level each person is at and what is required for different roles and position. The existence of a gap in competence for each individual, but also at the overall level, can then be identified and training programs be defined to address these.

8.2.7 Job Rotation (IP7)

Although this practice can have a positive effect on RET alignment, e.g. when rotating testers between the development team and system test team, it was also found to cause challenges. For example, the cost of losing an experienced team member was mentioned as not being worth the gains in increased insight and competence. Also, some team members indicated that rotating to a different role was not in-line with their personal preferences.

This practice is related to competence development and in particular in gaining practical experience (as opposed to merely theoretical knowledge) of other jobs and roles. Even though the practice may be hard to apply in a systematic way, unless there is a competence management program in place, managers should be aware of the potential benefits and consider this as one option when discussing personal development, but also when consider competency needs overall.

8.2.8 Consider Quality Upfront in Requirements Elicitation (IP8)

This practice can mitigate issues caused by misalignment of quality requirements between the requirements implemented by the team and the ones that the testers at user- and system-level are verifying. When considering quality aspects during the elicitation the cognitive distance between team members concerning priority of

these aspects are bridged and decreased, and the risk of missing potentially critical user requirements is reduced. This is one of the practices that the studied team have identified for implementation.

Quality requirements are a known challenge for software development, in particular for agile development, but also for traditional development models. These aspects are often seen as tacit when defining and implementing requirements, which can then result in issues with the delivered software. For example, the performance or the capacity of the system can be found to be unsatisfactory from a user perspective, i.e. issues that are often costly to address.

8.2.9 Agree on Quality Priority for Project (IP9)

The team agreed to that this practice can support improved alignment of quality requirements at the project level by agreeing to a priority between quality characteristics, i.e. defining goal-level requirements for quality. Such an agreed and clearly communicated set of priorities within the project can bridge and decrease cognitive distance for this aspect between different roles. However, the mandate for deciding to implement this practice lies with the project rather than with the development team.

RET alignment at the goal level can be further improved when this practice is combined with using the agreed quality aspects as a check list for testing (IP10, see below). The prioritised quality aspects are then considered and validated both in defining and detailing the requirements, and then verified by the testing activities.

8.2.10 Use Checklist of Quality Characteristics for Testing (IP10)

Testing according to a checklist of quality characteristics was received as a good practice for enhancing RET alignment by catching missed quality requirements through testing. This practice can bridge cognitive distance for the aspect of priorities towards the testers by providing information on which quality characteristics that are important (also see IP9 above). This increased awareness of important quality aspects can in turn have a positive effect on the requirements validation and enable locating requirements issues already during elicitation and requirements detailing. Although the team agreed to the importance of implementing this practice, the testing department also need to agree on this.

8.3 Support for Team Reflections (RQ3)

This study shows that the concept of gaps and distances provides a good metaphor for discussing RET alignment issues and practices to mitigate them within a development team. Presenting the distance types and visualising the obtained measures stimulated team discussions around alignment issues seen to be caused by these distances and practices for mitigating them. In addition, the group reflections enabled the team to identify new improvement areas.

For some distances, e.g. geographical distance, the measures confirmed a known issue and triggered action to improve on this by implementing the suggested practice. For other instances, e.g. psychological distance, the presentation of the distance and the measurements unearthed issues that had been observed individually but never discussed within the team. In this instance, presentation of

the (anonymised) measures enabled an objective discussion of what could otherwise be a sensitive topic, and both potential causes and improvements were discussed openly within the team.

Discussing the distances and suggested practices with all the roles within the team including the product owner (who did not usually attend the team retrospectives) was beneficial. In particularly, there were misconceptions around the product owner's view on certain practices to which he/she was more positive than the core team members thought. Furthermore, the product owner and the requirements analyst were already applying some of the practices sporadically although the rest of the team were unaware of this fact.

The concept of distance triggered some team members to identify further practices that could bridge or decrease various distances. This further illustrated the relevance of using distance as a metaphor when considering alignment within software development.

Finally, even though the gap workshop including the presentation of distances was found to support team reflection, the session ran out of time and the participants got tired. This was in part due to practical reasons (delayed meeting start and hot room), but possibly also due to the large amounts of data presented to them. Despite having selected a sub-set of the obtained measurements and visualised the majority of them using radar diagrams, tables and various colours, the impression was that it was hard for the participants to take in all the data, distance types and practices at once.

8.4 Improvements to the Gap Finder Method (RQ4)

One of the main outcomes of this formative evaluation is a number of potential improvements of the Gap Finder method. These can be divided into a) improving the set of measured RE distances, b) improving the identification of suitable improvement practices, c) ways to further increase the support for team reflections at the gap workshop and d) general improvements to the Gap Finder process.

8.4.1 The Set of Distances

Even though most of the applied distance measures were found to be relevant to RET alignment, some were less so and might be removed from the method. In particular this concerns the cognitive aspects of technical skill and organisational & process knowledge. Furthermore, two additional aspects were identified as relevant and useful when considering RET alignment. These are, 1) a cognitive aspect concerning the difference in knowledge of agreed requirements, e.g. between a development team and the system testers, and 2) an adherence aspect between the abstraction level of agreed requirements and the behaviour of the delivered software. Both of these are candidates for being added to the Gap Finder.

It may be possible to divide the RE distances and the various aspects of these into three categories, namely 1) indicators of weak or strong alignment, 2) factors affecting alignment or other distances, and 3) indicators of case characteristics. For example, the aspects of similarity and coverage for the semantic distance between requirements and test cases are indicators of how well the two artefacts are aligned and therefore belong to the first category. The people-related distances, e.g. cognitive and psychological, are examples of factors found to affect RET

alignment, i.e. these can be placed in the second category. The aspect of abstraction level is an example of the third category as an indicator of the selected development model (agile or traditional RE).

Finally, the use of self-assessing surveys to measure the artefact distances needs to be reconsidered since there is a high risk of bias with this approach. This is due to the fact that a long distance in similarity or coverage towards the documented and/or delivered requirements or test cases correlates with failure to capture and match the agreed requirements. The person responsible for ensuring that an artefact has sufficient similarity and coverage is likely not aware of there being a distance, and if so might be unwilling to admit that there is one. For this reason, an alternative measuring approach needs to be investigated for the adherence distance between agreed and documented requirements and the semantic distance between requirements and test artefacts.

8.4.2 Identifying Suitable Improvement Practices

Several of the practices proposed at the gap workshop were in fact already applied, although this had not been caught during the observations or the interviews. Despite this fact, suggesting existing practices also had a positive effect in that these were then discussed within the team and for some of them this meant that they were re-considered and agreed to be applied more frequently. However, the Gap Finder analysis could be improved by adding a step for identifying existing practices, e.g. through a survey, prior to gap analysis. This information can then be considered during the gap analysis, but should not exclude from suggesting existing but relevant practices. Rather for these practices, the outcome should be a suggestion to consider how to further improve on their implementation.

The suggested practices that were not seen to directly affect RET alignment, i.e. say in team seating (IP4) and competence development (IP6), and their impact on RE distance need to be investigated further. It may be the case that these practices need to be tailored further to address specific gaps. For example for competence development, if a gap for a specific competence is found this technical area and the involved individuals should be suggested for competence development. For the practice of team seating and the distance it is claimed to address, namely psychological distance, further insight is needed either from literature studies or from additional research into the impact of this practice.

8.4.3 Supporting Team Reflections

An important aspect of the gap workshop is to enable the team to jointly reflect on issues and improvement practices. The concept of distance was found to provide a good metaphor for supporting the team in discussing both known issues from a new perspective and previously un-discussed issues. However, these discussions could potentially be improved by focusing on key gaps particularly relevant to the assessed case. This would ensure a more effective use of meeting time while also reducing the set of distances and measured the participants are required to take in.

The visualisations used at the gap workshop supported the team in reflecting on the measured gaps and are an interesting area for further improvements. This includes investigating additional visualisation techniques and considering which

distance measures and combinations of measures that can further support software engineers in reflecting on factors that affect RET alignment.

Furthermore, the measures are currently presented per distance type. However, considering that several practices affect multiple distance types and that the distances affect each other, a potential improvement could be to analyse the full set of distances between each pair of entities. This may further support identifying specific improvement practices by analysing the combined set of measured distances for one relationship, e.g. between the tester and the product owner, or between the development team and the system test team. Furthermore, this could provide more specific focal points at the gap workshop, although discussing individual relationships could be sensitive even with objective data.

8.4.4 Process Improvement Through Applying Gap Finder

The main aim of the Gap Finder is to support enhanced RET alignment by providing practices that will improve on the RE integration with testing activities. This requires identifying suitable practices and supporting the team in deciding which practices to implement. For this study, the Gap Finder method was not re-iterated so the agreement on practices for implementation was gathered over a period of 4-6 weeks and mainly consisted of gathering this feedback through a survey. However, it is desirable to ensure an agreement that is committed to by the whole team and to gain this agreement in connection with the gap workshop to avoid delays in the process improvement work. This could be achieved by having two sessions of the gap workshop, i.e. one session similar to the existing one and one follow-on session with the main aim of reaching an agreement and an action plan for improving on practice. This would also allow the team members to reflect individually before meeting again to agree on which improvements to implement.

For this case three of the suggested practices were viewed by the team as improvements to implement, but the decision to adopt these practices lay outside of the team, e.g. with the line manager for the testers. This indicates that the mandate for implementing each of the practices identified through the gap analysis also needs to be considered and supported by the method. These mandates could be identified at an initial workshop session. The full set of stakeholders including the affected team members could then be invited to a follow-up workshop session. This would ensure sufficient mandate at the second gap workshop session to decide on which practices to implement and agree to an improvement plan.

9 Conclusions and Future Work

Software process improvement (SPI) aims to improve on the productivity of software development by tweaking the development process through identifying and implementing suitable improvements. While most existing SPI methods and framework focus on assessing and improving practice in general, the Gap Finder method is designed to support project teams to improve specifically on the alignment between requirements and testing. Furthermore, Gap Finder focuses on factors that can have an effect on RET alignment and proposes practices for mitigating these factors. These factors have been identified through previous

research and consist of RE distances between people and between artefacts. This approach has been evaluated through the case study reported in this paper by applying the Gap Finder method to an active development project and studying the outcome.

Through this formative evaluation three main insights have been gained concerning the relevance of Gap Finder for improving on RET alignment. Namely, concerning the relevance of the measured RE distances (RQ1), the relevance and suitability of the practices identified by applying Gap Finder (RQ2), and how the method can support team reflection in this area (RQ3).

All of the applied RE distances were found to be relevant to RET alignment, although the exact set of RE measurements needs to be reviewed. The results also indicate that there are three main categories of distances, namely distances that can a) have an effect on RET alignment, b) indicate weak or strong alignment or c) characterise the applied development process, e.g. agile or traditional RE process.

All of the improvement practices identified with the Gap Finder can support improved RET alignment, although further tailoring of some practices is needed for them to clearly address alignment specifically. In addition, the method can be further strengthened by adding a step for process improvement planning, which could include additional decision makers, e.g. responsible line managers.

The metaphor of distance used in the Gap Finder can support and stimulate team reflections on RET alignment and was found to enable the development team to identify new improvement areas. The concept of distance was found to provide the team with a new perspective and potential explanation of experienced issues. In addition, providing objective measures of distance can support an open and objective discussion, even of more sensitive subjects such as individual difficulties in communicating.

Future work includes considering the identified method improvements, as well as further exploring the visualisation of distances and iRE profiles. An interesting avenue to explore is to identify patterns in iRE profiles between related projects, e.g. projects that apply an agile development model, or distributed projects. Furthermore, the theoretical framework of the Gap Model will be improved further based on the new empirical evidence from this study on the connections between RE distances and RET alignment practices.

In conclusion, Gap Finder is found to support teams in improving on RET alignment in a novel way by providing an objective view of underlying factors, i.e. RE distances. By doing so the method allows practitioners to take a step back and consider underlying factors (as do inductive SPI methods) rather than focusing on problems with existing processes. The method combines an inductive and a prescriptive approach to SPI by first assessing the actual case and then comparing these findings in a structured way to an existing theoretical framework of best practices. In this way, the method supports identifying suitable improvements by starting at the bottom through insight into the specific project and combining this with existing empirical knowledge, i.e. *from the top*.

Acknowledgement
We would like to thank the development team members for enabling this study by sharing their time, thoughts and office space. This work was partly funded by EASE and by Ericsson Research.

References

Angermo Ringstad M, Dingsoyr T, Brede Moe N (2011) Agile Process improvement: Diagnosis and Planning to Improve Teamwork. Proc of 18th European Conf. on Systems, Software and Service Process Improvement (EuroSPI'11), Communications in Computer and Information Science Volume 172, 2011, pp. 167-178.

Barmi ZA, Ebrahimi AH, Feldt R (2011) Alignment of Requirements Specification and Testing: A Systematic Mapping Study. Proc. 4th Int. Conf. on Softw. Testing, Verification and Validation Workshops (ICSTW):476-485.

Basili VR (1985) Quantitative Evaluation of Software Methodology. Tech. report TR-1519, University of Maryland, College Park, Maryland.

Basili VR, Rombach HD (1988) The TAME Project: Towards Improvement-Oriented Software Environments, IEEE Transactions on Software Engineering, 14(6), 758–773.

Basili VR (1992) Software modelling and measurement: the Goal/Question/Metric paradigm, Technical report, University of Maryland at College Park College Park, MD, USA.

Benner P (1982) From Novice to Expert. American Journal of Nursing, 82(3), 402-407.

Bjarnason E, Wnuk K, Regnell B (2011) Requirements are Slipping Through the Gaps – A Case Study on Cause & Effects of Communication Gaps in Large-Scale Software Development. Proc. of 19th IEEE Int Requirements Engineering Conf., pp. 37-46.

Bjarnason E (2013a) Distances between Requirements Engineering and Later Software Development Activities: A Systematic Map. Proc. of Requirements Engineering for Software Quality Conference (REFSQ) 2013, pp. 292-307. 2013

Bjarnason E, Runeson P, Borg M et al. (2013b) Challenges and Practices in Aligning Requirements with Verification and Validation: A Case Study of Six Companies. Empirical Software Engineering, published on-line July 2013.

Bjarnason E (2013c) Integrated Requirements Engineering: Understanding and Bridging Gaps within Software Development, Ph. D. Thesis, November 2013.

Bjarnason E (2013d) Research material for Gap Finder evaluation study incl measurement instrument, interview guide etc. (latest access: 2013-10-22) http://serg.cs.lth.se/research/experiment_packages/GapFinder/

Boehm, BW (1981) Software Engineering Economics, Prentice Hall.

Brede Moe N, Dingsoyr T, Royrvik EA (2009) Putting Agile Teamwork to the Test – An Preliminary Instrument for Empirically Assessing and Improving Agile Software Development. Proc of XP 2009, LNBIP 31, pp. 114-123.

Briand L, El Emam K, Melo WL (1995) ANSI -- An Inductive Method for Software Process Improvement: Concrete Steps. Proc. of the ESI-ISCN'95: Measurement and Training Based Process Improvement, Sep. 11-12 1995, Vienna, Austria.

Briand LC, Labiche Y, O'Sullivan L, Sówka MM (2006) Automated impact analysis of UML models. Journal of Systems and Software 79(3), pp. 339-352.

Cataldo M, Herbsleb J, Carley K (2008) Socio-Technical Congruence: a Framework for Assessing the Impact of Technical and Work Dependencies on Software Development Productivity. Proc. of 2nd ACM-IEEE Int. Symp. on Empirical Softw. Engineering and Measurements (ESEM '08)

Chen JC, Huang SJ (2009) An empirical analysis of the impact of software development problem factors on software maintainability. Journal of Systems and Software, vol. 82, no. 6, pp. 981-992.

Chrissis MB, Konrad M, Shrum S (2007) CCMI for Development, v 1.2. Guidelines for Process Integration and Product Improvement (2nd edition), SEI Series in Software Engineering, Addison-Wesley.

Cleland-Huang J, Chang CK, Christensen M (2003) Event-Based Traceability for Managing Evolutionary Change. IEEE Transactions on Software, 29(9).

Collier B, DeMarco T, Fearey P (1996) A Defined Process for Project Postmortem Review, IEEE Software, vol. 13, issue 4, pp. 65-72.

Damian D, Chisan J, Vaidyanathasamy L, Pal Y (2005) Requirements Engineering and Downstream Software Development: Findings from a Case Study. Empirical Software Engineering, vol 10:255-283.

Damian D, Chisan J (2006) An Empirical Study of the Complex Relationship between Requirements Engineering Processes and Other Processes that Lead to Payoffs in Productivity, Quality, and Risk Management. IEEE Transactions on Software Engineering, 32(7):33 - 453.

De Lucia A, Fasano F, Oliveto R, Tortora G (2007) Recovering Traceability Links in Software Artifact Management Systems using Information Retrieval Methods. ACM Transactions on Softw. Engineering and Methodology, 16(4):Article 13.

Derby E, Larsen D (2006) Agile Retrospectives: Making Good Teams Great! Pragmatic Bookshelf, 2006.

Dias Neto AC, Arilo C, Subramanyan R, Vieira M, Travassos GH (2007) A Survey on Model-Based Testing Approaches: A Systematic Review. Proc of 1st ACM Int Workshop on Empirical Assessm. of Softw. Engineering Languages and Technologies, pp. 31-36.

Drury M, Conboy K, Power K (2011) Decision making in agile development: A Focus Group Study of Decisions and Obstacles. Proc. Of Agile Conference 2011, pp. 39-47.

Dybå T (2000) An Instrument for Measuring the Key Factors of Success in Software Process Improvement. Empirical Software Engineering, 5, pp. 357–390.

Ferguson RW, Lami G (2006) An Empirical Study on the Relationship between Defective Requirements and Test Failures. Proc of 30th Annual IEEE/NASA Software Engineering Workshop SEW-30 (SEW'06).

George M (2002) Lean Six Sigma: Combining Six Sigma Quality with Lean Production Speed. McGraw-Hill.

Gotel O, Finkelstein A (1994) An Analysis of the Requirements Traceability Problem. Proc. First Int Conf. Requirements Eng., pp. 94-101.

Grieskamp W, Kicillof N, Stobie K, Braberman V (2011) Model-based quality assurance of protocol documentation: tools and methodology. Softw. Test Verification Reliability. 21(1):55–71.

Hall T, Beecham S, Rainer A (2002) Requirements problems in twelve software companies: an empirical analysis. IEEE Software 149, 2002, pp. 153- 160.

Harter DE, Kemerer CF, Slaughter SA (2012) Does Software Process Improvement Reduce the Severity of Defects? A Longitudinal Field Study. Software Engineering, IEEE Transactions on , vol.38, no.4, pp.810,827, July-Aug. 2012 doi: 10.1109/TSE.2011.63

Hasling B, Goetz H, Beetz K (2008) Model Based Testing of System Requirements using UML Use Case Models. Proc of 2008 Int. Conf. on Software Testing, Verification, and Validation.

Hayes JH, Dekhtyar A, Sundaram SK, Holbrook EA, Vadlamudi S, April A (2007) REquirements TRacing On target (RETRO): Improving Software Maintenance Through Traceability Recovery. Innovations in Systems and Software Engineering, 3(3):193-202.

Humphrey WS (1989) Managing the Software Process. SEI Series in Software Engineering, Addison-Wesley.

Humphrey W (1997) Managing Technical People: Innovation, Teamwork, and the Software Process, Addison-Wesley.

ISO/IEC (2004-2011) ISO/IEC 15504 Information Technology – Process Assessment, parts 1-10.

Jarke M (1998) Requirements Traceability. Comm. ACM, vol. 41, no. 12, pp. 32-36, Dec. 1998.

Jilani LL, Desharnais J, Mili A (2001) Defining and Applying Measures of Distance Between Specifications. Journ. IEEE Transactions on Softw. Eng, 27(8), pp. 673-703.

Kandt RK (2009) Experiences in Improving Flight Software Development Processes. Software, IEEE, vol. 26, no. 3, pp. 58-64.

Kellner MI, Madachy RJ, Raffo DM (1999) Software Process Simulation Modeling: Why? What? How? Journal of Systems and Software, Volume 46, Issues 2–3, 15 April 1999, pp. 91-105.

Kukkanen J, Vakevainen K, Kauppinen M, Uusitalo E (2009) Applying a Systematic Approach to Link Requirements and Testing: A Case Study, Proc of Asia-Pacific Software Engineering Conference (APSEC '09):482 – 488.

Lavallée M, Robillard PN (2012) The Impacts of Software Process Improvement on Developers: A Systematic Review. Proc of 34th Int. Conf. on Software Engineering (ICSE), pp.113-122. doi: 10.1109/ICSE.2012.6227201

Lawson M, Karandikar HM (1994) A Survey of Concurrent Engineering. Concurrent Engineering 1994 2:1.

Lormans M, van Deursen A, Gross H (2008) An Industrial Case Study in Reconstructing Requirements Views. Empirical Software Engineering, December 2008, Vol. 13, Issue 6, pp. 727-760.

Martin R, Melnik G (2008) Tests and Requirements, Requirements and Tests a Möbius Strip. IEEE Software, 25(1):54-59.

Melnik G, Maurer F, Chiasson M (2006) Executable Acceptance Tests for Communicating Business Requirements: Customer Perspective. Proc. of Agile Conference, Minneapolis, USA, pp. 12-46.

Mohagheghi P, Dehlen V (2008) Where is the Proof?-A Review of Experiences from Applying MDE in Industry. Proc. of Model Driven Architecture–Foundations and Applications, pp. 432-443.

Nebut C, Fleurey F, Traon YL, Jézéquel J (2006) Automatic Test Generation: A Use Case Driven Approach. IEEE Trans. on Softw. Engineering, 32(3):140-155.

Lubars M, Potts C, Richter C (1993) A Review of the State of the Practice in Requirements Modelling. Proc. of 1st IEEE Int. Symposium on Requirements Engineering, pp. 2–14.

O'Beirne P (1997) Personal Software Process - does the PSP deliver its promise? INPIRE II, Process Improvement Training and Teaching for the Future. The British Computer Society.

Pettersson F, Ivarsson M, Gorschek T (2008) A Practitioner's Guide to Light Weight Software Process Assessment and Improvement Planning. Journal of Systems and Software 81(6):972-995

Post H, Sinz C, Merz F, Gorges T, Kropf T (2009) Linking Functional Requirements and Software Verification. Proceedings of 17th IEEE International Requirements Engineering Conference, pp. 295-302.

Ramesh B, Stubbs C, Powers T, Edwards M (1997) Requirements traceability: Theory and practice. Annals of Software Engineering, 3(1):397-415.

Ramesh B (1998) Factors Influencing Requirements Traceability Practice. Communications of the ACM CACM Homepage archive, 41(12):37-44.

Ramesh B, Cao L, Baskerville R (2010) Agile Requirements Engineering Practices and Challenges: An Empirical Study. Information Systems Journal, Volume 20, Issue 5, pages 449–480, September 2010.

Randell B (1969) Towards a Methodology of Computing System Design. NATO Working Conference on Software Engineering 1968, Report on a Conference Sponsored by NATO Scientific Committee, Garmisch, Germany, pp. 204-208.

Regnell B, Runeson P (1998) Combining Scenario-based Requirements with Static Verification and Dynamic Testing. Proc. 4th Int. Working Conf. Requirements Engineering: Foundation for Software Quality, pp.195–206.

Regnell B, Runeson P, Wohlin C (2000) Towards integration of use case modelling and usage-based testing. Journal of Systems and Softw. 50(2):117–130.

Robinson H, Segal J, Sharp H (2007) Ethnographically-Informed Empirical Studies of Software Practice. Information and Software Technology, 49, pp. 540-551.

Robson C (2002) Real World Research. 2nd ed. Blackwell Publishing.

Rogers Y, Sharp H, Preece J (2011) Interaction Design: Beyond Human - Computer Interaction, 3rd Edition. Wiley.

Runeson P, Höst M, Rainer A, Regnell B (2012). Case Study Research in Software Engineering – Guidelines and Examples. Wiley.

Sabaliauskaite G, Loconsole A, Engström E, Unterkalmsteiner M, Regnell B, Runeson P, Gorschek T, Feldt R (2010) Challenges in Aligning Requirements Engineering and Verification in a Large-Scale Industrial Context. Proceedings of REFSQ 2010.

Salo O, Abrahamsson P (2007) An Iterative Improvement process for Agile Software Development. Software Process Improvement and Practice, vol.12, issue 1, pp. 81-100.

Stapel K, Knauss E, Schneider K, Zazworka N (2011) FLOW Mapping: Planning and Managing Communication in Distributed Teams. Proc of 6th IEEE Int. Conf. On Global Software Engineering (ICGSE), pp 190-199.

Unterkalmsteiner M, Feldt R, Gorschek T (2013) A Taxonomy for Requirements Engineering and Software Test Alignment. Accepted for publication in ACM Transactions on Software Engineering and Methodology.

Uusitalo EJ, Komssi M, Kauppinen M et al. (2008) Linking Requirements and Testing in Practice. 16th IEEE Int Requirements Engineering Conf, NJ, USA, pp. 265-270

Watkins R, Neal M (1994) Why and How of Requirements Tracing, IEEE Software 11(4):104-106.

Wolf T, Nguyen T, Damian D (2008) Does Distance Still Matter? Journal of Impr. and Practice of Softw. Process, 13(6), pp. 493-510

Yu ESK, Mylopoulos J (1994) Understanding "Why" in Software Process Modelling, Analysis, and Design. Proc. of 16th Int. Conf. on Software engineering (ICSE '94). IEEE Computer Society Press, Los Alamitos, CA, USA, pp. 159-168.

Yue T, Briand LC, Labiche Y (2011) A Systematic Review of Transformation Approaches Between User Requirements and Analysis Models. Requirem. Engin., 16(2), pp. 75-99.