ABSTRACT

Background: The software development environment is growing increasingly complex, with a greater diversity of stakeholders involved in product development. Moves towards global software development with onshoring, offshoring, insourcing and outsourcing have seen a range of stakeholders introduced to the software development process, each with their own incentives and understanding of their product. These differences between the stakeholders can be especially problematic with regard to aspects of software quality. The aspects are often not clearly and explicitly defined for a product, but still essential for its long-term sustainability. Research shows that software projects are more likely to succeed when the stakeholders share a common understanding of software quality.

Objectives: This thesis has two main objectives. The first is to develop a method to determine the level of alignment between stakeholders with regard to the priority given to aspects of software quality. Given the ability to understand the levels of alignment between stakeholders, the second objective is to identify factors that support and impair this alignment. Both the method and the identified factors will help software development organisations create work environments that are better able to foster a common set of priorities with respect to software quality.

Method: The primary research method employed throughout this thesis is case study research. In total, six case studies are presented, all conducted in large or multinational companies. A range of data collection techniques have been used, including questionnaires, semi-structured interviews and workshops.

Results: A method to determine the level of alignment between stakeholders on the priority given to aspects of software quality is presented—the Stakeholder Alignment Assessment Method for Software Quality (SAAM-SQ). It is developed by drawing upon a systematic literature review and the experience of conducting a related case study. The method is then refined and extended through the experience gained from its repeated application in a series of case studies. These case studies are further used to identify factors that support and impair alignment in a range of different software development contexts. The contexts studied include onshore insourcing, onshore outsourcing, offshore insourcing and offshore outsourcing.

Conclusion: SAAM-SQ is found to be robust, being successfully applied to case studies covering a range of different software development contexts. The factors identified from the case studies as supporting or impairing alignment confirm and extend research in the global software development domain.
Software Quality Alignment:
Evaluation and Understanding

Sebastian Barney
Software Quality Alignment:  
Evaluation and Understanding

Sebastian Barney

School of Computing  
Blekinge Institute of Technology  
SWEDEN
Nowadays people know the price of everything and the value of nothing.
Oscar Wilde (1854 – 1900)
Abstract

Background: The software development environment is growing increasingly complex, with a greater diversity of stakeholders involved in product development. Moves towards global software development with onshoring, offshoring, insourcing and outsourcing have seen a range of stakeholders introduced to the software development process, each with their own incentives and understanding of their product. These differences between the stakeholders can be especially problematic with regard to aspects of software quality. The aspects are often not clearly and explicitly defined for a product, but still essential for its long-term sustainability. Research shows that software projects are more likely to succeed when the stakeholders share a common understanding of software quality.

Objectives: This thesis has two main objectives. The first is to develop a method to determine the level of alignment between stakeholders with regard to the priority given to aspects of software quality. Given the ability to understand the levels of alignment between stakeholders, the second objective is to identify factors that support and impair this alignment. Both the method and the identified factors will help software development organisations create work environments that are better able to foster a common set of priorities with respect to software quality.

Method: The primary research method employed throughout this thesis is case study research. In total, six case studies are presented, all conducted in large or multinational companies. A range of data collection techniques have been used, including questionnaires, semi-structured interviews and workshops.

Results: A method to determine the level of alignment between stakeholders on the priority given to aspects of software quality is presented—the Stakeholder Alignment Assessment Method for Software Quality (SAAM-SQ). It is developed by drawing upon a systematic literature review and the experience of conducting a related case study. The method is then refined and extended through the experience gained from its repeated application in a series of case
studies. These case studies are further used to identify factors that support and impair alignment in a range of different software development contexts. The contexts studied include onshore insourcing, onshore outsourcing, offshore insourcing and offshore outsourcing.

**Conclusion:** SAAM-SQ is found to be robust, being successfully applied to case studies covering a range of different software development contexts. The factors identified from the case studies as supporting or impairing alignment confirm and extend research in the global software development domain.
Acknowledgements

First I would like to sincerely thank my supervisor, Claes Wohlin, for his invaluable feedback, expertise and advice. Despite Claes’ busy schedule, he has always made time to ensure I get the support I need.

I would like to extend this gratitude to my co-supervisor, Aybüke Aurum. Aybüke’s direction and support lead me to my doctoral studies in Sweden, she had continued to support me, and is always there to provide a different perspective on my research activities.

Recognition must be given to all of my research collaborators and the SERL group for providing support and helping me grow as a researcher. In particular I would like to acknowledge the direct contribution made to this thesis by Varun Mohankumar, Lefteris Angelis, Panagiota Chatzipetrou, Kai Petersen, my co-supervisor Mikael Svanberg, and Hamish Barney.

I am grateful to everyone who has participated in this research—filling in questionnaires, providing feedback and analysis, and putting me in contact with the right people. Special thanks must be extended to Ericsson, and in particular my mentors, for their continued support and direction of my research.

Finally I would like to thank my family for their continued support. Especially to Oziel, for all his support, time and patience.

This work was partly funded by The Knowledge Foundation in Sweden under a research grant for the Blekinge Engineering Software Qualities (BESQ) project, and Vinnova, Sony Ericsson Mobile Communications, Ericsson, ST-Ericsson, Axis Communications and Softhouse Consulting Baltic as part of the the Industrial Excellence Center for Embedded Applications Software Engineering (EASE).
Overview of Papers

Papers in this Thesis


**Chapter 9:** Sebastian Barney and Claes Wohlin, “Balancing Software Product Investments,” in proceedings of *Empirical Software Engineering and*
Measurement (ESEM), Lake Buena Vista, Florida, USA, 15–16 October 2009.

Related Papers not in this Thesis


# Contents

Abstract vii

Acknowledgements ix

Overview of Publications xi
- Papers in this Thesis xi
- Related Papers not in this Thesis xii

Contents xiii

1 Introduction 1
- 1.1 Overview 1
- 1.2 Background 2
- 1.3 Research Questions 7
- 1.4 Methodology 9
- 1.5 Research Setting 13
- 1.6 Overview of Chapters 15
- 1.7 Authors’ Contribution 24
- 1.8 Thesis Contribution 24
- 1.9 Future Work 27
- 1.10 Reading Guide 29

2 Software Quality Trade-offs 31
- 2.1 Introduction 32
- 2.2 Background 33
- 2.3 Method 35
- 2.4 Systematic Map 44
xiv CONTENTS

2.5 Systematic Review .............................................. 51
2.6 General Discussion ........................................... 64
2.7 Conclusion ..................................................... 66

3 Requirements Selection ........................................ 69
  3.1 Introduction .................................................. 70
  3.2 Background ................................................... 72
  3.3 Methodology .................................................. 77
  3.4 Data Analysis and Results ................................. 84
  3.5 Discussion of Results ....................................... 101
  3.6 Validity Threats ............................................. 106
  3.7 Conclusions and Future Study ......................... 107

4 A Method for Determining Alignment .................... 111
  4.1 Introduction .................................................. 111
  4.2 Method ....................................................... 112
  4.3 Experiences and Discussion ............................. 123
  4.4 Validity Threats ............................................. 132
  4.5 Conclusions .................................................. 133

5 Quality Alignment Inside an Organisation ............ 137
  5.1 Introduction .................................................. 138
  5.2 Background ................................................... 139
  5.3 Methodology .................................................. 144
  5.4 Case Study .................................................... 149
  5.5 Discussion .................................................... 158
  5.6 Conclusion .................................................... 159

6 Quality Alignment with Subcontracted Developers ... 163
  6.1 Introduction .................................................. 164
  6.2 Background ................................................... 165
  6.3 Methodology .................................................. 168
  6.4 Case Study .................................................... 170
  6.5 Results ........................................................ 173
  6.6 Discussion .................................................... 178
  6.7 Conclusion .................................................... 181
# CONTENTS

7 Quality Alignment with Offshored Development 183  
7.1 Introduction ............................................. 184  
7.2 Background ............................................. 185  
7.3 Methodology ............................................ 187  
7.4 Case Study .............................................. 191  
7.5 Results ................................................... 194  
7.6 Discussion .............................................. 201  
7.7 Conclusion .............................................. 204  

8 Quality Alignment in a Global Organisation 207  
8.1 Introduction ............................................. 208  
8.2 Background ............................................. 208  
8.3 Methodology ............................................ 211  
8.4 Case Study .............................................. 212  
8.5 Priorities on Quality Aspects ......................... 216  
8.6 Alignment Between Key Stakeholder Groups .......... 216  
8.7 Issues Affecting Alignment and Misalignment ....... 222  
8.8 Discussion .............................................. 225  
8.9 Conclusion .............................................. 227  

9 Balancing Software Product Investment Options 229  
9.1 Introduction ............................................. 230  
9.2 Background ............................................. 231  
9.3 Methodology ............................................ 237  
9.4 Case Study .............................................. 242  
9.5 Discussion .............................................. 248  
9.6 Validity Threats ....................................... 249  
9.7 Conclusion .............................................. 249  

Bibliography 251  

A Software Quality Model—Ericsson 265  

B Software Quality Model—Global Finance 269  

C Software Investment Model—Ericsson 273
1.1 Overview

Trade-off decisions are found everywhere. When a crayfish foraging for food is faced with danger it takes milliseconds to decide between fleeing or freezing (Liden et al., 2010). Fleeing removes the crayfish from both the danger and the food, while freezing gives the crayfish a second chance at the food if it is not attacked. In the short time it takes to make a decision, a crayfish weighs up the food on offer, the speed of the attacker and the possibility of escape. This process has evolved over millions of years of evolution. Unfortunately trade-off decisions in the software development domain are not rapid, or innate or advanced.

The context in which software development occurs is growing increasingly complex. Modern software development involves stakeholders representing a myriad of different roles (Herbsleb, 2007)—for example, product managers, project managers, architects, developers, test manager, tester and users. Further, the organisational push for faster, cheaper and better software development has seen work outsourced and insourced, offshored and onshored (Dibbern et al., 2008). The result is often a complex hybrid mix of people—onshore outsourced project management with insourced offshore developers producing software for a globally distributed user base is but one simplified example.

Each stakeholder is involved with software development in different ways. These differences give each stakeholder a conflicting understanding software quality for the product being developed and create context-based incentives
peculiar to each stakeholder’s circumstances. For example, as developers are one of the only groups that work directly with the code, they are more likely to care about issues of maintainability as they are more affected by this aspect than other groups. Ultimately these differences in roles can lead to a conflicting understanding of what is required of the product in terms of software quality.

Misalignment between critical stakeholders on issues of software quality can lead to conflict and project failure (Chan and Reich, 2007). Ultimately the stakeholders need to work together to produce one product, and a common understanding of software quality has been found to be the most effective way for stakeholders to achieve this shared goal (Phongpaibul and Boehm, 2005).

In order to support people working in software development settings, this thesis has two objectives. The first objective is create a method capable of determining the level of alignment between success-critical stakeholders in terms of the priority given to the aspects that make up software quality. The second objective is to use this method to identify factors that support and impair the alignment between stakeholders involved in the software development process.

This thesis presents the Stakeholder Alignment Assessment Method for Software Quality (SAAM-SQ), a method for determining the level of alignment between success-critical stakeholders in terms of the priority given to software quality. The method has been applied in a series of case studies, with incremental improvements being made to the method based on the experiences of its use. The case studies involve a wide range of software development settings including all combinations of onshoring/offshoring and insourcing/outsourcing. The results of these case studies identify a number of factors that positively and negatively affect alignment in the software development process.

The remainder of this chapter is structured as follows. Key concepts are presented in Section 1.2. The research questions for the thesis are posed in Section 1.3. The methodology is described in Section 1.4 and the research setting studied in Section 1.5. The research studies included as part of this thesis are detailed in Section 1.6 with the authors’ contribution defined in Section 1.7. The contribution of the thesis is detailed in Section 1.8 and future work is posed in Section 1.9. The chapter concludes with a reading guide for the remainder of the thesis in Section 1.10.

1.2 Background

In the software development process there are four variables that need to be controlled—cost, time, quality and scope (Beck, 2000). Further there is an
1.2 Background

axiom that states that external forces can set at most three of these variables with the remainder needing to be set by the development team.

Scope, time and cost can all operate within acceptable ranges, but quality is a terrible control variable as it only allows very short term gains at a very high cost to all parties involved (Beck, 2000). That said, quality does not need to be perfect (Yourdon, 1995) but the development process is much simpler when the success-critical stakeholders agree on what action should be taken (Boehm and Ross, 1989).

This section examines different perspectives of quality and processes to manage quality in a software engineering context.

1.2.1 Software Quality

The definitions of quality are both many and conflicting, even when only examining the topic in relation to software engineering. Looking across different disciplines it is possible to see a complex multifaceted concept of quality that can be described from five different perspectives (Garvin, 1984):

- The transcendental perspective defines quality as something that can be recognized but not defined in advance.
- The user perspective defines quality as fit for purpose.
- The manufacturing perspective defines quality as conformance to specification.
- The product view defines quality in terms of essential characteristics of the product in question.
- The value-based view defines quality in terms of the amount a customer is willing to pay for it.

By far the most common perspectives taken in the software development industry are that of the user and manufacturer (Hoyer and Hoyer, 2001; Kitchenham and Pfleeger, 1996). However, there is an increasing body of literature that recognises the importance of taking advantage of all of the perspectives involved in software development. Theory-W states that success requires all of the success-critical stakeholders to compromise (Boehm and Ross, 1989), while requirements specification reading techniques that take advantage of different perspectives have been found to catch 35% more defects than non-directed alternatives (Basili, 1997; Boehm and Basili, 2001). Value-based software engineering
now recognises the benefit brought by involving different perspectives into the development process (Biffl et al., 2006).

Software quality is not only defined from the relevant perspectives, but also by the context in which it exists (Kitchenham and Pfleeger, 1996). Just as each line of cars has a target market, software quality must be planned to allow a development company to meet its business objectives. Less than perfect software quality can in fact be ideal (Yourdon, 1995), but how much less than perfect can only be decided in a given business context (Kitchenham and Pfleeger, 1996).

1.2.2 Quality Models for Software Development

Numerous models have been developed to support software quality. Examples of these models include McCall’s quality model, Boehm’s quality model, Dromey’s quality model and ISO 9126.

McCall’s quality model is the first of the modern software product quality models (Kitchenham and Pfleeger, 1996). The model uses a hierarchy of factors, criteria and metrics to address internal and external product quality. Eleven factors define an external or user perspective of quality. Each of these factors is linked to between two and five of 23 criteria that define an internal or development perspective of quality. Further metrics are associated with the factors allowing quality to be measured and managed.

McCall’s quality model was followed by Boehm’s quality model (Kitchenham and Pfleeger, 1996). Like McCall’s model, Boehm’s model presents product quality in a hierarchy with three high level characteristics linked to seven intermediate factors, which are in turn linked to 15 primitive characteristics. Boehm’s model has a wider scope than that of McCall’s, with more emphasis on the cost-effectiveness of maintenance (Milicic, 2005).

More recently work has been done to create an international standard for software product quality measurement—ISO9126 (2001). This standard is again organised in a hierarchy with six characteristics at the top level and 20 sub-characteristics with indicators used to measure the sub-characteristics. In addition to aspects of internal and external quality, covered by McCall and Boehm’s models, ISO 9126 includes quality characteristics of functionality (Milicic, 2005). Internal, external and functional qualities are also mixed at all levels of the hierarchy. However, ISO 9126 does not clearly state how quality should be measured (Kitchenham and Pfleeger, 1996).

None of these three models present a rationale for the selection of characteristics to be included in the quality model and it is not possible to tell if a model presents a complete or consistent definition of quality (Kitchenham and
Pfleeger, 1996). Further the placement of items appears arbitrary in ISO 9126, with no justification as to why Interoperability is not related to Portability.

Dromey (1996) presents a different type of model that attempts to address some of the issues presented and support developers to build product quality. Dromey believes that it is impossible to build high-level quality attributes like reliability or maintainability into a product, but developers must instead build properties that manifest in achieving these goals. The distinction this model makes is important, as using it will verify that it allows the quality required to be achieved (Kitchenham and Pfleeger, 1996). Before Dromey’s model can be successfully applied, the various groups involved in the development of a software product must agree on what quality attributes should be achieved and to what level. This process can be supported using other models.

1.2.3 Merging Perspectives on Software Quality

The alignment of stakeholders in software development has been defined as “convergent intentions, shared understanding and coordinated procedures” (Chan, 2002). Organisations continually try to align stakeholder groups as there is significant evidence that successfully aligned groups will outperform those that are not aligned (Chan and Reich, 2007). Alignment of stakeholders in software development allows them to collaborate more effectively and produce systems that support the long-term business strategies. Highly misaligned teams can cause conflict and eventually lead to the failure of a project.

Stakeholder alignment is a daunting task due to the fact that alignment is not a state, but an ongoing process (Chan, 2002). Keeping groups aligned requires ongoing work. A lack of awareness and belief in alignment have been identified as main contributors to misalignment (Chan and Reich, 2007).

A number of methods can be applied to help reconcile this situation and select the best way forward. These methods include expert judgement, the Non-Functional Requirements (NFR) Framework, Quality Functional Deployment and Theory-W.

Expert judgement involves one or more professionals using their experiences and knowledge to make a decision on an issue. The decisions are not necessarily supported by modeling or numerical assessment.

The NFR Framework uses diagrams to relate non-functional requirement goals with different decisions that can be made in the design and operation of a system that affect it positively or negatively, allowing trade-offs to be identified and made (Chung et al., 2000). While this method makes the results of a choice
to be made more explicit, it requires a set of common priorities to be identified to allow effective decisions to be made.

Quality function deployment (QFD) considers the priority of customer and technical requirements in achieving the goals of the system to help prioritize the requirements (Herzwurm et al., 2003). However, the other perspectives involved in the development of the software product are not considered.

Value-based software engineering (VBSE) recognises the problems created by conflicting perspectives in the software development process (Boehm and Jain, 2006). Central to resolving conflict in VBSE is Theory-W, which requires (Boehm and Ross, 1989):

1. Success-critical stakeholder groups to be identified;
2. The requirements of these groups to be elicited;
3. Negotiation between the groups to create a win-win situation; and
4. A control process to support success-critical stakeholder achieve the win-win situation and adapt to the changing environment.

The key advantage of Theory-W is that it explicitly brings all of the parties on whom success depends together to understand each other’s needs, compromise and agree. But in order to be successful, Theory-W must be managed to ensure the plans are achieved and any deviations from the plans are corrected (Boehm and Ross, 1989). Management requires an understanding of why the goals are being pursued, what is the required result, who is responsible for the result, how the result will be achieved and at what cost the result can be achieved. The answer to these questions will be specific to the context in which they are answered.

Getting alignment between stakeholders is becoming increasingly difficult as the software development environment continues to grow and become more complex. This is most visible with the rapid take-up of global software development (GSD) practices.

1.2.4 Global Software Development

GSD is defined as software development work undertaken across national boundaries at geographically separated locations. While teams are not co-located, they are still working towards a common goal with a commercially viable product. Groups are commonly classified against two criteria:
1.3 Research Questions

1. **Insourcing/Outsourcing** defines whether-or-not the work is undertaken by employees of the organisation.

2. **Onshore/Offshore** defines whether-or not the work is undertaken in the home country of the organisation.

GSD has become a common practice (Conchuir et al., 2009), with a number of benefits and risks. It allows companies to call upon a global talent pool to supplement a locally scarce resource pool (Wang et al., 2008). These specialized skills can have a positive impact on productivity and quality (Conchuir et al., 2009; Mockus and Herbsleb, 2001). Further, many studies cite the ability to reduce development costs (Conchuir et al., 2009), and to focus on strategic business functions with day-to-day operations off-loaded (Wang et al., 2008).

The most commonly cited challenges of GSD relate to communication (Mockus and Herbsleb, 2001) and coordination (Herbsleb, 2007). Co-located teams have a greater opportunity to share formal and informal discussions, which have been found to be an effective way of creating a shared understanding in relation to what is expected in software quality. Geographic distance reduces a team’s ability to communicate and collaborate (Noll et al., 2010). While new technologies have helped to reduce the barriers created by distance, it is emphasized that distance still matters (Carmel and Abbott, 2006).

Intercultural factors, such as power distance, individualism, and uncertainty avoidance (Hofstede, 1980) can also be problematic in GSD settings. Studies on the impact of national and cross-cultural issues on systems development emphasize the need to take organizational and national culture seriously when working in these environments (Hofstede, 1980). Further, cultural issues can exacerbate exiting communication and coordination problems (Herbsleb and Moitra, 2001).

### 1.3 Research Questions

The benefits of having the success-critical stakeholders and stakeholder groups aligned in the priorities they award to aspects of software quality are clear. A common understanding of the priorities given to software quality helps ensure the delivered product meets the expectations of all success-critical stakeholders. However, given the growing complexity in the software development environment, it is becoming increasingly complex for success-critical stakeholders to achieve. To help understand this situation, this thesis has two main research objectives.
As previously stated, the first objective of this thesis is to develop a method that can determine the alignment between success-critical stakeholders in terms of the priorities given to aspects of software quality. Such a method will help ensure the success-critical stakeholders are aligned, and support further action if they are not. This aim leads to the follow research questions and subquestions:

- **RQ1:** Which method can be used to determine the alignment between success-critical stakeholder groups in terms of the priorities given to aspects of software quality?
  - **RQ1.1:** How can individual priorities on software quality be elicited?
  - **RQ1.2:** How can individual priorities be aggregated to represent group priorities?
  - **RQ1.3:** How can the level of alignment between groups be determined?
  - **RQ1.4:** How can the level of alignment between the members of a group be determined

Given the level of alignment between success-critical stakeholders, it is important to understand the reasons behind any alignment and misalignment found in the priority given to aspects of software quality. This knowledge can be used to foster a common understanding of software quality, with all of the benefits this entails. This second research objective leads to the following research questions and subquestions:

- **RQ2:** Which factors can positively and negatively affect alignment between success-critical stakeholder groups?
  - **RQ2.1:** On which aspects of software quality do the success-critical stakeholders and stakeholder groups share a common priority?
  - **RQ2.2:** On which aspects of software quality do the success-critical stakeholders disagree on the priority?
  - **RQ2.3:** How do these results change when stakeholders describe what they perceive should be happening today in an ideal scenario?
  - **RQ2.4:** What are the reasons for the agreement and disagreement?

An overview of which chapters are used to answer each research question is shown in Table 1.1.

The next section discusses the research methodologies used to answer these research questions.
1.4 Methodology

Table 1.1: Cross reference between chapters and the research questions answered

<table>
<thead>
<tr>
<th>Research Question</th>
<th>Chapter 1.1</th>
<th>Chapter 1.2</th>
<th>Chapter 1.3</th>
<th>Chapter 1.4</th>
<th>Chapter 2.1</th>
<th>Chapter 2.2</th>
<th>Chapter 2.3</th>
<th>Chapter 2.4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter 2</td>
<td>◦</td>
<td>◦</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chapter 3</td>
<td>◦</td>
<td>◦</td>
<td>◦</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chapter 4</td>
<td>◦</td>
<td>◦</td>
<td>◦</td>
<td>◦</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chapter 5</td>
<td>◦</td>
<td>◦</td>
<td>◦</td>
<td>◦</td>
<td>◦</td>
<td>◦</td>
<td>◦</td>
<td>◦</td>
</tr>
<tr>
<td>Chapter 6</td>
<td>◦</td>
<td>◦</td>
<td>◦</td>
<td>◦</td>
<td>◦</td>
<td>◦</td>
<td>◦</td>
<td>◦</td>
</tr>
<tr>
<td>Chapter 7</td>
<td>◦</td>
<td>◦</td>
<td>◦</td>
<td>◦</td>
<td>◦</td>
<td>◦</td>
<td>◦</td>
<td>◦</td>
</tr>
<tr>
<td>Chapter 8</td>
<td>◦</td>
<td>◦</td>
<td>◦</td>
<td>◦</td>
<td>◦</td>
<td>◦</td>
<td>◦</td>
<td>◦</td>
</tr>
<tr>
<td>Chapter 9</td>
<td>◦</td>
<td>◦</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1.4 Methodology

In order to answer the research questions posed in this chapter it is essential to have a research methodology. A research methodology provides the link between research questions and the data used to answer them Leedy and Ormrod (2005). Thus a methodology must be chosen that provides the necessary data to answer the stated research questions. This section explores types of methodologies that can be used, with a focus on those that can be used to answer the research questions this thesis seeks to address.

Research methodologies can be categorised in many different ways. Empirical research is one of the main categories in which methodologies can be placed. Empirical research can be described as evidenced based research, where findings are verified through observation and experience, and it is contrasted with studies of pure logic. As the research questions in this thesis have applied and practical objectives, this thesis focuses on empirical research methodologies.

1.4.1 Approaches to Empirical Research

There are a number of approaches for conducting empirical research in the software engineering discipline. The most common approaches for gathering data are case studies, surveys and experiments (Kitchenham et al., 2002). However, there are also other methods that can be used.
Case Studies

A case study is used to investigate phenomena in their native context around the time they occur (Runeson and Höst, 2009). While the small sample size makes it difficult to close to impossible to generalise the results of case study research (Kitchenham et al., 1995), the approach is useful in exploratory research where little is known about an area. The methods for data collection will vary depending on the needs of the case study, but can be direct (eg. observations and interviews) or indirect (eg. document studies).

Surveys

Where case study research is conducted on a small scale, surveys are used to learn about a large population (Leedy and Ormrod, 2005). Much more data is collected with the aim of producing more generalisable results. As such it should be possible to tabulate or otherwise structure and order the results to describe characteristics of the situation studied. In order to generalise the results of a survey, careful attention must be paid to the selection of participants to reduce the risk of a bias in the results. The results are usually collected in questionnaires and interviews, with statistical methods applied to analyse the results.

Experiments

Experiments are situations created by researchers to investigate the relationship between different factors by controlling related variables (Leedy and Ormrod, 2005). This means a situation is repeatedly created with a controlled variance of some aspect or aspects to observe the resultant outcome.

1.4.2 Data Collection

With all of the approaches above there are multiple methodologies that can be employed to collect data. These methodologies are divided into two broad categories—quantitative and qualitative (Leedy and Ormrod, 2005). These are described in more detail in the following sections, with a discussion on choosing which one is the most appropriate for a given study.
1.4 Methodology

Quantitative Research

Quantitative research is generally used to answer questions about the relationships between variables for the purpose of explaining, predicting or controlling the phenomena (Leedy and Ormrod, 2005). Statistical methods are commonly used to establish or confirm hypotheses, and produce generalised findings—which is the greatest power of quantitative research.

Quantitative research is usually carefully structured with consideration for the variables that need to be observed or controlled. Objectives are often written in terms of hypotheses to be tested.

Qualitative Research

Qualitative research seeks to better understand and explain complex situations (Leedy and Ormrod, 2005). The researcher must enter the research with an open mind, as discoveries along the way can shape the research.

Qualitative researchers use observations and inductive reasoning to build theory from the ground up, where quantitative researchers use statistical methods to confirm proposed theories. One reason for this difference is an assumption by qualitative researchers that reality cannot be easily divided up into discrete measurable variables.

Deciding Between Quantitative and Qualitative Research

The choice between quantitative and qualitative methods is not a choice between right and wrong (Leedy and Ormrod, 2005). Both have their strengths and weaknesses, and each is more suited to certain situations. It is even possible to use both together, with qualitative methods being to understand why a situation is occurring, and quantitative methods to find out if these reasons are common.

1.4.3 Data Sources

In addition to the categorisations of methodologies detailed above, the researcher also needs to select the subjects or context to be studied.

Glass (1994) discussed the ‘software research crisis,’ concluding that the main problem with software research is that most is not relevant to industry. Since then there has been a movement toward more applied research in the software development industry, helping to ensure research is valuable to the people who develop software. However, recent work would suggest that progress is slow (Ivarsson and Gorschek, 2010).
However, it is not always possible to conduct research in an industrial context. For example, it may not be possible to control the required variables to conduct an experiment due to a business’ need to carry-on working. This means that while some research may be valuable in an industrial context, it is necessary to conduct the research in a laboratory setting.

It is also important to consolidate knowledge in a given research area. Systematic reviews are becoming an increasingly common technique to achieve this goal within the software engineering discipline (Kitchenham, 2007; Kitchenham et al., 2009).

1.4.4 Categorisation of Chapters in this Thesis

This thesis employs a mixed research methodology. An overview of the methodologies used in each chapter of this thesis is presented in this section, but further details on the methodologies used can be found in each chapter. The methodologies used are summarised in Table 1.2.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Chapters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quantitative</td>
<td>2 3 4 5 6 7 8 9</td>
</tr>
<tr>
<td>Qualitative</td>
<td>○ ○ ○ ○ ○ ○ ○ ○</td>
</tr>
<tr>
<td>Case study</td>
<td>○ ○ ○ ○ ○</td>
</tr>
<tr>
<td>Survey</td>
<td>○ ○ ○</td>
</tr>
<tr>
<td>Experiment</td>
<td>○ ○ ○ ○ ○ ○</td>
</tr>
<tr>
<td>Industry</td>
<td>○ ○ ○ ○ ○ ○</td>
</tr>
<tr>
<td>Laboratory</td>
<td>○ ○ ○ ○</td>
</tr>
<tr>
<td>Literature</td>
<td>○ ○</td>
</tr>
</tbody>
</table>

Chapter 2 presents a systematic map and systematic literature review. The method involved conducting a survey of the literature addressing software quality trade-offs. Relevant papers are classified according to a predefined schema, providing quantitative statistical information about the research publications in this area. A subset of these papers are selected for deeper review, with a descriptive study of the aims, method and empirical work that had been completed—providing qualitative results.
Chapter 3 uses qualitative methods to discover how companies are managing requirements to create value and quantitative methods to determine what values were being applied to the selection and prioritisation of requirements. A number of case studies are presented in the results, providing a small industrial survey.

Chapter 4 consolidates the knowledge and experience of applying the Stakeholder Alignment Assessment Method for Software Quality (SAAM-SQ) in a series of case studies. The chapter draws upon the published and unpublished literature describing each application of SAAM-SQ. Changes to SAAM-SQ and lessons learnt from applying SAAM-SQ are drawn out to provide a descriptive history of its development.

Chapters 5–9 each present case studies conducted in industry using SAAM-SQ to determine the priorities placed on different aspects of software product quality. Qualitative analysis of the results was conducted in a workshop setting following the analysis of the quantitative data.

1.5 Research Setting

This thesis presents a number of empirical studies set in industrial contexts. The settings used in the collection of data are presented in the following subsections. Further detail can be found in the appropriate chapters.

1.5.1 Collective Case Study—Anonymous Companies

Chapter 3 uses data collected from a set of companies. These companies are described, but for reasons of confidentiality remain anonymous.

In total three medium-sized companies are involved in this research:

- The first company is Australian-based and operates primarily within the Australian market. The company provides a number of business services, including a number of software products. This company has been developing software for over ten years.

- The second company is Australian-based, but has offices in other countries and sells products globally. It is a young company, founded in 2002, and creates a number of software products aimed at the professional software development and business markets.

- The third company is based in Germany, with sales offices in other countries and sells their products primarily within Europe. It offers a number
of software products and software intensive solutions in the telecommunications arena.

Collecting data from a number of industrial sources provides greater power in the ability to identify and draw more general trends and conclusions. However, this is not true for this research as the set of companies used is not representative of the software development industry at large. Although there are a number of advantages from gathering data from a number of different sources—this provides greater confidence for identifying more general trends, allowing future work to focus on areas of key interest without having to find a large number of companies able and willing to participate in a general study.

1.5.2 Case Studies—Ericsson

Ericsson is a world leading company in telecommunication, providing a wide range of products and solutions. Products are developed and sold as generic solutions offered to an open market, although customised versions of the products are also developed.

Ericsson is the industrial partner for the research presented in Chapters 5, 6, 7 and 9. Ericsson is active in this research relationship—shaping the research, providing access to collect data, analysing the results, and instigating change based on the results.

Of Ericsson’s diverse product portfolio, two major software-intensive product have been the subject of the research activities presented in this thesis. Neither should be considered representative of Ericsson, due to a number of common factors between the product in terms of management. The products are not named for reasons of confidentiality.

The research partnership with Ericsson provides benefits for both academia and industry—creating a win-win scenario with the success-critical stakeholder groups. Ericsson is able to get a external perspective that is grounded in research to examine areas perceived beneficial to the company; and it provides the researcher with industrially relevant topics, data and results.

1.5.3 Case Study—Anonymous Company

Chapter 8 uses data collected from one anonymous company. This company is a multinational financial institution, providing a wide range of payment, expense management and travel solutions. Its customers range from individual consumers to other multinational corporations. The company is based in the
US and employs approximately 60,000 people throughout the North American, South American, European, African, Middle Eastern and Asia-Pacific regions.

Focusing a study to one company allows for more in-depth data collection and analysis than is possible in a collective case study. Further, anonymity allows for results to be published that may otherwise be considered too sensitive to be linked to the organisation. Such studies are valuable to researchers and practitioners alike, as they can draw attention to challenges faced by organisations that would not otherwise be made public, allowing greater opportunities for understanding and resolution.

1.6 Overview of Chapters

This section provides a brief overview of each chapter in this thesis. Attention is given to how each chapter helps answer the research questions defined in Section 1.3.

The overall structure of this thesis in terms of the relationship between chapters is shown in Figure 1.1. Research activities are represented as rectangles, and thesis chapters are represented as documents. Lines are used to indicate how research activities and chapters are related.

The thesis begins with a systematic review of the literature addressing software quality trade-offs, providing state of art background for this thesis. The results of the systematic review are presented in Chapter 2.

A collective case study of the factors that success-critical stakeholder perceive as influencing requirements selection was conducted and is presented in Chapter 3.

Both Chapter 2 and Chapter 3 provide an important input for the Stakeholder Alignment Assessment Method for Software Quality (SAAM-SQ), which is developed and applied as part of this thesis to answer the research questions presented in Section 1.3. The method is applied in a series of case studies. This experience of applying the method allows for it to be both evaluated and refined—answering RQ1. Further, each case study provides a description of a software development setting in terms of priorities different groups place on aspects of software quality, allowing the reasons for similarities and differences to be explored—answering RQ2. The latest version of the method is presented in Chapter 4. Selected case studies applying the method are presented in Chapters 5, 6, 7, 8 and 9.

The remainder of this section provides an overview of each chapter.
1.6.1 Chapter 2: Software Quality Trade-offs

The interdependency between the aspects that make up software quality pose challenges for researchers and practitioners alike. The interplay between quality aspects, under investment and even over investment all pose challenges for software development, and if not well managed can create major problems. This interdependence is often overlooked as many researchers aim to advance individual aspects of software quality without fully considering the impact this sub-optimisation will have on other aspects of software quality.

In order to better understand and address the interdependence between the aspects that make up software quality, this chapter aims to provide a consolidated overview the literature addressing trade-offs between aspects of software quality. This provides an opportunity to understand what approaches are avail-
able to make trade-offs between aspects of quality, and the degree to which they have been validated and evaluated.

A systematic map of the literature addressing software quality trade-offs is conducted and the results are presented. This provides a high-level overview of the research area, highlighting the approaches used for trade-offs, the development artefacts on which trade-offs are made, the types of research used, key authors and key publication venues. The results show a wide range of solutions being considered to support trade-offs within software quality. Half of the identified research addresses architectural artefacts, with the requirements and process artefacts also commonly used to support trade-offs. However, there is no empirical validation or evaluation made of most approaches.

Given the aim of this thesis is to determine the level of alignment between success-critical stakeholders on the priority they give to aspects of software quality, a systematic review is conducted and presented of the empirical literature addressing the process and requirements artefacts, as identified from the systematic map. The reasons for the process and requirements artefacts being made the focus of this second phase relates to the importances of both of these artefacts in ensuring a common understanding of the software quality expectations—which is the focus of this thesis. Again the results show a wide range of approaches for trade-offs, each with limited empirical validation and evaluation. However, relevant findings are made that support this thesis. Many of the approaches emphasise the need to develop a model of software quality specific to the context under study, and to involve representatives of key stakeholder groups as they represent important and differing perspectives.

Further, these results highlight the need and value of empirical validation and evaluation of the approaches used to make trade-offs between aspects of software quality. This chapter address RQ1.1 and RQ1.2 of this thesis.

1.6.2 Chapter 3: Requirements Selection

Much of the value of a software product is in the requirements it fulfils—both functional and non-functional. Given the need for software companies to maximise the creation of value in their products, the process for selecting requirements for a release of software becomes a crucial one. Decisions ultimately have to balance a number of different perspectives and issues that concern the success-critical stakeholders.

This chapter extends previous research presented by Barney et al. (2006), and explores how the creation of software product value through requirements selection is understood in various industrial settings. It looks at the process for
selecting requirements for a release of software, and how this has evolved over time. It determines the priority placed on the values that shape the requirements selection and prioritisation decisions for a release of software; and uses these results to determine the relative influence of key stakeholder perspectives in this process.

A methodology is used involving interviews and a cumulative voting (CV) prioritisation task. The interviews seek to understand the requirement selection and prioritisation processes and how they have evolved. CV is used to find the relative priority on a number of values that influence the selection and prioritisation of requirements. Each value is associated with one success-critical stakeholder group, allowing the influence of each group to be determined.

Two of the three companies participating in the collective case study followed the same process to arrive at their current requirements management practices, which are very similar, despite coming from different countries and different industries. While there were differences in the values that shape the selection and prioritisation of requirements for a release, it appears that these values will change depending on a number of contextual factors—like number of customers and the profitability of a product. Finally the results show that while all success-critical stakeholders are important, their influence is not equal—with the business perspective being the more influential in the prioritisation and selection of requirements for a release as compared with the product and project perspectives.

The result show that there is no silver bullet in the selection and prioritisation of requirements—it is a difficult task to balance requirements and companies continue to improve in a step-wise process. Additionally, the values used to select requirements for a release vary with the context in which a software product exists—for example, a new product that is yet to turn a profit is treated differently to an established and profitable product; and the size of the customer base influences requirement selection.

This work in requirements selection and prioritisation has been extended beyond what is presented in this chapter. A much larger collective case study has been undertaken examining the values influencing the selection and prioritisation of companies operating in China (Barney et al., 2009b). The results show that Chinese companies with a domestic market, Chinese companies with an international market and Western companies operating in China have similar value systems—but Chinese companies with an international market perceive they have a strength in after-sales support.
1.6 Overview of Chapters

1.6.3 Chapter 4: A Method for Determining Alignment

The development of software products involves a number of groups working towards a set of common goals. However, as each group is involved with the product in a different way, each will be more aware of certain issues and less aware of other issues. In turn, this impacts how the different groups will prioritise the same set of issues. While features of a software product are usually explicitly stated in a requirement document, the expectations on software qualities are often not so clearly stated—leaving the interpretation up to the individuals working on the software product. As software product quality is one of the key areas to control in the development of a software product, it is important to ensure the people working on that product are working towards the same goal, as this has been found one of the most successful ways of meeting expectations.

This chapter aims to support the software development process by describing the Stakeholder Alignment Assessment Method for Software Quality (SAAM-SQ), a method to determine the level of alignment between key stakeholders and stakeholder groups in the priority given to different aspects that describe software quality. This in turn will allow the areas of shared understanding and conflict to be identified and understood.

SAAM-SQ is developed from prior research and refined through its repeated application in a series of case studies—shown in Figure 1.1. Key findings from the systematic review in Chapter 2 are integrated into the method and subsequent research—including the need to tailor a quality model to the context studied, to involve key stakeholder groups, and to provide empirical validation or evaluation of the method’s application. The proposed method builds upon the CV approach used in Chapter 3. In this research the CV technique was found to be capable of eliciting individual priorities and aggregating them to a group level with a high level of precision.

The chapter draws and is refined upon the results and experiences of successful and unsuccessful applications of SAAM-SQ. The lessons learnt from applying the method in these case studies are discussed to support researchers and practitioners in future applications of SAAM-SQ—answering RQ1. These improvements include using Hierarchical Cumulative Voting to ease elicitation of respondents priorities, the development of statistical methods to gain a deeper understanding of the results and a greater understanding of the factors that lead to the methods successful application in industrial contexts. A timeline of the chapters presented in this thesis that employ the method described in Chapter 4 is shown in Figure 1.2.

The remaining chapters in this thesis present case studies applying SAAM-
SQ throughout its development. The purpose of these chapters is to develop and refine the method based on empirical results—answering RQ1, and identify factors that positively and negatively affect alignment between the stakeholders—answering RQ2.

1.6.4 Chapter 5: Quality Alignment Inside an Organisation

This chapter presents a case study of one product at Ericsson in Sweden. An early version of SAAM-SQ is applied to determine the level of alignment between stakeholder groups defined by onshore insourcing for this product.

SAAM-SQ initially employed the CV technique to elicit the priorities given to the various aspects of software quality. However, this choice proved difficult for respondents during the pilot, and was changed to HCV in order to limit the comparisons between aspects that people found difficult to compare (e.g. usability and cost constraints). After this change the method was successfully applied, eliciting the priorities on the aspects of quality defined in the model of quality tailored for this context.

The results of this case study show the groups studied to be very closely aligned. When asked to define the priorities that should be ideally placed on the product today, each group wanted to see greater emphasis on the aspects of quality that most directly related to their role. For example, developers perceived greater need for improved maintainability and testers perceived greater
1.6 Overview of Chapters

need for improved testability.

The results presented in this chapter show that SAAM-SQ can be successfully applied to determine the alignment of key stakeholder groups—addressing RQ1.1, RQ1.2 and RQ1.3. Further the results highlight a potential conflict in the priorities given to software quality due to the different roles involved in software development—addressing RQ2.

1.6.5 Chapter 6: Quality Alignment with Subcontracted Developers

Issues with software product quality are commonly reported when organisations engage in outsourcing relationships. To address this issue, value-based software engineering literature emphasises the need for all success-critical stakeholder groups to work towards a mutually agreed goal.

This chapter extends the case study presented in Chapter 5 to include groups defined by onshore outsourcing, with two subcontracted development teams brought into the study. Thus the results are extended to include the original and new data.

SAAM-SQ was found to be sufficiently robust to extend the case study in the manner described, with no additional changes required to the method—answering RQ1.1, RQ1.2 and RQ1.3.

The results provide additional findings that answer RQ2. The subcontracted stakeholders priorities are the result of a merging of the priorities of both the software development organisation, and the firm providing the subcontracted developers. For example, the subcontracted developers are not involved in customer installations and perceived this quality aspect much less important than the stakeholders defined by insourcing. While some significant differences were found between the groups defined by insourcing and outsourcing, these differences had not caused any problems at the time of the study. However, with the growing complexity of the system, it was determined that the situation should continue to be monitored, and if problems arise due to any of the differences identified, it may be necessary to change the way the subcontracted developers are involved with the product. For the example provided, one solution would be to involve the subcontracted developers in customer installations so that they understand the challenges faced and make appropriate decisions to maintain this aspect of quality at a sufficient level.

Further, stakeholders from outsourced relationships had greater difficulty to define their perceived ideal balance of software product qualities. Each of the subcontracted developers identified challenges they had faced in their short
time working on the product that could be improved through some changes to the priorities on aspects of software quality. However, due to the limited time the subcontracted developers had been working on the product, each had faced different challenges and perceived different changes necessary to resolve their various issues. This result suggests a certain level of experience working with a software product is required to fully understand the implications of changing the priorities on software quality.

1.6.6 Chapter 7: Quality Alignment with Offshored Development

Software quality issues are commonly reported when offshoring software development. Value-based software engineering addresses this by ensuring key stakeholders have a common understanding of quality. However, this is recognised to be much more difficult given the geographic and temporal barriers to communication and collaboration.

This chapter extends the case study presented in Chapter 5 and Chapter 6 with the inclusion of success-critical stakeholder groups defined by offshore insourcing. It also presents a new case study, of a second telecommunications product produced by Ericsson, for which the context studied is also defined by offshore insourcing. The offshore location in both case studies presented in this chapter is India, with the onshore location being Sweden.

For this chapter additional statistical techniques are adopted that allow the alignment within a group to be calculated—answering RQ1.4. The statistic is based on inertia, and detail on its calculation are provided both in the description of the method in Chapter 4 and Chapter 7.

Low levels of alignment were found within between the groups studied for both products. The low levels of alignment were attributed to a number of factors. First, quality requirements are not sufficiently defined as part of work packages, creating an environment where stakeholders assume the level of quality that is required. Second, a cultural difference was observed between India and Sweden—where Swedish developers will push-back and question if they do not understand or agree with their manager, Indian workers respect their managers judgement and work to achieve the maximum possible given their constraints. Third, a different perspective of time was observed between the roles—the knowledge and understanding of groups like product support is based on the present and the past; other groups, like developers and architects, have a vision of the products longer-term sustainability.
1.6.7 Chapter 8: Quality Alignment in a Global Organisation

Global software development (GSD) is rapidly increasing in popularity. With additional key stakeholder groups involved in software development there are more opportunities for misunderstanding and misalignment on software quality issues.

This chapter presents a case study of a bespoke development project at a global financial services company. The stakeholders are globally distributed across three sites and cover insourcing and outsourcing. The site studied are in America, Australia and India.

For this study SAAM-SQ was adapted to collect data remotely. The prioritisation data for the other studies presented in this thesis were all collect during one-on-one interviews. A previous attempt had been made to collect the this data remotely, but proved unsuccessful. A discussion on the factors for successful application of the method, including in remote data collection scenarios, is provided in Chapter 4.

The results show the Australian site to be well aligned with both the American and Indian site. However, the Indian and American sites are not aligned with each other. In this case study alignment was positively influenced by access to business users, overlapping work hours and domain knowledge—however, access to these facets was not equal for all sites.

The results are used to identify a set of best practices that should help achieve alignment within a GSD context. These best practices support the findings of related GSD literature.

1.6.8 Chapter 9: Balancing Software Product Investment Options

The long-term sustainability of a software product depends on more than developing features and quality. Priorities are placed on aspects that support the development of software, like features, software quality, project constraints—time and cost, and even the development of intellectual capital (IC). A greater focus on any one aspect takes priority from another, but as each aspects delivers a different type of value, managers have trouble comparing and balancing these aspects.

This chapter aims to extend SAAM-SQ to make more general trade-offs between investment options in the software development process. Specifically this chapter aims to exploit the hierarchical nature of HCV, to combine the
priorities on software quality presented in Chapter 5, with the priorities on IC for the same case—presented in Barney et al. (2009a)—and new data on the relative importance of features, quality, IC, time constraints and cost constraints.

The results show that it is possible to exploit the hierarchical nature of HCV to combine different data sets. This is an important result for the method presented in Chapter 4, as it means that it is possible for different people to prioritise different sets of quality aspects. Given software systems are often sufficiently complex that no one person can have a complete understanding of the entire system, this result allows for experts to respond specifically to their area of expertise and still be able to aggregate the results at a higher level.

Further, the results show it is possible to apply the method more generally in contexts that extend beyond software quality.

1.7 Authors’ Contribution

Sebastian is the main author of each chapter in this thesis. The contribution of Sebastian to each chapter is discussed together, except Chapter 8, which is discussed separately.

As main author, Sebastian was responsible for coordinating the research process, data collection, analysis and most writing activities. The coauthors mainly supported through feedback on and discussion of study design, analysis and written papers. For the systematic review, presented in Chapter 2 the coauthors further assisted in providing second and third reviews of identified publications against the inclusion and exclusion criteria.

Chapter 8 is based on the work published in the honours thesis by Varun Mohankumar (2010). This thesis applies the method described in Chapter 4. Sebastian supported Varun’s supervisor, Aybüke Aurum, in an unofficial secondary supervisor capacity. Sebastian supported Varun with the literature review, design, execution and analysis of the work. Based on the work presented in the thesis and his experience providing support throughout the research process, Sebastian distilled and wrote the research paper presented. This paper draws heavily on the work in Mohankumar (2010), but some new analysis is provided.

1.8 Thesis Contribution

The general contribution of this thesis is is twofold:
• First, it provides an empirically evaluated and mature method, the *Stakeholder Alignment Assessment Method for Software Quality* (SAAM-SQ), to determine the level of alignment between success-critical stakeholders in terms of the priority given to aspects of software quality.

• Second, it identifies a number of factors that affect the level of alignment in terms of the priority given to aspects of software quality.

SAAM-SQ was developed to address RQ1—to determine the alignment between success-critical stakeholder groups in terms of the priorities given to aspects of software quality. The method draws on key findings from the systematic literature review on software quality trade-offs presented in Chapter 2—including the need to tailor a quality model to meet the context of the environment being studied, and to involve representatives of key stakeholder groups. It exploits the power of the cumulative voting technique used in Chapter 3 to elicit and aggregate stakeholders priorities to a group level.

SAAM-SQ has been applied to a series of case studies, with a subset of these studies included in this thesis—Chapters 5, 6, 7 and 8. The repeated application of SAAM-SQ allowed the authors to reflect upon and evolve the method based on empirical evidence. The opportunities for empirical-based evolution proved invaluable. The first version of the method, discussed in Chapter 5, had difficulty eliciting the priorities of the respondents. This was overcome by replacing cumulative voting (CV) with hierarchical cumulative voting (HCV). Chapters 7 and 8 continue the evolution of SAAM-SQ, each presenting improved data analysis techniques that provide more detail and greater confidence. Further, many of the lessons gained from the empirical evolution of the method can be applied to other research.

The latest version of SAAM-SQ, along with details about its evolution, is presented in Chapter 4. The repeated application of the method also fulfilled a need identified in the systematic review presented in Chapter 2—for greater empirical validation and evaluation of methods within this area of research.

SAAM-SQ has been found to be robust, describing a wide range of work contexts found in modern software development settings, as shown in Figure 1.3. The ability for the method to be applied to a diverse range of stakeholder groups has allowed a variety of software development contexts to be studied—helping to provide a richer answer for RQ2.

Further, in Chapter 9 and Barney et al. (2009a) SAAM-SQ has been successfully applied to balance priorities more generally. Barney et al. (2009a) apply the method to examine the alignment between success-critical stakeholder groups
in terms of the priorities on intellectual capital investment. Chapter 9 exploits the hierarchical nature of the method, to combine the results from (Barney et al., 2009a) and Chapter 5 with new data, and provide a description of the investment space for the software development environment studied.

The repeated application of SAAM-SQ in addressing RQ1 proved fruitful in addressing RQ2—to identify factors that can positively and negatively affect alignment between success-critical stakeholder groups in terms of software quality. Each application studied a different context, as shown in Figure 1.3, allow for a more general understanding of the factors. A summary of the findings addressing RQ2 is presented below, with references to the appropriate chapters provided for further information:

- **Role-based responsibilities:** Where some aspects of quality more directly affect one role, they are more likely to be more highly prioritised by that role. For example, developers are likely to prioritise maintainability higher than other groups. (Chapter 5)

- **Time and experience working on the product** provides opportunities for closer alignment as people will share a more homogeneous body of knowledge about the product. (Chapter 6)

- **Serving two masters:** The priorities of outsourced developers will represent a merging of the priorities of both the company employing their services

![Figure 1.3: Research setting studied in selected chapters of this thesis](image-url)
and the company employing the outsourced workers. Ultimately the workers need to satisfy both organisations. This has the potential to reduce alignment if the companies have competing goals. (Chapter 6)

- Lack of experience: Stakeholders can perceive the need for changes without fully understanding the consequences. Thus, this lack of experience can lead to conflicting priorities on the goals. (Chapter 6)

- Temporal/Role-based differences: Each roles perspective of the product is shaped from the different perspective from which it is viewed. For example, the experience of product support provides them with an understanding of the product’s present and past, but does not necessarily give them the ability to understand the affect of a change to the architecture. (Chapter 7)

- Cultural differences: Unexpected, unmanaged cultural differences leads to poor communication. For example, between India and Sweden—Swedish developers will push-back and question if they do not understand or agree with their manager, Indian workers respect their managers judgement and work to achieve the maximum possible given their constraints. (Chapter 7, Chapter 8)

- Failure to document and communicate changes can result in stakeholder being unaware of changed expectations. (Chapter 8)

- Insufficient domain knowledge limits the ability for key stakeholders to understand what is required. (Chapter 8)

- Inadequate documentation limits the ability for key stakeholders to understand what is required. (Chapter 8)

- Insufficient access to key stakeholders: Communication and collaboration is key to a common understanding of expectations. (Chapter 8)

- Time zone differences: These limit the possibility for casual and synchronous communication, which support have been found to foster a mutual understanding. (Chapter 8)

1.9 Future Work

Some future work was underway at the time of writing this thesis, but there remain a number of research activities that remain to be completed. While
the repeated application of SAAM-SQ has allowed the method to evolve based on empirical evidence, there remain the possibility of further enhancements. Further application of the method would also help identify additional factors that support and impair alignment in the priority given to software quality aspects.

SAAM-SQ currently clusters stakeholders by their role when testing the level of alignment. It is possible that role-based clustering does not best identify groups of aligned stakeholders. Adopting hierarchical factor analysis to identify the groups of stakeholders that are logically clustered provides the opportunity to reverse engineer the factors that support their shared priorities. Future applications of the method should apply such an approach to gain a deeper understanding of the factors supporting and impairing alignment. Additional information should also be collected from the stakeholder, such as experience, to identify the most influential factors in alignment.

SAAM-SQ also needs to provide greater support when applied in contexts employing software product lines. In these contexts it is possible that one component will be used in a number of systems—each with different quality requirements. Thus understanding the priorities on quality in these contexts is a much more difficult undertaking.

Currently SAAM-SQ is reliant on subjective measures of priority to determine the level of alignment, making the method very dependent on the human condition. There are numerous benefits for defining more objective measurements as part of this work—including reducing biases and providing the possibility for automating the method. However, the experience of many researchers working with quality models has shown it to be a difficult undertaking. However, it should still be an objective of future work.

Further, it would be beneficial to understand if the priorities on software quality are aligned with the strategic goals for the both the product and the software development company. There are two main choices for such work—to involve stakeholders that represent these perspectives, or to develop a metric-based assessment approach to provide a more objective measure. As previously stated, work to define clear and useful metrics to describe software quality that can be applied to multiple artefacts has had limited success.

In addition to improving SAAM-SQ, more can be done towards understanding the factors that positively and negatively affect alignment between success-critical stakeholder groups in terms of software quality.

Applying SAAM-SQ in additional contexts would help verify the generalisability of the factors supporting and impairing stakeholder alignment that were
identified as part of this thesis.

With a deeper understanding of the factors that support and impair alignment of software quality, it is possible to design work environments to better support alignment. This work would include identifying the factors that have the greatest effect on alignment, and working out how to integrate them into work environments if they support alignment, and mitigate the risks if they impair alignment.

Finally, given every context studied has identified some need for change, research should be conducted to understand what factors are limiting organisations ability to achieve these changes. If appropriate, greater support should be provided to help organisations overcome these factors.

1.10 Reading Guide

This thesis develops and refines a methodology, the *Stakeholder Alignment Assessment Method for Software Quality* (SAAM-SQ), to determine the alignment between success-critical stakeholders on the priority given to aspects of software quality for the software development process. It is applied in a series of case studies that are used to develop and refine the method, as shown in Figure 1.1.

The thesis is structured as a collection of papers. This means that most of the chapters are published or planned to be published in academic fora. Thus each chapter needed to contain sufficient detail on the background and methodology to be accepted for publication as a self contained paper. One of the results of this approach is that some elements are necessarily repeated between the chapters. Thus a reading guide is suggested in Table 1.3.

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Read</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter 2</td>
<td>All sections</td>
</tr>
<tr>
<td>Chapter 3</td>
<td>All sections</td>
</tr>
<tr>
<td>Chapter 4</td>
<td>All sections</td>
</tr>
<tr>
<td>Chapter 5</td>
<td>Start from Section 5.4</td>
</tr>
<tr>
<td>Chapter 6</td>
<td>Start from Section 6.4</td>
</tr>
<tr>
<td>Chapter 7</td>
<td>Start from Section 7.4</td>
</tr>
<tr>
<td>Chapter 8</td>
<td>Start from Section 8.4</td>
</tr>
<tr>
<td>Chapter 9</td>
<td>All sections except Section 9.2.2</td>
</tr>
</tbody>
</table>

As Chapter 1 and Chapter 4 cover the introduction, background and methodo-
ology of the case studies presented in Chapters 5, 6, 7 and 8, it is possible to skip these sections in the case studies.
Chapter 2

Software Quality Trade-offs

Abstract

Background: Software quality is complex with over investment, under investment and the interplay between aspects often being overlooked as many researchers aim to advance individual aspects of software quality.

Aim: This chapter aims to provide a consolidated overview the literature that addresses trade-offs between aspects of software product quality.

Method: A systematic literature map is employed to provide an overview of software quality trade-off literature. A deeper analysis is provided for the literature providing empirical research on the process and requirements artefacts.

Result: The results show a wide range of solution proposal being considered. However, there is insufficient empirical evidence to adequately evaluate and compare these proposals. Further a very large vocabulary has been found to describe software quality.

Conclusion: Greater empirical research is required to sufficiently evaluate and compare the wide range of solution proposals. This will allow researchers to focus on the proposals showing greater signs of success and better support industrial practitioners.

\(^0\) An extended version of this chapter is planned for publication as Sebastian Barney, Kai Petersen, Mikael Svahnberg, Aybiike Aurum and Hamish Barney, “Software Quality Trade-offs: A Systematic Map and Review.”
Software quality is far more complex than acknowledged by much of the literature addressing the topic. The interplay between software quality aspects are often overlooked as many researchers aim to increase individual aspects of software quality. The problems associated with under investment are obvious, but over investment is also problematic (Svahnberg et al., 2003; Regnell et al., 2008). Insufficient quality leads to a useless product. As the level of quality increases the product become useful, and some point the level of quality can give the product a competitive advantage. However, beyond a certain point the level of quality becomes excessive—requiring high levels of investment, but not providing any competitive advantage, or sufficient business value to offset the required cost.

Given both the importance and complexity involved in achieving the right balance of software quality there is value in consolidating the research addressing software quality trade-offs. Consolidating the knowledge will help practitioners access the approaches to software quality trade-off that are are most suitable to their given context, and identify research gaps that academics can focus greater study. Thus this chapter aims to provide an overview of the approaches used in software quality trade-offs, and the level of empirical evidence currently provided. Greater focus is then given to software quality trade-offs that focus on requirements and the software development process, as research has found a common understanding of quality helps achieve the expected quality requirements (Trienekens et al., 2010).

A systematic map of the software quality trade-off literature has been developed to understand the state of the research addressing this area. This systematic map was then used to identify the empirical research addressing software quality trade-offs specifically focused towards the software process and requirements, and the resultant publications were summarised and consolidated as part of a systematic review.

The remainder of this chapter is structured as follows. The background to the study is presented in Section 2.2. The systematic mapping and review process is detailed in Section 2.3. The results of the mapping study are presented in Section 2.4 and the systematic review is presented in Section 2.5. A general discussion is provided in Section 2.6, with conclusions drawn in Section 2.7.
2.2 Background

Quality is a complex and multifaceted concept (Garvin, 1984). This holds true for software.

The software development industry has traditionally defined software quality as ‘fit for purpose’ or ‘conforming to specification’ (Hoyer and Hoyer, 2001). Most research on software quality focuses on improving and optimising individual aspects of software product quality (e.g., maintainability, security, or usability). However, achieving the highest level of quality against a model or set of measures does not ensure a sufficient level of quality will be achieved (Svahnberg et al., 2003)—it is possible that the required level of quality is higher than that assumed by any given model or measure. Further, such sub-optimisation of software product improvements does not always make sense. Improving one aspect of software quality can cause improvements or degradation to other aspects of software quality (Johansson et al., 2001; Svahnberg et al., 2003).

There are numerous models of software quality to support the software development process (Kitchenham and Pfleeger, 1996). Commonly cited models include McCall’s Quality Model, Boehm’s Quality Model, Dromey’s Quality model and ISO9126 (2001). Most quality models are presented in a hierarchy of attribute, sub-attributes and occasionally metrics. Further, it is recommended that any quality model should be tailored to any specific context in which it is used.

More recently software engineering research has emphasised the need to consider software quality from a range of different perspectives and take a value-based approach to software quality (Boehm and Jain, 2006). This means ensuring that the level of quality delivered is acceptable to all the stakeholders upon whom the success of the product depends.

The value of a shared understanding of software quality between key stakeholders is increasingly being recognised. Organisations with highly aligned individuals and groups have been found to significantly outperform those organisations that do not share these high levels of alignment (Chan and Reich, 2007). However, achieving and maintaining alignment is an ongoing activity as priorities will naturally change over time for numerous reasons (Chan, 2002).

There are numerous approaches to reconciling conflicts between aspects of software product quality. Some of the most common include expert judgement, the Non Functional Requirements (NFR) Framework, Quality Function Deployment (QFD) and Theory W.

Expert judgement involves one or more experienced professionals using their experiences and knowledge to make a decision on an issue. The decisions are
not necessarily supported by modelling or numerical assessment.

The NFR Framework uses diagrams to relate non-functional requirement goals with different decisions that can be made in the design and operation of a system that affect it positively or negatively, allowing trade-offs to be identified and made (Chung et al., 2000). While this method makes the results of a choice to be made more explicit, it requires a set of common priorities to be identified to allow effective decisions to be made.

Quality function deployment (QFD) considers the priority of customer and technical requirements in achieving the goals of the system to help prioritise the requirements (Herzwurm et al., 2003). However, the other perspectives involved in the development of the software product are not considered.

Value-based software engineering (VBSE) recognises the problems created by conflicting perspectives in the software development process (Biffl et al., 2006). Central to resolving conflict in VBSE is Theory-W (Boehm and Jain, 2006), which involves identifying the success-critical stakeholder, determining a mutually acceptable set of objectives, and then working to ensure these objectives are achieved.

2.2.1 Summary of Previous Reviews

At the time of writing the authors were only able to find one systematic review addressing the issue of software quality trade-off decisions and analysis—published by Berntsson Svensson et al. (2010a). This publication addresses the more general topic of software quality management, but specifically aims to address issues of prioritisation between aspects of software quality. However, on the topic of software quality prioritisation this review was only able to identify three relevant articles. This weakness is caused by the limited number of keywords used to define the prioritisation concept within the search string.

The authors hope that by conducting a more focused and detailed study, the topic of software quality trade-off decisions and analysis can be better understood.

2.2.2 Research Questions

The aim of this chapter are twofold.

The first research question seeks to provide an general overview of the recent literature addressing software quality trade-offs. This question has been further subdivided into a series of sub-research questions:
2.3 Method

- **RQ1.** What is the state of the art with respect to software quality trade-off literature?
  - **RQ1.1.** What are the common publication fora?
  - **RQ1.2.** Who are the key researchers in the field?
  - **RQ1.3.** Which research approaches are employed?
  - **RQ1.4.** During which development phase does literature suggest for software quality trade-offs to take place?
  - **RQ1.5.** What approaches are being used to make software quality trade-offs?

The second research question seeks to provide a more detailed description of the empirical research addressing software quality trade-offs for the process and requirements artefacts. Both of these artefacts involve a range of stakeholders early in the development process and are at a relatively high level compared to architecture and coding. Achieving alignment as early as possible is important, as quality problems become increasingly expensive to fix the longer they go unaddressed. This creates the possibility for these methods to help align stakeholders working on the product, which has been found one of the most effective ways of achieving software quality. This research question has also been further subdivided into a series of sub-research questions:

- **RQ2.** What is the state of the art for empirical research addressing software quality trade-offs process and requirements artefacts?
  - **RQ2.1.** What approaches have been proposed?
  - **RQ2.2.** What is the purpose of these approaches?
  - **RQ2.3.** What level of empirical evidence is provided for this work?

2.3 Method

A systematic map and systematic literature review are used to address the research questions posed in this chapter. Addressing RQ1, a systematic map can provide an overview of the literature addressing software quality trade-offs. The systematic mapping process can also be used to identify empirical publications addressing software quality trade-off literature focused on the software development process and requirements engineering. A systematic review can then be used to provide a detailed description of the identified publications, answering RQ2.
2.3.1 Research Process
The methodology employed in this work is based on the systematic review guidelines presented by Kitchenham (2007). It has been further informed and shaped by the experiences of Dybå and Dingsøyr (2008); Petersen et al. (2008); Riaz et al. (2010).

The following procedure was followed in conducting the systematic mapping and systematic review:

1. First bibliographic databases were selected from which to search for publications. The rationale and choices are presented in Section 2.3.2.

2. Keywords describing the research area were identified, and search strings were created for the selected databases. This process is described in Section 2.3.2.

3. The search strings were then run against the databases, with the bibliographic information for all resultant publications extracted and saved.

4. The list of publications from each database were combined, with duplicate records for a publication removed.

5. The title and abstracts of each publication was then reviewed by two reviewers against a set of inclusion and exclusion criteria. This process is described in Section 2.3.3.

6. Publications that were marked for both inclusion and exclusion were then re-reviewed and selected for inclusion or exclusion, as described in Section 2.3.3.

7. The resultant list of publications were then classified for the systematic map. This process is described in Section 2.3.4.

8. Finally a data extraction was undertaken on the relevant publications. This process is described in Section 2.3.5.

A summary of this process is provided in Figure 2.1.

2.3.2 Databases, Keywords and Search Strings
After defining the research questions, the first step of conducting the systematic map and review was to identify the set of databases to be used to find publications. As this systematic review addresses software engineering, databases
2.3 Method

Identify Keywords → Search Strings → IEEE Xplore → ACM Digital Library → ... → List of Papers → Remove Duplicates → Updated List → Review Abstracts → Updated List → Review Conflicts → Updated List → Classify Papers → SLR Papers → Systematic Review → Review Data

Figure 2.1: Research process
were selected for their coverage and use in this domain. The following set of databases were selected to identify relevant publications:

- ACM Digital Library
- Compendix/Inspec
- IEEE Xplore
- ISI Web of Science
- Scopus

While the authors wanted to include SpringerLink, this database was excluded due to its inability to handle complex queries.

The next step was to identify the set of keywords. These keywords are used to create search strings, which are run against the selected publication databases.

Keywords were identified in an iterative approach with several steps. First, relevant papers were identified, and appropriate keywords were extracted. Then thesauri were used to identify additional keywords based on the extracted keywords. Search strings were then created and run against various databases to identify additional relevant publications. This process was repeated until there were no significant developments to the keywords. This process was conducted by the first author, and then reviewed by coauthors.

The keywords were classified into four groups—software (C1), engineering (C2), product quality (C3) and trade-offs (C4). These categories and the final list of keywords are presented in Table 2.1. To increase readability, all of the three word combinations starting non-functional or non functional are listed together without repeating the the first two words. Further, each pair of ISO 9126 aspects was inserted. The AND operator was used between the two items making each pair, while the OR operator was used between each pair.

The search string was created by putting the OR operator between all of the words or phases within each category, and putting the AND operator between each category. Thus the main search string was composed as:

C1 AND C2 AND C3 AND C4

Where, for example, C2 was composed as:

(engineering OR development OR product OR service OR system)
### 2.3 Method

#### Table 2.1: Search string keywords

<table>
<thead>
<tr>
<th>Category</th>
<th>Label</th>
<th>Keyword(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>Software</td>
<td>software, development, product, service, system</td>
</tr>
<tr>
<td>C2</td>
<td>Engineering</td>
<td>engineering, development, product, service, system</td>
</tr>
<tr>
<td>C3</td>
<td>Product Quality</td>
<td>ISO9126, ISO 9126, ISO-9126, IEC9126, IEC 9126, IEC-9126, non-functional+</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(artefacts, artifacts, aspects, attributes, capabilities, characteristics, factors, features, properties, requirements, traits),</td>
</tr>
<tr>
<td></td>
<td></td>
<td>all ISO 9126 quality aspect pairs</td>
</tr>
<tr>
<td>C4</td>
<td>Trade-offs</td>
<td>tradeoff, &quot;trade off&quot;, &quot;trade-off&quot;, priorit*, balance, collaborat*, compromis*, &quot;cumulative voting&quot;, &quot;analytic hierarchy process&quot;, AHP, vote, voting</td>
</tr>
</tbody>
</table>

The search queries were further limited to publications printed in 2005–2010. This authors considered this six-year period to cover ‘recent’ publication, and that any previously identified approach not revisited during this time unlikely to be still considered relevant.

Applying the searches created from this process to the selected databases yielded 2930 publication, as presented in Table 2.2. Combining the results from each database and removing duplicates reduced this list to 2153 publications. However, further duplicates were found later in the process.

#### 2.3.3 Inclusion and Exclusion Criteria

The 2153 publications identified by the search strings were then reviewed against a set of inclusion and exclusion criteria.

The inclusion and exclusion criteria were developed in a review of publications from 2009 by the first author. Articles were reviewed with rules being developed and refined to support the decision making process. These rules were then reviewed and agreed upon by the coauthors.

The final set of rules, applied to the review of all abstracts, is set out below:
Table 2.2: Results from each database

<table>
<thead>
<tr>
<th>Database</th>
<th>Publications</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACM Digital Library</td>
<td>113</td>
</tr>
<tr>
<td>Compendix/Inspec</td>
<td>493</td>
</tr>
<tr>
<td>IEEE Xplore</td>
<td>965</td>
</tr>
<tr>
<td>ISI Web of Science</td>
<td>108</td>
</tr>
<tr>
<td>Scopus</td>
<td>1248</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>2930</strong></td>
</tr>
</tbody>
</table>

- Included results must be in English.
- Included results must not come from an excluded source:
  - Books are excluded, however, book chapter from an edited volume may be included.
  - Forewards are excluded regardless of their source.
  - Editorials are excluded.
- Included results must address software development.
- Included results must address multiple aspects of software product quality (for example, see ISO9126 (2001)).
  - Aspects of product quality not relating to a software product are out of scope in this study (eg. the reliability of a car is out of scope, but the reliability of the software running a car is in scope).
  - Aspects of service quality that do not directly relate to software product quality are out of scope in this study (eg. reliability can be considered an aspect of service quality and product quality, customer service does not).
  - Aspects of process quality that do not directly relate to software product quality are out of scope in this study (eg. having a reliable process is different to having a reliable product).
- Included results must address trade-offs between at least two aspects of software product quality.
2.3 Method

- Results that allow direct comparisons between aspects of software product quality can be included in this study.

- Results only seeking to improve one or more aspects of software product quality without a trade-off between them are excluded from this study.

- Results that perform trade-offs between a single aspect of software product quality and some aspect non-software product quality aspects are excluded.

- Results that perform trade-offs between multiple aspects of software product quality and an other aspect or aspects are included.

Each publication received two independent reviews against the inclusion and exclusion criteria. The first author reviewed all publications, while the other four authors reviewed one quarter of the publications each. In this process 140 publications were marked in scope, while 1866 publications were marked as out of scope and 147 publications had conflicting reviews.

Where the two reviews were in agreement a publication no further action was taken, with the paper being marked in or out of scope as appropriate. The 147 publications with conflicting reviews were given to the second author to get a third and final assessment\(^1\). This resulted an additional 39 publications being marked in scope, resulting in 179 publications being marked in scope.

2.3.4 Classification for Systematic Map

Having identified the papers in scope, the next step was to classify the publications for the systematic map—answering RQ1. For each publication the following data was collected from the abstract, unless otherwise stated:

- Qualities: The aspects of quality considered as part of the trade-off in the publication.

- Trade-off approach: The approach used to support software quality trade-offs. If this could not be determined from the abstract, the introduction and conclusion where read.

\(^{1}\)Of the 147 publications in conflict, 14 were previously reviewed by the second author. The second author changed their opinion on four of the publications, suggesting they were not subject to the same biases as in their first review.
• Development Artefact: The development artefact to which the trade-off was applied. The options considered were process, requirements, architecture, code, test and runtime.

• Research approach: The research approach used according to the classification system proposed by Wieringa et al. (2006), as recommended in Petersen et al. (2008). The research approaches are opinion papers, philosophical papers, experience reports, solution proposals, validation research and evaluation research. Where this could not be confirmed, the introduction and conclusion were read.

• Authors: The authors who wrote the publication.

• Venue: The publishing journal, conference, workshop or book.

• Year: The year of publication.

During the classification process 11 publications were excluded for the reasons stated in Table 2.3. The final list of publications totalled 168.

<table>
<thead>
<tr>
<th>Reason</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Less than two quality aspects studied</td>
<td>3</td>
</tr>
<tr>
<td>Previously unidentified duplicates</td>
<td>2</td>
</tr>
<tr>
<td>Process quality, not software quality</td>
<td>2</td>
</tr>
<tr>
<td>Foreward (excluded source)</td>
<td>2</td>
</tr>
<tr>
<td>Publication not in English</td>
<td>1</td>
</tr>
<tr>
<td>Outside period studied</td>
<td>1</td>
</tr>
</tbody>
</table>

### 2.3.5 Data Extraction for Systematic Review

Each publication classified by the systematic mapping process as empirical research addressing software quality trade-offs for the process and requirements artefacts was further studied in a systematic review. This scope was selected as it covers the artefacts used to ensure that stakeholder involved in software development have a common understanding of software quality. Artefacts from later stages, such as architecture and code, are more technical in nature.

The full publication was reviewed, and the following information was collected:
2.3 Method

- The aim of the doing the software quality trade-off.
- The approach supporting the software quality trade-off.
- A brief description of the process used to conduct the software quality trade-off.
- A brief description of the empirical evidence presented in the paper.
- A classification of the paper against the rigour and relevance criteria for empirical research in software engineering, as proposed by Ivarsson and Gorschek (2010).

In total 23 publications were identified from the systematic map as within scope of the systematic review. However, on closer inspection, three of these papers were found to be out of scope, as discussed in Section 2.5.1. Further, the authors are still trying to get a copy of one of the publications that has not been made available online.

2.3.6 Validity Threats

This section presents the key validity threats in this research and how they have been addressed.

Threats to conclusion validity limit the ability to draw conclusions from the research (Wohlin et al., 2000). For a systematic review the main risk concerns the ability of the authors to accurately select papers for inclusion and extract data. To further complicate matters, a number of researchers who have completed systematic reviews have noted incorrectly used terms leading to incorrect classification of research (Petersen et al., 2008). However, this chapter does not endeavour to provide a deep understanding of the research field studied, but provide an overview that allows for more detailed analysis in future work. To this end the authors of the publications identified as part of this systematic review have been given the benefit of the doubt.

Construct validity concerns the ability to generalise the results (Wohlin et al., 2000). Given the limited nature of the analysis done as part of a the systematic mapping and review process, it is possible that aspects of the publications are not understood by the authors of the systematic review. This risk is reduced given that the aims of the research presented in this paper are to provide an overview of the research area. Further, the scope of the review is limited, reducing the possibility for misinterpretation.
The aim of a systematic review is to capture as much of literature on a topic as possible to avoid bias. This risk has been managed in several ways for this research. The search strings were developed iteratively with piloting to ensure sufficient keywords were captured. All papers were reviewed by at least two authors to reduce the risk of individual bias. While it is possible that a replication of the study would result in minor differences in the papers included and the classification, the authors perceive the differences would be minimal.

2.4 Systematic Map

In total 168 publications identified and analysed as part of the systematic map. Splitting the publications by the year in which they were published show a general increasing trend in the number of publications, peaking in 2009 with 40 publications. A breakdown of these results is shown in Figure 2.2. Comparing the 28 publications for 2010 against the 40 publications for 2009 provides insufficient data to know if this is the start of a downward trend or is an anomaly in the results.
2.4 Systematic Map

2.4.1 Key Publication Venues

The 168 publications identified as part of the systematic map were published at 121 different venues. Conferences were by far the most common venue type with 106 publications, followed by journals with 38 publications and then workshops with 24 publications.

The publication venues with more than two identified publications from the systematic map are listed in Table 2.4. This list of venues covers 24% of the identified publications. The venues include six conferences, four journal and one workshop.

Table 2.4: Venues with more than two identified publications

<table>
<thead>
<tr>
<th>Type</th>
<th>Venue</th>
<th>Publications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conference</td>
<td>ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing</td>
<td>5</td>
</tr>
<tr>
<td>Journal</td>
<td>Information and Software Technology</td>
<td>4</td>
</tr>
<tr>
<td>Journal</td>
<td>Journal of Systems and Software</td>
<td>4</td>
</tr>
<tr>
<td>Conference</td>
<td>Asia-Pacific Software Engineering Conference</td>
<td>4</td>
</tr>
<tr>
<td>Conference</td>
<td>International Conference on Software Engineering</td>
<td>4</td>
</tr>
<tr>
<td>Conference</td>
<td>International Working Conference on Requirements Engineering</td>
<td>4</td>
</tr>
<tr>
<td>Journal</td>
<td>IEEE Software</td>
<td>3</td>
</tr>
<tr>
<td>Journal</td>
<td>Software Quality Journal</td>
<td>3</td>
</tr>
<tr>
<td>Conference</td>
<td>Empirical Software Engineering and Measurement</td>
<td>3</td>
</tr>
<tr>
<td>Conference</td>
<td>Euromicro—Software Engineering and Advanced Applications</td>
<td>3</td>
</tr>
<tr>
<td>Workshop</td>
<td>International Workshop on Software Product Management</td>
<td>3</td>
</tr>
</tbody>
</table>

All of the venues with more than two identified publications are focused on various aspects of software engineering. Some venues, like the *Software Quality Journal* and the *International Working Conference on Requirements Engineering*, focus on software quality. However, none of the venues are focused specifically on trade-offs within software engineering.
The list of venues presented in Table 2.4 are all highly ranked in different fora. With the exception of the *International Working Conference on Requirements Engineering*, all these venues are or have been listed in the *ISI Web of Knowledge*, although not all years of all conferences and workshops are listed. Further, all venues are listed in the Australian Core Ranking for Information System, except for *ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*.

### 2.4.2 Key Researchers

The 168 publications included in the systematic map list 483 authors, which corresponds to 396 people. Most people were only listed on one publication, but 65 people were listed on two or more publications. The list of authors listed on three or more publications is presented in Table 2.5, with the number of publications with which they are associated from the systematic map.

<table>
<thead>
<tr>
<th>Rank</th>
<th>Author</th>
<th>Publications</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Berntsson Svensson, Richard</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>Regnell, Björn</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>Reussner, Ralf</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>Martens, Anne</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>Barney, Sebastian</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>Daneva, Maya</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>Davidsson, Paul</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>Huang, Gang</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>Lee, Dan Hyung</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>Mei, Hong</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>Ramdane-Cherif, Amar</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>Svaahnberg, Mikael</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>Wohlin, Claes</td>
<td>3</td>
</tr>
</tbody>
</table>

The results show a number research collaborations. Berntsson Svensson and Regnell coauthored all six of the publications listed under their name; Reussner and Martins coauthored four publications; Huang and Mei coauthored three publications; Barney and Wohlin coauthored two publications; Davidsson and Svaahnberg coauthored two publications; and Svaahnberg and Wohlin coauthored one publication.
2.4.3 Research Approach and Focal Development Artefacts

This section breaks the 168 identified publications by the research approach taken and development artefact that is the focus of the publication. The classification of research approach is done using the classification system of Wieringa et al. (2006). Publications were also linked to the development artefact to its most strongly related—process, requirements, architecture, code, testing or runtime. The results of this analysis are presented visually in Figure 2.3.

![Figure 2.3: Number of publications per research facet and development artefact](image)

Within recent software quality trade-off research, solution proposals stand out as the dominant research approach. This approach is used in 61% of the
identified publications. By contrast, the second most common research approach used is *evaluation research*, covering 21% of the publications. Of the remaining publications, 8% employ *validation research*, 5% present *philosophical papers*, 3% present *opinion papers* and 2% present *experience reports*.

When breaking down the papers by development artefact, another strong tendency is observed. Half of the identified publications look at software quality trade-offs within the architecture domain. The second most common development artefact address by recent software quality trade-off is the software development *process*, with 24% of publications, followed by *requirements engineering* with 17% of publications. Little attention has been given to software quality trade-offs at *runtime*, in *code* and during *testing*, with 5%, 3% and 2% of the identified publications respectively addressing these development artefacts.

Looking at the identified publications in terms of both the research approach and the development artefact provides some interesting insights. Given the clear focus on the *architecture* development artefact, one might expect a greater level of maturity with research, with a higher proportion of empirical research. This was not the case, however, with only 23% of publications addressing the *architecture* development aspects employing *validation* or *evaluation* research. The comparison values for the *process* and *requirements* development artefacts are 34% and 32% respectively.

### 2.4.4 Quality Aspects Studied

Publications were classified according the the aspects of software quality covered in the abstract. A break-down of these results is provided in Table 2.6.

<table>
<thead>
<tr>
<th>Quality Aspects</th>
<th>Publications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software Quality</td>
<td>121</td>
</tr>
<tr>
<td>Requirements</td>
<td>23</td>
</tr>
<tr>
<td>ISO 9125</td>
<td>11</td>
</tr>
<tr>
<td>Specific Pair</td>
<td>9</td>
</tr>
<tr>
<td>Specific Group</td>
<td>4</td>
</tr>
</tbody>
</table>

From the abstracts, most papers (72%) address software quality aspects generally—this includes aspects of quality (eg. ISO 9126) and quality requirements (eg. boot time must be less than three seconds). For most publications it was not possible to ascertain which specific qualities were studied from the
abstract. Quality aspects and quality requirements were combined as it was not possible to differentiate this level of detail from the abstracts given the range and manner in which terms used to describe software quality. Papers specifically mentioning ISO 9126 in the abstract were separated from this group, with 11 publications identified (7%) using this quality model specifically.

Requirements more generally, and specifically including quality requirements, are addressed in 23 publications (14%).

In total 9 publications (5%) addressed specific pairs of quality. Only the performance/reliability pair was studied more than once, with two publications. However, performance was the most commonly studied quality aspect in this group, included in five publications.

Specific groups of quality were studied in four publications (2%). Of these two addressed groups of quality aspects relating to security, with the others addressing different sets of aspects.

2.4.5 Trade-off Approaches

A very wide array of trade-off approaches have been employed in the identified publications. The most commonly employed approaches are listed below, with the corresponding number of publications in brackets. This list covers 55% of the identified publications.

- Model-based approaches (23)
- Approaches based on Analytical Hierarchy Process (AHP) (21)
- Approaches employing multiple approaches (11)
- Algorithmic-based approaches (9)
- Approaches based on Architecture Tradeoff Analysis Method (ATAM) (8)
- Metrics-based approaches (7)
- Expert opinion (5)
- QFD-based (5)
- Prototyping (4)
However, this list presents a simplified view of the approaches taken. For example, within the model-based approaches different models are used (eg. UML, other architectural models) and the models are used in different ways to make the trade-off. Similarly the AHP-based trade-off approaches varied in their implementation. Expanding out the list of approaches to identify each unique approach would result in almost as many approaches as identified publications.

A selection of some of the approaches used in the other papers include an agent-based approach, artificial intelligence (AI) based approaches, game theory, a goal-question-metric (GQM) based approach, optimization theory, Pareto curves, a perspective-based approach, a risk-driven approach, search-based techniques, utility theory and various statistical methods.

### 2.4.6 Systematic Mapping Discussion

The greatest surprise for the authors in conducting this research was the range of trade-off approaches used in the publications identified. While the authors had hoped to develop a classification of different approaches for software quality trade-offs, this proved neither feasible nor useful given the diverse range. This result suggests that there is an immaturity in the field in that no trade-off approach or set of approaches have emerged as candidates to dominate the research space.

While it is positive to see such a wide range of trade-off approaches are being considered, it creates concern that a clear majority of the research (62%) is classified as solution proposals. By contrast only 28% of the publications identified present empirically validated results, either validation research or evaluation research. While it is important in immature fields to consider a wide range of options, the lack of empirical research means only limited comparisons can be made between the strengths and weaknesses of the various techniques. More information on how each method performs in practice would allow for greater focus on the methods displaying greater strength. Ultimately, the aim of any engineering should be develop and refine something of use.

The next question is to ask why so many of the solution proposals are not being submitted to empirical validation. There are a number of possible reasons for this result. Firstly, it is easier to propose a solution, than to propose and validate/evaluation a solution. However, methods that could be of value should be used, confirming if indeed this is the case. Secondly, it is generally easier to publish positive results than negative ones. This situation is unfortunate, as negative results would allow researchers and practitioners alike to better navigate the set of solution proposals.
Another interesting result is the very dominant focus of the quality trade-off approaches on the earlier phases of software development—process, requirements and architecture. This is

It is interesting to see that the research is focused in the early phases. This makes sense, as problems are very costly to fix later. For example, if the architecture is wrong, then trying to fix the problem at runtime may not be possible.

2.5 Systematic Review

This section presents an overview of the 23 publications that present empirical results towards addressing software quality trade-offs for the process and requirements artefacts. First, the papers are classified by the approach used for the trade-off, with a brief description of the aim, method and empirical results presented in each paper. Then an analysis is made of the rigour and relevance of the research using the model proposed by Ivarsson and Gorschek (2010). The section concludes with a discussion of the results.

2.5.1 Overview of Approaches and Publications

This section provides an overview of all 23 publications identified as within scope of the systematic review.

Software Quality Trade-offs in Practice

Berntsson Svensson et al. (2009) sought to understand how practitioners find the right balance among competing quality requirements in the software development industry. An interview study was conducted for this research. In total five companies participated in the research, with one product manager and one project manager from each. The results showed that most companies did neither do any analysis of the interdependencies between quality requirements, nor between quality requirements and functional requirements. This meant that trade-offs could not be made between different quality requirements. Further, the results highlighted some differences between the product manager and the project manager in terms of the priorities they gave to different aspects of software quality.
The Analytic Hierarchy Process (AHP) proved to be the widely used tool for software quality trade-offs within the scope of the systematic review. It was used in three publications.

Trienekens et al. (2010) aim to implement a quality model in the software development process for a combat management system (CMS). As part of this work, they aim to determine the relative importance of the software quality aspects to support the realisation of these qualities in the development process.

The authors propose a method to tailor the ISO 9126 model to the specific needs of the software product studied. Working with the various stakeholder group aspects of quality were defined in detail for the system. The need to involve different perspectives was identified. The relative importance of the aspects that make up the tailored quality model where then calculated using the AHP method.

This information was found to help all stakeholders—from helping users express their expectations to helping developers design solutions. The authors report that the method is now adopted more widely within the studied organisation.

Onut and Efendigil (2010) propose the use of AHP to evaluate different Enterprise Resource Planning (ERP) systems in terms of aspects of cost, quality and vendor reputation.

The method first describes the criteria used to assess the different solutions. Each solution is evaluated against these criteria. Then a group of stakeholders representing a range of perspectives use the AHP method to determine the relative importance of the different criteria used to assess the solutions. The AHP method is then able to calculate a value for each solution based on the relative importance of each criterion and the assessment of each product against the criteria.

A case study is presented with a Turkish manufacturing company applying the method to help select an ERP. In the case study two approaches are taken—using conventional AHP and using fuzzy AHP. The two AHP-based approaches both resulted in different values being calculated for the different system, but the preference order was the same for both approaches. From the authors’ experience, the fuzzy approach was better from the perspective of the users of the method. However, it is not stated what role the result had in the final selection of an ERP system.
Liu and Pang (2010) aim to provide a method for evaluating the factors necessary to evaluate software quality.

The proposed method uses fuzzy matter element (FME) to assess candidate solutions against a set of qualities that are defined as part of this process. The results are presented in a matrix, with each column representing a solution, each row representing a quality, and each cell representing the level of quality for the solution. The Analytic Hierarchy Process (AHP) is then used to determine the relative importance of each of the qualities, and to determine a single quality number for each of the proposed solutions. The method has been designed to be generic, and is not linked to any specific software artefact, and does not define who should be involved in this process.

A case study is conducted to evaluate the method—selecting the highest quality software product. However, no information is given about the context in which the case study was conducted, nor explanation of how the numbers were determined. No evaluation of how the method performed is provided, other than to say it was successful.

**QUPER Model**

The QUPER Model was found to be one of the most researched approaches for software quality trade-off. It is presented in Regnell et al. (2007); Berntsson Svensson et al. (2008, 2010b). Parts are further used in Vanhanen et al. (2009), which is described in the next section. This model aims to support releases for the quality aspects of a software product. It applies to upstream requirements engineering activities, such as road-mapping, release planning and platform scoping.

The QUPER Model provides three views on quality—the benefit view, the roadmap view and the cost view. It recognises that the benefits of increasing quality are not linear, splitting the benefits into four categories—useless, useful, competitive and excessive. It also recognises that cost of improving quality is not linear, with some small improvements coming at great cost while some big improvements can be made cheaply.

To use the QUPER Model, quality indicators are selected that measure the aspects of quality of interest. An analysis of the product with respect to the cost and benefit views is made. The authors also look at the level of benefit offered by competitors. This information is then used to plan the level of quality to be achieved in future releases.
Regnell et al. (2007) present the results of an interview study with six software engineering practitioners from the mobile telecommunications handset domain. Each were asked to review the QUPER Model within the context of their own software product development activities. The results showed the respondents to be positive to the model, but they were uncertain how many quality indicators would be required in practice, and perceived the need for greater support when making trade-offs between different qualities.

In Berntsson Svensson et al. (2008) the QUPER Model was applied by industry practitioners in four case studies from the mobile telecommunications handset domain. The experiences of the practitioners gave them a greater understanding of what constituted a sufficient level of quality. They also recognised that changing technologies and market conditions could alter the benefit and cost breakpoints over time. The users of the model stated it was easy to learn and use.

Berntsson Svensson et al. (2010b) present the first use of the QUPER Model outside the mobile telecommunications handset domain, with a case study in a company processing electronic payments. The practitioners from this company found the model easy to use and the support provided useful. However, as they had moved from having no support, they found a new process time consuming. With regard to the benefit view, it was found difficult to gather competitor information and determine the quality saturation breakpoint. With regard to cost barriers, it was found far easier to determine the next cost-barrier than the cost barriers beyond this point.

Quality Attribute Workshop

Vanhanen et al. (2009) present a method to elicit, prioritise and elaborate software qualities goals. It is aimed at market-driven software product development, with the intention of using the results to practices to improve the identified quality goals.

The proposed method is based on Quality Attribute Workshops (QAW), used for quality trade-offs affecting architecture. The proposed approach brings together people representing different quality perspectives (eg. business, end user and project manager) into a workshop. The workshop has three main steps, (1) to elicit quality goals through brainstorming, (2) to prioritise the identified quality goals through voting, and (3) to elaborate on the most important quality goals—including drawing on the QUPER Model’s benefit and cost views.

The method has been applied in four industrial case studies. The results from each study were used to develop and refine the method. While each of
the four companies responded positively to the method, the results differ to the authors expectations. The workshops took 20–40 hours, and participants often focused on quality aspects where they perceived the company as weak—which did not necessarily correlate with those that were most important.

2.5 Systematic Review

Groupware Supported Negotiation and Collaboration

A number of publications pick up on the need for negotiation and collaboration between various stakeholders when considering software quality trade-off situations. Groupware was suggested by two papers as one possible way to help facilitate this process.

Ramires et al. (2005) propose the use of groupware to support the implementation of software Quality Function Deployment (QFD) and subsequent negotiations.

QFD aims to test the alignment between user requirements and technical specifications. ‘Correlation’ values are used to indicate how well the technical specifications meet the user requirements. One of the methods for determining the correlation values is in a group meeting, which can be time consuming and difficult to organise. Groupware is seen as one possibility to avoid the problem created with the meeting.

A prototype of the groupware was created as part of this research, with an experiment being made with industrial practitioners. Even on a reduced-scale problem, the experiment participants found the software difficult to use, but perceived that it did support the intended process and that achieving a mutual consensus was beneficial.

Linhares et al. (2009) propose using a groupware solution to support negotiation and collaboration during formal technical reviews. This research is heavily based on the work of Ramires et al. (2005).

Linhares et al. (2009) recognise that different roles develop conflicting understanding of functional and non-functional requirements. This situation necessitates negotiation and collaboration, with the best solutions being achieved through consensus. Groupware can support this process by creating an environment that can bring together a diverse range of perspectives, and support the development of a collective understanding. The groupware solution supports stakeholders collaborate by sharing information and proposed solutions. From here it is possible to review and negotiate with arguments for or against proposed solutions to create win-win scenarios. Each solution proposal is reviewed
with reviewers stating giving arguments on why the solution is acceptable or not acceptable, with possible improvements.

A prototype of the software was developed, and an industrial case study was conducted. The review team was split, with one group conducting a traditional technical review, while the other group used the software prototype. The strength of the method is based on the number of solution proposals, arguments, and times people changed their position on whether a solution was acceptable. While the groupware-based solution won on all three criteria, the group using this solution had more members. Further, the study was limited to two aspects of the system studies, making the study small. Although an interview study showed that the participants found the approach to be good.

**Hierarchical Cumulative Voting**

Drawing on the need to align key stakeholder groups, the publication presented in Chapter 5 presents a method to determine the alignment between these groups on the priority given to aspects of software product quality. Understanding shared and conflicting priorities with respect to quality is an important step in being able to bring the key stakeholders together to work towards a common goal.

The method proposes three main steps; (1) to define a model of software quality appropriate to the context being studied, (2) to identify the success-critical stakeholders priorities with respect to the aspects in the software model created using the Hierarchical Cumulative Voting (HCV) technique, and (3) to analyse the results to determine and understand the reasons for aspects of quality with shared or dissimilar priorities.

The results of an industrial case study are presented in the publication. While the identified stakeholder groups showed high levels of alignment, the method is still able to observe differences associated with the participants role (eg. developers perceive maintainability to be more important than other groups).

The publication presented in Chapter 9 extends the work done in Chapter 5 to describes a method to quantify the priorities on aspects making up scope, quality, intellectual capital investments, time constraints and cost constraints for a software product. The resultant values highlight the aspects where success-critical stakeholder groups perceive the need for increased investment, decreased investment or no change in the investment levels.

The method presented in Chapter 9 exploits the hierarchical nature of the
2.5 Systematic Review

method and data presented in Chapter 5 to bring together priorities relating
to the scope, quality, intellectual capital investment, time constraints and cost
constraints. After describing what is happening, and what participants perceive
should be happening, it is possible to compare the results and determine areas
where change is perceived beneficial.

The approach is applied to a case study, extending the results presented
Chapter 5 with new data. This method can be applied at any stage during the
development process. It is recognised by the authors that the priorities on the
aspects examined are likely to change over the life of the product.

Levels of Importance

Sibisi and Van Waveren (2007) recognise the value of quality models like ISO 9126,
and the need for these models to be tailored to be useful in a specific context.
Thus these authors propose a framework to tailor a generic quality model to
meet the needs of specific organisations, and perspectives within these organi-
sations.

While this proposal is not directly inline with software quality trade-offs,
the proposal uses values showing relative importance in the development of the
method—providing trade-off information in the process.

The framework provides a method for developing a questionnaire, which aims
to determine the level of importance of each quality aspect included. Using this
information it is possible to exclude aspects below a certain importance level,
as it is the other aspects that should be the focus of any development work.

The results of two industrial case studies are presented in the publication.
The results show that the models of software quality are sufficient at the char-
acteristic level, but not at the sub-characteristic level. More work is needed to
evaluate the benefits and cost-effectiveness of the approach. It also needs to be
verified in other contexts.

Multi-criteria Methodology Decision Aiding

Lacerda et al. (2009) aim to develop a methodology to identify, measure and in-
tegrate project success factors. Given the high rate of project failure, this paper
questions whether project success and failure has been adequately defined, as it
is possible for all stakeholders to be satisfied by a project that technically fails
to meet some evaluation criteria, such as not all scope being fully implemented.
When addressing quality, the authors perceive a need to move towards a more
nuanced and perspective-based understanding of quality.
The proposed methodology is built on Multi-criteria Methodology Decision Aiding–Constructivist (MDAC-C). It helps to develop a model handling multiple criteria in three phases—structuring, evaluation and recommendations. First key stakeholders are identified, and they identify the areas they are most interested in evaluating. These areas are then clustered and put into logical hierarchies. In the evaluation phase each of the areas are defined metrics on which areas that can be evaluated. In the final phase alternative options are considered with the impact on the metrics evaluated.

The method was applied in an industrial case study. It is claimed the methodology helped decision-makers focus on key areas, and improving project success in terms of the new metrics. However, there is insufficient detail to sufficiently understand this case.

Pragmatic Quality Factors

Yahaya and Deraman (2010) propose the use of pragmatic quality factors (PQF) to balance technical quality (e.g., metrics) with human-centred quality.

The method described in the publication involves a literature study and industrial survey to help identify and define the aspects of quality of technical and human-centred quality. Weights are then defined for the different aspects of quality based on their importance. The approach to assign these weights is not prescribed in the paper, but references are provided.

The approach was applied in a case study, although little information is given about the context of the study. The case study shows that the method can be applied, but its industrial value is not evaluated.

Automated Negotiation

Patankar and Hewett (2008) present an approach for procuring web-services. Web-service procurements is split into three phases, (1) discovery of web-services, (2) negotiation and (3) contracting. The proposed method is focused on the negotiation phase, proposing a set of algorithms to support the negotiation process between the party seeking to procure a web-service and parties providing that web-service.

The party seeking to procure a web-service can set desired and required levels of quality and cost constraints. Web-service providers are able to advertise the levels of quality they can meet for given prices. The method then proposes algorithms to make offers and counter offers until an agreement is reached—varying levels of quality and price to try and find a mutually acceptable service.
The possibility that no service provider can meet the required levels of quality or price is recognised, with options to accept lower levels of quality or higher price, or to terminate negotiations.

The authors have applied the approach to an case study in the medical insurance industry, however, it is unclear to the reader if this is an actual case study or a theoretical example.

SQUARE

Mead and Stehney (2005) provide an evaluation of the System Quality Requirements Engineering Model (SQUARE), which aims to provide a method to elicit and prioritise security requirements.

SQUARE defines nine steps to elicit and prioritise security requirements. For each step, the inputs, techniques, participants and outputs are defined. The main phases are (1) to agree on definitions, (2) identify security goals, (3) develop supporting artefacts, (4) perform risk assessment, (5) select elicitation techniques, (6) elicit security requirements, (7) categorise requirements, (8) prioritise requirements, and (9) conduct requirements inspections.

With regard to step (8), prioritise requirements, the SQUARE model is not prescriptive on which approach should be used. Mead and Stehney (2005) cite triage and win-win as two methods, but acknowledge there are more that could be used. The participants required for this step are the most diverse, just described as stakeholders.

The evaluation of the method is provided by an application of the SQUARE Model in a company for a product currently on the market. University students applied the model, but after failing to complete all the required steps to a satisfactory level a second group of students was assigned the task. The results show that the students had difficulty completing the task within the allocated time. The first groups submission was not perceived valuable to the company as it did not align with their business goals. However, these issues were largely addressed by the second group. After this empirical work, some minor changes were made to the SQUARE model, and it is thought that it may be more valuable if conducted prior to the commencement of software development, but more work is needed.

Prospect Theory

Fogelström et al. (2009) identify prospect theory as a possible bias when selecting and prioritizing requirements for a release of software. The proposition
behind prospect theory is that people do not perceive the value from gains and losses linearly—that is losses will be perceived larger than the same size gains, and that doubling a gain or loss does not double the perceived value of the gain or loss. This publication seeks to understand if and how this decision-making bias applies to requirements selection and prioritisation activities.

The publication is focused on software product managers in a market-driven software product development setting, but could apply to anyone responsible for prioritising and/or selecting requirements for a release of software.

The bias is of interest during the requirements selection and prioritisation process.

Fogelström et al. (2009) seek to understand the effect of the bias through an experiment involving masters students in software engineering and simplified requirements prioritisation decisions. Although not as strongly as predicted, the results support the hypothesis; showing the students to be more risk adverse when selecting between requirements described in terms of revenue and more risk taking when selecting between requirements described in terms of cost. This is translated into a general preference order for requirements types:

- Customer specific features (ie. requested by a customer of the product)
- Innovative requirements
- External quality requirements (eg. usability)
- Internal quality requirements, (eg. architecture)

The authors of this publication recommend a knowledge and understanding of the decision-making biases posed by prospect theory to help ensure it does not negatively affect software product development. However, the authors recognise the need for further work, studying the bias in an industrial setting.

Differences in Embedded Systems

Oliveira et al. (2008) are focused on quality trade-offs faced in the embedded software domain. In the embedded software domain there is a strong interdependence between hardware and software. Further, there is a perceived conflict between ‘traditional’ software quality metrics and embedded system metrics. Traditional software quality metrics include reuse, abstraction, coupling, cohesion; while embedded quality metrics include performance, memory footprint and power consumption. In this publication the authors aim to understand the relationship between the two types of quality metrics.
2.5 Systematic Review

An experiment was conducted with four systems being designed and implemented to fulfil a set of functional requirements. Each system was designed and developed to emphasise different quality aspects within the software and embedded domains. The results show a small negative correlation between the those aspects of quality defined for software, and those defined for embedded systems. However, the authors conclude that it is better to optimise the software quality metrics, as they lead to other benefits like increased reuse and decreased time to market—as long as it is done in a smart and considered manner.

Unavailable

The publication by Al Balushi et al. (2008) could not be found online\(^2\).

Excluded

Three publications were identified as being within scope in the context of the systematic mapping process, however, on closer inspection during the systematic review, these publications were found not to meet the criteria required for inclusion in this stage of the process. Reasons for their exclusion are provided in this section.

Kassab et al. (2009) propose an architecture and relational database design to support the collection, analysis and trade-off between functional and non-functional requirements. While the abstract claims to present a case study, it only presents a small example implemented by the authors. The authors recognise that there is no empirical evidence in their conclusion.

Nepal et al. (2010) propose and use a fuzzy AHP-based approach to achieve higher levels of customer satisfaction with respect to product quality. However, the case study related to the automotive product development domain, with qualities such as ‘fuel economy.’ As the approach only drew of techniques used in the software development industry, and did not propose a solution for the paper was excluded from the systematic review.

Wijnants et al. (2009) present a method that aim to find relationships and correlations between aspects that make up quality of service (QoS) and aspects that make up quality of experience (QoE). The authors describe an experiment set-up involving a multi-platform game with network monitors used to determine the QoS and undisclosed techniques to monitor QoE while people played the game. A network shaper will be used to control the QoS, allowing the authors

\(^2\)An author was contacted to request a copy of the publication. An offer of a hard copy was made, but had not been received at the time of publication of this thesis.
to monitor the effect on QoE of the players. However, the experiment had not been implemented at the time of publication, so this paper does not provide any empirical evidence.

2.5.2 Rigour and Relevance

Each paper included in the systematic review was given a rating according to its rigour and relevance, as defined by Ivarsson and Gorschek (2010). Rigour refers to the the extent and detail that context is explained in the publication. It is possible for a publication to score poorly if the research was rigorous, but insufficient information was detailed in the publication. Relevance evaluates the potential for the results to impact industry by considering the realism of the scenario used for evaluation. The approach proposed in Ivarsson and Gorschek (2010) awards points for achieving certain criteria that define rigour and relevance, with a maximum of three points for rigour and four points for relevance. A summary of the results is shown in Figure 2.4.

![Figure 2.4: Rigour and relevance evaluation of systematic review publications](image)

These results are positive, showing significantly higher levels of rigour and
relevance than are seen in Ivarsson and Gorschek (2010). However, there is still a large range in these results.

Of the areas assessed using this method, one stood out as falling short in comparison to the other aspects. Validity was usually not discussed in the publications.

2.5.3 Systematic Review Discussion

The systematic review seek to answer RQ2—to provide an overview of the state of the art for empirical research addressing software quality trade-offs process and requirements artefacts.

The results show a diverse range of research objectives within the research space surveyed. The most commonly cited objectives were to determine the priorities on the qualities to better understand the quality requirements. Some papers sought to take this further, by using this information to provide improvement suggestions, or put quality requirements on the development roadmap.

A number of publications were more research orientated in the objective presented. Two publications sought to test methods in their ability to make software quality trade-offs. Another publication sought to understand industrial practitioners approach these problems, while one publication aimed to understand if people are subject to decision-making biases when making trade-off decisions. Two publications had the objective to make trade-offs between quality aspects to determine the best software product in the procurement process.

The results show a diverse range of approaches to address software quality trade-offs with respect to the software development process and requirements. AHP is the most widely applied approach, being used by three different groups of researchers in three different contexts. The QUPER model is empirically assessed in an equal number of publications, but all this work has been done by the same group of researchers and two of the papers study a similar context.

None of the approaches stands out as dominating the research landscape. In the absence of a clear direction it appears that researchers are looking for new and improved solution to empirically validate. However, the two publications that assess software quality trade-offs in different software packages for the procurement process use the AHP method.

There appear to be a number of common elements in the approaches used for software quality trade-offs. Firstly a number of papers cite the need to create or tailor a quality model to a specific context before it can be used. Many papers also highly recommend the involvement of a diverse range of stakeholders when
developing or tailoring a quality model and making trade-off decisions. Many groups are affected, and each in different ways—so by bringing everyone together it helps ensure a more mutually acceptable goal.

The research also highlights that any trade-off decision is very particular to its context. Embedded systems face different challenges to web-based systems and combat management systems. This emphasizes the need for greater empirical work in this area, as methods will need to be robust in dealing with such a diverse range of contexts.

While it was positive to find higher levels of research rigour and relevance than was expected by the authors, the publications are by no means perfect. An attempt was made to classify the research against the model of context in industrial software engineering research proposed by Petersen and Wohlin (2009). Such a classification would provide much detail into the contexts that have been studied and the areas where further work is required. Unfortunately so much information was not published in the papers that no meaningful result could be obtained on the contexts that have been studied and those that have not.

As previously mentioned, discussions of research validity were commonly missing or lacked sufficient detail.

2.6 General Discussion

One of the greatest challenges in conducting the systematic review related to research publication databases. In total three separate software packages were used to extract and manage the bibliographic records. ACM was the most difficult database from which to extract records, as there is no bulk save/export option. However, the Mendeley\textsuperscript{3} software can be used to extract the data page-by-page.

There were also many errors in the bibliographic information extracted from the databases, which necessitated manual correction and compilation of data. These differences also made finding duplicate publications very difficult and records are correctly identified as not matching. The following scenarios were found within the extracted bibliographic records:

- Information is stored in the incorrect fields. (eg.

\textsuperscript{3}http://www.mendeley.com/
Authors with more than one surname often have their first surname included as a middle initial. Such surnames are common in Spanish speaking countries.

The order of the first name and surname is sometimes swapped for Asian names in some databases.

Publication titles can vary. For example, in one database the time will use the word ‘and,’ while in another the ampersand symbol (&) is used.

Special characters in (eg. Å) can be replaced by meaningless alphanumeric codes.

Articles published in Springers’ Lecture Notes in Computer Science series often do not state the conference with which they are affiliated. This necessitating the manual collation of important data.

While the authors were surprised by the number of approaches proposed and used in software quality trade-offs, the greater problem came when trying to classify them. While some approaches had names, many did not—which lead to the use of long descriptions. At times the authors were envious of the structure nomenclatures used in the natural sciences. While a structured nomenclature may not be feasible for software engineering, the advantage of authors giving new methods unique names is that it makes these methods easier to identify and reference.

The opposite issues was found when it came to the terms used to describe software quality. In total 11 different ‘non-functional things’ were found to describe software quality—artefacts, artefacts, aspects, attributes, capabilities, characteristics, factors, features, properties, requirements and traits. Adding in all of the permutations created with the use of non functional, non-functional and nonfunctional and the list of terms become very long. This is in addition to other phrases like ‘software quality.’ The risk of using such a diverse vocabulary to describe one thing is that researchers and practitioners may not find important publications. Just as author’s should be liberal in giving names to new things, effort should be taken to ensure the existing nomenclature is reused where it can.
2.7 Conclusion

This chapter seeks to answer two research questions with respect to software quality trade-offs.

The first research question, RQ1, aims to identify the state of the art with respect to software quality trade-off literature. A systematic map of 168 publications has been conducted to answer this question. The map covered the key research venues, key researchers, research approaches applied, development artefacts on which trade-offs are applied and the trade-off approaches studied.

A clear majority of the research is focused in the early phases of the development cycle—with 90% of the publications address process, requirements and architecture. Architecture stands out as the artefact on which most software quality trade-off research is undertaken, with 50% of the identified publications focusing on this artefact.

The results show the software quality trade-off research area to still be maturing, with 61% of the research providing non-empirically assessed solution proposals. Only 29% of the publications provide empirical evidence. Further, a very diverse range of approaches for conducting software quality trade-offs are still being proposed and explored. AHP is the most research approach, employed in 13% of the publications. Model-based approaches are also popular, but these vary greatly in their implementation.

The quality trade-off approaches cover a wide solution space. Researchers are drawing from computer science, software engineering, economics, mathematics and statistics to find suitable approaches. However, most research on an approach seems to stop once the solution has been proposed. In order to be able to confirm the viability and compare these approaches, empirical research must be conducted. Without empirical research, practitioners and researchers alike are unable to determine which approach is the most suitable for a given context.

The second research question, RQ2, seeks to provide a more detailed overview with respect to empirical research addressing software quality trade-offs for the process and requirements artefacts. The systematic map identified 23 publications within the scope of this question, and a systematic review of these publications was conducted. After reading and classifying the publications only 19 were found to be within scope of the systematic review.

The aims for conducting software quality trade-offs are varied. Most commonly this work is done to help elicit priorities, which can then be explicitly or implicitly turned into requirements. Some publications emphasised the need for
a common understanding to achieve software quality success, with one claiming that knowledge of the priorities on quality can provide a sufficient understanding of the quality requirements.

As with the systematic map, the results of the systematic review highlight that there is no clear approach used for software quality trade-offs. The systematic review emphasises this point with respect to the process and requirements artefacts.

However, there are some commonalities between the various approaches to software quality trade-offs. Many publications emphasise the need to create a model of software quality tailored to the context in which it is being applied, and to involve a diverse range of perspectives in any trade-off process to help ensure all aspects of the product development are properly considered. Further, AHP is the trade-off approach of choice when choosing between software packages as part of the procurement process.

This research found a very wide vocabulary used to describe software quality. In using such a wide research vocabulary, researchers make it difficult to people find relevant publications. Going forward it would be advantageous to develop a standard vocabulary to describe software quality to help overcome these issues.

Further, the authors would like to recommend authors who develop solution proposals to name these proposals. This allows different approaches to be quickly and easily identified. Without a name for solution proposals, it can become very difficult to search for reviews, validation and evaluation of this work.

Finally, the results highlight the need for greater empirical research. While a wide range of solutions are being proposed, the lack of empirical evidence means only limited comparisons and evaluations can be made between the methods. Without this understanding it not possible to determine and transfer best-practice to industrial practitioners.
Chapter 3

Requirements Selection

This chapter is published as:


Abstract

It is important for a software company to maximize value creation for a given investment. The purpose of requirements engineering activities is to add business value that is accounted for in terms of return-on-investment of a software product. This chapter provides insight into the release planning processes used in the software industry to create software product value, by presenting three case studies. It examines how IT professionals perceive value creation through requirements engineering and how the release planning process is conducted to create software product value. It also presents to what degree the major stakeholders’ perspectives are represented in the decision making process. Our findings show that the client and market base of the software product represents the most influential group in the decision to implement specific requirements. This is reflected both in terms of deciding the processes followed and the decision-making criteria applied when selecting requirements for the product. Furthermore, the management of software product value is dependant on the
context in which the product exists. Factors, such as the maturity of the product, the marketplace in which it exists, and the development tools and methods available, influence the criteria that decide whether a requirement is included in a specific project or release.

3.1 Introduction

Incremental software development is becoming an increasingly commonplace practice among software companies as they have discovered the potential of this approach to reduce the amount of effort and time that needs to be invested in a product development prior to its release (Karlsson et al., 1998; Carlshamre, 2002; Ruhe and Greer, 2003). Examples of organisations that have adopted time-to-market as their principal product development criterion include Ericsson, Nokia, General Electric and Hewlett Packard. The ability for software developers to develop a product that meets customer requirements while offering high value to both their own business and to their customers provides increased reassurance of market success, provided that the product is released at the appropriate time and offers a superior level of quality relative to competitors.

Global competition forces companies to become more competitive and responsive to consumers and market developments. Moreover, rapidly changing market requirements as well as environmental and governmental regulations have stressed the urgency of dramatic changes in software companies for future economic survival. Thus, creating value for software companies is more important than ever before.

Value is created when a company makes a profit. It is critical for a software company to maximize value creation for a given investment. Hence, it is essential to understand the relationships between the technical decisions and the business strategy that drives the value. Boehm (2006a) stated software engineering (SE) is largely practiced in a value neutral setting with every requirement, use case, test case, defect and object being considered equally important. Traditionally, there is a separation of concerns – software developers are confined to turning requirements into verified code. Yet Bullock (2000) found that eighty percent of business value comes from only twenty percent of software components. Furthermore, there is often a mismatch between the decision criteria used by software developers at the organizational level, and the value creation criteria used by software development organizations (Boehm and Sullivan, 2000). In other words, the alignment of product, project and business decisions is a major problem in the software industry.
3.1 Introduction

The value-based approach in requirements engineering (RE) promotes the alignment of product, project and business decisions (Aurum et al., 2006; Aurum and Wohlin, 2007), and the involvement of multiple stakeholders’ perspectives in the creation of product, project and business value, while aiming to maximize the value of a release of software through the selection and prioritization of requirements (Wohlin and Aurum, 2005, 2006; Biffi et al., 2006). Despite the fact that most release planning literature covers prioritization and dependencies between requirements (Carlshamre, 2002; Karlsson et al., 1998; Ruhe and Greer, 2003), there has been little research into the criteria used in this decision-making process around requirement selection (Wohlin and Aurum, 2005, 2006; Hu et al., 2006).

This chapter reports a continuation and extended version of the work presented in Barney et al. (2006). The main objective of this chapter is to provide insight into the release planning processes used in industry to create software product value. In Barney et al. (2006), we conducted a case study, by examining two products from an Australian company, and investigated the decision making process when creating product value through requirements selection. In this chapter, we extend our research to three case studies from one German and two Australian companies. Data is collected from six different products through interviews and questionnaires. The study addresses both the process followed and the criteria used in the decision-making process. It also highlights different stakeholders’ roles and their influences on this process. The contribution of this chapter is twofold; a) it studies the values applied in prioritization of requirements for different products in one specific company and investigates the release planning process, and b) it examines to what degree stakeholders’ perspectives influence the prioritization of requirements in three different companies and then provides a comparison between the companies.

The remainder of this chapter is organized as follows: Section 3.2 covers background knowledge on the value-based approach in software development. Section 3.3 presents the methodology used in the studies. Section 3.4 presents the results of data analysis. Section 3.5 provides a detailed discussion while Section 3.6 addresses validity threats. Finally, Section 3.7 concludes the chapter.
3.2 Background

3.2.1 The Concept of Value

Economic theory defines value at an abstract level, in terms of use and exchange value. A use value is what the customer is willing to pay for the product, and an exchange value is the market value of the product (Mill and Ashley, 1909). ‘Value-adding’ became very popular in the early 20th century where the focus on product development became the product itself and customer value was seen as being inherent in the product. By the end of the 1980s the product development was focused on the relationships between the customer service and customer needs. Value was created in cooperation with the customer who was seen as an active participant in value creation activities (Storbacka and Lehtinen, 2001; Heinonen, 2004).

The concept of the value-based approach in SE was introduced in late 1990s in the context of decision-making about product lines (Faulk et al., 2000), managing investments in reusable software (Favaro, 2003) and software economics (Boehm and Sullivan, 2000). Since then, the value-based approach has attracted both software practitioners and academics leading to the integration of value considerations into existing and emerging SE principles and practices (Wohlin and Aurum, 2006).

3.2.2 Defining Value

Value constructions in economic theory are based on customer satisfaction, loyalty and re-purchasing behaviour (Heinonen, 2004). By borrowing the economic theory, we address three aspects of value, namely product value, a customer’s perceived value and relationship value. Product value is related to the product price and influenced by the quality attributes of the software product. The value of a product increases in direct proportion to its advantage over competitive products or decreases in proportion to its disadvantage (Alwis et al., 2003). A customer’s perceived value is the benefit derived from the product and is a measure of how much a customer is willing to pay for it, i.e. \[ \text{perceived value} = \frac{\text{perceived benefits}}{\text{perceived price}}, \] where \( \text{perceived benefits} \) and \( \text{price} \) are both measured relative to competing products (Weinstein and Johnson, 1999). A customer’s perceived value is influenced by his/her needs, expectations, past experiences, and culture. Relationship value is created through the social relationships between the software company and the customer. It exists through the product and customer’s perceived value (Henneberg et al.,
A customer views a purchase as a bargain, if perceived $value > price$. If $price > cost$ then the software company will make profit for their sale. The critical success factor for software companies is their ability to develop a product that meets customer requirements while offering high value that provides a certain guarantee of market success (Boehm and Sullivan, 2000; Messerschmittis and Szyperski, 2004). Since the ultimate aim for a software company is to maximize value creation for a given investment, it is essential to understand the relationships between product, project and business level decisions and the business strategy that drives the value (Aurum et al., 2006; Aurum and Wohlin, 2007; Starovic et al., 2004).

### 3.2.3 Value Based Approach in Requirements Process

Today software has a major effect on the cost, value and schedule of projects (Boehm, 2006a). However, an organization’s success in terms of profitability or market capitalisation does not necessarily correlate with their level of investment in IT (Thorp, 2003). That is because money spent does not always translate into the realisation of benefits. Most studies into the critical success factors, in successful and failed projects, find that the primary critical success factors lie in the value domain. Most projects fail due to a lack of user input, incomplete requirements, lack of resources, unrealistic expectations, unclear objectives and/or unclear timeframes (Boehm, 2006a). This is because projects are tracked by monitoring project cost and schedule (Boehm, 2006b). This approach, unfortunately, does not consider stakeholder or business value. A project can be successful in terms of cost (i.e. if it is finalized within its budget), but may fail to provide any business value. This can be due to not effectively tracking a project when the project plan changes rapidly, there are flaws in user acceptability, operationally the system is not cost-effective or timely market entry is required (Boehm, 2006a).

Favaro (2003) argues that the purpose of the requirements process is to add business value. As global competition forces companies to become increasingly competitive and responsive to consumers and market developments, the emerging discussions in SE indicate that a value-based approach makes all the difference in creating successful product and value for software business because it puts the requirements engineer in the position of managing requirements to make the most strategic opportunities.

Although companies put a great amount of effort into increasing customers’ perceived value in the product development process, determining how and when
value is added is still a challenge even in marketing and management science (Gordijn and Akkermans, 2003). This is because value creation strategies are highly contextual (Kotler, 1973) and must be analysed as part of a multidimensional array of variables (Poladian et al., 2003). Unfortunately, there is no prescribed approach to achieving this perception. There are some companies that adopt a single strategy that best suits their circumstances and are quite successful in value creation. Examples of this approach include Sony-Ericsson and Siemens-Nokia marriages for their mobile phone products.

Kotler (1973) states that strategy and product management should change with market demand. Although this is only one of many factors that could influence the relative importance for criteria in the selection of requirements, we were unable to identify a complete list of factors from the literature.

Techniques to reconcile conflicts include requirement prioritization techniques, business case analysis techniques, and stakeholder identification and requirements and negotiation techniques (Boehm, 2006b). It is also important to consider that the value a market attaches to different requirements changes over time (Maurice et al., 2006), but it is not understood how they will change (Wohlin and Aurum, 2005, 2006). The critical success factor for software vendors is to respond quickly to changing requirements while maintaining a focus on their value proposition, which may, for example, yield a quicker return on investment.

Shaw (2002) adds that features in software only need to be good enough to meet the needs of the users. Recognising that bug fixes come at a cost. The paper highlights that people are able to handle a certain level of unintended behaviour, and making valuable software includes finding this optimal level of quality.

Value-based requirements engineering (VBRE) aims to maximize the value of a release of software to success-critical stakeholders through the selection of requirements (Wohlin and Aurum, 2005, 2006; Aurum and Wohlin, 2007). VBRE is a very young area of academic study, however, the problem of creating product value through requirements selection and prioritization is real to the development of software products in today’s competitive environment. Companies have been forced to change their practices due to current market forces, but there is little theory providing an approach for development of IT intensive solutions that are valuable to all stakeholders (Gordijn and Akkermans, 2003). In this article we examine this topic in industry through three case studies.
3.2 Background

3.2.4 Incremental Development and Release Planning

Incremental software development is a top-down approach to development in which a minimal software product is developed and released in the first increment, and a function(s) or a requirement(s) is added in each successive increment until the product is complete. Each increment or product release may contain all previously elicited requirements in addition to some new requirements or functions allowing the product to cumulatively grow. This approach to software development requires the analysis of requirements, assigning them to increments (Greer and Ruhe, 2004) and releasing each increment with an aim of meeting the expectations and values of stakeholders who are involved in product development.

Release planning is the process of selecting an optimal subset of requirements for realization in a certain release in incremental software development (Carlshamre, 2002). The aim of release planning is to determine the optimal set of requirements, when they should be released and at what cost this should be achieved (Maurice et al., 2006). Researchers agree that release planning is a crucial determinant of the success of the software product as it determines which requirements will go the next release (Carlshamre et al., 2001). Release planning can only be conducted after a product’s requirements have been elicited, analysed and specified (Wohlin and Aurum, 2006). If release planning is done badly it increases risk (Maurice et al., 2006). For example, leaving critical features or difficult tasks to last or ignoring dependencies and interdependencies can result in time and budget overruns and a loss of market share.

Karlsson et al. (1998) suggested that release planning be approached through the prioritization of requirements. Carlshamre et al. (2001) and Dahlstedt and Persson (2003) furthered prioritization by recognising and accommodating the dependencies that exist between requirements. Wiegers (1999) proposed a method of prioritization that recognised and combined benefits and penalties of proposed functionality between multiple stakeholder perspectives, relative costs and relative risks.

3.2.5 Aligning Release Planning and Stakeholders’ Value Perspectives

Value creation in software development is supported by aligning product, project and business level decisions throughout the development process (Aurum et al., 2006; Wohlin and Aurum, 2005, 2006). Decisions are traditionally made in a client neutral setting as it is easier to assume that all clients have the same or
similar expectations in the solution a product will supply (Berry and Aurum, 2006). This does not result in the best solution for all stakeholders. Additionally software developers are increasingly trying to expand the target market for software products (Wohlin and Aurum, 2005; Aurum and Wohlin, 2007). Requirements come from a diverse set of stakeholders are not fed into a specific project, but are managed at a product level. Product managers initiate development projects based on a selected set of requirements. We argue that the following value perspectives are important to software developers when creating value:

**Business perspective:** Business value stems from product sales.

**Product perspective:** Product value stems from customer and market requirements.

**Project perspective:** Project value stems from project budget/timing/delivery etc.

The above value perspectives need to be aligned with product, project and business level decisions made during the software development process. Each stakeholder group’s perception of value is different (Keller, 1995; Boehm, 2006b). For example a project sponsor defines value in terms of the cost of the software and the benefits it provides, while a software company measures value in profit. Stakeholders values are often incompatible, and must be reconciled (Boehm, 2006b). Boehm (2006b) argues that:

- Users of the software want many features, while the project sponsor wants to limit cost by minimising the development effort
- Developers want stable requirements, but users want to be able to change requirements
- The system maintainers want their job to be made easier, but the developers and project sponsors want control over the solution provided and the ability to make changes whenever they want

In summary, while product managers are focussing on business perspective and developers are focussing on product perspectives, sales/project managers will be more concerned with project perspective. Interestingly stakeholders are not always aware of their own value propositions, which can often only be elicited through experience in the problem domain (Anderson and Narus, 1998).
3.3 Methodology

Additionally only a limited number of companies in the software business are able to define or measure value from their customers’ perspectives despite many describing the achievement of this as having “never been more important” (Anderson and Narus, 1998). Maurice et al. (2006) recognised that while iterative development facilitates early customer feedback, allowing faster delivery and a more interactive process; it also creates difficulties with reconciling conflicting stakeholder perspectives.

Stakeholder management, which involves a focus on the relationship between an organisation and its stakeholders over the transactions been an organisation and its stakeholder has found to create and improve the value of the organisation for shareholders (Hillman and Keim, 2001). While taking a more liberal definition of stakeholder management, Berman et al. (1999) found that two aspects of stakeholder management that are most influential on the financial performance of an organisation – employees, and product safety and quality.

Value-based Requirements Engineering (VBRE) exploits the concept of economic value during the RE process (Gordijn and Akkermans, 2003). Boehm (2006a) states that in order to achieve this, VBRE must include practices and principles for a) identifying a system’s success-critical stakeholders; b) Eliciting their value propositions with respect to the system and c) reconciling these value propositions into a set of mutually agreed objectives for the system.

3.3 Methodology

The main purpose of this study is to understand how software product value is created through the RE process, and identify the relative influence of the decision criteria and major stakeholders’ perspectives during requirements selection stage of market driven incremental software development. This chapter extends the exploratory research conducted in Barney et al. (2006). The following research questions, presented in three groups, are investigated with respect to the collective case study conducted:

1. Value-based approach in release planning
   (a) How is release planning conducted in order to create product value?

2. Values influencing requirement selection
   (a) To what degree do different value propositions influence the selection of requirement for a release of software?
(b) Does the manufacturer see potential to improve product value by applying the criteria differently in the selection of requirements for a release?

3. Perspectives influencing requirement selection

(a) To what degree do the perspectives of the major stakeholders influence the requirements selection and prioritization process in software industry?

(b) Does the manufacturer see potential to improve product value by changing the level of influence of the major stakeholders?

3.3.1 Research Model

The research framework in this study is built on Boehm’s Theory-W model (Boehm, 2006a) that proposes a process for VBRE. According to this model, the process follows identification of success-critical stakeholders, elicitation of their value propositions and requirements, and finally reconciliation of value into a mutually agreed set of requirements.

In this study we used the stakeholder groups identified by Wohlin and Aurum (2005, 2006) who participate in requirements selection for the next software release. Note that this model addresses stakeholders’ perspectives from software company point of view only. These stakeholders group include business, project and product perspectives as illustrated in Figure 3.1. Each stakeholder group’s perception of value is different. For example a project sponsor defines value in terms of the cost of the software and the benefits it provides, while a software company measures value in profit. Based on premises that requirements selection process will be influenced by stakeholders’ perception of value, this study aims to investigate the most prominent decision criterion during requirements selection process.

This study used the criteria identified in Wohlin and Aurum (2005, 2006). In these studies the researchers aimed to identify a set of orthogonal criteria that influenced the selection of requirements for a release of software. It was determined this was only possible if the criteria were kept at a high level of abstraction. Criteria were identified and confirmed in two main stages: (i) through a in-depth brainstorming session involving three researchers in requirements engineering with industrial contacts, and (ii) the industry-based research participants were asked to confirm the relevance of the criteria identified by the researchers and identify any missing criteria.
3.3 Methodology

1. Identify success-critical stakeholder groups

- Business
  - Competitors
  - Requirements issuer
  - Stakeholder priority of requirement
  - Function is promised/sold
  - Volatility

- Project
  - Support for education and training
  - Development cost-benefit
  - Resource competencies
  - Delivery date calendar time

- Product
  - System impact
  - Complexity
  - Requirements dependencies
  - Evolution
  - Maintenance

2. Elicit requirement and value propositions of stakeholders

3. Reconcile requirements and value propositions

Prioritised and selected requirements for the next product release

Figure 3.1: Value-based requirements engineering research model
Additionally this chapter combined both the criteria identified by the Wohlin and Aurum (Wohlin and Aurum, 2005, 2006) and the results provided by the participants. All of the criteria originally identified were adopted, with one additional criterion being selected from those identified by the research participants: function is promised/sold. It was felt there was sufficient overlap between the other criteria identified by the participants and those identified by the researcher.

Further this study asked participants to confirm that each criterion was relevant and identify any criteria missing from list provided. The participants were all IT professionals working in the software development industry.

This research is conducted by using three industry-based case studies since case studies are “especially suitable for learning more about a little known or poorly understood situation” (Leedy and Ormrod, 2005). This study employed a mixed research methodology to conduct each case study which is supported by semi-structured interviews, questionnaires and unstructured interviews. Data is collected from two Australian companies (Company A and Company C), and one German company (Company B). Any cultural issues are outside the scope of this study. The researchers of this study do not perceive cultural issues as a factor in this research as both companies are Western companies, with the German company having a multicultural workforce.

Semi-Structured Interviews

In the first phase semi-structured interviews were conducted to understand the processes used in RE and release planning to create product value – research question 1. The interview questions (Barney et al., 2006) covered the interviewee’s background; company’s background; product background; RE process; requirements elicitation process; requirements interpretation, verification and validation process; requirements prioritization and selection; and views on value-based software engineering, and was piloted with academics and professionals within the IT industry.

Each interview went for a period ranging between 25 and 40 minutes. The interviews were held at times suitable for both the researcher and participant. Each participant was given an introduction, providing an overview of the research, and detailing the participant’s rights and responsibilities. Then each participant was asked a series of questions, with the researcher and participant seeking clarification or more information where required. The proceedings of each interview were recorded verbatim electronically. The construction of the interview questions was based on the key activities in the RE, release plan-
3.3 Methodology

ning processes and the interviewees’ perception of how to create product value through RE. The interview questions also included company, product and personal (interviewee) background details.

Questionnaire

In the second phase, a questionnaire was created to understand the value-system applied to the decision making process in release planning to create product value – research questions two and three. The questionnaire (Barney et al., 2006) provided the list of criteria presented in this section and asked respondents to identify any criteria missing from the list; mark the criteria relevant to selection of requirements for a release; specify the relative importance of each criteria as they influence the process today; and specify the relative importance of the criteria as the respondent perceived in an optimal situation. The questionnaire was piloted with academics and professionals with in the IT industry.

The questionnaire was targeted at employees who are in a decision-making capacity for the selection of requirements for the product studied. The participants were asked to respond for the product they were identified as being involved with by their company. The company contact at each company selected participants they regarded as appropriate to participate in the questionnaire. The questionnaire focused on the 14 decision criteria which covered three different perspectives, as shown in Figure 3.1, i.e. business, project and product perspectives.

Unstructured Interviews

In the third phase the results from the semi-structured interview and questionnaire were then presented to selected participants (including project manager, product manager and development team leader) for their comments in unstructured interviews. The participants were asked to verify and explain the results.

3.3.2 Case Studies

As illustrated in Table 3.1, data is collected from three different companies where each company is regarded as one case study. Due to the exploratory nature of this research and limited access to industry, each company was selected using convenience sampling, with a personal with the authors.
Table 3.1: Descriptions of companies used in the collective case study

<table>
<thead>
<tr>
<th>Company</th>
<th>Location</th>
<th>Market Geographic</th>
<th>Industry served</th>
<th>Employees R&amp;D</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Australia</td>
<td>Australia</td>
<td>Telecommunications, finance, government, retail</td>
<td>200</td>
<td>1000</td>
</tr>
<tr>
<td>B</td>
<td>Germany</td>
<td>Europe</td>
<td>Telecommunications</td>
<td>All</td>
<td>100</td>
</tr>
<tr>
<td>C</td>
<td>Australia</td>
<td>Global</td>
<td>Corporate groupware, software development</td>
<td>All</td>
<td>55</td>
</tr>
</tbody>
</table>
3.3 Methodology

Company A

In Company A we studied two products (Product A1 and Product A2) in detail. Product A1 is an enterprise grade document and data repository developed and supported by Company A. The product has been undergoing iterative development for over ten years. It was in its third major release, with the fourth release due months after the study was conducted. Product A2 provided a solution for managing the collection, analysis and processing of paper requests and responses such as election polls, surveys, exams, insurance claims, remittances and general inbound mail. It is developed, hosted and supported by Company A. The product has been undergoing iterative development over two years. At the time of the study the company was preparing for the first major release of this product since it was first released. This product was implemented to replace the existing solutions within the organization that no longer met the requirements of the business.

Company B

Data is collected on only one product from Company B, called Product B, which was a real-time service for mobile telecommunications networks, developed, manufactured and supported. Once implemented this product allows mobile subscribers to decide what each caller hears before the subscriber answers his/her phone call, replacing the traditional ring back tone. The product has been undergoing iterative development for over two years. It has been implemented for over ten mobile telecommunications carriers, and continues to be highly popular with new clients of Company B. Product B needs to be set-up for each mobile telecommunications carrier by Company B. While implementations of this product have a common core, it needs to interface with a number of proprietary systems of the mobile telecommunications carrier. These include the mobile telecommunications network and account management system. Interfaces to these systems, and integration of the service into the existing infrastructure need to be developed and customized. The product has a highly customizable front end for user interaction with the system, the core product software in a middle layer, with a database and specialised telephony switches at the backend. Product B was originally built for a specific client contract, but was designed to allow it to be implemented in other environments.
Company C

We collected data on three products from this company; Products C1, C2 and C3. Product C1 is an enterprise grade issue tracking system developed and maintained by Company C. It is a web-based application with a database backend. The product has been undergoing iterative development for approximately three years. It is used by almost 4,000 organizations in 55 countries. This product can be set-up and integrated by an organization that purchases the software, however, Company C provides support for this product. Product C2 is an enterprise grade wiki developed and maintained by Company C. It is a web-based application with a database backend. This product has been undergoing iterative development for approximately two years. It is used by over 1,800 organizations. This product can be set-up and integrated by an organization that purchases the software, however, Company C provides support for this product. Product C3 is a continuous integration service developed and maintained by Company C. It has just been released for alpha testing. This is used by software developers to build and test code every time it is updated.

3.4 Data Analysis and Results

Researchers have proposed numerous methods to create software product value through RE and release planning, such as evolve and trade-off analysis techniques. While it is recognized that there is a disparity between formal methods and the processes used in industry, little research has been conducted into the process companies in the software industry use to create software product value.

3.4.1 Value-Based Approach in Release Planning

This section focuses on decision criteria and release planning applied at the product level based on data from Company A only. The interviews with management level people provided insights to release planning for these two products. We conducted four interviews with people from Product A1 and two interviews from Product A2.

The aim of interview was to understand how the software industry conducted release planning in order to create product value. The interview questions also addressed stakeholder groups, the role of product and requirements engineer, change management issues, and RE activities. The participants from Product A1 were the product manager, R&D managers, IT consultant and System and
Program team manager and were the product and R&D managers from Product A2.

**Stakeholder groups**

The participants identified the clients as the most important stakeholder group in determining what requirements were implemented for a release of software because they represent the revenue stream. However, it was noted that to understand a clients’ needs, Company A needed to understand their business and customers. Sales, implementation, operations, development, product management and marketing were also seen as influential groups, but there was disagreement as to the level of influence these stakeholders played.

**Product Management**

In the past at Company A the sales team dealt directly with the Research and Development (R&D) team to create customer specific versions of Product A1. This created unnecessary rework as similar solutions we redeveloped from scratch, so the role of Product Manager was created to merge these changes into a core product with a strategy to take it into the market place. Additionally both development and sales are biased in their approach to development. The development team is technology focused, the sales team is revenue focused and operations just need to get the job done. One of the participants described how a Product Manager agnostically has a view of all areas and can make a balanced judgment.

**The Role of Requirements Engineer**

The role of a requirements engineer is not clearly defined in Company A. The Product Manager was responsible for business requirements, while the Development Manager (or research and development team) was responsible for technical requirements.

**Change Management**

In Company A changes to the product are mainly managed according to their size. One of the participants identified small changes as being rolled out in bug fixes, while bigger changes have a more formal project. The availability of the resources was also identified as the biggest constraint to development.
Requirements Elicitation (for Product Release)

Company A held an annual meeting bringing together the sales teams, production team and other vested parties to develop a business strategy for the upcoming year. The objective of this meeting was also to provide feedback on what are the things that are going to be driving the business that year to development team. In this meeting, the key products to fulfilling the business strategy would be brought into focus. When a product was brought into focus, it would undergo a requirements elicitation phase involving sales, developers and support teams. The sales team represented general markets requirements and identified product development opportunities. Additionally, industry forms, expos, trade shows and conferences were as ways to keep their product aligned with their competitors, by the marketing and development teams. The requirements elicited from these processes were documented at feature level.

Requirements Interpretation, Verification and Validation

Once business level requirements were defined, a workshop was held with the Product Manager, development team and support team to determine what the requirement elicited means to product in hand. They looked at what must be delivered, what could be delivered and in what timeframe it should be delivered. This workshop was used to develop an understanding of what requirements needed to be implemented. For small discussions phone and email were also used. Since feature descriptions could often be vague business people were invited to meetings. As a result of these meetings the development team determined the system architecture and who would be responsible for what task. Once the functional specification was complete, the client was responsible for verifying and validating that the system specified met their requirements.

Requirements Selection and Prioritization

In Company A, traditionally requirements selection and prioritization has been the role of the R&D Team. Recently, the Product Manager has taken on this role for Product A1. As for Product A2, Product Manager and Development Manager were responsible from this role. Requirements selection and prioritization was a part of the workshop mentioned in the above section. The first stage in requirements selection and prioritization was to ask if the requirements could be implemented. In both products, revenue, product differentiation and sales were the key aims in selecting and prioritizing requirements according to the Product Manager.
The value of requirements from the perspective of Company A and their clients formed part of the requirements selection and prioritization process. One of the participants listed key questions asked as part of this process:

What is the value to (the company)? Is it something that is strategic? Is it eye candy? Is it limited to one particular market place? So we don’t only look at the amount of effort required by (Company A), but what it delivers to our customers.

Another participant acknowledged if a client requested a requirement be implemented, but if this requirement was not seen as marketable to other clients, then it would not be rolled into the core product, but a client specific version would be created.

While the input of the workshop played an important role, the final decision as the selection and prioritization of requirements laid with the Product Manager working closely with the Development Manager. Once agreed, the resultant list of requirements was usually made available. As Product A2 was relatively new, a large part of the development effort was rolling customer specific features that have been developed back into the core product as generic components. It was the product manager and development manager’s job to start by looking at what should be rolled back into the core product. In order to facilitate an open working environment, the Product Manager wrote a monthly report that detailed all work being done by the R&D team.

**Release Planning**

Company A has a defined direction for Product A1, but new and existing customers defining new requirements heavily influence this direction. In addition to the formal RE and release-planning project the product evolves with customer specific versions being developed between major releases. If there are features that need to be implemented to win a contract, then they will be made available in either the next release or a customer specific version, in which case they will be rolled back into the core product at a later date. Any requirements that are perceived to have a sufficient market are rolled back into the core product.

The product managers preferred method is to get all of the stakeholders into a meeting to discuss and analyze the business requirements. This process was described by one the product managers:

So we’ll sit down and put together a development brief and answer all of the requirements of what we’re developing, how long is it going
to take, are we going to build for it, how much are we going to sell it for, who else can use it – so all these other questions will form that brief, we’ll submit that to R&D and they’ll take a look at it and say yep, it’ll take two hours, it will cost this much, cost that much, they’ll do the building, all the testing that they need to do and sort-of roll it out for that client and then later on roll it out in a major release.

Company A planned a major release of Product A1 every 18 to 24 months. They would try and include the features selected for inclusion. However, if a release needed to be out by certain date, then Company A would make a major release and follow it with a minor release. Although Company A normally worked on a two-year cycle for major releases of its software products, as Product A2 was relatively young, it was currently undergoing six monthly releases. As the time gets closer to a release date, Company A would renegotiate what would be in the release. If it was decided that certain features could not make the desired release date, an additional release with the postponed functionality would be planned to follow.

### 3.4.2 Values Influencing Requirement Selection

The objective of this section is to capture and understand any differences on the product level when it comes to which criteria that are important when selecting which requirements to include in a release. The questionnaire allowed us to capture the decision criteria used in requirements selection for Product A1, Product A2, Product B and at Company C. The return rate for the questionnaires is described in Table 3.2.

<table>
<thead>
<tr>
<th>Product/Company</th>
<th>Distributed</th>
<th>Returned</th>
<th>Return Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product A1</td>
<td>7</td>
<td>7</td>
<td>100%</td>
</tr>
<tr>
<td>Product A2</td>
<td>7</td>
<td>5</td>
<td>71%</td>
</tr>
<tr>
<td>Product B</td>
<td>16</td>
<td>9</td>
<td>56%</td>
</tr>
<tr>
<td>Company C</td>
<td>40</td>
<td>5</td>
<td>20%</td>
</tr>
</tbody>
</table>

In their study in Sweden, Wohlin and Aurum (2005, 2006) identified thirteen criteria, which influenced the selection and/or prioritization of a requirement
covering the main stakeholders. While our pilot studies identified several more additional criteria, only one was substantially different to the others. This was the criteria “Function is promised/sold”. Hence, this study used fourteen criteria, based on the response from the participants as illustrated in Figure 3.1. First the respondents were asked to identify any additional criteria not listed. Then the respondents were asked to mark the criteria they felt were relevant in deciding whether to include a requirement in a release or project. Finally the respondents were asked to provide relative weights regarding the importance of the criteria in two sets, currently (today) and if the criteria were applied optimally (future). The respondents had 1000 points to spend amongst the criteria. A high number of points meant a criterion was important.

Participants, for Product A1 identified three additional criteria as being important in the decision making process in release planning: (i) creation of competitive advantage, (ii) preferred operating architecture, and (iii) adherence to corporate software design parameters. Participants of Product A2 identified two additional criteria as influencing the release planning process (i) non custom application (Resell a solution, save costs); (ii) future financial worth/new business applications.

The researchers decided these were covered in the existing set of criteria provided in the questionnaire.

The results for all case studies clearly showed that some criteria are more influential than others in the selection and prioritization of requirements. For products A1, A2 and B the value propositions of the business perspective are paramount. However, while the most influential criterion for Product C represents the business perspective the next four all represent the product and project perspective.

Some change in the relative importance of the criteria is also perceived beneficial in all cases. There are no criteria that move consistently across all case studies, but there are some notable similarities between products A1 and A2. The participants for both felt that the influence of the criterion function is promised or sold was overly influential today, while not enough consideration was being given to the impact of delivery date/calendar time.

For each case study the order and relative importance of the different decision criteria for “today” and “future” is presented. The “movement” column whether a particular criterion moves up or down in the future situation when comparing with today. The “perspective” column shows to which perspective each criteria belongs.
Decision Criteria for Product A1 (Today and Future)

The results for Product A1 are presented in Table 3.3.

The business perspective of Product A1 has the most significant influence on the selection and prioritization of requirements for inclusion in the software. When the criteria were ranked in order of influence, all four criteria representing business perspective of Product A1 appear in the first five places. Some project perspective criteria were considered much more important than others. Both the development cost-benefit of the requirement and the impact the requirement has on delivery date had percentage values above 9%, while others were below 3.5% in their importance. Three product perspective criteria were considered of higher importance – (11) the complexity of the requirement, (10) the impact on the system, and any (12) requirement dependencies.

The participants for Product A1 optimally saw a tighter distribution of the criteria. In describing the situation today the criteria were distributed over 8.8 percentage points, compared with an optimal distribution over 6.6 percentage points. The optimal application of the criteria remained business perspective focused; however, this area still reported a significant change. The importance of the criterion if the function has been promised or sold fell five places in the ranking of criteria. The development cost-benefit of the requirement, raised five places when the criteria were ranked how the participants would like to see them applied in future. The most significant change in the criteria representing the product perspective was a decrease in the importance of (11) the complexity of the requirement. This criterion fell five places when ranked against how the participants would like to see them applied.

Decision Criteria for Product A2 (Today and Future)

The results for Product A2 are presented in Table 3.4.

The business perspective of Product A2 has the most significant influence on the selection and prioritization of requirements for inclusion in the software. When the criteria were ranked in order of influence, all four criteria representing the business perspective of Product A2 appear in the first five places. Unlike Product A1, the most important criteria for Product A2 represent both the business and project perspectives when ranked by importance in release planning. Product A2 also differed from the others in the project perspective. The most important issue from the project perspective was the available resources and their competencies. Two product perspective criterions considered of higher importance: (10) the impact on the system, and (11) the complexity of the re-
Table 3.3: Relative importance of different criteria for Product A1 today and in future

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Perspective</th>
<th>Today (%)</th>
<th>Future (%)</th>
<th>Movement</th>
</tr>
</thead>
<tbody>
<tr>
<td>3. Stakeholder Priority of Requirement</td>
<td>Business</td>
<td>11.5</td>
<td>10.4</td>
<td>−</td>
</tr>
<tr>
<td>2. Requirement’s Issuer</td>
<td>Business</td>
<td>11.5</td>
<td>10.3</td>
<td>−</td>
</tr>
<tr>
<td>4. Function is Promised/Sold</td>
<td>Business</td>
<td>9.0</td>
<td>6.7</td>
<td>−5</td>
</tr>
<tr>
<td>11. Complexity</td>
<td>Product</td>
<td>8.9</td>
<td>6.4</td>
<td>−5</td>
</tr>
<tr>
<td>1. Competitors</td>
<td>Business</td>
<td>8.8</td>
<td>8.9</td>
<td>+1</td>
</tr>
<tr>
<td>10. System Impact</td>
<td>Product</td>
<td>7.8</td>
<td>8.8</td>
<td>+1</td>
</tr>
<tr>
<td>12. Requirements Dependencies</td>
<td>Product</td>
<td>7.6</td>
<td>7.3</td>
<td>−</td>
</tr>
<tr>
<td>7. Development Cost-Benefit</td>
<td>Project</td>
<td>7.3</td>
<td>9.8</td>
<td>+5</td>
</tr>
<tr>
<td>9. Delivery Date/Calendar Time</td>
<td>Project</td>
<td>7.3</td>
<td>7.8</td>
<td>+3</td>
</tr>
<tr>
<td>8. Resources/Competencies</td>
<td>Project</td>
<td>4.9</td>
<td>4.4</td>
<td>−3</td>
</tr>
<tr>
<td>14. Maintenance</td>
<td>Product</td>
<td>4.8</td>
<td>5.1</td>
<td>−</td>
</tr>
<tr>
<td>13. Evolution</td>
<td>Product</td>
<td>4.5</td>
<td>5.8</td>
<td>+2</td>
</tr>
<tr>
<td>6. Support/Education/Training</td>
<td>Project</td>
<td>3.4</td>
<td>3.8</td>
<td>−1</td>
</tr>
<tr>
<td>5. Volatility</td>
<td>Business</td>
<td>2.7</td>
<td>4.4</td>
<td>−3</td>
</tr>
</tbody>
</table>
Table 3.4: Relative importance of different criteria for Product A2 today and future

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Perspective</th>
<th>Today (%)</th>
<th>Future (%)</th>
<th>Movement</th>
</tr>
</thead>
<tbody>
<tr>
<td>4. Function is Promised/Sold</td>
<td>Business</td>
<td>14.5</td>
<td>5.8</td>
<td>−6</td>
</tr>
<tr>
<td>8. Resources/Competencies</td>
<td>Project</td>
<td>11.9</td>
<td>10.7</td>
<td>−1</td>
</tr>
<tr>
<td>1. Competitors</td>
<td>Business</td>
<td>11.3</td>
<td>13.9</td>
<td>+2</td>
</tr>
<tr>
<td>7. Development Cost-Benefit</td>
<td>Project</td>
<td>10.1</td>
<td>7.5</td>
<td>−2</td>
</tr>
<tr>
<td>3. Stakeholder Priority of Requirement</td>
<td>Business</td>
<td>8.5</td>
<td>12.3</td>
<td>+3</td>
</tr>
<tr>
<td>2. Requirement’s Issuer</td>
<td>Business</td>
<td>8.3</td>
<td>7.8</td>
<td>+1</td>
</tr>
<tr>
<td>11. Complexity</td>
<td>Product</td>
<td>7.1</td>
<td>5.1</td>
<td>−2</td>
</tr>
<tr>
<td>9. Delivery Date/Calendar Time</td>
<td>Project</td>
<td>6.6</td>
<td>10.3</td>
<td>+4</td>
</tr>
<tr>
<td>10. System Impact</td>
<td>Product</td>
<td>6.4</td>
<td>4.0</td>
<td>−4</td>
</tr>
<tr>
<td>5. Volatility</td>
<td>Business</td>
<td>4.5</td>
<td>5.0</td>
<td>−1</td>
</tr>
<tr>
<td>12. Requirements Dependencies</td>
<td>Product</td>
<td>3.1</td>
<td>4.1</td>
<td>−1</td>
</tr>
<tr>
<td>14. Maintenance</td>
<td>Product</td>
<td>2.9</td>
<td>5.6</td>
<td>+4</td>
</tr>
<tr>
<td>6. Support/Education/Training</td>
<td>Project</td>
<td>2.6</td>
<td>5.1</td>
<td>+3</td>
</tr>
<tr>
<td>13. Evolution</td>
<td>Product</td>
<td>2.2</td>
<td>2.7</td>
<td>−</td>
</tr>
</tbody>
</table>
3.4 Data Analysis and Results

The optimal (future) application of the criteria remained business focused; however, this area reported a significant change in its makeup. The importance of the criteria, (4) if the function has been promised or sold, fell six places in the ranking of criteria going from the most important to the least important business perspective criterion. The order of the business perspective criteria was otherwise unchanged, with the status of (1) competitors with respect to the requirement, and the market’s priority of the requirement being the most significant in first and second place of all criteria respectively. The project perspective criterion (9) delivery date/calendar time rose four places when the criteria were ranked against how the participants would like to see them applied. The participants from Company A optimally saw a tighter distribution of the development criteria. In describing the current situation the criteria were distributed over 4.7 percentage points, compared with an optimal distribution over 1.4 percentage points.

Decision Criteria in Company B (Today versus Future)

The results for Product B clearly indicate that some criteria are more important than others in the selection and prioritization of requirements for a release. It is worth noting that three of the criteria have percentage values above 10% and five have values below 5%. The order and relative importance of the different criteria can be seen in Table 3.5.

The business perspective of Product B has the most significant influence on the selection and prioritization of requirements for inclusion in the software. The three most important criteria all represent the business perspective of Product B; in order these are (4) if the function has been promised or sold, (3) the market’s priority of the requirement, and (2) the stakeholder responsible for issuing the requirement, Requirement’s Issuer.

Some project perspective criteria were considered much more important than others. Both (7) the development cost-benefit of the requirement and (9) the impact the requirement has on delivery date had percentage values above 9%, while (5) a requirement’s volatility and (6) the ability to provide technical support, education and training for the requirement were below 3.5% in their importance.

Product perspective criteria were considered of fairly equal importance, with all criteria representing these groups clustered together in the results between 5.9% and 4.6%.

The results for how the value criteria should be optimally (future) applied in requirements selection and prioritization for Product B indicate that some
Table 3.5: Relative importance of different criteria for Product B today and future

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Perspective</th>
<th>Today (%)</th>
<th>Future (%)</th>
<th>Movement</th>
</tr>
</thead>
<tbody>
<tr>
<td>4. Function is Promised/Sold</td>
<td>Business</td>
<td>14.1</td>
<td>9.6</td>
<td>-1</td>
</tr>
<tr>
<td>3. Stakeholder Priority of Requirement</td>
<td>Business</td>
<td>12.1</td>
<td>12.3</td>
<td>+1</td>
</tr>
<tr>
<td>2. Requirement’s Issuer</td>
<td>Business</td>
<td>10.1</td>
<td>10.1</td>
<td>-1</td>
</tr>
<tr>
<td>9. Delivery Date/Calendar Time</td>
<td>Project</td>
<td>9.4</td>
<td>9.4</td>
<td>+1</td>
</tr>
<tr>
<td>7. Development Cost-Benefit</td>
<td>Project</td>
<td>9.1</td>
<td>9.1</td>
<td>-</td>
</tr>
<tr>
<td>8. Resources/Competencies</td>
<td>Project</td>
<td>6.6</td>
<td>6.6</td>
<td>-3</td>
</tr>
<tr>
<td>1. Competitors</td>
<td>Business</td>
<td>6.5</td>
<td>6.5</td>
<td>-3</td>
</tr>
<tr>
<td>11. Complexity</td>
<td>Product</td>
<td>5.9</td>
<td>5.9</td>
<td>-3</td>
</tr>
<tr>
<td>10. System Impact</td>
<td>Product</td>
<td>5.6</td>
<td>5.6</td>
<td>+1</td>
</tr>
<tr>
<td>12. Requirements Dependencies</td>
<td>Product</td>
<td>4.9</td>
<td>4.9</td>
<td>-4</td>
</tr>
<tr>
<td>14. Maintenance</td>
<td>Product</td>
<td>4.9</td>
<td>4.9</td>
<td>+5</td>
</tr>
<tr>
<td>13. Evolution</td>
<td>Product</td>
<td>4.6</td>
<td>4.6</td>
<td>+5</td>
</tr>
<tr>
<td>6. Support/Education/Training</td>
<td>Project</td>
<td>3.4</td>
<td>3.4</td>
<td>+3</td>
</tr>
<tr>
<td>5. Volatility</td>
<td>Business</td>
<td>3.0</td>
<td>3.0</td>
<td>-4</td>
</tr>
</tbody>
</table>
change would be perceived as being beneficial. The participants from Company B optimally saw a tighter distribution of the criteria. In describing the current situation the criteria were distributed over 11.1 percentage points, compared with an optimal distribution over 7.9 percentage points. The optimal application of the criteria remained business focused; however, the focus within this area has changed. The first two criteria swapped place, with Company B preferring to see (3) the stakeholder’s priority of the requirement as the most important criteria over (4) whether the function has been promised or sold. The optimal application of the criteria also saw the project perspective criteria of (9) the impact the requirement has on delivery date, overtake the importance of (2) the stakeholder responsible for issuing the requirement. It is also of interest to note that (6) the ability to provide technical support, education and training for the requirement rose three places, while (8) the available resources and their competencies fell three places in the ranking.

The criteria representing the product perspective became more evenly distributed when applied optimally. Instead of being distributed over 1.3 percentage points, they were distributed over 3.2 percentage points, despite the general reduction in range. The effect of a requirement on both (14) maintenance and (13) system evolution jumped five places in the ranking of the criteria, while considerations of (12), requirement dependencies fell four places in the ranking.

**Decision Criteria in Company C (Today versus Future)**

The order and relative importance of the different criteria can be seen in Table 3.6.

None of the stakeholder groups is prominent in their influence over the selection and prioritization of requirements at Company C. While the most influential criterion represents the view of the business perspective, the next two criteria represent project perspective criteria, followed by two product perspective criteria. This mixed distribution is continued with the remaining criteria. The results for how the value criteria should be optimally applied in requirements selection and prioritization for Product C1, Product C2 and Product C3 indicate that some change would be perceived as being beneficial. The results can be seen in the table above.

The criteria representing the product perspective have all increased in importance, except for (12) requirements dependencies, showing a general feeling that the issues representing this perspective are currently undervalued. Of most note was (13) evolution, which moved up nine places. It should be noted that
Table 3.6: Relative importance of different criteria in Company C today and future

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Perspective</th>
<th>Today (%)</th>
<th>Future (%)</th>
<th>Movement</th>
</tr>
</thead>
<tbody>
<tr>
<td>3. Stakeholder priority of requirement</td>
<td>Business</td>
<td>11.2</td>
<td>10.7</td>
<td>-1</td>
</tr>
<tr>
<td>9. Delivery date/Calendar time</td>
<td>Project</td>
<td>9.8</td>
<td>7.8</td>
<td>-4</td>
</tr>
<tr>
<td>7. Development cost-benefit</td>
<td>Project</td>
<td>9.7</td>
<td>8.8</td>
<td>-1</td>
</tr>
<tr>
<td>10. System impact</td>
<td>Product</td>
<td>9.2</td>
<td>12.8</td>
<td>+3</td>
</tr>
<tr>
<td>12. Requirements dependencies</td>
<td>Product</td>
<td>8.3</td>
<td>5.3</td>
<td>-6</td>
</tr>
<tr>
<td>4. Function is promised/sold</td>
<td>Business</td>
<td>7.2</td>
<td>6.8</td>
<td>-1</td>
</tr>
<tr>
<td>8. Resources/competencies</td>
<td>Project</td>
<td>7.0</td>
<td>4.3</td>
<td>-5</td>
</tr>
<tr>
<td>11. Complexity</td>
<td>Product</td>
<td>6.7</td>
<td>5.4</td>
<td>-1</td>
</tr>
<tr>
<td>14. Maintenance</td>
<td>Product</td>
<td>6.7</td>
<td>8.6</td>
<td>+4</td>
</tr>
<tr>
<td>6. Support/Education/Training</td>
<td>Project</td>
<td>5.8</td>
<td>6.2</td>
<td>+2</td>
</tr>
<tr>
<td>2. Requirement’s issuer</td>
<td>Business</td>
<td>5.3</td>
<td>3.3</td>
<td>-1</td>
</tr>
<tr>
<td>13. Evolution</td>
<td>Product</td>
<td>5.2</td>
<td>10.6</td>
<td>+9</td>
</tr>
<tr>
<td>5. Volatility</td>
<td>Business</td>
<td>4.1</td>
<td>4.1</td>
<td>-</td>
</tr>
<tr>
<td>1. Competitors</td>
<td>Business</td>
<td>3.8</td>
<td>5.3</td>
<td>+4</td>
</tr>
</tbody>
</table>
Company C only employs people with technical backgrounds. The developers swap between full time development and support, with the time in support used to determine market requirements. The other big movers were the influence of (1) competitors with respect to a requirement, which rose 4 places; and a decrease in importance of (8) resources/competencies, which fell five places.

### 3.4.3 Perspectives Influencing Requirement Selection

The relative influence of the different stakeholder groups in each case can be seen in Table 3.7. The same information is also presented graphically in Figure 3.2 and Figure 3.3.

![Figure 3.2: Stakeholders’ perspective at business, project and product level (Today)](image)

The results for all cases, when grouped by stakeholder group, clearly indicate that some stakeholder groups are more important than others in the selection and prioritization of requirements for a release. The results for how the value criteria should be optimally applied in requirements selection and prioritization for all products indicate that some change would be perceived beneficial.
Table 3.7: Relative importance of different criteria in Company C today and future

<table>
<thead>
<tr>
<th>Perspective</th>
<th>Product A1</th>
<th>Product A2</th>
<th>Product B</th>
<th>Company C</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Today</td>
<td>Future</td>
<td>Today</td>
<td>Future</td>
</tr>
<tr>
<td>Business</td>
<td>45</td>
<td>42</td>
<td>46</td>
<td>40</td>
</tr>
<tr>
<td>Project</td>
<td>23</td>
<td>27</td>
<td>33</td>
<td>37</td>
</tr>
<tr>
<td>Product</td>
<td>32</td>
<td>31</td>
<td>21</td>
<td>22</td>
</tr>
</tbody>
</table>


Figure 3.3: Stakeholders’ perspective at business, project and product level (Future)
The business perspective is the most influential in all cases, both today and in the future, except for Company C. Additionally in all cases the participants felt that the business perspective was overly influential to the detriment of other perspectives.

The product perspective is second most important in selecting and prioritizing requirements for Product A1, with the project perspective is the least important. While the participants in the study felt that optimally, the ranking of these groups would remain the same, it was felt that the opinions of the groups should be valued more equally.

The results for Product A2 indicate that some changes in future would be perceived beneficial when value decision criteria are applied in requirements selection and prioritization. The project perspective is second most important in selecting and prioritizing requirements, while product perspective is least important. The participants in the study saw the release planning being conducted more effectively with a small decrease in the influence of the business perspective and an increase in the influence of the project perspective.

As illustrated in Table 3.7, the project perspective is higher than product perspective for Product A1, while the opposite is true for Product A2. We believe that this difference in requirements selection can be attributed to the respective maturity levels of these two products. This issue is further discussed in Section 3.5.

The project perspective is second most important in selecting and prioritizing requirements for Product B, while product perspective is least important. While the participants in the study felt that optimally, the ranking of these groups would remain the same, it was felt that the opinions of the groups should be valued more equally.

In Company C, participants perceived that both business perspective and project perspective are influencing the selection and prioritization of requirements to the detriment of the product perspective. The optimal application of the criteria ultimately saw a decrease in the influence of both the business and project perspective to give greater priority to product related issues. Company C is the only Company in this survey that did not have the business perspective as the most influential group, both today and optimally, in the selection and prioritization of requirements.
3.5 Discussion of Results

Our results highlighted several interesting aspects of decision making in requirements selection process in incremental development. While between them participants of our study raised the perspectives of each of the key stakeholder groups, it appeared that the value propositions that influenced their decision making process were more intrinsic than part of an explicitly planned process.

3.5.1 Value-Based Approach in Release Planning

Release Planning

Company A in the past developed products for the market without the support of a specific client. This approach was unsuccessful for both clients with the products not meeting the needs of the market and therefore not selling. In order to create software product value, Company A have shifted the focus from fulfilling the perceived value propositions of the market to the much safer known value propositions of clients in a relationship with the company. This “real” client base is a more secure source of revenue for the software development company. There is a relationship between the two organizations and the software development company is creating software to meet the actual needs of the client. Company A moved to a development strategy that requires a client to request a product or feature before it will be developed. This approach created a situation where new features were only made available in customer specific implementations. This approach created two main problems; new features developed for a customer-specific implementation could not be made available to other clients easily, and when changes were made to the core product many customer-specific implementations need to be released involving a large human effort.

Company A created positions for product managers. The Product Manager for Product A1 and A2 saw his primary role as facilitating the rolling up of customer specific functionality into the core product so that it could be released to the market. An interview participant for Product B expressed a hope that the product manager for Product B would do likewise. A role of the product manager at both companies is to monitor similar products so that the company can react more competitively in bids and marketing material without the need to have functionality implemented. Both companies felt that with a better understanding of the market place sales bids would be more successful.

Company A expressed a need to remain client implementation driven – only implementing requirements when requested by a client. The elicitation of new
requirements for a client for all products involves the client and a project manager and/or business analyst from Company A. Where the requirements are technically complex key developers, and in the case of Company A, the Product Manager will also get involved to ensure a common understanding is reached between all of the stakeholders in the requirements. It should be noted that whenever new functionality is to be implemented for Product A1 or Product A2 the client requesting the functionality is used to validate that it is correctly understood and when implemented, that it functions work as required. This ensures that the functionality is meeting the real market need, and not what Company A perceives as the market need.

When a software product has a small client base it is easier to keep the product in line with individual clients’ requirements. The client base of Product A2 is both measured in the tens, while the client base of Product A1 is a few hundred. It is much more feasible to create client specific versions of Product A2 with the intension to roll this functionality back into the core product at a later date as the number of customer specific versions of the software can be kept relatively low. This is reflected through the approaches to product development for the three products; RE for Product A1 follows an implementation project lifecycle, whereas Product A2 follow a product release plan except where a client has specific needs that must be met in the short term.

Where development is implementation project driven the requirements selection and release planning are largely defined by the client and will usually be very similar as the client requires the software to meet their needs. However, where software is developed as a product and includes the requirements of many customers the requirements selection and release planning are more easily separated. Dependant on the situation of the Product Manager for Products A1 and A2, issues like time to market are more critical when releasing a product not for a specific customer and it may be desirable to release a version of software that does not fulfil all the requirements with the intension to follow a release with updates. Failing to meet all requirements is unlikely to affect all customers; there are more market opportunities to be considered in this approach.

Developing a software product to individual clients’ requirements is beneficial in the early stages as it facilitates the products growth in line with the needs of the client base and the general market. All features that are developed are requirements of the clients, and nothing is developed that is not a requirement of a client. This practice minimises the time to market with a fully featured product, bringing forward revenue generation. The product manager of Product Manager for Product A1 and A2 noted:
It gives us more credibility if we release the product soon; it gives it more functionality, better saleable product.

However, when a company only develops functionality requested, they risk losing competitive advantage where other marketing offerings contain features that have not been requested, but are sufficient to win a contract. In both Products A1 and A2, release planning was very focused towards meeting customer objectives. Customer specific versions of the software acted as a testing ground for requirements and are considered to represent real needs of the market. Selective and controlled rolling of these requirements into the software product core allowed the most suitable requirements to be released to a market that is better understood. The processes followed for both products A1 and A2 involved bringing the success critical stakeholders together so that a common understanding of requirements could be found. The understanding gained then acted as a base for further analysis. The product manager was then responsible for taking all of these perspectives and information on board in making the final decision as to what should be implemented.

We observed the similar approach to product development with Company B in Germany, who also changed their market driven approach to customer specific development. On the other hand, Company C, which was producing to a larger market (4600 customers across 60 countries), paid more attention to market requirements and their focus on decision criteria in requirements selection was primarily on product perspective, which was different from Companies A and B.

**Product Maturity**

In two products that we studied at Company A, Product A2 was much younger than Product A1 i.e. Product A1 had been evolving over the past 10 years, whereas Product A2 was much newer, at two years, and it was still trying to gain credibility within the market place. This can be seen through the most important criteria representing this stakeholder group, for Product A1 these were (3) the stakeholders’ prioritization of the requirement and (2) the party responsible for issuing the requirement, while for Product A2 these were (4) whether the function has been promised or sold and (1) the status of competitors with respect to the requirement. Similarly (8) the resources and competencies of development personnel were more important for Product A2 than Product A1. As Product A1 was more mature, greater expertise existed within the company to support this product. However, Product A2 took the company through “un-
known waters”. This is inline with the research that says a company’s strategy and product management will change with market demands (Kotler, 1973).

Development cost-benefit (7) was also a bigger issue for Product A2 as a lot of money has been spent in its early development and management was more cautious about revenue prospects, whereas Product A1 had become highly profitable for Company A. Looking at the optimal application of criteria, the product manager felt that the (7) development cost-benefit for Product A1, was perceived as undervalued due to the time and resources being put into other projects. There was a perception within the organization that the highly profitable Product A1 would lose competitive advantage while resources were focused elsewhere. However, the PM also conceded that there was a greater desire to build functionality with less concern over the revenue it would generate.

The increase in importance of (9) the impact a criterion has on the delivery date for both Product A1 and Product A2 can be explained with reference to how it helps to create a positive market perception when the company is first to bring new functionality to the market place. The massive decrease in the importance of (4) the functionality being promised or sold can be explained with reference to the fact that Company A would like to start taking a more structured approach to bringing new functionality into the market place. Rather than developing client specific versions of the software, the company would like to see more controlled release planning, which implements functionality and releases it to the general market.

3.5.2 Values Influencing Requirement Selection

Decision Criteria in Requirements Selection

The in depth analysis of two products from Company A has provided noteworthy results. Most people were unable to express how value was created through requirements selection and prioritization in their company; however working with customers was seen as the most effective way to make product valuable. The decision criteria and release planning were affected by several factors e.g. the product maturity, customer importance, the size of the customer-base, the services that were provided to customers etc.
3.5 Discussion of Results

3.5.3 Perspectives Influencing Requirement Selection

Stakeholder Influence in Requirements Selection

The results of the second research question, based on the data collected from three companies, further highlighted the results of the first research question. The business perspective was the most critical group in creating software product value for Company A and B where the product development involved both market-driven and customer specific approaches. As for Company C, the most influential perspective was the product perspective. Although the ranking in decision criteria changed between the products as well as the companies, the overall results clearly pointed out the influence of business perspective in requirements selection, i.e. the first criterion in ranking always related to the business perspective. There was no major disparity between product and project perspectives for Company A and B. The results showed that practitioners were quite aware of the fact that the requirements selection had to be aligned with business strategies which drive the value creation for their companies. Overall, the results showed that there was no significant differences between what practitioners were applying today and how they would like to see it in future in relation to requirements selection. In both companies A and B, the business perspective came out as the most influential perspective overall (today and future). However, there was a slight push down to the business perspective and push up in both project and product perspectives in future criteria selection. On the other hand, the product perspective was the most influential perspective in requirements selection for Company C (today and future), however, there was a slight push up for project perspective in future criteria selection. Furthermore, in future criteria selection, the product and project perspectives were more evenly distributed. This indicates that software practitioners accept the importance of the business perspective in requirement selection; however, in future, they would like to see the product perspective receiving more attention during requirements selection.

Customer importance

Just as different stakeholder groups exert different amounts of pressure on the selection and prioritization of requirements, not all customers are equal in the level of influence. Company B identified that some customers are strategically more important than others to the software development company. The factors identified that determine whether a customer is strategically important include the prestige, credibility and marketability of having a particular high profile
customer; the ability for larger clients to pay higher premiums, and in the case of multinationals, there is a potential for repeat work in many countries with similar requirements. For these reasons, the software development company is more likely to accommodate changes to the base system as part of the contract for strategically important customers.

For small customers [Company B] won’t make a lot of customizations:

... we will ... force them down a track that is easier for our developers to do. But if they are a really big customer, either from the respect of paying a lot of money or basically they are important to the company, we will go in there and gather all the requirements come back and work out what it is we can and cannot do and then make a decision based on man-hours and the improvement to the product if it is worth development for that customers.

3.6 Validity Threats

The following presents the validity threats to the findings, considering four kinds of validity, including conclusion validity, internal validity, construct validity, and external validity.

**Conclusion validity:** Threats to conclusion validity, are lack of statistical calculations or misuse of statistical assumptions that leads to incorrect conclusions made by the researcher. There may be a risk that conclusions from this study are inaccurate due to low statistical power. We did not use statistical calculations to find patterns in the result. Instead, deductive logic was used because of limited data points. To receive high reliable measures and to avoid poor question and poor layout several pilot studies were conducted. The results of this study must be interpreted with caution as the three companies that we investigated are not a representative sample of the software development industry. The small number of people involved in the decision making process for the inclusion of a requirement in a software system has limited the number of possible responses for each company.

**Internal validity:** The research instrument in this study was developed with a close reference to literature. In addition, we conducted pilot studies to ensure that the questions relate to the stated objectives of this study. The participants in this study were experienced software practitioners. Two of the participants were involved in both the interview and questionnaire for both Product A1 and Product A2 affecting the internal validity of the results. Each of the products
has the same Product Manager and Development Manager. It is possible that these participants applied the same values to product development in general. However, comparing the the Product Manager’s responses for the two products showed that more than half of the points were awarded differently both for the criteria today and the optimal application of the criteria, 54% and 59% respectively. The differences for the Development Manager were not as high, with 27% and 30% of the points being awarded respectively. Another potential threat to internal validity is related to the questionnaire. It is always difficult to know whether the respondents have understood the questions as intended and in a similar fashion to one another. This threat was partially address during the third stage of this study, where the results were presented to the product manager in an unstructured interview with the researcher for confirmation.

Construct validity: We asked participants to evaluate the need for further criteria, however it is easier for the participants to agree with the set of criteria identified by the researchers than disagree. It was easier for the participants to agree with the set of criteria identified by the researchers in advance than disagree, because they knew that the criteria on the list indicated that the researcher considered them as relevant. In addition, it was easier to stick to the stated criteria than proposing new criteria. This is partially taken care of by allowing the participants to assign a relative importance of zero or propose new criteria where they see fit. However, the effect of examining the criteria at a different level of abstraction has not been considered for this study.

External Validity: This threat can cause incorrect conclusions to be drawn from empirical study. There are two threats to external validity that are relevant for this study: interaction of selection and treatment, and interaction of history and treatment (Wohlin et al., 2000). Participants were selected from different geographic locations in Australia and Germany. Both male and female participants were represented. The small sample size was another threat to the external validity of this study. The sample size may affect the conclusions; hence they may not be generalized for the whole SE industry population.

3.7 Conclusions and Future Study

This article addresses value based approach in requirements engineering when creating product value through requirements selection for a software release. It particularly addresses the issues related to (a) value-based approach in release planning, (b) values influencing requirements selection and (c) stakeholders’ perspectives influencing requirements selection.
Interviews at an Australian software development company showed that there is no silver bullet for a value-based approach to release planning. While the interviewees raised the perspectives of each of the key stakeholder groups involved in the development process, it appeared that the value propositions that influenced the selection and prioritisation of requirements were more intrinsic than part of an explicitly planned process.

With respect to the values that influencing requirements selection, VBRE is a new area of study and is still in the theory building stage. Questionnaires at three software companies operating in either Australia or Germany have shown that some criteria are more important than others in the selection of requirements to include in a specific project or release. Moreover, this study has shown that the business perspectives are more influential than the project and product perspectives in general; however this may change due to product and project requirements. The following aspects influenced the requirements selections.

- Maturity of the product
- Requirements source, customer type (big or small – importance: potential for future business, can change a premium on larger companies), contract type
- Size of the customer base – this aspect influences the process more than the result

Furthermore, our findings showed that requirements selection is tightly linked to the business situation and release planning. The requirements selection process for the product was affected by the release data. In other words, in the event that a release is going to be very late, the requirements selection would be different and client business situation would be used to select what they would prefer to release. These findings were particularly true for Company A and Company B where the requirements were defined by specific customers. On the other hand for Company C, where the product is developed based on a wider set of market requirements with many more customers, it was a different story. The selection and prioritization is made by a committee comprised of employees of the software development team representing the views of customers. As the customer base is so large no individual client has the level of influence or control that can be seen in Company A and Company B. The choice is made to provide “the greatest bang for the buck,” to maximize the value for the customer base at large.
With respect to stakeholders’ perspectives influencing requirements selection, the following can be said about alignment of product, project and business perspectives:

- It is interesting to see how these perspectives are represented in the entire process. For example, at Company C the business perspective is represented by the developers through their interaction with users in support forums. All developers rotate between support and coding. At companies A and B the developers are more distant from the requirements and these are generally expressed by the project manager (albeit through the product manager).

- With Company C’s approach they like to think of the customers’ problem as the developers’ problem – because the developer will have to deal with it – so is more likely to correct it, add the feature, etc ... Whereas you do not get this opportunity with the approach by the other companies.

- Because of the way the customers’ requirements are presented, they have always been filtered by at least one other stakeholder group’s perspective.

These conclusions have some implications. First, as the creation of software product value through requirements selection is not very well understood, it cannot be managed in the most effective way. Greater insight into how value is created through release planning would allow this process to be more effectively managed. Secondly the management of software product value is dependant on the context in which the product exists. Factors such as: the maturity of the product, the marketplace in which it exists and the development tools and methods available influence the criteria that decide whether a requirement is included in a specific project or release. A young product will have a greater need to recoup initial development costs, a more competitive marketplace requires more adaptive development, and standard software tools and techniques speed up development and reduce development resource issues. However, further research is required to determine what aspects of a software product’s context influence the decision-making criteria and how these contextual elements influence the priority given to the different decision-making criteria. These issues pose great challenges when it comes to creating software product value through requirements selection.
Chapter 4

A Method for Determining Alignment

Overview

There are many groups of people involved in the software development process. Each group represents a role with different responsibilities. This means that each group understands and is affected by software quality in different ways. However, there is an increasing body of work that recognises the need for success-critical stakeholders to have a common understanding of software quality in order to deliver a sustainable software product.

This chapter aims to support the software development process by describing the Stakeholder Alignment Assessment Method for Software Quality (SAAM-SQ), a method to understand the level of alignment between key stakeholders in the priority given to different aspects that describe software quality.

SAAM-SQ has been applied in a series of case studies. The lessons learnt from applying the method in these case studies are discussed to support researchers and practitioners in future applications of the method.

4.1 Introduction

Software development involves people working together to achieve a goal. However, the understanding of software quality is shaped by each stakeholders’ role,
experience and knowledge. This creates a potential for misalignment between success-critical stakeholders on issues of software quality, which can lead to conflict and project failure (Chan and Reich, 2007).

Possible perspective-based conflict is discussed in Chapter 3 of this thesis. It is shown that some roles are more important and influential than others when it comes to ensuring the profitability and long-term sustainability of the products in question. It also recognised that different factors affect people with different roles or perspectives in different ways.

The challenges of stakeholder alignment coupled with the subjectiveness of software quality in modern software development settings is an issue faced by numerous organisations (Johansson et al., 2001; Morgan, 2004). However, a common understanding of the expectations on software quality aspects has been found to be the most effective way for groups to achieve a common goal in a GSD setting (Phongpaibul and Boehm, 2005).

This chapter presents the Stakeholder Alignment Assessment Method for Software Quality (SAAM-SQ), a method to determine the alignment within and between success-critical stakeholder groups on priority given to software quality aspects. It draws upon the experiences of author in a series of successful and unsuccessful case studies aiming to meet this same objective.

The remainder of this chapter is structured as follows. The latest version of the SAAM-SQ is presented in Section 4.2. Experiences using the method in the various case studies are discussed in Section 4.3. The validity threats of the method are presented in Section 4.4. Finally, conclusions and recommendations for use of the method are presented in Section 4.5.

4.2 Method

This section presents a set of research objectives, an approach to meet the objectives, and methods to facilitate the analysis of the results of the approach.

4.2.1 Research Objectives

The objective of the method presented in this chapter, the Stakeholder Alignment Assessment Method for Software Quality (SAAM-SQ), is to determine the level of alignment between the success-critical stakeholder groups involved in software engineering. The method should be able to:

- Identify the degree to which the groups are aligned in how they perceive operations today with respect to quality aspects, and
4.2 Method

- Elicit how each group thinks the organisation should be operating today with respect to quality aspects—highlighting differences of opinion and potential improvement areas. This is referred to as the perceived ideal situation.

This leads to the following research questions:

**RQ1:** Is SAAM-SQ capable of identifying the degree to which the success-critical stakeholder groups are aligned in how they perceive the priorities on software quality aspects today?

However, alignment itself only ensures that the success-critical stakeholder groups have a common understanding of what is happening today, it does not mean the groups agree this is what should be happening today. As each group represents a different, and potentially conflicting, perspective on software quality aspects it is important to discover what each of these groups perceive should be happening in the situation today, in a hypothesised ideal situation. This is addressed by the second research question:

**RQ2:** Is SAAM-SQ capable of identifying what the different internal success-critical stakeholder groups perceive as the ideal set of priorities on software qualities in the situation today? To what degree are the groups aligned? And what changes do they perceive necessary?

Further, SAAM-SQ should be able to identify how united the internal success-critical stakeholder groups are in whom they perceive as managing the different software qualities.

**RQ3:** Is SAAM-SQ capable of identifying the level of consensus between the success-critical stakeholder groups as to who is responsible for managing software quality aspects? And what types of ambiguity exist, if any?

### 4.2.2 Approach

This chapter defines the SAAM-SQ, which was developed and used by the authors to determine the level of alignment between key stakeholder groups. Given the success of this method in its numerous applications to meet the aims of the study it has been refined and reused.
SAAM-SQ draws on the first steps of Theory-W (Boehm and Jain, 2006), which aims to create win-win scenarios by getting success-critical stakeholder groups to agree on how product development should proceed. The first steps of Theory-W are to identify the success-critical stakeholders, identify how they want to win, negotiate plans to achieve win-win scenarios and control the process to achieve these scenarios.

The main steps of SAAM-SQ are described below.

1. Select a company and product

As each product can have different quality requirements and quality expectations by success-critical stakeholders, it is essential to ensure the study is sufficiently focused.

2. Identify success-critical stakeholder groups

Success-critical stakeholder groups, are groups of people upon whom the success of the product depends—for example product managers and developers. Identifying these groups ensures critical perspectives are not lost, while less important perspectives cannot dominate.

3. Develop a quality model

Literature addressing software quality recognizes that quality and quality aspects depend on both the perspective of the observer and the actual software product in question. That is quality and quality aspects will be defined differently by different people, and quality and quality aspect will be defined differently for different products. As such, using any model of software quality as it appears in the literature risks not adequately defining quality in the context being studied. To use one of the quality models mentioned in Section 1.2 is a good starting point, but company and product specific needs must be taken into account (Chapter 5).

4. Develop a questionnaire

This method proposes the use of the hierarchical cumulative voting technique (HCV) (Berander and Jönsson, 2006) to elicit the priorities given the various software quality aspects. This method allows participants to state the relative importance of the aspects being studied by assigning a fix number of points across the set of aspects. This chapter assumes 1000 points will be assigned
to each table. Past research has shown that participants have trouble comparing some software quality aspects at a low level directly (Chapter 5); and HCV provides a method for breaking the problem into a series of smaller direct comparison exercise, with a method to join these results back together.

The third research question is addressed by asking the participants to consider each quality aspect individually and answer:

- The degree to which they thought the quality aspect was being managed today, with the answer presented on a five point Likert scale;
- The group perceived as ultimately responsible for managing the quality aspect; and
- The group perceived as managing the quality aspect on a day-to-day basis.

5. Conduct the questionnaire

The questionnaire should be completed by representatives of each of the identified success-critical stakeholder groups. Doing this in a one-on-one structured interview allows richer information to be collected from the participants, providing greater understanding of the results obtained. This also helps ensure participants have common understandings of the questionnaire with the interviewer being able to assist with questions or problems faced by the participants.

6. Analyse the results

A detailed explanation on how to analyse the results is provided in Section 4.2.3.

7. Workshop the results

Finally the results should be presented to participants from each of the success-critical stakeholder groups, asking them for their response to the following questions:

- Do these results look reasonable?
- Do the differences make sense?
- Why do these differences exist?
4.2.3 Analysis

Converting HCV Data to CV Data

HCV and Cumulative Voting (CV) allows participants’ responses to be grouped logically for analysis—for this method into the success-critical stakeholder groups. The results of each participant in a group can be averaged, ultimately producing lists that show how important each aspect is to each group. In order to conduct the analysis each participant’s response needs to be changed from HCV to CV, converting the cumulating voting lists that make up the HCV exercise into a single CV list.

Converting a HCV dataset to a CV dataset has been described by Berander and Jönsson (2006); Kuzniarz and Angelis (2011). A HCV dataset contains a master list, $HCV$. Each aspect of the master list refers to a sublist, $CV_i$. Thus the master list can be defined as $HCV = CV_1, ..., CV_n$, with $n$ sublists. Each sublist, denoted $CV_i$, contains a $n_i$ aspects. Each aspect is denoted $CV_{ij}$, thus $CV_i = CV_{i1}, ..., CV_{in_i}$. During a HCV exercise, points are awarded to each aspect, $CV_{ij}$ and each list $CV_i$ to donate the relative importance of the aspects in a list, and the relative importance of the lists to each other.

The first step of converting HCV values a single CV dataset involves calculating new values for each aspect studied. For each aspect $CV_{ij}$ we will denote a new value, $M_{ij}$. This new value is calculated by multiplying the points awarded to an aspect, $CV_{ij}$, by the number of points in the list containing that aspect, $n_i$ and the number of points to the list containing that aspect, $CV_i$. Thus:

$$M_{ij} = CV_{ij} \times n_i \times CV_i$$ (4.1)

Once this has been done for all aspects, it is possible to scale these values so that they sum to 1000 points. Defining a new value for aspect studied, $S_{ij}$ that corresponds to $CV_{ij}$ and $M_{ij}$, we can define:

$$S_{ij} = \frac{1000 \times M_{ij}}{\sum_{i=1}^{n} \sum_{j=1}^{n_i} M_{ij}}$$ (4.2)

Transforming each $CV_{ij}$ value to its corresponding $S_{ij}$ value, it is possible to complete the remainder of the analysis.

Analysis of Alignment within each Group

It is possible to determine the alignment between stakeholders within each group by measuring the inertia of the group. Inertia measures the variation in a
4.2 Method

dataset (Greenacre, 1984). It uses contingency tables, where two categorical variables are cross-tabulated and the chi-square statistic is computed to test the hypothesis of their independence. The inertia is computed by dividing the chi-square statistic by the total number of cases that are distributed in the cells of the contingency table.

For this method each group of stakeholders is considered separately. Given a group of \( S \) stakeholders and a set of \( I \) aspects, it is possible to denote \( n_{ij} \) as the amount allocated to each aspect \((j = 1, ..., I)\) by the stakeholder \((i=1..S)\) in the cumulative voting results. Thus, the mean value of the amounts allocated by each stakeholders within a group to issue \( j \) can be defined as:

\[
\bar{n}_j = \frac{1}{S} \sum_{i=1}^{S} n_{ij} \tag{4.3}
\]

Inertia checks the level of variation between the responses of the stakeholders in a group. When there is little variation between members of the group, all of the members prioritize the aspects in more or less the same way, which is very close to the means values calculated by Equation 4.3. The next step to calculating the inertia is to calculate the chi-square statistic for the group being studied:

\[
\chi^2 = \sum_{i=1}^{S} \sum_{j=1}^{I} \frac{(n_{ij} - \bar{n}_j)^2}{\bar{n}_j} \tag{4.4}
\]

The chi-square statistic tests the significance of the variation within the group. The test used is the common chi-square test, which compares the value computed in Equation 4.4 with the critical value of the theoretical chi-square distribution with \((S - 1) \times (I - 1)\) degrees of freedom. If the significance of the test is \(< 0.05\), then we can infer that the stakeholders within a group have significant variations in the priorities given to the aspects studied.

In order to compare between different groups it is necessary to normalize the result given by Equation 4.4 as this result is dependent on the number of stakeholders and aspects studied. This can be achieved by dividing the chi-square result by the total number of points allocated by all stakeholders to all aspects within the group studied:

\[
\text{inertia} = \frac{\chi^2}{\sum_{i=1}^{S} \sum_{j=1}^{I} n_{ij}} \tag{4.5}
\]
As the cumulative voting method gives 1000 points to assign across all the aspects, it is possible to simply this equation to:

\[
\text{inertia} = \frac{\chi^2}{1000 \times S} \tag{4.6}
\]

When the inertia of one group is larger than the inertia of a second group, then we can interpret it that there higher variability within the first group than within the second group.

Analysis of Alignment Between Groups

After the results have been transformed from an HCV dataset to a CV dataset it is possible to calculate the priority each group places on each quality aspect studied. Calculate the average value awarded to each quality aspect by each group. This should be done separately for the situation today and the perceived ideal situation.

The results can then be formatted as shown in Figure 4.1 to assist with analysis. A description of each section is provided below.

- **Q** Quality aspects: Keep the values associated with each quality aspect in one row.
- **Tp** Today’s priorities: Using separate columns for each group, list the number of points awarded to each aspect studied.
- **Tr** Today’s ranks: Using separate columns for each group, list the rank of each quality aspect by the group in question.
- **Tcp** Today’s correlation and p-values: Calculate a Spearman rank correlation matrix and corresponding p-value matrix, showing the level of alignment between the groups in how they describe the situation today (McDonald, 2009).
- **Ip** Ideal priorities: Using separate columns for each group, list the number of points awarded to each aspect studied.
- **Ir** Ideal ranks: Using separate columns for each group, list the rank of each quality aspect by the group in question.
- **Icp** Ideal correlation and p-values: Calculate a Spearman rank correlation matrix and corresponding p-value matrix, showing the level of alignment
4.2 Method

Figure 4.1: Format of data for analysis

between the groups in how they describe the ideal situation (McDonald, 2009).

Dr Difference ranks: Show how many places each aspect studied has risen or fallen between the situation today and the ideal situation for each group.

Dcp Difference correlation and p-values: Calculate a Spearman rank correlation values and corresponding p-values, the alignment between the priorities identified by each group in the situation today and the ideal situation (McDonald, 2009).

The researcher should then seek to understand the following:

- The overall priorities for the situation today.
- Key differences between the groups in their priorities today.
- The level of alignment between groups in the situation today.
• The overall priorities for the ideal situation.

• Key differences between the groups with their ideal priorities.

• The level of alignment between groups in the ideal situation.

• The amount of change each group would like to see to achieve their perceived ideal situation.

• The changes overall that all groups would like to see moving towards the ideal situation.

• The specific changes each groups would like to see moving towards the ideal situation.

The authors have found the use of sorting and colours to assist in the analysis of the data. For example, if all of the rank data uses a colour gradient to indicate the values, then sorting the data by the overall priorities or priorities of one group will quickly highlight any groups that disagree—with a colour outside its normal position in the gradient. Microsoft Excel provides all this functionality.

**Framework for Management Analysis**

Participants are asked to identify the group or groups within the organisation ultimately responsible for and managing each of the software qualities in order to address RQ3. A group is defined as managing a quality aspect if they are looking after the implementation of the quality aspect on a day-to-day basis and are likely to serve as the first point of contact if an issue arose. To be defined ultimately responsible a group should be heavily involved in setting the direction of and strategy for that quality aspect.

While thresholds are needed to structure the results, these are not defined as part of the methodology as purpose and requirements may change depending on the context of the case studied.

Ambiguity in leadership can be presented in four ways. The results are analysed using the framework described in this section.

*Blank responses:* A blank response indicates that a participant does not know who is in one of the leadership roles for a quality aspect.

*Insufficient votes:* It is also possible that no group or groups will be identified using the defined thresholds.

*Multiple groups:* While participants are able to select multiple groups as responsible for or managing each quality aspect, it is possible that multiple
groups are identified where the participants disagree over which group is in either leadership role for a quality aspect.

**Pushing/Pulling:** It is important to understand whether the group or groups in a leadership role have taken on that role or think that it belongs to another group. This is a two-step process:

1. Look at the responses from all participants to determine which group or groups are taking on a leadership role for a given quality aspect as defined previously.

2. Look at the responses of each of the identified groups to see which group these participants identified as taking on the same leadership role. If the group selects itself, it agrees with the majority and pulls this responsibility to itself. Alternatively, the identified group could push the responsibility onto another group.

For example, the results could indicate that project management is ultimately responsible for one quality aspect. If we look at the project managers responses there are two possibilities; (a) if a majority of the project managers identify their own role as ultimately responsible then the group pulls responsibility onto itself, (b) alternatively it is possible that a majority will identify another group — pushing responsibility to that group.

Pushing and pulling of roles can be broken down into several subgroups:

1. **Pull:** Where all participants identify one or more groups as responsible and each of the identified groups also identifies itself as responsible. Figure 4.2 shows an example where two groups—A and B—were identified as responsible, and each of these groups identifies itself as responsible. There are two types of pulling:

   (a) Single group: a single group is identified as responsible for or managing a quality aspect, and the identified group agrees with this finding.

   (b) Multiple groups: multiple groups are identified by all participants as responsible for or managing a quality aspect and each group identified itself as responsible or sharing the responsibility with the other identified groups. An example of this can be seen in Figure 4.2, with groups A and B.
2. Pull/Push: This situation is where multiple groups are identified as responsible for or managing a quality aspect, with one or more of the identified groups identifying another group as responsible, and one or more groups identifying themselves as responsible. For example, given groups A, B and C it is possible that groups A and B are identified as responsible for scalability. Group B might identify itself as responsible for scalability — pulling the responsibility to themselves, while group A might also identify group B as responsible — thus pushing the responsibility onto another group.

(a) Push to pullers: Where one or more of the identified groups nominates themselves as responsible and the remaining groups nominate the one or more of the groups as pulling responsibility. An example of this scenario can be seen in Figure 4.3.

(b) Combination of 2a and any of the scenarios described under 3, below.

3. Push: Where each group identified as responsible for or managing a quality aspect perceives another group taking on this role. For example, groups
A, B and C identify groups A and B as responsible for security. Group A identifies group B as responsible for security, and group B identifies group A as responsible.

(a) To group not identified: Where the group or groups identified by all the participants as responsible for or managing a quality aspect perceive another group or groups as responsible. An example of this scenario can be seen in Figure 4.4.

(b) Push to pushers: Where each group identified as responsible for or managing a quality aspect identifies another group that also pushes their identified responsibility.

(c) Combination of 3a and 3b.

![Figure 4.4: Example of (3a) push to a group not identified](image)

The most ideal situation is (1a), with a single group being identified and pulling that responsibility onto themselves as the responsibility is clearly defined. Second is (2a) the situation where multiple groups are identified, but between them they agree one group holding responsibility as the responsibility is clearly defined to those holding or perceived as holding it. Third is (1b) where multiple groups identify themselves as responsible, as responsibility is still being taken, but there is potential for conflicts with many groups. Finally situations involving only (3) pushing are the most dangerous as no group is taking responsibility.

### 4.3 Experiences and Discussion

This section presents the case studies that have employed the method described in Section 4.2, and then discusses the experiences following each step of the methodology.
4.3.1 Case Studies

SAAM-SQ has been applied in seven case studies, as shown in Figure 4.5. These case studies have primarily tested the alignment between success-critical stakeholder groups on the priority given to software quality aspects, but the method has also been applied other areas. A description of each of these studies is provided in the following sections.

![Figure 4.5: Case studies employing SAAM-SQ](image)

**S1. Onshore Insourcing—Quality**

Study 1 is presented in Chapter 5 of this thesis. This study presents a case study exploring the alignment of groups on the priority of software quality aspects. The participants are fit into the onshore insourcing category for one product at Ericsson, Sweden.

**S2. Onshore Outsourcing—Quality**

Study 2 is presented in Chapter 6 of this thesis. This study extends Study 1 to cover stakeholders fitting into the onshore outsourcing category for the same product. For this study developers from two consulting companies were asked to
complete the same questionnaire as in the first case study, and then a comparison was made between the two case studies.

S3. Offshore Insourcing—Quality

Study 3 is presented in Chapter 7 of this thesis. It further extends Study 1 and Study 2 to include stakeholders by offshore insourcing—that is Ericsson employees outside of Sweden, specifically India for this case study. The study examined two products, one new and one the same as previously studied. Participants completed the same questionnaire as in the first two studies. Divergent results between the participants limited the comparisons that could be made to the previous studies.

S4. Onshore Insourcing—Intellectual Capital

In Study 4 the method was applied to understand the alignment between groups on the investment priorities for intellectual capital for the same product and groups studied in Study 1. The results of this study are presented in Barney et al. (2009a) and highlight that each group has different intellectual capital needs.

S5. Onshore Insourcing—Investments

Study 5 is presented in Chapter 9 of this thesis. It combines the results of Study 1 and Study 4 with new data to provide a richer understanding of the investment space needed in software product development.

S6. Global Insourcing—Quality

In Study 6 the method was applied to a third product for groups defined by both onshore and offshore insourcing. Insufficient data points were collected to analyse the results.

S7. Global Software Development—Quality

Study 7 is presented in Chapter 8 of this thesis. In this study the method was used to examine software development on one product across three countries and involving insourcing and outsourcing.
4.3.2 Experiences Applying SAAM-SQ

The following sections present and discuss the experiences of the researchers who have applied SAAM-SQ in the aforementioned case studies. These experiences are divided into eight sections, the first examining the application of the method as a whole and the remainder going through each step of the method.

Conducting the method

The most critical success-factor for conducting this research is management support. Due to the cross-functional nature of this work, this usually means the support of multiple managers. Management support makes people more positive about participating in the research, and will lead to it being prioritised more highly by the participants.

SAAM-SQ is dependent on a set of success-critical stakeholder groups. These groups can have only few members or even one member. This can make the method very dependent on key people. The authors’ experience has found that if one key stakeholder withholds their support from this work, it can severely limit the value of the results obtained.

1. Select a company and product

To date this research has been primarily supported by Ericsson. Managers at Ericsson have been involved in the direction and scope of this research. Six of the seven case studies employing the method described in Section 4.2 have been conducted at Ericsson. The seventh, presented in Chapter 8, was conducted at a firm in which one of the coauthors was working.

The authors who have conducted these case studies recognise a need for any study employing this method to be sufficiently focused. Different products have different quality requirements, and even different parts of the same product can have different quality requirements. Making the scope of an individual study too large, will mean including people who have good reasons to have differences in priorities. While it is possible for the method to identify these differences, this research is not inline with the aims of the method, and making comparisons within or between groups will not make sense when one can reasonably expect there to be differences.

Taking the example of a mobile phone, engineers developing the hardware interface drivers to for phone call functionality will have very different reliability and usability requirements than engineers developing a game.
2. Identify key stakeholder groups

In each case study the authors worked with managers to identify the success-critical stakeholder groups. This has not been perceived as a difficult undertaking by the authors or the managers.

One open question has been around when the research is sufficiently mature to involve customers. To date customers have not been involved in the research, but roles representing the customer perspective have. However, it is expected that there will be some differences in the priorities between the customers and roles like customer support and product management, as each of these roles has additional concerns that do not directly impact the customer. For example, the product manager will be much more concerned about the profitability of the product than the customer.

3. Select/Tailor a quality model

The process of selecting and tailoring a quality model for the case study presented in Study 1 was found to be a very valuable activity on its own. Representatives of all the success-critical stakeholder groups participated in two workshops to tailor the ISO9126 (2001) model to the organisational needs, and create definitions for the aspects upon which they decided.

The workshop discussions highlighted that different groups used different words to describe the same quality aspect, and different groups used the same words to describe different quality aspects. The discussion proved to be enlightening for many of the participants. It also highlighted areas where there had been misunderstandings, which could now be addressed. These discussions were very useful, as it gave the participants a common vocabulary from which to discuss software quality going forward. The same rewards was not seen for the other quality studies in Ericsson, as the same model of quality was reused to save time and cost.

The findings of the workshop were documented and used for all of the studies at Ericsson to-date. While statistics were not kept, the authors’ were surprised by the number of questionnaire participants who requested to keep a copy of the model and definitions created in the workshop. This result was seen in all studies. These requests came from all groups, but developers and testers stood out as the groups requesting the most copies. Reasons for wanting to keep a copy of the model and definitions included:

- For use as a personal check-list to help improve development and testing skills.
• To help ensure all issues had been considered and reduce the risk of unan-
ticipated issues.

• To help put forward more reasoned and structured arguments when it was
felt more time was needed to deliver a piece of functionality adequately.

For Study 7 it was not possible to get the same level of input on the develop-
ment of the quality model. One of the authors worked with a project manager
and a business analyst to create the quality model. The feedback from the ques-
tionnaire participants found the model to be lacking, especially in the area of
security—which was perceived as a key quality aspect. This emphasizes the need
and value of involving representatives of all success-critical stakeholder groups
in the process—in Ericsson this step of the method was considered valuable
activity, independent of the rest of the process.

In Study 4, examining the alignment on aspects of intellectual capital, par-
ticipants had much more trouble approaching the concepts that were the focus
of the study. One likely reason is that the participants learn about, experience
and understand software quality and how it relates to their job. But without
a formal education in aspects relating to intellectual capital, they feel less able
to comment on these issues.

4. Develop a questionnaire

The use of the CV method to prioritize the various aspects studied was first
proposed. However, the pilot undertaken as part of the first study applying the
method found participants having great difficulty making direct comparisons
between what they perceived as separate concepts. The method was updated
to use the HCV method, which was found more successful when used with ques-
tionnaire participants. All subsequent studies have employed the HCV method.

One of the benefits of the HCV method is that it allows different result sets to
be combined, providing they can be presented in a compatible hierarchy form.
This technique was used to understand the investment option space between
functionality, scope and intellectual capital within the constraints of time and
cost in Study 5. This chapter was able to bring together the priorities identified
Study 1 and Study 4.

The technique used in Study 5 could be used in a complex system, where no
one felt able to describe the priorities on quality aspects at the level of detail
required. Some groups could describe the situation at higher levels, while other
groups could describe the situation at lower levels. The HCV method then
allows for these results to be brought together.
Not all participants like using the HCV method. It is perceived as complex and time consuming to get the totals to sum to 1000 while maintaining the relative importances at the desired levels. However, the authors considered the major alternatives—such as ranking, AHP or Incomplete Pairwise Comparison (IPC) (Johansson et al., 2001)—but concluded that HCV offers the best compromise.

Ranking quality aspects does not allow results to be combined for further analysis. For example, it is not possible to work out a single preference rank of software quality aspects given the individual ranks of six different developers. It would be possible to consider getting one single ranked list from each group studied, but it has been shown that members of a group are not always in agreement Study 3—which itself is an important result.

AHP gets participants to make comparisons between each pair of attributes. For the study in Ericsson this would have meant 325 comparisons for the 26 aspects studied. Further, the pilot showed participants unable or unwilling to make comparisons between some pairs of aspects, making the use of this method more troubling.

Similarly it was felt that IPC was difficult to operationalise. Further, it could still suffer from the same problem as AHP, with participants needing to make comparisons between aspects they did not feel willing or able to make.

5. Conduct the questionnaire

In total two approaches have been used to conduct the questionnaire. For Study 1 to Study 5 participants were invited to attend one-on-one interviews in which they completed the questionnaire with support from one of the authors. For Study 6 and Study 7 instructions and surveys were sent out for the participants to complete at a time of their own convenience. The one-on-one interview approach yielded a much higher response rate at 70–100%, while the online method received a 28% response rate at the Ericsson.

Feedback was sought and collected from Ericsson workers selected to participate in the online studies on their experience using the method. Responses were gathered from both people who returned a completed questionnaire and people who did not return a questionnaire. The overwhelming response was that the activity took much longer than they expected, with most giving up.

Outside the interview situation people did not time-box the activity. Interviews were scheduled to last 30 minutes, but in reality lasted between 20 minutes and one hour. While online participants were told that the task should take approximately 30 minutes to complete, many admitted to spending hours
working on the task. Due to this extended time to complete the task many gave up.

Many of the online participants in the online forum further feared their answers were not sufficiently accurate—with a number indicating they lacked a sufficient objective measurement to make the comparisons. In the interview situation, however, participants felt compelled to complete what they started to the level of quality achievable in the time allocated. Further, in the interview situation the author was able to reassure the participant that the response that was completed in the allocated time was sufficient for the purpose of the study.

Fearing their results were not accurate, participants in the online interviews sought “correct” or “expert opinions” from managers and other workers. This went against the objectives of the study, which aimed to determine the alignment within and between groups on their understanding of software quality aspects. This approach was most commonly seen with part of the questionnaire addressing RQ3.

A couple of participants in the online study stated that they had difficulty translating the quality model into the real-life situation. One noted that while there is a “prioritization between different aspects but we do not think of it in these terms.”

The second study using online data collection, Study 7, was more successful. The reason for this is thought to be that one of the authors was working in the project and had a professional relationship with all of the participants in the study. This would likely lead to the participants feeling a greater obligation to return the questionnaires, as they know the person they would be letting down and would have a continuing relationship with them.

For the study at Ericsson using the online approach to data collection, an introductory email was sent out containing the following elements:

1. Introductory email inviting the person to participate in the study.

2. Introduction video\(^1\): Presents the aims of the study and briefly describes what is required.

3. A copy of the survey to complete.

4. A copy of the model of software quality, with definitions for each of the terms.

\(^1\)http://www.youtube.com/user/sebastianbarney#p/a/u/0/cNTrjjxwhuw
5. HCV instructions video\textsuperscript{2}: Explains how to complete the HCV activities in more detail.

6. Other instructions video\textsuperscript{3}: Support people having trouble with the other management of the survey.

From the online study at Ericsson, 16 responses were received from the 62 candidate participants. The number of views of each video is shown in Table 4.1. It is interesting to note the high decay rate, with only two-thirds watching the introductory video, one third watching the HCV instructions and a quarter watching the other instructions. These results suggest maintaining interest in an activity like this one is a difficult undertaking in a remote scenario.

Table 4.1: Use of materials in online surveys

<table>
<thead>
<tr>
<th>Video</th>
<th>Views</th>
<th>Participants</th>
<th>View Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>40</td>
<td>62</td>
<td>65%</td>
</tr>
<tr>
<td>HCV instructions</td>
<td>20</td>
<td>62</td>
<td>32%</td>
</tr>
<tr>
<td>Other instructions</td>
<td>15</td>
<td>62</td>
<td>24%</td>
</tr>
</tbody>
</table>

6. Analyse the results

The techniques used to analyse the results have improved as the method has been used, reused and refined.

When the method described in Section 4.2 was first applied in Study 1, there was no definitive answer on what weighting factors should be used for each sublist with the HCV method (Berander and Jönsson, 2006). This issue has since been resolved, and it is recommended to use the number of items in the group as the weighting factor (Kuzniarz and Angelis, 2011). All studies have used this method as it was initially identified as the best approach (Berander and Jönsson, 2006).

The authors' believed that there were varying levels of alignment within each group studied, however, this was difficult to confirm numerically. Additional researchers were brought in to help analyse this situation for Study 3, and a method was developed that allows for the level of alignment to be calculated.

\textsuperscript{2}http://www.youtube.com/user/sebastianbarney#p/a/u/2/x4fnKfO-0sg
\textsuperscript{3}http://www.youtube.com/user/sebastianbarney#p/a/u/1/mrEHuSz8kCs
for each group. This alignment figure is normalised so that it can be compared between groups. Further it is possible to calculate the significance level of the alignment, making it possible to say if the level of alignment is significant for a group. This method is presented in Section 4.2.3.

The method has been further refined to include significance levels when looking at the level of alignment between groups, and the amount of change desired by each group. Using p-value statistics, make it possible to say with statistical significance if two groups are aligned, or if one group wants significant changes to the set of priorities as they are understood today.

7. Workshop the results

Many interesting results can be found analysing the numbers, but a deep understand of the results requires talking to representatives of all of the success-critical stakeholder groups. In particular where there are differences, it is important to hear from people who hold this differing point of view why they perceive things differently.

The workshops have been found to run best when they have a clear agenda. The facilitator should present the results, and have a list of issues on which they wish to gain a deeper understanding. More focused questions on the part of the researcher has been generally found to lead to more focused answers.

4.3.3 Summary

A summary of the key improvements made to the method in each of the studies is presented in Table 4.3.3. These improvements are listed in chronological order.

4.4 Validity Threats

When developing the model and definitions to describe software quality for the product in question, it is easier for the workshop participants to agree with the set of criteria identified by the researchers than disagree. This could lead to an over-theoretical model that does not represent the understanding of software quality within the organisation. This is partially taken care of by allowing the participants to assign a relative importance of zero to any aspect.

It is also difficult to know whether the respondents have understood the questions as intended and in a similar fashion to one another. This threat is partially
4.5 Conclusions

This section presents findings and recommendations from applications of the Stakeholder Alignment Assessment Method for Software Quality (SAAM-SQ), outlines the remainder of this thesis and presents future work.

Table 4.2: Chronological listing of improvements to the method

<table>
<thead>
<tr>
<th>Study</th>
<th>Improvements</th>
</tr>
</thead>
</table>
| S1    | Foundation of set and applied.  
Use of HCV over CV. |
| S4    | Possible scope of method expanded, with study of intellectual capital. |
| S5    | Exploited HCV method to allow different people to prioritise different parts of the hierarchy. |
| S6    | Method adapted to be collect information online. |
| S7    | Inclusion of inertia statistic to test alignment within a group. |
| S3    | Inclusion of p-value statistics to assess confidence. |

addressed in the final stage of this analysis, where the results were presented in the final workshop for confirmation and discussion with the researcher.

SAAM-SQ does not account for the level of influence each role and individual has in shaping software quality. It is possible that some roles and/or individuals are more influential than others. The results overall and for each group represent an average of the priorities of the members of the same group. While members of each group were not always aligned, the average represents a possible mutual consensus assuming all individual are equal. It is also important to remember that the purpose of this method is to measure alignment—so if a lack of alignment is found within a group or between groups, is important to ask what further analysis can be done. It may be that if members of a group are not aligned, or groups are not aligned, then it does not make sense to determine overall priorities from the data. The method further recommends workshops with representatives of the key stakeholder groups are held to help ensure reasonable results were gathered.

The results of any individual study have limited generalizability (Runeson and Höst, 2009). It is expected that priorities will vary between groups, products and at different stages of a product’s lifecycle. However, the method has been proven to work in different contexts and can be applied to other cases.
4.5.1 Findings and recommendations

This chapter aims to present a best-of-breed method for determining the alignment between success-critical stakeholder groups on the priority given to software quality aspects. The paper draws upon the experiences using SAAM-SQ in seven case studies. The method has successfully been able to determine the level of alignment between success-critical stakeholders in the priority they give to software quality aspects both in the situation today and in the ideal situation, answering RQ1 and RQ2. However, only Chapter 5 sought to answer RQ3—which it did successfully—due to a focus on RQ1 and RQ2 by the research collaborators.

From the experiences applying SAAM-SQ, a set of best practices have been identified to support future applications of the method:

- The full value of applying SAAM-SQ can only be achieved with the inclusion of all success-critical stakeholder groups. This is likely to require the support of multiple managers. It is important to recognise that the exclusion of one of more key roles can severely limit the value of applying the method.

- Any study employing SAAM-SQ must be sufficiently focused. Large products may consist of various components, each with different priorities for the same set of quality aspects, and in this situation it is reasonable that groups will not be aligned.

- Tailoring a quality model to describe software quality as it is understood for the software product alone is a valuable exercise if done correctly. Bringing representatives of all stakeholders together to discuss software quality and quality aspects can itself highlight differences in understanding that can cause confusion and miscommunication about software quality and quality aspects.

- Using the HCV method in the questionnaire allows the greatest flexibility in the analysis of the results. It allows groups of results to be brought together, and for the results to be extended by exploiting the hierarchical nature of HCV.

- Much greater success has been achieved conducting the questionnaire in a one-on-one style interview. However, personal relationships have also been found successful at getting timely responses from participants. As
the method is slightly complex and subjective, participants can feel uncomfortable completing the task individually.

- A set of methods has been developed and refined to statistically and qualitatively analyse the data. These are presented in Section 4.2.
- Finally, conducting workshops with representatives of all the success-critical stakeholder groups allows for a much deeper understanding of the results.

### 4.5.2 Application of the method

The remainder of this thesis presents a selection of the case studies introduced in this chapter that employ SAAM-SQ. Due to the evolution of the method over the period in which these case studies were conducted and this thesis being a collection of publications, not all chapters presented exploit all parts of the method as it is presented in this chapter. However, the chapters do show the development and maturity of the method over this time.

Chapter 5, Chapter 6 and Chapter 7 present a case study of the alignment of key stakeholders on software quality for one product at Ericsson. The chapters respectively explore the situations described by onshore insourcing, onshore outsourcing/insourcing and offshore insourcing. Chapter 7 further presents a case study of a second product at Ericsson described by offshore insourcing.

Chapter 8 presents a case study applying SAAM-SQ at a multinational financial services organisation. This case involves all combinations of insourcing/outsourcing and onshore/offshore.

Finally, Chapter 9 presents an extension of SAAM-SQ—bringing together results from Chapter 5 and Barney et al. (2009a) with the collection of new data.

### 4.5.3 Future work

With an understanding of the levels of alignment, and differences in priorities, more work needs to be done to support software development organisations reach a mutual consensus on how they can best move forward. This work includes techniques to understand and analyse the decision space to make better trade-off decisions between the various software quality aspects. Additionally, work needs to be done to bring the success-critical stakeholder groups together, so that they understand and agree why the chosen strategy is in the best interest of the development organisation.
Chapter 5

Quality Alignment Inside an Organisation

An abridged version of this chapter is published as:


Abstract

Software qualities are in many cases tacit and hard to measure. Thus, there is a potential risk that they get lower priority than deadlines, cost and functionality. Yet software qualities impact customers, profits and even developer efficiency. This chapter presents a method to evaluate the priority of software qualities in an industrial context. The method is applied in an exploratory case study, where the ISO 9126 model for software quality is combined with Theory-W to create a process for evaluating the alignment between success-critical stakeholder groups in the area of software product quality. The results of the case study using this tool is then presented and discussed. It is shown that the method provides valuable information about software qualities.
5.1 Introduction

Software quality forms an important part of a product offering. But which qualities that are valuable is a very context dependant problem, changing with both the product and perspective you bring. Maximising the value of a product’s quality involves reconciling any conflicts between the key stakeholder groups — including customer, business and technical perspectives — so that these groups can work together effectively towards a common goal. However, there is always a risk that qualities get a lower priority than delivery date, cost and functionality. This risk comes from the fact that the qualities are most difficult to measure in relation to delivery date, cost and functionality. The balance between delivery time, cost, scope and quality is discussed as part of XP (Beck, 2000).

This obvious risk forms the starting point of the research presented in this chapter. The main objective is to understand the priorities, responsibilities and management of software qualities in an industrial context. This chapter makes three main contributions. First of all, a method for analysing qualities in an industrial organisation is presented. Second, the method is applied and an industrial study exploring the alignment of priorities given to various software qualities between the groups involved in the software development process is presented. The third contribution is related to exploring the ambiguities present in the management of a software product.

Asking what defines an adequate level of quality in a software system is a highly context dependent question (Kitchenham and Pfleeger, 1996). Software quality affects more than just the user of the software and each group involved with a software product brings its own perspective on quality (Kitchenham and Pfleeger, 1996). Customers, developers, product managers, project managers and testers can all value the same qualities of the same product in different ways. Looking at the software supporting a social networking site, one could reasonably expect that customers would value usability higher than the other groups, developers value maintainability due to the dynamic nature of the product and management values efficiency due to the scale and resources required of the application. Ultimately these value stances need to be reconciled.

The qualities each group values will also change depending on the product. Functionality is becoming increasingly important for mobile phones, reliability is more important in financial and medical domains, as is portability for web-based applications.

Understanding both (a) the groups impacted by a software product and (b) the value provided to each group by the various software product qualities is useful information for companies developing software. As any development
5.2 Background

In the software development process there are four variables that need to be controlled — cost, time, quality and scope (Beck, 2000). Further there is an axiom that states that external forces can set at most three of these variables with the remainder being set by the development team.

Time, cost and scope can all operate within acceptable ranges, but quality is a terrible control variable as it only allows very short term gains at a very high cost to all parties involved (Beck, 2000). That said, quality does not need to be perfect (Yourdon, 1995) but the development process is much simpler when the success-critical stakeholders agree on what action should be taken (Boehm and Ross, 1989).

This section examines different perspectives of quality, processes to manage quality in a software engineering context, and the research objectives.

5.2.1 Definitions of Quality

The definitions of quality are both many and conflicting, even when only examining the topic in relation to software engineering. Looking across different disciplines it is possible to see a complex multifaceted concept of quality that can be described from five different perspectives (Garvin, 1984):
The transcendental perspective defines quality as something that can be recognized but not defined in advance.

The user perspective defines quality as fit for purpose.

The manufacturing perspective defines quality as conformance to specification.

The product view defines quality in terms of essential characteristics of the product in question.

The value-based view defines quality in terms of the amount a customer is willing to pay for it.

By far the most common perspectives taken in the software development industry are that of the user and manufacturer. (Kitchenham and Pfleeger, 1996; Hoyer and Hoyer, 2001). However, there is an increasing body of literature that recognises the importance of taking advantage of all of the perspectives involved in software development. Theory-W states that success requires all of the success-critical stakeholders to compromise (Boehm and Ross, 1989), while requirement specification reading techniques that take advantage of different perspectives have been found to catch 35% more defects than non-directed alternatives (Basili, 1997; Boehm and Basili, 2001), and value-based software engineering now recognises the value brought by different perspectives into the development process (Kitchenham and Pfleeger, 1996).

Software quality is not only defined by the relevant perspectives, but also by the context in which it exists (Kitchenham and Pfleeger, 1996). Just as each line of cars has a target market, software quality must be planned to allow a development company to meet its business objectives. Less than perfect software quality can in fact be ideal (Yourdon, 1995), but deciding how much less than perfect can only be decided in a given business context (Kitchenham and Pfleeger, 1996).

5.2.2 Quality Models for Software Development

Numerous models have been developed to support software quality. Examples of these models include McCall’s quality model, Boehm’s quality model, Dromey’s quality model and ISO 9126.

McCall’s quality model is the first of the modern software product quality models (Kitchenham and Pfleeger, 1996). The model uses a hierarchy of factors,
5.2 Background

criteria and metrics to address internal and external product quality. Eleven factors define an external or user perspective of quality. Each of these factors is linked to between two and five of 23 criteria that define an internal or development perspective of quality. Further metrics are associated with the factors allowing quality to be measured and managed.

McCall’s quality model was followed by Boehm’s quality model (Kitchenham and Pfleeger, 1996). Like McCall’s model, Boehm’s model presents product quality in a hierarchy with three high level characteristics linked to seven intermediate factors, which are in turn linked to 15 primitive characteristics. Boehm’s model has a wider scope than that of McCall’s, with more emphasis on the cost-effectiveness of maintenance (Milicic, 2005).

More recently work has been done to create an international standard for software product quality measurement — ISO9126 (2001). This standard is again organised in a hierarchy with six characteristics at the top level and 20 sub-characteristics with indicators used to measure the sub-characteristics. In addition to aspects of internal and external quality, covered by McCall and Boehm’s models, ISO 9126 includes quality characteristics of functionality (Milicic, 2005). Internal, external and functional qualities are also mixed at all levels of the hierarchy. However, ISO 9126 does not clearly state how quality should be measured (Kitchenham and Pfleeger, 1996).

None of these three models present a rationale for the selection of characteristics to be included in the quality model and it is not possible to tell if a model presents a complete or consistent definition of quality (Kitchenham and Pfleeger, 1996). Further the placement of items appears arbitrary in ISO 9126, for example, with no justification as to why Interoperability is not related to Portability.

Dromey presents a different type of model that attempts to address some of the issues presented and support developers to build product quality (Dromey, 1996). Dromey believes that it is impossible to build high-level quality attributes like reliability or maintainability into a product, but developers must instead build properties that manifest in achieving these goals. The distinction this model makes is important, as using the model will verify that the model allows the quality required to be achieved (Kitchenham and Pfleeger, 1996). Before Dromey’s model can be successfully applied, the various groups involved in the development of a software product must agree on what quality attributes should be achieved and to what level. This process can be supported using other models.
5.2.3 Merging Perspectives on Software Quality

Software product quality can easily become an area of problems and conflict, as each stakeholder group has its own perspective on what is important. A number of methods can be applied to help reconcile this situation and select the best way forward. These methods include expert judgement, the NFR Framework, Quality Functional Deployment and Theory-W.

Expert judgement involves one or more experienced professionals using their experiences and knowledge to make a decision on an issue. The decisions are not necessarily supported by modelling or numerical assessment.

The NFR Framework uses diagrams to relate non-functional requirement goals with different decisions that can be made in the design and operation of a system that affect it positively or negatively, allowing trade-offs to be identified and made (Chung et al., 2000). While this method makes the results of a choice to be made more explicit, it requires a set of common priorities to be identified to allow effective decisions to be made.

Quality function deployment (QFD) considers the priority of customer and technical requirements in achieving the goals of the system to help prioritize the requirements (Herzwurm et al., 2003). However, the other perspectives involved in the development of the software product are not considered.

Value-based software engineering (VBSE) recognises the problems created by conflicting perspectives in the software development process (Boehm and Jain, 2006). Central to resolving conflict in VBSE is Theory-W, which requires (Boehm and Ross, 1989):

1. Success-critical stakeholder groups to be identified;
2. The requirements of these groups to be elicited;
3. Negotiation between the groups to create a win-win situation; and
4. A control process to support success-critical stakeholder win-win realisation and adaption to a changing environment.

The key advantage of Theory-W is that it explicitly brings all of the parties on whom success depends together to understand each other’s needs, compromise and agree. But in order to be successful Theory-W must be managed to ensure the plans are achieved and any deviations from the plans are corrected (Boehm and Ross, 1989). Management requires an understanding of why the goals are being pursued, what is the required result, who is responsible for the result, how the result will be achieved and at what cost the result can be
achieved. The answer to these questions will be specific to the context in which they are answered.

5.2.4 Research Objectives

The objective of the research presented in this chapter is to create and validate a method capable of determining the level of alignment between the internal success-critical stakeholder groups. The method should be able to:

• Identify the degree to which the groups are aligned in how they perceive operations today with respect to quality, and

• Elicit how each group thinks the organisation should be operating today with respect to quality — highlighting differences of opinion and potential improvement areas. This is referred to as the ideal situation.

This method should be evaluated in an industrial case, answering the research questions presented in this section.

**RQ1:** Is the method proposed in this chapter capable of identifying the degree to which the internal success-critical stakeholder groups are aligned in how they perceive the priorities on software product quality today?

However, alignment itself only ensures that the success-critical stakeholder groups have a common understanding of what is happening today, it does not mean the groups agree this is what should be happening today. As each group represents a different, and potentially conflicting, perspective on software product quality it is important to discover what each of these groups perceive should be happening the situation today, in a hypothesised ideal situation. This is addressed by the second research question:

**RQ2:** Is the method proposed in this chapter capable of identifying what the different internal success-critical stakeholder groups perceive as the ideal set of priorities on software product qualities in the situation today? And to what degree are the groups aligned?

Further, the method should be able to identify how united the internal success-critical stakeholder groups are in whom they perceive as managing the different software product qualities.
RQ3: Is the method proposed in this chapter capable of identifying the level of consensus between the success-critical stakeholder groups as to who is responsible for managing software product quality attributes? And what types of ambiguity exist, if any?

5.3 Methodology

To address the research questions a method using Theory-W as a starting point is developed. By exploiting the early phases of Theory-W it is possible to determine the level of alignment between the internal success-critical stakeholder groups – addressing the first two research questions. This involves identifying the internal success-critical stakeholder groups and eliciting their value propositions with respect to quality.

Further, by collecting information on who each of the internal success-critical stakeholder groups identify as responsible for and managing each quality, it is possible to analyse the level of consensus between these groups — addressing RQ3.

The method should support the continued application of Theory-W, to negotiate between the success-critical stakeholders to achieve a better situation and realise this goal through clearer management.

5.3.1 Quality Model

The literature on software product quality recognises that quality depends both on the perspective of the observer and the actual software product in question. As such, using any model as it appears in the literature risks not adequately defining quality in the context being studied. To use one of the quality models briefly introduced in Section 5.2 is a good starting point, but company specific needs have to be taken into account, as illustrated in the case study in Section 5.4.

5.3.2 Questionnaire

This method proposes the cumulative voting (CV) (Leffingwell and Widrig, 1999) technique to elicit both (a) how important each quality is today, and (b) how important each quality should be today in a perceived ideal situation – answering the first two research questions. CV asks participants to spend 1000 points across all of the qualities previously identified, to represent their relative
influence. For example, if a participant thought testability does not matter at all today and security was twice as important as scalability they might award these qualities zero, 200 and 100 respectively, leaving 700 points to be spent on the remaining qualities.

The third research question is addressed by asking the participants to consider each quality individually and answer:

- The degree to which they thought the quality was being managed today, with the answer presented on a five point Likert scale;
- The group perceived as ultimately responsible for managing the quality; and
- The group perceived as managing the quality on a day-to-day basis.

Additionally, participants are given the opportunity to make comments about any quality, or general comments.

### 5.3.3 Analysis

The analysis techniques used to address the research questions are presented in the following subsections.

#### Importance of Qualities

CV allows participants’ responses to be grouped logically for analysis — for this method into the success-critical stakeholder groups. The results of each participant in the group can be averaged for each quality, ultimately producing a list that shows each quality and the averaged notion of its importance.

From here it is possible to rank the qualities from most to least influential for each success-critical stakeholder group. The degree to which the groups are aligned can then be calculated pairwise using a Spearman rank correlation matrix.

#### Framework for Management Analysis

Participants are asked to identify the group or groups within the organisation ultimately responsible for and managing each of the software product qualities in order to address RQ3. A group is defined as managing a quality if they are looking after the implementation of the quality on a day-to-day basis and are likely to serve as the first point of contact if an issue arose. To be defined
ultimately responsible a group should be heavily involved in setting the direction of and strategy for that quality.

While thresholds are needed to structure the results, these are not defined as part of the methodology as purpose and requirements may change depending on the context of the case studied.

Ambiguity in leadership can be presented in four ways. The results are analysed using the framework described in this section.

Blank responses: A blank response indicates that a participant does not know who is in one of the leadership roles for a quality.

Insufficient votes: It is also possible that no group or groups will be identified using the defined thresholds.

Multiple groups: While participants are able to select multiple groups as responsible for or managing each quality, it is possible that multiple groups are identified where the participants disagree over which group is in either leadership role for a quality.

Pushing/Pulling: It is important to understand whether the group or groups in a leadership role have taken on that role or think that it belongs to another group. This is a two-step process:

1. Look at the responses from all participants to determine which group or groups are taking on a leadership role for a given quality as defined previously.

2. Look at the responses of each of the identified groups to see which group these participants identified as taking on the same leadership role. If the group selects itself, it agrees with the majority and pulls this responsibility to itself. Alternatively, the identified group could push the responsibility onto another group.

For example, the results could indicate that project management is ultimately responsible for one quality. If we look at the project managers responses there are two possibilities; (a) if a majority of the project managers identify their own role as ultimately responsible then the group pulls responsibility onto themselves, (b) alternatively it is possible that a majority will identify another group — pushing responsibility to that group.

Pushing and pulling of roles can be broken down into several subgroups:

1. Pull: Where all participants identify one or more groups as responsible and each of the identified groups also identifies itself as responsible. Figure 5.1 shows an example where two groups — A and B — were identified as
responsible, and each of these groups identifies itself as responsible. There are two types of pulling:

(a) Single group: a single group is identified as responsible for or managing a quality, and the identified group agrees with this finding.

(b) Multiple groups: multiple groups are identified by all participants as responsible for or managing a quality and each group identified itself as responsible or sharing the responsibility with the other identified groups. An example of this can be seen in Figure 5.1.

![Identified Groups](image)

Figure 5.1: Example of (1b) pulling by multiple identified groups

2. Pull/Push: This situation is where multiple groups are identified as responsible for or managing a quality, with one or more of the identified groups identifying another group as responsible, and one or more groups identifying themselves as responsible. For example, given groups A, B and C it is possible that groups A and B are identified as responsible for scalability. Group B might identify itself as responsible for scalability — pulling the responsibility to themselves, while group A might also identify group B as responsible — thus pushing the responsibility onto another group.

(a) Push to pullers: Where one or more of the identified groups nominates themselves as responsible and the remaining groups nominate the one or more of the groups as pulling responsibility. An example of this scenario can be seen in Figure 5.2.

(b) Combination of 2a and 3a, 3b or 3c.

3. Push: Where each group identified as responsible for or managing a quality perceives another group taking on this role. For example, groups A, B and
Identified Groups

A

B

Figure 5.2: Example of (2a) push to pullers within identified groups

C identify groups A and B as responsible for security. Group A identifies group B as responsible for security, and group B identifies group A as responsible.

(a) To group not identified: Where the group or groups identified by all the participants as responsible for or managing a quality perceive another group or groups as responsible. An example of this scenario can be seen in Figure 5.3.

(b) Push to pushers: Where each group identified as responsible for or managing a quality identifies another group that also pushes their identified responsibility.

(c) Combination of 3a and 3b.

Figure 5.3: Example of (3a) push to a group not identified

The most ideal situation is (1a), with a single group being identified and pulling that responsibility onto themselves as the responsibility is clearly defined. Second is (2a) the situation where multiple groups are identified, but between them they agree one group holding responsibility as the responsibility is clearly defined to those holding or perceived as holding it. Third is (1b) where multiple
groups identify themselves as responsible, as responsibility is still being taken, but there is potential for conflicts with many groups. Finally situations involving only (3) pushing are the most dangerous as no group is taking responsibility.

To answer the research questions, this method can now be piloted and trialled. The results are presented in Section 5.4.

5.4 Case Study

The case study was conducted during autumn 2007 for one product at Ericsson. Ericsson is a world leading company in telecommunication, providing a wide range of products and solutions. Products are developed and sold as generic solutions offered to an open market, although customized versions of the products are also developed.

5.4.1 Success-Critical Stakeholder Groups

High-level R&D management supported the authors to identify internal success-critical stakeholder groups for this case study. Participants in the case study represent:

- **Strategic Product Management (SPM)** has the strategic product responsibility and decides the overall product development direction.

- **Project Management (PM)** is responsible for planning and executing projects aligned with the priorities of the strategic product management.

- **Tactical Product Management (TPM)** supports the strategic product management with expert knowledge of the systems and their architecture. It is also responsible for providing analysis of pre-project requirements in the form of feasibility, impact and technical dependencies.

- **Development and Testing (R&D)** are responsible for the implementation, verification and validation of requirements.

The high-level management further recommended that the results of SPM and PM be combined when determining the priorities given to the software product qualities, the first research question. These groups work closely together; with SPM prioritising the development activities that PM is responsible for planning.
A description of this case study was sent out to the managers of the identified success-critical stakeholder groups requesting volunteers from their teams to take part in the case study.

In total 44 potential participants were identified to take part in this case study, with 31 usable results being obtained. A breakdown of the participants can be seen in Table 5.1. Two of the participants identified felt they were not appropriate and identified other people in their team to replace themselves, two people declined to participate, one questionnaire result was lost in an Excel crash and nine people could not find time to complete the questionnaire.

<table>
<thead>
<tr>
<th>Group</th>
<th>Candidates</th>
<th>Replacements</th>
<th>Responses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strategic Product Management</td>
<td>15</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>Project Management</td>
<td>6</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>Tactical Product Management</td>
<td>9</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>Development and Testing</td>
<td>14</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>44</strong></td>
<td><strong>2</strong></td>
<td><strong>31</strong></td>
</tr>
</tbody>
</table>

The low response rate for Strategic Product Managers was anticipated, so extra participants for this role were selected to ensure a sufficient number of responses.

The questionnaire was conducted as a one-on-one structured interview, which each participant taking between 30 and 75 minutes. The interviews were conducted over a two-month period.

### 5.4.2 Software Product Qualities

The process of defining a model of software product qualities was a collaborative exercise involving the academic and industrial perspectives. The list of qualities was defined specifically for the product studied at Ericsson, thus maximising the relevance for this industrial partner and possibilities for using the results to support improvements within the company.

However, just as some qualities can be more important than others, there are other aspects of the development process that compete with the implementation of software product quality. To understand the importance of software product quality it must be placed in the context of all aspects of software product de-
5.4 Case Study

Development that are controlled. These are time, cost, quality and scope (Beck, 2000).

These four control variables have been complemented with ISO 9126, the international standard for software product quality, providing more detail on the components of quality and scope. The authors then wrote preliminary definitions for these terms.

A workshop was held within Ericsson to review and refine the terms defining software product quality. The aim was to ensure the final list of terms and definitions would be complete, meaningful and useful to Ericsson. The model was split into three categories — the ISO 9126 qualities relating to functionality, the ISO 9126 qualities relating to system properties and project management to cover time and cost. Moreover, security was moved from functionality to system properties. Two new qualities were identified and added to system properties; these are scalability and performance management/statistics. Finally, five of the quality terms were complemented with alternative names used in Ericsson.

The terms and definitions used in the case study presented in this chapter can be found in Appendix A.

5.4.3 Pilot Study

A questionnaire was developed using the methodology described in Section 5.3 and piloted. The participants in the pilot had trouble making comparisons between qualities related to features, system properties and aspects of project management. An example of such a comparison could include accuracy of features, resource behaviour of the system and development cost. In order to address this issue the authors modified the questionnaire to use a hierarchical cumulative voting (HCV) method as described by Berander and Jönsson (2006). This effectively splits the questionnaire up into four independent CV exercises; one parent list that includes the three category terms — features, system properties and project management — and one list for each of the categories, each containing the relevant qualities. A pilot of the new questionnaire found that the participants’ capacity to respond was much improved. In order to conduct the subsequent analysis each participant’s response needs to be changed from HCV to CV, converting the four cumulating voting lists into one that covers all of the qualities.

Remember that one of the four lists includes the categories features, system properties and project management, while the remaining lists each detail the qualities that make up one of the categories. This allows the number of points awarded to each quality to be multiplied with the category from which it came
152 Quality Alignment Inside an Organisation

(Berander and Jönsson, 2006). It is also necessary to multiply each of these results by the number of qualities from the same category as the resultant value. Finally the set of number for each quality can be scaled so that the sum is 1000.

For example, if 200 points are awarded to project management and 600 points are awarded to time, then \( \text{category} \times \text{quality} \times \text{number of qualities} = 200 \times 600 \times 2 = 240,000 \). The scaling of this result then depends on the other values, but if the other values were to sum to 4,800,000 then the number would be scaled to \( \frac{\text{result}}{\text{totalsum}} \times 1000 = 240,000 / 4,800,000 \times 1000 = 50 \).

It is necessary to multiply each value by the number of qualities in the same category to ensure that qualities with many categories are not underrepresented and that categories with few qualities are not overrepresented (Berander and Jönsson, 2006).

The individual responses can now be grouped and averaged, allowing ranks to be determined and the Spearman rank correlation can then be calculated.

The final version of the questionnaire is available online (Barney and Wohlin, 2008).

5.4.4 Software Product Quality Priorities

The first objective of this case study is to determine the degree to which the key stakeholders are aligned regarding how they see software product quality today. The results show that the groups are very aligned, with Spearman’s rank correlation values between 0.80 and 0.90 indicating each group ranked the qualities in a very similar order. The full results are presented in Table 5.2.

Table 5.2: Correlation matrix showing the degree to which the groups are aligned in how they perceive the priorities today

<table>
<thead>
<tr>
<th></th>
<th>SPM &amp; PM</th>
<th>TPM</th>
<th>R&amp;D</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPM &amp; PM</td>
<td>1.00</td>
<td>0.80</td>
<td>0.90</td>
</tr>
<tr>
<td>TPM</td>
<td>1.00</td>
<td></td>
<td>0.86</td>
</tr>
<tr>
<td>R&amp;D</td>
<td></td>
<td>1.00</td>
<td></td>
</tr>
</tbody>
</table>

Similarly the key stakeholder groups are aligned in how they ranked the software qualities should be today, in their perceived ideal situation. The results in Table 5.3 show correlation values between 0.65 and 0.74 between the groups.

The similarities between the perceived situation today and the perceived ideal situation were striking. Looking at all responses the correlation between
Table 5.3: Correlation matrix showing the degree to which the groups are aligned in how they perceive the priorities should be today (ideal)

<table>
<thead>
<tr>
<th></th>
<th>SPM &amp; PM</th>
<th>TPM</th>
<th>R&amp;D</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPM &amp; PM</td>
<td>1.00</td>
<td>0.74</td>
<td>0.71</td>
</tr>
<tr>
<td>TPM</td>
<td>1.00</td>
<td>0.65</td>
<td></td>
</tr>
<tr>
<td>R&amp;D</td>
<td></td>
<td>1.00</td>
<td></td>
</tr>
</tbody>
</table>

the two situations is 0.82. However, each group individually saw the need for more changes. The results in Table 5.4 show, for example, a correlation of 0.62 comparing what R&D perceived as the priorities today against what they thought the priorities should ideally be today.

Table 5.4: Correlation between perceived situation today and perceived ideal situation

<table>
<thead>
<tr>
<th>Groups</th>
<th>Correlation</th>
</tr>
</thead>
<tbody>
<tr>
<td>All groups</td>
<td>0.82</td>
</tr>
<tr>
<td>SPM &amp; PM</td>
<td>0.72</td>
</tr>
<tr>
<td>TPM</td>
<td>0.77</td>
</tr>
<tr>
<td>R&amp;D</td>
<td>0.62</td>
</tr>
</tbody>
</table>

While there was variation between participants of the same perspective, there was no individual that stood out as being consistently different in their results to the other members of their group.

Looking at the underlying data it is possible to further understand the differences and similarities between the groups. The remainder of this section highlights key aspects of similarity and difference.

The qualities studied have been grouped into three categories — features, project management and system properties. All of the groups today ranked these groupings in the same order, with features as the most important category, followed by project management and finally system properties. Interestingly all groups would like to see system properties overtake project management in their perception of the ideal situation. This helps explain the high correlation values attained.
154 Quality Alignment Inside an Organisation

However, looking at the individual qualities it is possible to explain why the collective results from all participants shows less need for change than the result of any of the groups individually. While confidentiality does not allow the ranked qualities to be published in this chapter, some groups are more affected by some of the qualities than other groups; so they perceive these qualities as more important, while the other groups perceive the same quality as less important. For example, Time Behaviour is ranked seventh today, with all groups placing it in the seventh or eighth position today. In the perceived ideal situation the overall rank is only increased one place to sixth, but the same criterion is ranked second most important by R&D, sixth most important by TPM and tenth most important by SPM & PM. This situation acts to reduce the correlation coefficients for the individual groups, but still keeps it high when examining all results together.

There was a high level of agreement in the ranks given to qualities relating to features and project management both today and in the ideal situation. The area of greatest contention between the groups concerns qualities relating to system properties. SPM differs the most when examining the results of the three groups. The results highlight which qualities the groups agree on as important — such as Scaleability, Time Behaviour, Robustness/Stability, Configurability/Product Customisability/Adaptability and Resource Behaviour — and which qualities for which there are differing priorities — Recoverability, Operability, Performance Management/Statistics, Upgradeability/Replaceability, Analysability, Testability, Containment/ISP/Fault Tolerance and Security.

5.4.5 Responsibility and Management

While the main aim of the third research question is to determine ambiguity in responsibility and management of the software qualities, it was also considered important to ask participants the degree to which each area was being managed. For example, it is possible that an important quality has a clear management line, but the managers are not sufficiently with respect to the quality. Respondents answered on a Likert scale from 1 (Not managed) to 5 (Clearly and explicitly managed).

In general, the more important a quality is today, the greater the degree to which it is managed. This can be seen with the Spearman rank correlation of 0.81 between the importance of a quality today and the degree to which it is being managed today. The results can also be seen in Figure 5.4, showing the criteria in order from most important today at the left to least important today at the right. The line of best fit shows that the less important qualities are less
managed. It should also be noted that the correlation between the perceived ideal situation and the degree to which the qualities are being managed today is 0.56.

Figure 5.4: Importance of qualities today and the degree to which they are managed

Ambiguity in who is ultimately responsible for or managing a quality can be presented in many different ways, as discussed in Section 5.3.3. The remainder of this section discusses these ambiguities with respect to the results collected.

For a group to be identified in one of these two leadership roles in this case study it must receive at least half of the non-blank responses. For two groups to be identified, each must receive more than a third of the non-blank responses; and for three groups to be identified each group must receive more than a quarter of the non-blank responses. This chapter highlights the qualities where one third or more of the responses are blank.

Blank responses: There were four qualities with a significant number of blank responses as to which group had a leadership role. These can be seen in Table 5.5, with the qualities showing a significant number of blank responses marked with a tick. For example we can see maturity has a significant number of blank responses for both types of leadership. It is interesting to note that with the exception of maturity, all of the ambiguity presented through blank responses concerned the group that was ultimately responsible for a quality. It should also be noted that each of these qualities comes from the category
of system properties. These qualities are also ranked quite low, thus being relatively less important.

Table 5.5: Qualities with a significant number of blank responses

<table>
<thead>
<tr>
<th>Quality (Rank: Today/Ideal)</th>
<th>Responsible</th>
<th>Managing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maturity (19/23)</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Understandability (21/19)</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Resource Behaviour (14/12)</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Analysability (24/22)</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

Insufficient votes: Similarly four qualities received insufficient votes to pass the thresholds. The qualities are listed in Table 5.6, which show the number of votes received by the most selected group where the threshold was not met. For example one group received 11 of the 23 non-blank responses for the management of Understandability. Again almost all ambiguity exists in who is ultimately responsible for the qualities. However, it should be noted that the result is only slightly below the threshold for one group to be identified in each case. The qualities with insufficient votes represent the categories system properties and project management.

Table 5.6: Qualities with insufficient votes

<table>
<thead>
<tr>
<th>Quality (Rank: Today/Ideal)</th>
<th>Responsible</th>
<th>Managing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maturity (19/23)</td>
<td>11/23</td>
<td></td>
</tr>
<tr>
<td>Understandability (21/19)</td>
<td>11/23</td>
<td></td>
</tr>
<tr>
<td>Resource Behaviour (14/12)</td>
<td>12/27</td>
<td></td>
</tr>
<tr>
<td>Analysability (24/22)</td>
<td>10/27</td>
<td>13/28</td>
</tr>
</tbody>
</table>

Pushing: Ambiguity is also presented in how a group identified as having a leadership role for a quality perceives their role with respect to this task. In the case study there is a strong tendency for the identified groups to pull responsibility to themselves, even when multiple groups were identified.

From Table 5.7 it is possible to see that the most common scenario in describing the ultimate responsibly and management of a software product quality is (1a) for a single group to be identified and for that group to pull this
responsibility to themselves. For example, the SPMs were identified as ultimately responsible for *Configurability/Product Customisability/Adaptability* and the SPMs identified themselves as ultimately responsible for this same quality. Further R&D was identified as managing this quality and also pulled this responsibility onto themselves.

Table 5.7: Pushing and pulling of ultimate responsibility and management

<table>
<thead>
<tr>
<th>Management</th>
<th>Responsible</th>
<th>Managing</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a. Pull, single group</td>
<td>10</td>
<td>16</td>
</tr>
<tr>
<td>1b. Pull, multiple groups</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>2a. Push/Pull, push to puller(s)</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>2b. Combination of 2 &amp; 3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3a. Push to unidentified group(s)</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3b. Push to pusher(s)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3c. Combination of 3a &amp; 3b</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

The second most common scenario found for both ultimate responsibility and management was (1b) where multiple groups were identified as taking on this role and all of the identified groups pulled this responsibility onto themselves. There were only two types of pushing found in the case study — types 2a and 3a.

Of type 2a, all but one of the cases fitted the same form: two groups were identified as ultimately responsible and/or managing a quality. One of the groups was always R&D, the other group was SPM or TPM, R&D always pulled responsibility to itself while the other group pushed responsibility to R&D. The exception to this case was for *Upgradeability/Replaceability* where SPM, TPM and R&D were all identified as ultimately responsible, with R&D and SPM pulling this responsibility onto themselves and TPM pushing the responsibility onto R&D.

Finally, one quality was identified as having (3a) one group as ultimately responsible for that quality, but the identified group perceived another group as being ultimately responsible. The management of this quality was being reviewed and changed while the case study was being conducted. Thus, it should be noted that the application of a method, such as the one presented in Section 5.3, allows for identification of this type of situation and hence also appropriate actions.
5.5 Discussion

The methodology proposed in Section 5.3 has been applied to identify the level of alignment between internal success-critical stakeholders with the modification made after the pilot study described in Section 5.4.3. While the results from the case presented in this chapter are not generalisable, it highlights what situations can be detected by the method and acts as a reference point for future applications of the method.

While in the case study some changes to the priorities given to the software product qualities would be perceived as beneficial by each of the internal success-critical stakeholder groups, the extent of the changes required is reduced when considering all perspectives together. This can be seen most clearly with a number of qualities where the groups agree on their importance today, but some of the groups think some should be more important while other groups think they should be less important and they thus end up in almost the same place. This result shows Theory-W in action, with the organisation having to balance conflicting stakeholder perspectives in order to achieve the optimal balance.

While the current processes seem to have done a reasonable job to balance the various concerns of software product quality, this is not explicitly visible to all of the stakeholders who felt that their needs were not being adequately addressed. One of the ongoing aims within Ericsson is to use these results to foster a greater understanding and dialogue between the internal success-critical stakeholders in terms of each other’s needs.

Respondents in the case study had greater difficulty in identifying and agreeing upon who was ultimately responsible for a quality than who was managing a quality at a more hands-on level. This does not indicate a problem, as people generally know to whom they need to turn regarding an issue of software product quality. What they were less aware of was to whom the manager could turn regarding the same issue.

Discussions of the results within Ericsson confirmed that some multiple groups were responsible for some software product qualities. This was identified as a potential risk, with no one feeling the need to be an advocate for some qualities. The responsibility of these qualities has since been more clearly defined. The same was not felt true of the day-to-day management of the qualities as different parts of the organisation could turn to the area most appropriate to them provided there is a clearly communicated and common strategy.

In terms of the three categories of software product qualities — features, project management and system properties — the results highlighted that most ambiguity exists around system properties. This is primarily the result of two
5.6 Conclusion

factors; system properties are the least important of the categories today, yet account for 18 of the 24 qualities studied. However, the internal success-critical stakeholder groups all agree that these qualities should be more important when going forward. These results supported discussions to clear up the management lines and helped present these messages to relevant parties. Most of the ambiguities raised in the ultimate responsibility of project management were because people were not sure whether to identify PM as ultimately responsible for ensuring that the product was delivered on time and cost, or SPM as the group that approves a project.

While it was anticipated that a blank response meant the respondent was unaware who was responsible, it became apparent that this response was also used to indicate that a respondent perceived no one was responsible. The interpretation of this result should be made with caution and future work should allow responses in the questionnaire to differentiate between the two cases. The pushing and pulling of responsibility and management of software product qualities also provided some interesting insights. There is a strong culture of groups taking on a management role with respect to qualities. This is a positive attribute with employees taking care of the product they produce and doing what they can to ensure the best outcome.

While some pushing of responsibility and management was observed, in almost all cases the groups identified agreed between themselves who was responsible — one group would pull responsibility to themselves, while the other groups pushed responsibility to this group.

It is also interesting to note the types of pulling and pushing not observed in the results of this case study. There was only one case of (3a) pushing without a group pulling and there were no cases of (3b, 3c) multiple groups pushing. As (3) pushing poses the greatest risks to the software development organisation this is a positive result. It means, for example, that there are no cases where group A believes group B to be responsible while group B believes group A to be responsible. Two qualities were identified by a number of participants of this case study as missing from the list of qualities and should be considered for future work. These were the packaging of requirements into releases and client contracts — as both impact upon the value that is delivered to customers.

5.6 Conclusion

This chapter presents a methodology and results of a case study for examining the alignment between the internal success-critical stakeholder groups in
software product quality. The results obtained by the method were interesting, valuable and very positive from the perspective of the industrial partner, Ericsson, with:

- The groups found to be aligned in perceiving the priorities placed on different software product quality today;
- Overall the participants in the case study perceive few changes necessary to improve the current situation;
- The most important qualities are the most clearly and explicitly managed; and
- The majority of qualities have clear management.

The case study results highlight that different stakeholder groups have different priorities, and companies must be able to balance these differing opinions in order to achieve an optimal outcome. Key to achieving this outcome appears to be open and transparent dialogue and cross group communication and understanding. The results also provide an understanding of the context of software product quality for future work within the case setting.

The results of the case study have helped the company make their employees more aware of the priorities other groups face in the development of their software product, and the need to gain a balance to achieve an optimal outcome. The results have also been used to support changes to the management of software qualities to help ensure better outcomes.

The case study presented in this chapter may not be representative of the software development industry, only involving one product from one company. Still, it provides some insights into how qualities are handled in an industrial context. Furthermore, the method can be applied in other situations to support the alignment of success-critical stakeholders in issues of software product quality and identify potential management issues. In turn, these additional results can help determine which of the results, if any, can be generalised.

This research will be used in three ways:

- This work is the first in a series of studies examining, with the intention to help improve, the alignment of company strategy, product strategy, product management and development efforts.
- This work is also the first in another series that is looking at different investment options and trade-offs in software development — like features,
quality and staff training. Going forward the aim of this work is to support organisations improve the investment choices they make.

- The authors are also hoping to replicate parts of this study at different organisations to help achieve greater alignment in issues of software product quality and draw more general conclusions in this topic area.
Chapter 6

Quality Alignment with Subcontracted Developers

This chapter is published as:


Abstract

**Background:** Issues with software product quality are commonly reported when organisations engage in outsourcing relationships. To address this issue, value-based software engineering literature emphasises the need for all success-critical stakeholder groups to work towards a mutually agreed goal.

**Aim:** This chapter presents a case study that aims to compare and contrast the priority two groups place on software product quality—stakeholders internal to the development organisation, and stakeholders from outsourcing relationships.

**Method:** A model of software product quality was developed and used for this study based on ISO 9126 standard. Questionnaires were collected from 38 representatives of the two key stakeholder groups, in which each person rates the relative importance of aspects of software product quality using the hierarchical
cumulative voting (HCV) technique. The results of these two groups were then analysed and compared.

**Results:** The results show the stakeholders priorities to be a merging of the priorities from both the software development organisation, and the firm providing the outsourced services. Further, stakeholders from outsourced relationships had greater difficulty define an ideal future balance of software product qualities.

**Conclusions:** One of the keys to success when outsourcing is to ensure both the internal and external groups understand the needs of each other—and ensure they can work towards a sufficiently compatible goal. It may be necessary to change the way work is outsourced to align the goals of both firms to be compatible.

### 6.1 Introduction

Many organisations are becoming more global by either establishing themselves in different countries or outsourcing to sub-contractors, for example in China (Barney et al., 2009b). The reasons for this are different, but include cost reduction, proximity to market and making use of the competence around the world (Smite et al., 2010). This puts new requirements on software development, including different aspects on coordination, communication and control (˚Agerfalk et al., 2005). Furthermore, outsourcing may or may not include geographical, temporal and cultural challenges depending on the locations of the parties involved. While there are many benefits from outsourcing, this change has not been without problems.

One of the main challenges faced when outsourcing is ensuring a sufficient level of quality is developed in the software produced by the outsourced developers (Morgan, 2004). Thus, one part is to be able to communicate the priorities on different qualities between parties involved for example from a customer to a supplier. The qualities include both product qualities such as functionality, reliability, security and maintainability as well as more project-oriented attribute such as cost and delivery date. It was hypothesised that minimizing the challenges when it comes to geographical, temporal and cultural differences would be a more favourable situation than having customer and sub-contractor far apart, i.e. it ought to be easier to communicate how to balance and prioritise different qualities.

The benefits of having of having a common understanding is highlighted in research by Phongpaibul and Boehm (2005). They find that a common
understanding of software product quality is the most effective way for multiple
groups to be successful in achieving a common goal. Initially the objective
was to study the alignment between internal groups within a company. Thus,
a method for studying the balance and priority of different company internal
groups such as development, testing and different management roles was created
(Chapter 5). It was created to determine the degree to which internal success-
critical stakeholders are aligned on priorities given to various aspects of software
product quality as well as other aspects such as time and cost. However, it
was realized that the method could be used also to understand the alignment
between customer and sub-contractors.

Thus, this chapter extends the previous work to assess the alignment between
internal and outsourced development effort with regard to the priority placed
on software related qualities. Subcontracted developers from two separate firms
who are both working on the same product as the internal developers and co-
located with the internal developers were interviewed as part of this research.
This chapter extends the methodology and case study previously presented to
both:

- Determine the level of alignment between the internal and outsourced
  stakeholders, and
- Explore the reasons for the differences in software product quality priori-
ties between these two groups.

The remainder of this chapter is structured as follows. Key literature intro-
ducing the topic is provided in Section 6.2. Section 6.3 presents the research
question and methodology used in this chapter. Details of the case study are
provided in Section 6.4. The results are in Section 6.5, with a discussion of the
results in Section 6.6. Conclusions are presented in Section 6.7.

6.2 Background

This section introduces key concepts and work related to this chapter. First soft-
ware product quality is defined in Section 6.2.1, with models that describe the
aspects that make up software product quality presented in Section 6.2.2. The
models of software product quality are important to this work as the alignment
of the stakeholders groups is checked against a such a model. Using the broader
definitions of software product quality presented in the models, Section 6.2.3 ex-
amines some of the issues of software product quality when outsourcing develop-
ment effort. Finally, recognising that there are problems of alignment between
6.2.1 Software Product Quality

The most commonly used definitions used in software engineering domain define the users’ perspective—‘fit for purpose’—and the manufacturing perspective—‘conformance to specification’ (Hoyer and Hoyer, 2001; Kitchenham and Pfleeger, 1996). However, there is an increasing body of literature that recognises the value of the many perspectives involved in the development of a software product. This is presented most clearly in the value-based approach, which requires the success-critical stakeholders to come to a mutual consensus on the best way to move forward with the development of the product (Boehm and Ross, 1989; Boehm and Jain, 2006).

The value-based approach does not define quality as something absolute. It must be defined for a specific context or instance (Kitchenham and Pfleeger, 1996). This means the optimal level of quality may not be perfect (Yourdon, 1995), but how much less than perfect can only be decided in a given business context (Kitchenham and Pfleeger, 1996). One of the risks with outsourcing is that the parties involved may have conflicting notions of what constitutes an acceptable level of quality for the product on which they are working together. Thus it is the value-based definition of software product quality that is used in this chapter.

6.2.2 Models of Software Product Quality

There are a number of models of software product quality that help define and manage quality. Common models include McCall’s quality model, Boehm’s quality model, Dromey’s quality model and ISO 9126. With the exception of Dromey’s model, each of these models defines software product quality in a multi-level hierarchy.

ISO9126 (2001) has three layers with six top-level characteristics, 20 sub-characteristics and indicators. In addition to the internal and external quality characteristics presented in Boehm and McCall’s quality models, ISO 9126 also presents aspects of quality related to functionality (Milicic, 2005). However, the major criticism of ISO 9126 is that it does not clearly state how the qualities it defines can be measured (Kitchenham and Pfleeger, 1996).
6.2 Background

6.2.3 Software Product Quality and Outsourcing

Common models and definitions of software product quality improve the likelihood of success when a company outsources software development (Phongpaibul and Boehm, 2005). These things have been found more effective than definitions of quality requiring compliance to processes or specifications.

This is an important finding, as many organisations that have chosen outsourcing to reduce costs have found that decreases in quality were a common side-effect (Morgan, 2004). For example, Capiluppi et al. (2006) found that the complexity of code produced by outsourced developers is higher than developers internal to the organisation developing the software.

One of the reasons for lower software product quality in cases of outsourcing is a lack of trust between the two organisations (Moe and Šmite, 2007). (Boehm et al., 2008) are calling for more research into the affect of cultural issues on software product quality given the rise of outsourcing. However, the work of Phongpaibul and Boehm (2005) suggests the lack of a common understanding of software product quality alone is causing problems.

Ultimately if outsourced developers skimp on quality, then it is a loose-loose situation (Boehm and Jain, 2006). The development companies do not get what they need, and the outsourced developers fail to meet the expectations of their employers.

This research to date emphasises the need for all developers working on a software product to have a common set of priorities with respect to software product quality. Hence there is a need to check the degree to which internal and external parties are aligned with the priorities they each hold.

6.2.4 Merging Perspectives on Software Product Quality

There are a number of methods used to help reconcile conflicting stakeholders priorities. These methods include expert judgment, the NFR Framework, Quality Functional Deployment and Theory-W.

Central to resolving conflict in the value-based approach to software engineering is Theory-W (Boehm and Ross, 1989; Boehm and Jain, 2006). The aim of Theory W is to create a win-win scenario for all stakeholders by:

1. Identifying the success-critical stakeholders;
2. Eliciting the requirements of these groups;
3. Creating a win-win situation by negotiating with these groups; and
4. Realising the negotiated solution through a controlled process.

The research presented in this chapter seeks to help reconcile conflicts in software product quality priorities between internal and outsourced stakeholders by applying Theory-W.

6.3 Methodology

The methodology presented in this chapter seeks to understand the priorities of internal and external stakeholders on aspects of software product quality. This section presents the research objectives, questions, and method used to conduct the research presented in this chapter.

6.3.1 Research Objectives and Questions

There are many aspects of software product quality, and many groups that influence software product quality. While the authors’ research has focused on co-located internal success-critical stakeholders to date (Chapter 5), companies are increasingly outsourcing software development to supplement and complement internal development effort. The change to outsourced development has been found to impact the quality of a software product negatively, while difficulties in communicating quality requirements are well documented (Morgan, 2004).

The objective of this case study is to determine the degree to which outsourced developers are aligned with internal success-critical stakeholders in how they perceive priorities on software product quality.

It is valuable to understand both if there is a common understanding of both what the priorities are today, and what the priorities should ideally be today. Thus the research question has been broken into two sub-questions—to understand the alignment of the stakeholder groups with respect to software product quality:

- RQ1: As they perceive the priorities are today.
- RQ2: As they perceive the priorities should ideally be today.

6.3.2 Method

In the study presented in this chapter the authors follow a methodology they previously developed and tested to determine the alignment of internal success-
critical stakeholder groups with respect to software product quality (Chapter 5). Given the success of this method in this domain, it has been reused.

The method draws heavily on early phases Theory-W (Boehm and Ross, 1989; Boehm and Jain, 2006). The aim of Theory-W is to create a win-win scenario between the success-critical stakeholders, the people upon whom the success of a project is dependent. Theory-W states that in order to achieve such a scenario one must identify the success-critical stakeholders, identify how they want to win, negotiate win-win plans, and control the process to achieve the win-win situation.

They key steps in method used to answer the research question have been developed from (Chapter 5) and are detailed in the following seven subsections.

Select a company and product

As each product can have different quality requirements, it is essential to ensure the study is sufficiently focused. The case study presented in this chapter extends the work previously done by (Chapter 5), focusing on the same product.

Identify success-critical stakeholder groups

Success-critical stakeholder groups are groups upon whom the success of the product depends—for example, product managers and developers. This ensures the most important perspectives are covered, while less important perspectives cannot dominate.

Develop quality model

The literature on software product quality recognizes that quality depends both on the perspective of the observer and the actual software product in question. As such, using any model as it appears in the literature risks not adequately defining quality in the context being studied. To use one of the quality models briefly introduced in Section 6.2.2 is a good starting point, but company specific needs have to be taken into account, as illustrated in the authors’ previous research (Chapter 5).

Develop a questionnaire

This method proposes the use of the hierarchical cumulative voting technique (HCV) (Berander and Jönsson, 2006) to elicit the relative importance of each aspect of software product quality. This method allows respondents to state the
relative importance of the aspects being studied. Past research has shown that respondents have trouble comparing some aspects of software product quality at a low level directly (Chapter 5); HCV provides a method for breaking the problem into a series of smaller direct comparison exercise, with a method to join these results back together.

Conduct the questionnaire

The required information should be collected by getting representatives of each of the identified success-critical stakeholder groups to complete the questionnaire. Doing this in a one-on-one structured interview allows richer information to be collected. This also helps ensure participants have a common understandings of the questionnaire with the interviewer able to assist with questions or problems faced by the respondents.

Analyse the results

Using the method developed by Berander and Jönsson (2006) it is possible to turn the results of the HCV exercise into cumulative voting (CV) results. From here it is possible to group the results by success-critical stakeholder group, and average the points awarded to each aspect of software product quality. These results can be used to calculate Spearman rank correlation coefficients, determining the level of alignment between the success-critical stakeholder groups.

Workshop the results

Presenting the results to participants from each of the success-critical stakeholder groups and asking for their response allows for a deeper understanding of the results. The following questions should be answered as best as possible:

- Do these results look reasonable?
- Do the differences make sense?
- Why do these differences exist?

6.4 Case Study

This chapter presents a case study for one of the major products at Ericsson. Ericsson is a world leading company in telecommunications, providing a wide
range of products and services. These are developed and sold as generic solutions to an open market, although customized versions of the products are also developed for key customers.

An exploratory case study has been employed for this research, as it seeks to gain insights and understanding of the current situation within Ericsson (Rune-son and Höst, 2009).

The study was conducted in two phases, with the second phase being the focus of this chapter. Each phase is described in more detail in the following subsections.

### 6.4.1 Phase 1: Internal Stakeholders

The first phase of the case study was conducted in autumn 2007 (Chapter 5). It was focused on the alignment of internal success-critical stakeholders with respect to software product quality. The success-critical stakeholder identified in this study were:

- **Strategic Product Managers (SPM)** have the strategic product responsibility and decides the overall product development direction.
- **Project Managers (PM)** are responsible for planning and executing projects aligned with the priorities of the strategic product management.
- **Tactical Product Managers (TPM)** supports the strategic product managers with expert knowledge of the systems and their architecture. It is also responsible for providing analysis of pre-project requirements in the form of feasibility, impact and technical dependencies.
- **Developers and Testers (R&TD)** are responsible for the implementation, verification and validation of requirements.

A model of software product quality was created specifically for this study. The model was developed over a series of three two-hour meetings between one of the authors and representatives of each internal success-critical stakeholder group. The model was based on both the ISO9126 (2001) Standard and the control variables in software development identified by Beck (2000)—functionality, quality, cost and time. A number of changes were made to the ISO 9126 model to make it more meaningful and useful in the organisational setting being studied. Beck’s control variables were used to recognise the business context in which software development occurs.
A questionnaire was then developed using the model, with HCV used to determine the relative priority of each aspect of software product quality—both as perceived today and in the ideal situation. The model of software product quality and questionnaire used in the study is available online (Barney and Wohlin, 2008).

In total 31 complete responses were obtained from this first phase, with between four and twelve responses per group. The results found the groups to be aligned in the priority they placed on the various aspects of software product quality both today and their perceived ideal situation (Chapter 5). Further the groups perceived only small changes needed to be made to the priorities in place today.

### 6.4.2 Phase 2: External Stakeholders

During the first phase all of the software development was done internally by Ericsson employees for the product studied. However, in the year following the first phase some of the development effort was outsourced to a number of consultancy firms, as shown in Figure 6.1, with these subcontracted providers placing self-contained teams onsite in the Ericsson office.

![Figure 6.1: Development shared between internal and external stakeholders](image)

Prior to this change, new developers were brought into existing teams. This meant new developers were surrounded by the support, knowledge and experience of developers with a deep understanding the product. However, the new subcontracted teams often comprised of people with no previous experience of the product. While they had experts within the organisation they could call upon for support, these people were not part of their development teams and worked with many groups.

Understanding the difficulty in communicating software product quality priorities, senior management at Ericsson wanted to confirm if the organisation was
able to successfully communicate their priorities to the subcontracted teams. These subcontracted software development providers clearly had the ability to impact the software product quality, but given the difficulty in expressing quality requirements, management within Ericsson sought to determine if they were successful in achieving this aim.

To determine if the subcontractors were aligned with the internal success-critical stakeholder groups a second phase was planned to follow the first. This second phase was focused on the same product as in Phase 1. The list of success-critical stakeholder groups was expanded to include two of the subcontracted providers—referred to as Provider 1 and Provider 2 in this chapter. A decision was made to use the model of software product quality and questionnaire from Phase 1, as both were found to work effectively and this choice allows comparisons between the two studies.

A decision was made to reuse the results of the internal success-critical stakeholders from Phase 1. A related study showed the priorities of these internal groups to be unchanged in autumn 2008 (Chapter 9).

Interviews for the second phase were conducted during spring 2009—six months after the subcontracted developers started working on the Ericsson product studied. In total eight people were selected to participate in the study—four from each of the selected subcontracted firms. One person was unable to participate, with seven results being used in the analysis presented in this chapter. The seven participants had all worked on the product at Ericsson for six months. One participant from each firm was a recent graduate, with less than 12 months industrial experience. The remaining five developers had between three and six years of experience in the software development industry.

Figure 6.2 shows the scope of the two studies in relation to the groups examined.

6.5 Results

The objective of this study is to determine the degree to which internal stakeholder groups and subcontracted software developers are aligned in the priorities they place on aspects of software product quality. This results section brings together previous research into the alignment of internal stakeholder groups (Chapter 5), with new data collected on the priorities of external stakeholder groups, as shown in Figure 6.2. Bringing this information together allows comparisons between both sets of groups.

The research was broken into two sub-questions, presented in Section 6.3.1.
RQ1 sought to determine the degree to which the internal and subcontracted success-critical stakeholder groups are aligned in how they perceive the priorities on aspects of software product quality today. The results to this sub-question are presented in Table 6.1. The table contains Spearman rank correlation coefficients for each possible pairs of groups. In this table the number one indicates complete agreement between two groups in their priorities on software product quality, while minus-one indicates two groups think the priorities of the other group are completely back-to-front. The results of the two subcontracted software development service providers are shown individually, and are also shown collectively in the \textit{subcontracted} column.

The internal success-critical stakeholders are in strong agreement as to what the current priorities are on software product quality (Chapter 5), with correlation coefficients between 0.80 and 0.90.

The subcontracted software providers also identified a very similar set of priorities on software product quality to the internal groups in the situation as it they perceive it today, although not as closely aligned as the internal stakeholder groups. Comparing the priorities of Provider 1 against the internal stakeholder groups obtained correlation coefficients of 0.64 to 0.68, while Provider 2 obtained
Table 6.1: Correlation matrix showing the degree to which the groups are aligned in how they perceive the priorities today.

<table>
<thead>
<tr>
<th></th>
<th>Provider 1</th>
<th>Provider 2</th>
<th>Subcontracted</th>
<th>TPM</th>
<th>R&amp;D</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPM &amp; PM</td>
<td>1.00</td>
<td>0.90</td>
<td>0.80</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>TPM</td>
<td>0.79</td>
<td>0.73</td>
<td>0.86</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>R&amp;D</td>
<td>0.73</td>
<td>0.79</td>
<td>0.86</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Subcontracted</td>
<td>0.66</td>
<td>0.68</td>
<td>0.73</td>
<td>1.00</td>
<td>1.00</td>
</tr>
</tbody>
</table>
results between 0.65 and 0.79.

There were a number of noteworthy differences between the priorities of the internal and subcontracted stakeholder groups studied. The subcontracted groups ranked the qualities *testability, performance management/statistics* and *cost* higher than the internal groups. The subcontracted groups ranked *resource behaviour, time* and *upgradability/replaceability* lower than the internal groups. Possible reasons for these differences are discussed in Section 6.6 of this chapter.

The second research question, RQ2, sought to understand how each success-critical stakeholder groups thought the aspects of software product quality should be prioritised today, and to what degree these groups were aligned. The results to this sub-question are presented in Table 6.2. These results have the same format as Table 6.1, with Spearman rank correlation coefficients showing the degree of alignment between the different success-critical stakeholder groups.

The internal groups were relatively well aligned (Chapter 5), with Spearman rank correlation coefficients between 0.65 and 0.74. The differences in the priorities of these groups can be explained—each internal group prioritised a subset of qualities higher than the other internal groups based on what most directly impacted them. For example, the R&D group prioritised maintainability higher than the other groups, as they are the only group that directly with the code.

The results showed the subcontracted software providers to be not as clearly aligned with the internal groups, as the internal groups are with each other. However, examining the priorities of the subcontracted software providers in more detail showed the member of this group to be in disagreement with each other as to what should be important.

Despite the inconsistency between the perceived ideal priorities of the outsourced software providers, there were some priorities for which this group shared a consistent vision. The outsourced software providers would like to see:

- A much greater focus on *testability*;
- A greater focus on *changeability*; and
- Less focus on *installability*.

Possible reasons for an emphasis on these aspects of software product quality are discussed in Section 6.6 of this chapter.
Table 6.2: Correlation matrix showing the degree to which the groups are aligned in how they perceive the priorities should be today (ideal)

<table>
<thead>
<tr>
<th></th>
<th>SPM &amp; PM</th>
<th>TPM</th>
<th>R&amp;D</th>
<th>Subcontracted</th>
<th>Provider 1</th>
<th>Provider 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPM &amp; PM</td>
<td>1.00</td>
<td>0.74</td>
<td>0.71</td>
<td>0.03</td>
<td>0.62</td>
<td>0.36</td>
</tr>
<tr>
<td>TPM</td>
<td>1.00</td>
<td>0.65</td>
<td></td>
<td>0.41</td>
<td>0.62</td>
<td>0.11</td>
</tr>
<tr>
<td>R&amp;D</td>
<td>1.00</td>
<td></td>
<td></td>
<td>0.37</td>
<td>0.62</td>
<td>0.06</td>
</tr>
<tr>
<td>Subcontracted</td>
<td></td>
<td></td>
<td></td>
<td>1.00</td>
<td>0.62</td>
<td>0.62</td>
</tr>
<tr>
<td>Provider 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.00</td>
<td>0.11</td>
</tr>
<tr>
<td>Provider 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.00</td>
</tr>
</tbody>
</table>
6.6 Discussion

This study sought to understand if Ericsson was able to successfully communicate its priorities with respect to software product quality to subcontracted software development providers. The results were positive for Ericsson, with the subcontracted software development providers being reasonably well aligned with the internal success-critical stakeholder groups in their priority on aspects of software product quality. However, a number of interesting points were found in the results that will be examined in this section.

6.6.1 Merging of Priorities

The priorities of the subcontracted developers are a combination of the priorities of both Ericsson and the firms providing the subcontracted services, and are shaped by the way they are involved in the software development process. The results clearly indicate the subcontracted developers can identify the priorities placed on software product quality by the internal stakeholders today with correlation coefficients between 0.64 and 0.79 when comparing the priorities of the two groups. However, there were a number of consistent differences in priorities, which show the subcontracted developers adopting the priorities of the firms providing the subcontracted services.

The priorities of the firms providing the subcontracted developers are most visible where there is a consistent difference between the priorities of the internal stakeholders and the subcontracted stakeholders. The key differences in the case study presented in this chapter are listed here with explanations elicited from follow-up workshops that were used to review the results.

Testability and performance management/statistics are two aspects of quality given a much higher priority by the subcontractors than the internal stakeholders. In order to allow Ericsson to easily see that the subcontracted teams are doing good work, these teams must make their work easy to assess. These aspects are more important to the subcontracted development teams as they have a greater need to prove themselves to secure their position than internal development teams.

Another interesting differences relates to the project management control variables of time and cost. These two variables are very closely related as the time spent on software engineering activities accounts for much of the costs of developing software. However, where internal stakeholders prioritise time, the subcontracted developers prioritise cost. The reasons for this are clear, with internal stakeholder providing and working to time-based estimates, and
subcontracted developers providing and working to cost-based estimates. The subcontracted developers know the allocated budget to different activities.

Further, two aspects of software product quality were given a lower priority by the subcontracted developers than internal stakeholders—resource behaviour and upgradability/replaceability. The subcontracted developers are given the responsibility for delivering functionality, but are not involved with customer delivery. As such they do not directly face some of the issues dealt with by the internal stakeholder groups.

In summary, it seems like the subcontracted developers have a combination of the internal view of their customer and their own priorities to show their value.

### 6.6.2 The Ideal Situation

In addition to asking participants to identify the priorities today, the survey also asked participants to identify what they felt the priorities should be today. In this section of the questionnaire the internal participant groups indicated they wanted largely the same priorities as they perceived in the situation today, but with a greater emphasis on the aspects of quality that most affected their group (Chapter 5). For example, the developers wanted to see the aspects of maintainability given a higher priority than either their perception of the situation today or the other groups perception of the ideal situation. This shows that the participants were able to differentiate between what they perceived the priorities to be today, and what they wanted the priorities to be today.

The results of Phase 2 show that the subcontracted developers would also like some change to the current situation. Unlike the internal stakeholders, however, the subcontracted developers do not share a common vision of what priorities would be most beneficial for the product studied. Possible reasons for this result were elicited during the workshops to understand the results:

- The subcontracted developers do not have a complete vision of the product, target market or customers. They are excluded from certain aspects of product development and are not included in strategic planning. This means that they cannot approach product development from a holistic perspective. However, groups internal to the software organisation have more experience dealing with the whole product.

- The current priorities have been institutionalised in the internal stakeholders ways over an extended period of time, so the internal groups are more cautious of change.
However, the subcontracted developers were consistent on three aspects of software product quality. They would like to see a greater emphasis on *testability* and *changeability*, but less emphasis on *installability*. *Testability* is perceived undervalued by the subcontractors as they want to have accountability and prove themselves professionally to Ericsson. The need for a greater emphasis on *changeability* is justified as this would make their development tasks easier to complete. However, the subcontracted developers are not involved in the set-up of specific systems, so would not face the consequences of a reduced focus on *installability*.

These consistent differences emphasize the alternate agendas of the internal and subcontracted groups. While these differences have not caused problems in the case studied, it is possible for them to conflict and cause problems.

### 6.6.3 Validity Threats

The authors were limited in the number of responses they could collect representing the views of subcontractors. Thus the results are not necessarily representative of this group. However, all but one of the selected respondents participated in the study and the results for the situation today was described consistently between the participants.

Subcontracted developers were not included in the development of the quality model used to describe this product as this role did not exist at the time the model was developed. This makes it possible that some view on quality has been missed that relates directly to the subcontracted role. However, this study is primarily concerned with the degree to which the subcontracted stakeholders understand the priorities of software product quality of the internal stakeholders and their understanding of software product quality. Further, all results were collected in a one-on-one interview situation, with the participants given the opportunity to provide general comments and feedback.

As an exploratory case study was conducted the specific results of this study may have limited generalizability (Runeson and Höst, 2009). The authors believe the priorities placed on aspects of software product quality by different groups involved in the development process will not only vary between different software products, but also throughout the life of a software product. However, the method can be applied to other cases, with more results providing the possibility for more general findings.
6.7 Conclusion

This chapter aimed to understand the priorities of subcontracted software developers with respect to software product quality, and if they are aligned with the priorities of internal stakeholder groups working on the same product. The chapter presents a case study of one of the major products at Ericsson. While positive, the results show a need to take great care in managing the quality output of outsourced development to achieve a desired outcome.

The results show that Ericsson has successfully integrated the subcontracted developers into the company as they have a strong understanding of the current priorities with respect to software product quality. However, it does not appear that the subcontracted developers always understand why this balance is desired. The subcontracted developers do see opportunities for improvement, but their lack of vision of the entire product, process and market means their stated preferences do not take into consideration all relevant issues.

Further, the priorities of the outsourced developers represented a merging of the priorities of the company developing the product and the firm providing the subcontracted development services. Areas affecting how the subcontracted developers would be assessed were prioritised higher by this group than the internal groups. Some aspects of quality were down-prioritised by the subcontracted developer as the developers were not involved in activities that benefited from higher quality in these areas.

The organisational setting presented in this chapter, co-located task-based outsourcing, allows for a closer working relationship between the internal and external groups than other forms of outsourcing, near-shoring and off-shoring—primarily as all developers are working at the one site. In other types of outsourcing, near-shoring and off-shoring relationships it is likely to be more difficult for a company to impart their software product quality priorities. This means the partner organisation is likely to develop to their own priorities, which may not necessarily be aligned with those of the development company.

An interesting area of future research is to look at other relationships—such as outsourcing, near-shoring and off-shoring. A greater understanding of the priorities stakeholders in these groups have in developing software will lead to more effective relationships and software development.

The research by Phongpaibul and Boehm (2005) found that a common understanding of software product quality priorities between groups was a more effective way of achieving the desired level of quality than through process or product descriptions. Thus it is important for an organisation to ensure these internal and outsourced workers share a common value system with respect to
software product quality. The method presented in this chapter has been demonstrated to be able to elicit and compare the priorities of numerous groups. The authors’ experience has shown this is a necessary step in getting the groups to understand each other better, and work together effectively.
Chapter 7

Quality Alignment with Offshored Development

This chapter is submitted as:


Abstract

**Background:** Software quality issues are commonly reported when offshoring software development. Value-based software engineering addresses this by ensuring key stakeholders have a common understanding of quality.

**Aim:** This work seeks to understand the levels of alignment between key stakeholders on aspects of software quality for two products developed as part of an offshore insourcing arrangement. The study further aims to explain the levels of alignment identified.

**Method:** Representatives of key stakeholder groups for both products ranked aspects of software quality. The results were discussed with the groups to gain a deeper understanding.
Results: Low levels of alignment were found between the groups studied. This is associated with insufficiently defined quality requirements, a culture that does not question management and conflicting temporal reflections on the product’s quality.

Conclusions: The work emphasizes the need for greater support to align success-critical stakeholder groups in their understanding of quality when offshoring software development.

7.1 Introduction

The rapid rise of global software development (GSD) (Dibbern et al., 2008) has brought with it new benefits and challenges. The main drivers for this practice are cost reduction, proximity to markets and making use of different competencies (Šmite et al., 2010), but there are also many challenges. The most commonly cited challenges in GSD contexts concern communication (Mockus and Herbsleb, 2001) and coordination (Herbsleb, 2007), both essential elements in creating alignment between stakeholders (Carmel and Abbott, 2006).

There is a wide body of evidence showing that organisations that can create alignment through “convergent intentions, shared understanding and coordinated procedures” will outperform organisations that cannot create this alignment (Chan, 2002; Chan and Reich, 2007). Alignment of stakeholders allows them to collaborate more effectively and produce results that support the long-term business strategies, while highly misaligned teams can cause conflict and eventually lead to the failure of a project.

This chapter aims to expand the author’s previous work into the alignment of success-critical stakeholder groups on software quality (Chapter 5, Chapter 6) by looking beyond the onshore groups previously studied to include offshore software development. A common understanding of software quality has been found to be the most effective way for groups to achieve a common goal in a GSD setting (Phongpaibul and Boehm, 2005). Thus this study explores the offshore insourcing setting within GSD with the aim to understand (1) the levels of alignment between sites in terms of software quality, and (2) the reasons for the level of alignment.

The remainder of this chapter is structured as follows. Key literature introducing the topic is presented in Section 7.2. The research questions and methodology are presented in Section 7.3. Information about the case study is presented in Section 7.4. Results are presented in Section 7.5 and discussed in Section 7.6. Finally conclusions are made in Section 7.7.
7.2 Background

This section introduced key concepts and related work.

7.2.1 Software Quality

There are many definitions of software quality (Kitchenham and Pfleeger, 1996), with the most common in software engineering being ‘conformance to specification’ and ‘fit for purpose.’ The growing body of value-based software engineering (VBSE) literature recognizes that perceptions of software quality are individual and shaped by experiences (Johansson et al., 2001). VBSE suggests that the most successful way to move forward with software development is for the success-critical stakeholder groups to reach mutual consensus. The success-critical stakeholder groups are the groups upon whom the success of the product depends (Boehm and Jain, 2006).

Software quality does not need to be perfect (Yourdon, 1995). The tough question to answer is, ‘how much less than perfect is sufficient?’ There is no single answer to this question, as any answer must consider the context in which it is being asked. The Quper model helps answer this question by defining a relationship between the level of quality, the benefits and the costs (Regnell et al., 2008). The model defines a series of quality/benefit levels, in which a software product can be categorised as useless, useful, competitive or excessive. It recognizes that under-investment leads to an unusable product, but over-investment costs more than the benefits gained.

7.2.2 Models of Software Quality

There are many models that describe software quality. The most common representations of software quality present a hierarchy of quality attributes. Examples of such models including McCall’s quality model, Boehm’s quality model and ISO 9126 (Kitchenham and Pfleeger, 1996). All of these models are criticized for various deficiencies. The major complaints about ISO 9126 are missing or insufficiently detailed aspects of quality or insufficient information on measuring the aspect defined.

An alternative approach to defining quality models was undertaken in Dromey’s quality model (Dromey, 1996). This work proposing defining the actions required to achieve the desired level of quality rather than describing the quality itself. As such this approach provides developers with concrete actions that will achieve the desired quality.
7.2.3 Key Stakeholder Alignment

Stakeholder alignment has been defined as “convergent intentions, shared understanding and coordinated procedures” (Chan, 2002), inline with goals of VBSE. There is overwhelming evidence showing aligned groups outperform those who are not aligned (Chan and Reich, 2007). Alignment allow stakeholder to collaborate more effectively and produce systems that support the long-term business strategies. Highly misaligned teams can cause conflict and eventually lead to the failure of a project.

Stakeholder alignment requires ongoing effort to ensure the group remain aligned (Chan, 2002). Communication, coordination, control, supervision, creating social bonds and building trust are key to creating alignment between stakeholders (Carmel and Abbott, 2006). A lack of awareness and belief in alignment have been identified as main contributors to misalignment (Chan and Reich, 2007).

While GSD creates many opportunities, it also poses challenges that can negatively impact alignment if not properly addressed.

7.2.4 Global Software Development

GSD is defined as software development work undertaken across national boundaries at geographically separated locations. While teams are not co-located, they are still working towards a common goal with a commercially viable product. Groups are commonly classified against two criteria:

1. **Insourcing/Outsourcing** defines whether-or-not the work is undertaken by employees of the organisation.

2. **Onshore/Offshore** defines whether-or not the work is undertaken in the home country of the organisation.

GSD has become a common practice (Conchuir et al., 2009), with a number of benefits and risks. It allows companies to call upon a global talent pool to supplement a locally scarce resource pool (Wang et al., 2008). These specialized skills can have a positive impact on productivity and quality (Conchuir et al., 2009; Mockus and Herbsleb, 2001). Further, many studies cite the ability to reduce development costs (Conchuir et al., 2009), and to focus on strategic business functions with day-to-day operations off-loaded (Wang et al., 2008).

The most commonly cited challenges of GSD relate to communication (Mockus and Herbsleb, 2001) and coordination (Herbsleb, 2007). Co-located teams
have a greater opportunity to share formal and informal discussions, which have been found to be an effective way of creating a shared understanding in relation to what is expected in software quality. Geographic distance reduces a team’s ability to communicate and collaborate (Noll et al., 2010). While new technologies have helped to reduce the barriers created by distance, it is emphasized that distance still matters (Carmel and Abbott, 2006).

Intercultural factors, such as power distance, individualism, and uncertainty avoidance (Hofstede, 1980) can also be problematic in GSD settings. Studies on the impact of national and cross-cultural issues on systems development emphasize the need to take organizational and national culture seriously when working in these environments (Hofstede, 1980). Further, cultural issues can exacerbate exiting communication and coordination problems (Herbsleb and Moitra, 2001).

7.3 Methodology

This chapter aims to determine the level of alignment between key stakeholder groups involved in offshore software product development on the priorities they give to aspects of software product quality. It then seeks to understand the reasons behind the level of alignment between the groups.

7.3.1 Research Questions

This chapter addresses two research questions:

- **RQ 1.** To what degree are success-critical stakeholder groups aligned in offshore software product development on the priorities given to aspects of software product quality:
  - **RQ 1.1.** In the perception of the situation *today*?
  - **RQ 1.2.** In the perception of the *ideal* situation? That is what should the priorities be today.

- **RQ 2.** What are the reasons for the level of alignment seen between these groups?

7.3.2 Method

This chapter employs a method previously developed and used by the authors to determine the level of alignment between key stakeholder groups (Chapter 5,
Chapter 6). Given the success of this method to meet the aims of the study it is reused.

The method draws heavily on Theory-W (Boehm and Jain, 2006), which aims to create win-win scenarios by getting success-critical stakeholder groups to agree on how product development should proceed. This first steps of Theory-W are to identify the success-critical stakeholders, identify how they want to win, negotiate plans to achieve win-win scenarios and control the process to achieve these scenarios.

The seven main steps of the method used to answer the research question are described below.

**Select a company and product**

As each product can have different quality requirements and quality expectations by success-critical stakeholders, it is essential to ensure the study is sufficiently focused.

**Identify success-critical stakeholder groups**

Success-critical stakeholder groups, are groups of people upon whom the success of the product depends—for example product managers and developers. Identifying these groups ensures critical perspectives are not lost, while less important perspectives cannot dominate.

**Develop a quality model**

Literature addressing software quality recognizes that quality depends on both the perspective of the observer and the actual software product in question. That is quality will be defined different by different people, and quality will be defined differently for different products. As such, using any model of software quality as it appears in the literature risks not adequately defining quality in the context being studied. To use one of the quality models mentioned in Section 7.2.2 is a good starting point, but company and product specific needs must be taken into account (Chapter 5).

**Develop a questionnaire**

This method proposes the use of the hierarchical cumulative voting technique (HCV) (Berander and Jönsson, 2006) to elicit the priorities given the various aspects software product quality. This method allows respondents to state the
7.3 Methodology

relative importance of the aspects being studied. Past research has shown that respondents have trouble comparing some aspects of software product quality at a low level directly (Chapter 5); and HCV provides a method for breaking the problem into a series of smaller direct comparison exercise, with a method to join these results back together.

Conduct the questionnaire

The questionnaire should be completed by representatives of each of the identified success-critical stakeholder groups. Doing this in a one-on-one structured interview allows richer information to be collected from the participants, providing greater understanding of the results obtained. This also helps ensure participants have a common understandings of the questionnaire with the interviewer able to assist with questions or problems faced by the respondents.

Analyse the results

It is possible to transform the results of the HCV exercise into cumulative voting (CV) results (Berander and Jönsson, 2006). From here it is possible to group the results by success-critical stakeholder group, and calculate the mean number of points awarded to each aspect of software product quality. These results can be used to calculate Spearman rank correlation coefficients and corresponding p-values (McDonald, 2009), determining the level of alignment between the success-critical stakeholder groups.

The method is extended in this chapter to further examine the alignment between the members of each success-critical stakeholder group. As this has not previously been done, the process is described in Section 7.3.3.

Workshop the results

Finally the results should be presented to participants from each of the success-critical stakeholder groups, asking them for their response to the following questions:

- Do these results look reasonable?
- Do the differences make sense?
- Why do these differences exist?
7.3.3 Determining alignment within a group

Inertia measures the variation in a dataset (Greenacre, 1984). It uses contingency tables, where two categorical variables are cross-tabulated and the chi-square statistic is computed to test the hypothesis of their independence. The inertia is computed by dividing the chi-square statistic by the total number of cases that are distributed in the cells of the contingency table.

For this method each group of stakeholders is considered separately. Given a group of \( S \) stakeholders and a set of \( I \) aspects, it is possible to denote \( n_{ij} \) as the amount allocated to each aspect \((j = 1, \ldots, I)\) by the stakeholder \((i = 1, \ldots, S)\) in the cumulative voting results. Thus, the mean value of the amounts allocated by each stakeholders within a group to issue \( j \) can be defined as:

\[
\bar{n}_j = \frac{1}{S} \sum_{i=1}^{S} n_{ij}
\]

(7.1)

Inertia checks the level of variation between the responses of the stakeholders in a group. When there is little variation between members of the group, all of the members prioritize the aspects in more or less the same way, which is very close to the means values calculated by Equation 7.1. The next step to calculating the inertia is to calculate the chi-square statistic for the group being studied:

\[
\chi^2 = \sum_{i=1}^{S} \sum_{j=1}^{I} \frac{(n_{ij} - \bar{n}_j)^2}{\bar{n}_j}
\]

(7.2)

The chi-square statistic tests the significance of the variation within the group. The test used is the common chi-square test, which compares the value computed in Equation 7.2 with the critical value of the theoretical chi-square distribution with \((S - 1) \times (I - 1)\) degrees of freedom. If the significance of the test is \(< 0.05\), then we can infer that the stakeholders within a group have significant variations in the priorities given to the aspects studied.

In order to compare between different groups it is necessary to normalize the result given by Equation 7.2 as this result is dependent on the number of stakeholders and aspects studied. This can be achieved by dividing the chi-square result by the total number of points allocated by all stakeholders to all aspects within the group studied:

\[
\text{inertia} = \frac{\chi^2}{\sum_{i=1}^{S} \sum_{j=1}^{I} n_{ij}}
\]

(7.3)
As the cumulative voting method gives 1000 points to assign across all the aspects, it is possible to simply this equation to:

\[
inertia = \frac{\chi^2}{1000 \times S}
\]  

Inertia values for each group can be compared. If one group has a larger inertia value it signifies that there is a higher degree of variability within the group.

7.4 Case Study

This chapter presents a case study of two major products at Ericsson. Ericsson is a world leading company in telecommunications, providing a wide range of products and services. These are developed and sold as generic solutions, although customized versions of the products are also developed for key customers.

This chapter employes an exploratory case study on two products at Ericsson to gain insight and understanding from the current situation (Runeson and Höst, 2009). The key decisions made in the implementation of the methodology in the specific context are described in the following sections.

7.4.1 Product

The authors collaborated with a group of managers at Ericsson in India and Sweden to study the alignment between success-critical stakeholder groups described by “offshore insourcing” scenario for two products. Both products are leading solutions for the market they service. These products are referred to as Product 1 and Product 2 throughout this chapter for reasons of confidentiality.

Global software development practices are employed with both products. Work is distributed between the sites using a module-based approach. Sets of modules that make up the product are distributed to each site, reducing the coupling and cohesion between the sites. However, development of individual modules has been moved between sites to maximize the use of available resources.

Product 1 has been developed and sold by Ericsson continuously over more than 10 years. The product is developed at Ericsson sites in India and Sweden. This chapter represents Phase 3 of this research on Product 1. Phase 1 examined the alignment between success-critical stakeholder groups employed by Ericsson
on Product 1 and based in Sweden (Chapter 5). Phase 2 expanded this work to include subcontracted developers based out of the development centre in Sweden (Chapter 6). The third phase, presented in this chapter, examines the alignment between success-critical stakeholder groups employed by Ericsson on Product 1 and based in India. The relationship between these studies is shown in Figure 7.1.

![Figure 7.1: Case studies relating to Product 1](image)

*Product 2* has been under development for approximately two years and sold to customers for approximately one year. This product is developed at sites in China, India and Sweden. It has not been previously studied as part of this work.

The focus of the research presented in this chapter will be on the alignment between success-critical stakeholder groups for the development work based in India for both products. In addition to the Indian-based stakeholders, a couple of Swedish-based *Strategic Product Managers* are included in the research.

### 7.4.2 Success-critical Stakeholder Groups

The first author worked with three senior managers to identify success-critical stakeholder groups. As the same organizational structure is employed for both Product 1 and Product 2, the same roles and responsibilities were identified for both products. The *identified groups* are:

- *Architects* are responsible for the overall product design, assigning requirements to modules of the product.
- *Developers* are responsible for the implementation of requirements.
- *Product Support* provides support to users and product owners at customer sites.
• *Project Management* is responsible for planning and executing projects aligned with the priorities of the strategic product management.

• *Strategic Product Management* has the strategic product responsibility and decides the overall product development direction.

• *Tactical Product Management* supports the strategic product management with expert knowledge of the systems and their architecture. It is also responsible for providing analysis of pre-project requirements in the form of feasibility, impact and technical dependencies.

• *Testing* are responsible for the verification and validation of requirements.

The number of candidates identified by Ericsson and the number of interviewees for each group is shown in Table 7.1 for both products. For example, two *Tactical Product Managers* were identified for each product, but only one was interviewed for *Product 1*. The first author’s limited time in India meant a number of interviews were cancelled due to travel, sickness or unplanned leave. Most groups contained more members than the list of candidates provided by Ericsson, but in these cases the people that supplied the list of candidates aimed to provide a representative sample.

<table>
<thead>
<tr>
<th>Role</th>
<th>Abbrev.</th>
<th>Product 1</th>
<th>Product 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architects</td>
<td>Arch</td>
<td>5/5</td>
<td>3/4</td>
</tr>
<tr>
<td>Developers</td>
<td>Dev</td>
<td>7/10</td>
<td>6/7</td>
</tr>
<tr>
<td>Product Support</td>
<td>PS</td>
<td>2/2</td>
<td>1/1</td>
</tr>
<tr>
<td>Project Managers</td>
<td>PM</td>
<td>3/4</td>
<td>3/3</td>
</tr>
<tr>
<td>Strategic Product Managers</td>
<td>SPM</td>
<td>2/2</td>
<td>3/4</td>
</tr>
<tr>
<td>Tactical Product Managers</td>
<td>TPM</td>
<td>1/2</td>
<td>2/2</td>
</tr>
<tr>
<td>Testers</td>
<td>Test</td>
<td>7/8</td>
<td>4/5</td>
</tr>
</tbody>
</table>

### 7.4.3 Quality Model and Questionnaire

Given this work expands two previously conducted case studies of *Product 1* (Chapter 5, Chapter 6) the software quality model previously used was reviewed by a group of senior managers and selected for reuse in this case study. The managers deemed the model appropriate for both *Product 1* and *Product 2*. 
The terms and definitions defining the model for the case study presented in this chapter can be found in Appendix A.

7.4.4 Conduct the questionnaire

Interviews were conducted by the first author in a one-on-one interview situation. Respondents were first introduced to the aims of the study and then an explanation was provided on how to complete the questionnaire. Participants were invited to share any comments or feedback with the author, and upon completing the questionnaire the interviewer reviewed and discussed the respondents completed questionnaire to confirm and further understand the results. Almost all interviews were conducted face-to-face, with the exception of a couple of Strategic Product Managers who selected to have a phone interview with screen sharing due to conflicting commitments. Interviews took on average 45 minutes.

7.5 Results

This section presents the results for Product 1 and Product 2 individually.

7.5.1 Product 1

First, RQ1.1 aims to determine the degree to which the offshore success-critical stakeholder groups are aligned in how they perceive the priorities on aspects of software product quality today. The results for this question are presented in Table 7.2 and Table 7.3. Table 7.2 contains Spearman rank correlation coefficients for each pair of groups. In this table results can range from one (1) to minus 1 (−1). The closer a result for two groups is to one, the higher the level of agreement between the two groups in the ranking given to aspects studied. Table 7.3 contains p-values for the Spearman rank correlation coefficients. Values indicating the presence of a significant relationship between the priorities of the two groups at the \( p < 0.05 \) level are marked in bold.

The results for Product 1 show lower levels of alignment than seen in the previous case studies of this product when describing the situation today. At the \( p < 0.05 \) level, there is a significant relationship between Product Support and a number of other roles: Developers, Project Managers, Strategic Product Managers and Testers. This could be related to the requirement for Product Support to liaise with the other roles in the identification and resolution of issues,
Table 7.2: Spearman rank correlation coefficients describing Product 1 today

<table>
<thead>
<tr>
<th></th>
<th>Arch</th>
<th>Dev</th>
<th>PS</th>
<th>PM</th>
<th>SPM</th>
<th>TPM</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arch</td>
<td>1.00</td>
<td>0.18</td>
<td>0.35</td>
<td>0.16</td>
<td>0.39</td>
<td>0.03</td>
<td>0.15</td>
</tr>
<tr>
<td>Dev</td>
<td>1.00</td>
<td>0.42</td>
<td>0.30</td>
<td>0.23</td>
<td>0.24</td>
<td>0.42</td>
<td></td>
</tr>
<tr>
<td>PS</td>
<td>1.00</td>
<td>0.68</td>
<td>0.52</td>
<td>-0.09</td>
<td>0.53</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PM</td>
<td>1.00</td>
<td>0.26</td>
<td>-0.12</td>
<td>0.23</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPM</td>
<td>1.00</td>
<td>0.26</td>
<td>0.35</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TPM</td>
<td>1.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.00</td>
</tr>
</tbody>
</table>

Table 7.3: P-values for Spearman rank correlation coefficients describing Product 1 today

<table>
<thead>
<tr>
<th></th>
<th>Arch</th>
<th>Dev</th>
<th>PS</th>
<th>PM</th>
<th>SPM</th>
<th>TPM</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arch</td>
<td>0.000</td>
<td>0.393</td>
<td>0.093</td>
<td>0.464</td>
<td>0.057</td>
<td>0.876</td>
<td>0.483</td>
</tr>
<tr>
<td>Dev</td>
<td>0.000</td>
<td>0.042</td>
<td>0.150</td>
<td>0.282</td>
<td>0.264</td>
<td>0.039</td>
<td></td>
</tr>
<tr>
<td>PS</td>
<td>0.000</td>
<td>&lt;0.001</td>
<td>0.009</td>
<td>0.681</td>
<td>0.008</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PM</td>
<td>0.000</td>
<td>0.221</td>
<td>0.570</td>
<td>0.281</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPM</td>
<td>0.000</td>
<td>0.221</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TPM</td>
<td>0.000</td>
<td>0.154</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test</td>
<td></td>
<td></td>
<td></td>
<td>0.000</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
giving them a cross-functional understanding of the product. There is also a significant relationship between the priorities of the Developers and Testers, who are located together within agile teams. Levels of alignment between the Tactical Product Managers and other roles is low.

The results answering RQ1.2, describing the alignment between success-critical stakeholders in the perceived ideal situation, for Product 1 are presented in Table 7.4 and Table 7.5. The results show lower levels of alignment between the groups when compared to the onshore insourcing case study of the same product, however, the results are inline with those seen in the onshore outsourcing case study. Further it should be noted that in both of the previous case studies lower levels of alignment were seen in the ideal situation when compared with the situation today. This has been put down to different groups seeing different problems of quality that related specifically to their role (e.g. Testers wanting a greater emphasis on the testability quality attribute than the other groups). In this study no groups were found to be aligned at the $p < 0.05$ level.

Table 7.4: Spearman rank correlation coefficients describing Product 1 ideal situation

<table>
<thead>
<tr>
<th></th>
<th>Arch</th>
<th>Dev</th>
<th>PS</th>
<th>PM</th>
<th>SPM</th>
<th>TPM</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arch</td>
<td>1.00</td>
<td>-0.27</td>
<td>0.30</td>
<td>0.02</td>
<td>-0.06</td>
<td>0.19</td>
<td>0.14</td>
</tr>
<tr>
<td>Dev</td>
<td>1.00</td>
<td>0.10</td>
<td>0.21</td>
<td>0.39</td>
<td>-0.17</td>
<td>0.36</td>
<td></td>
</tr>
<tr>
<td>PS</td>
<td>1.00</td>
<td>0.27</td>
<td>0.08</td>
<td>0.29</td>
<td>0.40</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PM</td>
<td>1.00</td>
<td>0.00</td>
<td>0.17</td>
<td>0.23</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPM</td>
<td>1.00</td>
<td>0.08</td>
<td>0.19</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TPM</td>
<td>1.00</td>
<td>0.37</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test</td>
<td>1.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

To determine the degree to which members of a group were aligned, the inertia of each group was calculated. The closer an the inertia value is to zero, the more aligned the member of the group in their priorities. Table 7.6 provides the inertia and the significance of the chi-square test (p-value) for Product 1 for all stakeholders (overall) and for each group separately. These results show that the variability within each group studied is statistically significant at the $p < 0.05$ level, suggesting low levels of alignment between the stakeholders in each group. The inertia could not be calculated for the Tactical Product Managers, as only one person was in this group and thus, by definition, was in complete alignment. These results suggest the members of each group do not
Table 7.5: P-values for Spearman rank correlation coefficients describing Product 1 ideal situation

<table>
<thead>
<tr>
<th></th>
<th>Arch</th>
<th>Dev</th>
<th>PS</th>
<th>PM</th>
<th>SPM</th>
<th>TPM</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arch</td>
<td>0.000</td>
<td>0.194</td>
<td>0.921</td>
<td>0.787</td>
<td>0.369</td>
<td>0.518</td>
<td></td>
</tr>
<tr>
<td>Dev</td>
<td>0.000</td>
<td>0.634</td>
<td>0.329</td>
<td>0.057</td>
<td>0.434</td>
<td>0.082</td>
<td></td>
</tr>
<tr>
<td>PS</td>
<td>0.000</td>
<td>0.194</td>
<td>0.726</td>
<td>0.173</td>
<td>0.052</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PM</td>
<td>0.000</td>
<td>0.999</td>
<td>0.433</td>
<td>0.289</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPM</td>
<td>0.000</td>
<td>0.699</td>
<td>0.371</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TPM</td>
<td></td>
<td>0.000</td>
<td>0.071</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test</td>
<td></td>
<td></td>
<td>0.000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

agree on the priorities on quality for the product.

Table 7.6: Inertia within each group for Product 1

<table>
<thead>
<tr>
<th></th>
<th>Inertia</th>
<th>P-value</th>
<th>Inertia</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overall</td>
<td>0.28</td>
<td>&lt;0.001</td>
<td>0.22</td>
<td>&lt;0.001</td>
</tr>
<tr>
<td>Arch</td>
<td>0.26</td>
<td>&lt;0.001</td>
<td>0.19</td>
<td>&lt;0.001</td>
</tr>
<tr>
<td>Dev</td>
<td>0.21</td>
<td>&lt;0.001</td>
<td>0.15</td>
<td>&lt;0.001</td>
</tr>
<tr>
<td>PS</td>
<td>0.36</td>
<td>&lt;0.001</td>
<td>0.39</td>
<td>&lt;0.001</td>
</tr>
<tr>
<td>PM</td>
<td>0.18</td>
<td>&lt;0.001</td>
<td>0.21</td>
<td>&lt;0.001</td>
</tr>
<tr>
<td>SPM</td>
<td>0.09</td>
<td>&lt;0.001</td>
<td>0.05</td>
<td>&lt;0.001</td>
</tr>
<tr>
<td>TPM</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Test</td>
<td>0.15</td>
<td>&lt;0.001</td>
<td>0.08</td>
<td>&lt;0.001</td>
</tr>
</tbody>
</table>

It is also possible to compare the results from the participants descriptions of the situation today and the ideal situation, giving the level of change they would like to see on the priorities given to the aspects studied. The results for both products are presented in Table 7.7, with Spearman rank correlation coefficients and corresponding p-values. The Spearman rank correlation coefficients range from one (1) to minus one (−1). The greater the deviation from one, the more change desired by that group. P-values have also been calculated, with those at the p < 0.05 significance level marked in bold.

The results for Product 1 show that all groups perceive some need for change. However, there is a significant relationship between the priorities as they are understood in the situation today and in the ideal situation when looking at most groups—Overall, Product Support, Project Managers, Strategic Product
Table 7.7: Desired change correlation coefficients and p-values for Product 1

<table>
<thead>
<tr>
<th></th>
<th>Correlation</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overall</td>
<td>0.43</td>
<td>0.035</td>
</tr>
<tr>
<td>Arch</td>
<td>−0.18</td>
<td>0.390</td>
</tr>
<tr>
<td>Dev</td>
<td>0.06</td>
<td>0.784</td>
</tr>
<tr>
<td>PS</td>
<td>0.41</td>
<td>0.048</td>
</tr>
<tr>
<td>PM</td>
<td>0.77</td>
<td>&lt;0.001</td>
</tr>
<tr>
<td>SPM</td>
<td>0.56</td>
<td>0.005</td>
</tr>
<tr>
<td>TPM</td>
<td>0.51</td>
<td>0.011</td>
</tr>
<tr>
<td>Test</td>
<td>0.26</td>
<td>0.220</td>
</tr>
</tbody>
</table>

Managers and Tactical Product Managers. The groups desiring the greatest level of change in priorities are the Architects and Developers. Reasons for this will be discussed in Section 7.6.

7.5.2 Product 2

The results answering RQ1.1 for Product 2, determining the level of alignment between success-critical stakeholder groups in the perceived situation today, are presented in Table 7.8 and Table 7.9. Overall the results show moderate levels of alignment. At the \( p < 0.05 \) level, there is a significant relationship between the priorities of the Testers and five other roles—Architects, Developers, Strategic Product Managers and Tactical Product Managers. The reason identified in the workshop for this result is that Testers are required to have a functional and technical understanding of the product and through project management are well aware of time and cost limitations, while other roles can be more insulated from some aspects of the product. A significant relationship is also seen between the priorities of the Developers and Project Managers.

The results for RQ1.2 for Product 2, determining the level of alignment between success-critical stakeholder groups in the perceived ideal situation, are presented in Table 7.10 and Table 7.11. A significant relationship between the priorities was found at the \( p < 0.05 \) level between Testers and four other roles—Developers, Product Support, Strategic Product Managers and Tactical Product Managers. Significant alignment is also seen between Developers and Architects; Product Support and Strategic Product Managers; and Strategic Product Managers and Project Managers.

Table 7.12 provides the inertia and the significance of the chi-square test (p-
Table 7.8: Spearman rank correlation coefficients describing Product 2 today

<table>
<thead>
<tr>
<th></th>
<th>Arch</th>
<th>Dev</th>
<th>PS</th>
<th>PM</th>
<th>SPM</th>
<th>TPM</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arch</td>
<td>1.00</td>
<td>0.30</td>
<td>0.01</td>
<td>0.32</td>
<td>0.22</td>
<td>0.39</td>
<td>0.47</td>
</tr>
<tr>
<td>Dev</td>
<td>1.00</td>
<td>0.03</td>
<td>0.42</td>
<td>0.39</td>
<td>0.29</td>
<td>0.45</td>
<td></td>
</tr>
<tr>
<td>PS</td>
<td>1.00</td>
<td>0.08</td>
<td>0.23</td>
<td>0.15</td>
<td>0.22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PM</td>
<td>1.00</td>
<td>0.36</td>
<td>0.09</td>
<td>0.32</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPM</td>
<td>1.00</td>
<td>0.23</td>
<td>0.64</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TPM</td>
<td></td>
<td>1.00</td>
<td>0.48</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test</td>
<td></td>
<td></td>
<td>1.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 7.9: P-values for Spearman rank correlation coefficients describing Product 2 today

<table>
<thead>
<tr>
<th></th>
<th>Arch</th>
<th>Dev</th>
<th>PS</th>
<th>PM</th>
<th>SPM</th>
<th>TPM</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arch</td>
<td>0.000</td>
<td>0.152</td>
<td>0.954</td>
<td>0.132</td>
<td>0.295</td>
<td>0.059</td>
<td><strong>0.021</strong></td>
</tr>
<tr>
<td>Dev</td>
<td>0.000</td>
<td>0.882</td>
<td><strong>0.040</strong></td>
<td>0.058</td>
<td>0.164</td>
<td><strong>0.029</strong></td>
<td></td>
</tr>
<tr>
<td>PS</td>
<td>0.000</td>
<td>0.725</td>
<td>0.275</td>
<td>0.498</td>
<td>0.301</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PM</td>
<td>0.000</td>
<td>0.081</td>
<td>0.671</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPM</td>
<td>0.000</td>
<td>0.280</td>
<td></td>
<td></td>
<td></td>
<td><strong>0.001</strong></td>
<td></td>
</tr>
<tr>
<td>TPM</td>
<td></td>
<td>0.000</td>
<td></td>
<td></td>
<td></td>
<td><strong>0.017</strong></td>
<td></td>
</tr>
<tr>
<td>Test</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.000</td>
</tr>
</tbody>
</table>

Table 7.10: Spearman rank correlation coefficients describing Product 2 ideal situation

<table>
<thead>
<tr>
<th></th>
<th>Arch</th>
<th>Dev</th>
<th>PS</th>
<th>PM</th>
<th>SPM</th>
<th>TPM</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arch</td>
<td>1.00</td>
<td>0.66</td>
<td>−0.11</td>
<td>0.30</td>
<td>0.24</td>
<td>0.38</td>
<td>0.37</td>
</tr>
<tr>
<td>Dev</td>
<td>1.00</td>
<td>0.07</td>
<td>0.40</td>
<td>0.05</td>
<td>0.23</td>
<td>0.49</td>
<td></td>
</tr>
<tr>
<td>PS</td>
<td>1.00</td>
<td>0.34</td>
<td>0.52</td>
<td>0.25</td>
<td>0.50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PM</td>
<td>1.00</td>
<td>0.42</td>
<td>0.36</td>
<td>0.27</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPM</td>
<td>1.00</td>
<td>0.29</td>
<td>0.49</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TPM</td>
<td></td>
<td>1.00</td>
<td>0.58</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test</td>
<td></td>
<td></td>
<td>1.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 7.11: P-values for Spearman rank correlation coefficients describing Product 2 ideal situation

<table>
<thead>
<tr>
<th></th>
<th>Arch</th>
<th>Dev</th>
<th>PS</th>
<th>PM</th>
<th>SPM</th>
<th>TPM</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arch</td>
<td>0.000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dev</td>
<td>0.000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PS</td>
<td>0.000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PM</td>
<td>0.000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPM</td>
<td>0.000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TPM</td>
<td>0.000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test</td>
<td>0.000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

value) for Product 2 for all stakeholders (overall) and for each group separately. These results show that the variability within each group studied is statistically significant at the $p < 0.05$ level, with the exception of the Project Managers in their description of the ideal situation. With one exception, this suggests that the members of each groups do not agree on the priorities on quality for the product. The inertia could not be calculated for Product Support, as only one person was in this group and thus, by definition, was in complete alignment.

Table 7.12: Inertia within each group for Product 2

<table>
<thead>
<tr>
<th></th>
<th>Today</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Arch</td>
<td>0.26</td>
<td>&lt;0.001</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dev</td>
<td>0.19</td>
<td>&lt;0.001</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PS</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PM</td>
<td>0.08</td>
<td>&lt;0.001</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPM</td>
<td>0.19</td>
<td>&lt;0.001</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TPM</td>
<td>0.08</td>
<td>&lt;0.001</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test</td>
<td>0.16</td>
<td>&lt;0.001</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Finally, Table 7.13 shows that while some change is desired by each group, the level of desired change is small. There is a significant relationship between the priorities today and the perceived ideal priorities for each group at the $p < 0.05$ level.
Table 7.13: Desired change correlation coefficients and p-values for Product 2

<table>
<thead>
<tr>
<th></th>
<th>Correlation</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overall</td>
<td>0.65</td>
<td>0.001</td>
</tr>
<tr>
<td>Arch</td>
<td>0.69</td>
<td>&lt;0.001</td>
</tr>
<tr>
<td>Dev</td>
<td>0.69</td>
<td>&lt;0.001</td>
</tr>
<tr>
<td>PS</td>
<td>0.76</td>
<td>&lt;0.001</td>
</tr>
<tr>
<td>PM</td>
<td>0.60</td>
<td>0.002</td>
</tr>
<tr>
<td>SPM</td>
<td>0.65</td>
<td>0.001</td>
</tr>
<tr>
<td>TPM</td>
<td>0.41</td>
<td>0.050</td>
</tr>
<tr>
<td>Test</td>
<td>0.73</td>
<td>&lt;0.001</td>
</tr>
</tbody>
</table>

7.6 Discussion

The results show that each of the success-critical stakeholder groups for Product 1 and Product 2 are in some degree of conflict as to what the priorities are on quality both today and in their perceived ideal situation. There are still results that stand out from each group, however, such as a desire to reduce the time and/or cost limitations on development activities.

When describing the situation today, the results for Product 1 show some alignment between some of the success-critical stakeholder groups. Product Support stands out as the group that is aligned with the priorities of most groups. However, each group has different ideas of quality when it comes to describing their perceived ideal situation. This divergence in the ideal situation has been seen in the previous studies, where each group prioritizes issues that more directly affect them and pay less attention to the issues that affect other groups (Chapter 5, Chapter 6). While all groups want to see some change, only the Architects and Developers want to see significant changes made to the priorities today.

The results for Product 2 show some alignment between the success-critical stakeholder groups when describing both the situation today and the ideal situation. For this product it is the Testers that stand out as aligned with the priorities of the other groups. While all groups want to see some change, the level of change desired was not significant for any group—indicating all groups are reasonably content with the situation today.

During the interviews representatives of all groups expressed that time-based deadlines and cost restrictions were the major factor limiting the ability to achieve higher levels of quality for Product 1 and Product 2. This was most
vocally expressed by the Developers and Testers, who felt less constraints were needed in product development to allow a higher level of quality to be achieved.

To answer RQ2, the following sections explore the reasons behind the alignment and misalignment, along with the tension between scope/quality and time/cost.

7.6.1 Control Variables

First, this section will examine the case study in terms of the control variables used in the agile development environment. Beck (2000) identified four control variables—scope, quality, time and cost—and notes that those developing the software must be able to set at least one of these control variables to ensure they are able to deliver the other three.

The Developers and Testers for Product 1 and Product 2 described a situation where they are assigned functional requirements (scope) to develop and test by a given deadline (time) with a fixed-size team (cost). This leaves Developers and Testers with quality as the control variable to ensure they deliver the scope within time and cost.

Beck (2000) identified quality to be the least optimal control variable. His research found this to be the most expensive to correct if planners were too aggressive with the scope, time and cost. It appears all groups are conscious of this situation for both products, with all groups wanting to reduce the limitations of time and/or cost for both Product 1 and Product 2 when describing their perceived ideal situation.

7.6.2 Cultural Factors

While time and cost were identified as limiting factors in the first case study (Chapter 5), the constraints this had on quality were not as evident as in the case study presented in this chapter. One possible reason for this difference relates to differing cultural factors at play in Sweden and India.

India and Sweden are very different when it comes to power distance (Hofstede, 1980), which refers to the degree to which “the less powerful members of organizations ... accept and expect that power is distributed unevenly.” Of the countries studied by Hofstede, India was found to have the greatest power distance, while Sweden has a small power distance. This was evident in the interviews with Indian Developers and Testers, with many saying that when overambitious work-tasks were assigned to them they would deliver what they could by the deadline set, but would not challenge their managers judgement.
of what they could achieve. The workshop discussions showed that Swedish Developers and Testers saw the assignment of work tasks as a negotiation, in which they had a responsibility to push back if they did not think they could complete the task satisfactorily. The approach taken in Sweden lead to discussion of quality expectations that helped Developers and Testers reshape their understanding of the requirements.

Another difference between India and Sweden refers to the masculinity (Hofstede, 1980), which concerns personal attributes like assertiveness and competitiveness. Hofstede's results show India to be one of the most masculine countries, while Sweden is one of the least. It is likely that a less masculine culture finds it easier to collaborate and reach a mutual consensus on software quality.

India was found to have a shorter-term orientation than Sweden (Hofstede, 1980). A shorter-term orientation is associated with respect for tradition, fulfilling social obligations and protecting one's 'face.' A longer-term orientation is associated with thrift and perseverance. This emphasizes the culture seen in India of not challenging one's manager, and working hard to complete what has been requested by one's manager.

7.6.3 Role-based Temporal Perspective

Shorter-term and longer-term orientations were also observed on a role-based basis. For example, the Architects for both products identified one aspect of quality identified as being under-prioritized and requiring the greatest amount of additional attention. Product Support acknowledged internal concern from some technical groups for this aspects of quality, but noted that this aspect of quality had never been raised as an issue or potential issue by a customer.

In the aforementioned scenario the Architects are concerned about the possibility of products failing to meet customers' future quality requirements. Product Support draws from their experience customers past experiences.

While Product Support are aware of customers immediate needs and concerns, it appears other roles have insight into a longer-term perspective of the products sustainability. This shows a potential incongruence between the cultural aspects and the roles, where roles may have dominance.

7.6.4 Quality Cost-Benefit

Given that both Product 1 and Product 2 have been market leading solutions since they were first released, each fits into the competitive state of the Quper model. During the interviews Strategic Product Managers acknowledged they
were satisfied with the level of quality being delivered to customers, adding that they would like to see development times reduced for the current level of quality. These results were inline with the experiences of the Product Support.

The view of the Strategic Product Managers and Product Support contrasts with that of the, Architects, Developers and Testers, who perceive the need for additional investment in software quality. Given the Strategic Product Managers and Product Support have regular direct customer contact and the current state of the product and its ongoing success, it is likely that the level of quality that the Developers and Testers want to achieve is in the excessive range. This also supports the finding that Architects and Developers of Product 1 want to see significant changes to their perception of the priorities today.

7.6.5 Validity Threats

Not all stakeholders were included in this study and it does not account for the level of influence each role has in shaping software quality. The results used for each group represent an average of the priorities of the members of the same group. While members of each group were not always aligned, the average represents a possible mutual consensus assuming all individual are equal. To help ensure reasonable results were gathered, the results were presented to key stakeholders for their input and feedback.

The quality model used in this chapter was developed without the involvement of the offshore groups included in this chapter, making it possible for some aspects to have been overlooked. Given this model was developed for the same product for a similar group of success-critical stakeholder groups, and interviews participants were given an opportunity for feedback and comments this risk is considered low.

The results in this chapter have limited generalizability (Runeson and Höst, 2009). Priorities will vary between groups, products and at different stages of a product's lifecycle. However, the method has been proven to work in different contexts and can be applied to other cases. More results will provide the possibility for more general findings.

7.7 Conclusion

This chapter presents a case study of the offshore development effort of two products. For these products it seeks to determine the alignment between success-critical stakeholders groups on the priority given to aspects of software product
quality. It then seeks to understand the reasons for the level of alignment between the groups.

The results for RQ1 show low levels of alignment between identified groups in the priorities they give to the aspects that define software quality for both Product 1 and Product 2. Given the low levels of alignment, it was deemed not appropriate to make comparisons between the sites as there is no representative viewpoint from which to compare.

The results for RQ2 point to a number of factors that could be leading to the levels of alignment seen. While the company has continued to create and deliver customer value through Product 1 and Product 2, not all stakeholders are aware how value is delivered to the customers in terms of the scope delivered, the quality delivered, the delivery date and the cost. The main reasons behind the level of alignment are listed below.

- Quality requirements were often not explicitly stated as part of work assignments. This created ambiguity as to what was expected, which created an environment where Developers and Testers assumed the level of quality required. This was not always inline with customer or management expectations.

- A cultural difference was also observed between Developers and Testers in India and Sweden. When Developers and Testers in Sweden are assigned work, they see it as an opportunity to clarify their understanding and negotiate what they can achieve within given deadlines. The approach taken by employees in India respects the ability of management to know what can be achieved, and they do their best to deliver what has been requested. The approach in Sweden helps the groups reach a more common understanding, while the same groups in India are unsatisfied with what they are able to achieve in the time even though, unbeknownst to them, it meets customer needs.

- A difference was also observed in the period considered by different roles when reflecting on quality requirements. For example, Product Support draws upon a knowledge and experiences based on information about the product today and in the past. Other roles are more aware of future customer needs, and the longer-term implication of design decisions. Given a product that is succeeding today is not guaranteed success tomorrow, it is important to draw on both perspectives.

With insufficient direction as to the quality requirements, it is understandable that technical people will aim for technically perfect solutions. However,
such solutions are not feasible given the commercial realities of software development. It has long been recognized that software does not need to be perfect to be commercially viable (Yourdon, 1995), and that an over-investment in software quality requires an excessive investment without appropriate financial reward (Regnell et al., 2008).

The results indicate an opportunity for improvement. Ensuring the success-critical stakeholder groups have a common understanding of software quality will help them work together more effectively towards that goal (Johansson et al., 2001; Phongpaibul and Boehm, 2005). This would improve the job satisfaction, as people would be more reconciled with the work that they deliver, especially knowing it met customers requirements.

This study further emphasized the importance of drawing upon customer experience to shape the product, but care must be taken to this approach does not blind it to looking forward. How far a company chooses to look forward, however, must be taken on a case-by-case basis as it will change between different products.

While these results are not generalizable, the solutions provides a tested methodology from which further academic and industrial work can be done. This chapter extends the previous work in this area, most notably by looking into offshore insourcing, and by providing a method to calculate the level of alignment within a group—not just between groups, as has been done before. This also shows the methodology to work in range of differing contexts.

Going forward research needs to focus on methods to help groups reach a mutual understanding of software quality. This research highlights some of the challenges that are faced in different cultural environments, and in bringing different roles together.
Chapter 8

Quality Alignment in a Global Organisation

This chapter has been submitted for publication as:


Abstract

Global software development (GSD) is rapidly increasing in popularity. With additional key stakeholder groups involved in software development there are more opportunities for misunderstanding and misalignment on software quality issues. This paper aims to identify the levels of alignment between sites involved in a GSD project on software quality issues and explain reasons for misalignment. A case study is conducted of a software development project with sites America, Australia and India for a multinational financial services company. Questionnaires used to identify software quality priorities, and interviews are used to understand the reasons for alignment and misalignment. The Australian site is aligned well with both the American and Indian sites, but these sites are not well aligned with each other. A number of factors are identified that lead to this situation. A set of best practices is identified to help
insure better alignment for future projects.

8.1 Introduction

As global software development (GSD) increases in popularity (Dibbern et al., 2008), the number of stakeholders involved in a software project has also rapidly increased (Herbsleb, 2007). There are various reasons for the uptake in GSD, including cost reduction, proximity to market and making use of different competencies (Smite et al., 2010).

This increase in the number of stakeholders and the distributed setting add complexity to software development. Each group may prioritize the aspects that make up software quality from their own perspective. Software quality covers aspects like reliability, usability efficiency, maintainability and portability. The subjectiveness of software quality coupled with the challenges of alignment in GSD is an issue faced by numerous organisations (Johansson et al., 2001).

Misalignment between critical stakeholders on issues of software quality can lead to conflict and project failure (Chan and Reich, 2007). A common understanding of the expectations in terms of software quality has been found to be the most effective way for groups to achieve a common goal in a GSD setting (Phongpaibul and Boehm, 2005). Thus this study explores a GSD setting with the aim to understand (1) the levels of alignment between sites in terms of software quality, and (2) the reasons for alignment and/or misalignment.

The remainder of this paper is structured as follows. The background and research questions are detailed in Section 8.2. The method is described in Section 8.3. The case study details are provided in Section 8.4. The alignment between sites in the case study are shown in Section 8.6, with the reasons for alignment and misalignment identified in Section 8.7. A discussion is provided in Section 8.8 and conclusions are drawn in Section 8.9.

8.2 Background

This section first introduces the key concepts and work related to this paper and ends with the research questions.

8.2.1 Software Quality

There are many definitions of software quality (Kitchenham and Pfleeger, 1996), with the most common being ‘conformance to specification’ and ‘fit for purpose.’
However, there is an increasing body of literature that recognizes people’s perception of software quality is subjective and shaped by a person’s experiences (Johansson et al., 2001). This is most evident in the value-based software engineering research, which recognizes that the best way to move forward with software development is with mutual consensus of the success-critical stakeholders (Boehm and Jain, 2006). Software does not need to be perfect, it just needs to be good enough (Yourdon, 1995).

There are a number of models that help describe software quality. Most models present a hierarchy of quality attributes, with examples of such models including McCall’s quality model, Boehm’s quality model and ISO 9126 (Kitchenham and Pfleeger, 1996). However, deficiencies have been identified with each of these models, such as missing or insufficiently detailed aspects of quality or insufficient information on measuring each aspect.

### 8.2.2 Key Stakeholder Alignment

Inline with the value-based approach to software development, the alignment of stakeholders in software development has been defined as “convergent intentions, shared understanding and coordinated procedures” (Chan, 2002). Organizations continually try to align stakeholder groups as there is overwhelming evidence that successfully aligned groups will outperform those that are not aligned (Chan and Reich, 2007). Alignment of stakeholders in software development allows them to collaborate more effectively and produce systems that support the long-term business strategies. Highly misaligned teams can cause conflict and eventually lead to the failure of a project.

Stakeholder alignment is a daunting task due to the fact that alignment is not a state (Chan, 2002). Keeping groups aligned requires ongoing work. A lack of awareness and belief in alignment have been identified as main contributors to misalignment (Chan and Reich, 2007).

While GSD creates many opportunities, it also poses challenges that can negatively impact alignment if not properly addressed.

### 8.2.3 Global Software Development

GSD is defined as software development work undertaken across national boundaries at geographically separated locations. While teams are not co-located, they are still working towards a common goal with a commercially viable product.

GSD is becoming increasingly common practice. Ten years ago 185 of the Fortune 500 companies outsourced development effort to India alone (Mockus
and Herbsleb, 2001). In the 10 years since then there has been a 25-fold market increase in GSD (Conchuir et al., 2009).

GSD has a number of benefits. Companies can call upon a global talent pool where resources may be locally scarce (Wang et al., 2008). These specialized skills can have a positive impact on productivity and quality (Conchuir et al., 2009; Mockus and Herbsleb, 2001). Further, many studies cite the abilities to reduce development costs (Conchuir et al., 2009), and to focus on strategic business functions with day-to-day operations off-loaded (Wang et al., 2008).

However, GSD brings with it challenges not normally faced by co-located teams. The most commonly cited issues in a GSD setting relate to communication (Mockus and Herbsleb, 2001) and coordination (Herbsleb, 2007). Co-located teams have a greater opportunity to share formal and informal discussions, which have been found to be an effective way of creating a shared understanding in relation to what is expected in software quality. Geographic distance reduces a team’s ability to communicate and collaborate (Noll et al., 2010). While new technologies have helped to reduce the barriers created by distance, it is emphasized that distance still matters (Carmel and Abbott, 2006).

Intercultural factors, such as power distance, individualism, and uncertainty avoidance (Hofstede, 1980) can also be problematic in GSD settings. Studies on the impact of national and cross-cultural issues on systems development emphasize the need to take organizational and national culture seriously when working in these environments (Hofstede, 1980). Cultural issues can exacerbate existing communication and coordination problems (Herbsleb and Moitra, 2001).

8.2.4 Research Questions

Previous research has studied levels of alignment of software quality in co-located settings, but to the best of the authors’ knowledge no study has specifically sought to understand levels of alignment in a GSD setting. Given the growing popularity of GSD, and the challenges faced by practitioners in these settings, it is of key importance to understand how well different sites are aligned in their understanding of software quality, and what factors affect the levels of alignment.

Thus the first research question aims to undertake exploratory research through a case study in order to answer:

**RQ1:** What is the level of alignment between sites in the priorities given to aspects of software quality in a global software development setting?
While answering this question will identify if there are differences between sites in the priorities given to aspects of software quality, it will not identify the reasons behind these differences. Thus, for the same case, the second research question seeks to answer:

RQ2: What are the reasons behind key differences between the sites in how they prioritize aspects of software quality?

The next section presents a methodology to answer these research questions.

8.3 Methodology

This section presents the methodology in two parts—to determine the level of alignment and the reasons for misalignment between groups.

8.3.1 Determining Alignment

To answer RQ1 and determine the alignment between sites, this paper employs a method previously developed and used by the authors—as presented in Chapter 4. The main steps of this method are:

1. Select a company and product: As each product can have different quality requirements it is essential the study is sufficiently focused.

2. Identify key stakeholder groups: These are the groups upon whom the success of the product depends.

3. Select/Tailor a quality model: Literature on software quality recognizes that quality depends on the product in question. It is possible any standard model will have deficiencies in the setting studied.

4. Develop questionnaire: To elicit the importance of each aspect of software quality identified in the model, use the hierarchical cumulative voting method (Chapter 2, Chapter 5). This method allows respondents to show the relative importance of each aspect.

5. Conduct questionnaire: Elicit the priorities from representatives of the key stakeholder groups using the questionnaire developed.
6. **Analyze results:** Using the methods described by Berander and Jönsson (2006) it is possible consolidate the results for each group and then compare the groups using the Spearman rank correlation matrix. Further, inertia measures the variation in a dataset (Greenacre, 1984) and is used to measure the alignment within each group. It uses contingency tables, where two categorical variables are cross-tabulated and the chi-square statistic is computed to test the hypothesis of their independence. The inertia is computed by dividing the chi-square statistic by the total number of cases that are distributed in the cells of the contingency table.

7. **Workshop:** Discuss the results with representatives of the group to develop a deeper understanding of the results.

While the results of this work will identify the level of alignment between sites, an interview study will be conducted to explain the alignment or misalignment between the groups.

### 8.3.2 Understanding Alignment

Semi-structured interviews are used to identify reasons for alignment and misalignment between sites. This approach allows for unexpected and insightful information to be discovered (Hair et al., 2003) and it allows the researcher to probe and identify possible hidden reasons for a specific behaviour.

A semi-structured interview scheduled was prepared by one of the authors. It covers (a) the interviewee’s perception of the product, (b) the elicitation of quality requirements, (c) the prioritization of quality requirements, and (d) communication of quality requirements. The interview schedule was then reviewed and updated by two of the other authors, and finally was reviewed and updated based on a pilot study.

The methodology described in this section was then applied to a case study.

### 8.4 Case Study

This section describes the case study, with a focus on the elements required by the methodology.
8.4 Case Study

8.4.1 Company and Product Information

The organisation involved in the case study is a multinational financial institution. For reasons of confidentiality the company will be referred to by the pseudonym Global Finance.

Global Finance provides a wide range of payment, expense management and travel solutions. Its customers range from individual consumers to other multinational corporations. Global Finance is based in the US and employs approximately 60,000 people globally. It operates throughout the North American, South American, European, African, Middle Eastern and Asia-Pacific regions.

Specifically this case study focuses on a product that is intended to replace and extend several back-end billing and bill management systems for travel services provided by Global Finance. This product is referred to as TravelBill in this paper.

The TravelBill development project follows Global Finance’s methodology for large projects. This methodology is loosely-based on the waterfall development methodology with overlapping phases—for example, the design phase will still be underway when the build phase begins.

Further, the product is being rolled out in a series of markets following a phase-based approach. The first phase was to roll this product out in the US market. While the US rollout is underway the Australian rollout was started. The case study was conduct after the Australian roll-out began. Further rollouts in other countries is planned for the near future.

The Global Finance TravelBill product was chosen to be the subject of this case study as it was aligned with the objectives of the study and one of the authors had access to the company as a business analyst on the project studied.

8.4.2 Success-Critical Stakeholders

The success-critical stakeholders for TravelBill were identified in collaboration with a high-level manager involved in the product development. These were classified into three perspectives:

- The business perspective covers business users and system sponsors. Everyone in this group is an internal employee of Global Finance.

- The product perspective includes business analysts, architects, developers and testers. The business analysts and architects are both internal employees and external contractors. Developers all come from IndiaSoft.
The project perspective covers project managers. This role is fulfilled by a mixture of internal employees and external contractors.

IndiaSoft provides outsourced software development services. It was founded in India, employs over 160,000 employees and operates in 42 countries. Global Finance has used IndiaSoft’s services for approximately 10 years, during which time they have developed a good working relationship. While Global Finance has relationships with other software development providers, IndiaSoft was chosen to work on TravelBill for their experience with existing legacy system.

8.4.3 Development Sites

When the case study was conducted three sites were involved with the product:

- **Global Finance, America** is the main site for Global Finance. It hosts people from the business, product and project perspectives. Everyone is co-located within one building, however, the business stakeholders are situated in a different part of the building to the other groups.

- **Global Finance, Australia** closely resembles the American site in both the roles represented, and their physical location within the building.

- **IndiaSoft, India** hosts product and project stakeholders from IndiaSoft that are contracted to Global Finance.

The first phase of product development aimed to roll-out the product in the American site. This phase, however, involved the American, Australian and Indian sites. This phase of the project took much longer, and cost much more than expected. The second phase was underway at the time of the study, which aimed to roll-out the product to the Australian site.

8.4.4 Quality Model

Global Finance does not use any specific models to describe software quality, and none had been defined for the product or projects prior to our study.

Given the absence of a quality model, the authors proposed the use of the ISO9126 (2001) quality model as a foundation. One of the authors then worked with two senior managers from the product and project perspectives to tailor it to the specific needs of the product over two one-hour discussions. Based on the feedback to the ISO 9126 quality model the following changes were made:
8.4 Case Study

- Security was raised one-level in the hierarchy and given the sub-attributes Security Accuracy and Security Compliance. The managers reviewing the model felt that the profile of security with both Global Finance and the TravelBill product meant it should be seen as a top-level attribute.

- The sub-attribute of Compliance existed for each of the top-level aspects of quality. These were removed to avoid confusion, with the exception of security where it was still felt necessary.

- An additional sub-attribute was added under portability—Conformance.

8.4.5 Participants

The number of people involved with the Product during the study is shown in Table 8.1. This table breaks down the number by site and perspective. The number of people representing the business perspective changes with demand, so minimum levels have been provided.

<table>
<thead>
<tr>
<th>Perspective</th>
<th>America</th>
<th>Australia</th>
<th>India</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business</td>
<td>&gt; 15</td>
<td>&gt; 10</td>
<td>&gt; 7</td>
</tr>
<tr>
<td>Product</td>
<td>11</td>
<td>7</td>
<td>12</td>
</tr>
<tr>
<td>Project</td>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

Questionnaires and instructions on how to complete them were sent out to 44 candidate participants via email. In total 28 responses were received—Table 8.2 shows a breakdown of the returned results by site and perspective.

Interviewees were selected to create a diverse group in terms of both perspective and location. Where possible preference was given to people who had been working on the product longer, as it was thought these people would have more experiences from which to draw during the interview. A breakdown of the interviewees by site and perspective is shown in Table 8.3.

Interviews were conducted in a one-on-one setting between the author working at the Global Finance and the interviewees. Interviews in Australia were conducted face-to-face during office hours, while interviews with people in America and India were conducted over the phone at a time convenient to both the interviewer and interviewee. Interviews lasted approximately 30 minutes each.
Each interview was recorded, transcribed and emailed to the interviewee for validation.

### 8.5 Priorities on Quality Aspects

This section presents the ranking of the qualities studied. The results are presented for each site, and overall. The ranks given to each aspect in the situation as it is perceived *today* for each location and overall are presented in Table 8.4.

The ranks given to each aspect in the perceived *ideal* situation for each location and overall are presented in Table 8.5.

The movement in each quality from the situation *today* to the *ideal* situation for each site and overall are presented in Table 8.6.

### 8.6 Alignment Between Key Stakeholder Groups

After the questionnaires were completed and returned for the TravelBill product, the ranking of the quality attributes was calculated for each site, and overall for all sites. This was done for both a description of the situation as it was perceived *today*, and the description of the perceived *ideal* situation—what the participants thought should be happening today.

The alignment between the sites was calculated using the Spearman rank
Table 8.4: Ranking of qualities today

<table>
<thead>
<tr>
<th></th>
<th>All</th>
<th>US</th>
<th>AU</th>
<th>IN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Suitability</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>Time behaviour</td>
<td>2</td>
<td>6</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Security Compliance</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>Accuracy</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>17</td>
</tr>
<tr>
<td>Security Accuracy</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>12</td>
</tr>
<tr>
<td>Interoperability</td>
<td>6</td>
<td>4</td>
<td>7</td>
<td>15</td>
</tr>
<tr>
<td>Understandability</td>
<td>7</td>
<td>16</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>Operability</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>Compliance</td>
<td>9</td>
<td>13</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>Learnability</td>
<td>10</td>
<td>19</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>Maturity</td>
<td>11</td>
<td>22</td>
<td>14</td>
<td>2</td>
</tr>
<tr>
<td>Recoverability</td>
<td>12</td>
<td>21</td>
<td>12</td>
<td>3</td>
</tr>
<tr>
<td>Cost</td>
<td>13</td>
<td>14</td>
<td>20</td>
<td>9</td>
</tr>
<tr>
<td>Installability</td>
<td>14</td>
<td>9</td>
<td>16</td>
<td>23</td>
</tr>
<tr>
<td>Changeability</td>
<td>15</td>
<td>7</td>
<td>18</td>
<td>21</td>
</tr>
<tr>
<td>Adaptability</td>
<td>16</td>
<td>10</td>
<td>15</td>
<td>19</td>
</tr>
<tr>
<td>Analysability</td>
<td>17</td>
<td>17</td>
<td>13</td>
<td>21</td>
</tr>
<tr>
<td>Fault tolerance</td>
<td>18</td>
<td>12</td>
<td>25</td>
<td>10</td>
</tr>
<tr>
<td>Time</td>
<td>19</td>
<td>23</td>
<td>11</td>
<td>20</td>
</tr>
<tr>
<td>Testability</td>
<td>20</td>
<td>11</td>
<td>22</td>
<td>23</td>
</tr>
<tr>
<td>Attractiveness</td>
<td>21</td>
<td>15</td>
<td>24</td>
<td>13</td>
</tr>
<tr>
<td>Resource behaviour</td>
<td>22</td>
<td>20</td>
<td>17</td>
<td>14</td>
</tr>
<tr>
<td>Stability</td>
<td>23</td>
<td>18</td>
<td>21</td>
<td>18</td>
</tr>
<tr>
<td>Conformance</td>
<td>24</td>
<td>25</td>
<td>19</td>
<td>16</td>
</tr>
<tr>
<td>Replaceability</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>25</td>
</tr>
</tbody>
</table>
Table 8.5: Ranking of qualities in perceived ideal situation

<table>
<thead>
<tr>
<th>Quality</th>
<th>All</th>
<th>US</th>
<th>AU</th>
<th>IN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>Time behaviour</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>Suitability</td>
<td>3</td>
<td>7</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>Security Compliance</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>Security Accuracy</td>
<td>5</td>
<td>2</td>
<td>6</td>
<td>11</td>
</tr>
<tr>
<td>Recoverability</td>
<td>6</td>
<td>13</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>Operability</td>
<td>7</td>
<td>9</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>Maturity</td>
<td>8</td>
<td>14</td>
<td>13</td>
<td>2</td>
</tr>
<tr>
<td>Understandability</td>
<td>9</td>
<td>16</td>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td>Compliance</td>
<td>10</td>
<td>8</td>
<td>10</td>
<td>13</td>
</tr>
<tr>
<td>Fault tolerance</td>
<td>11</td>
<td>12</td>
<td>21</td>
<td>6</td>
</tr>
<tr>
<td>Stability</td>
<td>12</td>
<td>4</td>
<td>18</td>
<td>18</td>
</tr>
<tr>
<td>Learnability</td>
<td>13</td>
<td>19</td>
<td>8</td>
<td>14</td>
</tr>
<tr>
<td>Interoperability</td>
<td>14</td>
<td>11</td>
<td>12</td>
<td>16</td>
</tr>
<tr>
<td>Adaptability</td>
<td>15</td>
<td>6</td>
<td>17</td>
<td>19</td>
</tr>
<tr>
<td>Resource behaviour</td>
<td>16</td>
<td>15</td>
<td>11</td>
<td>17</td>
</tr>
<tr>
<td>Time</td>
<td>17</td>
<td>17</td>
<td>20</td>
<td>8</td>
</tr>
<tr>
<td>Changeability</td>
<td>18</td>
<td>10</td>
<td>14</td>
<td>22</td>
</tr>
<tr>
<td>Conformance</td>
<td>19</td>
<td>24</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>Cost</td>
<td>20</td>
<td>18</td>
<td>23</td>
<td>12</td>
</tr>
<tr>
<td>Analysability</td>
<td>21</td>
<td>20</td>
<td>16</td>
<td>22</td>
</tr>
<tr>
<td>Testability</td>
<td>22</td>
<td>22</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>Installability</td>
<td>23</td>
<td>21</td>
<td>22</td>
<td>22</td>
</tr>
<tr>
<td>Attractiveness</td>
<td>24</td>
<td>23</td>
<td>24</td>
<td>21</td>
</tr>
<tr>
<td>Replaceability</td>
<td>25</td>
<td>25</td>
<td>25</td>
<td>25</td>
</tr>
</tbody>
</table>
### 8.6 Alignment Between Key Stakeholder Groups

Table 8.6: Movement in ranks between situation today and perceived ideal situation

<table>
<thead>
<tr>
<th></th>
<th>All</th>
<th>US</th>
<th>AU</th>
<th>IN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Installability</td>
<td>−9</td>
<td>−12</td>
<td>−6</td>
<td>1</td>
</tr>
<tr>
<td>Interoperability</td>
<td>−8</td>
<td>−7</td>
<td>−5</td>
<td>−1</td>
</tr>
<tr>
<td>Cost</td>
<td>−7</td>
<td>−4</td>
<td>−3</td>
<td>−3</td>
</tr>
<tr>
<td>Analysability</td>
<td>−4</td>
<td>−3</td>
<td>−3</td>
<td>−1</td>
</tr>
<tr>
<td>Learnability</td>
<td>−3</td>
<td>0</td>
<td>2</td>
<td>−9</td>
</tr>
<tr>
<td>Changeability</td>
<td>−3</td>
<td>−3</td>
<td>4</td>
<td>−1</td>
</tr>
<tr>
<td>Attractiveness</td>
<td>−3</td>
<td>−8</td>
<td>0</td>
<td>−8</td>
</tr>
<tr>
<td>Suitability</td>
<td>−2</td>
<td>−6</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Understandability</td>
<td>−2</td>
<td>0</td>
<td>−2</td>
<td>−3</td>
</tr>
<tr>
<td>Testability</td>
<td>−2</td>
<td>−11</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Security Compliance</td>
<td>−1</td>
<td>−2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Compliance</td>
<td>−1</td>
<td>5</td>
<td>−1</td>
<td>−9</td>
</tr>
<tr>
<td>Time behaviour</td>
<td>0</td>
<td>3</td>
<td>−2</td>
<td>0</td>
</tr>
<tr>
<td>Security Accuracy</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Replaceability</td>
<td>0</td>
<td>−1</td>
<td>−2</td>
<td>0</td>
</tr>
<tr>
<td>Operability</td>
<td>1</td>
<td>−1</td>
<td>−1</td>
<td>3</td>
</tr>
<tr>
<td>Adaptability</td>
<td>1</td>
<td>4</td>
<td>−2</td>
<td>0</td>
</tr>
<tr>
<td>Time</td>
<td>2</td>
<td>6</td>
<td>−9</td>
<td>12</td>
</tr>
<tr>
<td>Accuracy</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>Maturity</td>
<td>3</td>
<td>8</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Conformance</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>Recoverability</td>
<td>6</td>
<td>8</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>Resource behaviour</td>
<td>6</td>
<td>5</td>
<td>6</td>
<td>−3</td>
</tr>
<tr>
<td>Fault tolerance</td>
<td>7</td>
<td>0</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Stability</td>
<td>11</td>
<td>14</td>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>
correlation coefficients. The alignment between each site in how they describe the situation today is shown in Table 8.7. Similarly the alignment between the sites in how they describe the ideal situation is shown in Table 8.8. In both tables a result of one (1) means both sites have ordered the priorities in the exact same way, zero (0) means there is no relationship in how the two sites priorities, and minus one (−1) means the two sites rank the priorities in the opposite way. Thus a result closer to one indicates a higher level of alignment.

Table 8.7: Alignment today: Correlation coefficients and corresponding p-values

<table>
<thead>
<tr>
<th>Correlations</th>
<th>P-values</th>
</tr>
</thead>
<tbody>
<tr>
<td>US AU IN</td>
<td>US AU IN</td>
</tr>
<tr>
<td>US 1.00 0.51 0.05</td>
<td>0.000 0.009 0.828</td>
</tr>
<tr>
<td>AU 1.00 0.47</td>
<td>0.000 0.018</td>
</tr>
<tr>
<td>IN 1.00</td>
<td></td>
</tr>
</tbody>
</table>

Table 8.8: Ideal scenario alignment: Correlation coefficients and corresponding p-values

<table>
<thead>
<tr>
<th>Correlations</th>
<th>P-values</th>
</tr>
</thead>
<tbody>
<tr>
<td>US AU IN</td>
<td>US AU IN</td>
</tr>
<tr>
<td>US 1.00 0.65 0.49</td>
<td>&lt; 0.001 0.012</td>
</tr>
<tr>
<td>AU 1.00 0.60</td>
<td>0.000 0.002</td>
</tr>
<tr>
<td>IN 1.00</td>
<td>0.000</td>
</tr>
</tbody>
</table>

The results for the situation today show the American and Australian sites to be reasonably aligned, and show the Australian and Indian sites to be reasonably aligned. However, the American and Indian site are not aligned.

The Indian site gave a much higher priority to maturity, recoverability, standardisation/compliance and learnability than the American site. A much lower priority was given by the Indian site to adaptability, time, changeability, instability and testability than the American site. The Australian site, however, sat somewhere between the American site and Indian site on all these issues.

The results for the perceived ideal situation show all of the sites to be in a reasonable level of alignment on the prioritization of software quality issues.

Reasons for differences between the sites will be explored in Section 8.7.

It is possible to see the level of alignment between the participants within a site or between all participants overall by calculating the inertia. Table 8.9
shows the inertia for both the situation as it is understood today, and the perceived ideal situation. A value of zero (0) means that all of the participants in the group are in complete agreement, while higher values show the level of disagreement between the participants in a given group. The results show the participants to be more homogeneous in how they describe the ideal situation than how they describe the situation today. This differs with previous studies (Chapter 5, Chapter 6).

The amount of change desired overall and the participants at each site is shown in Table 8.10. The table presents Spearman rank correlation coefficients comparing the priorities in the situation as it is understood today and the perceived ideal situation. The results in this table are on a scale from one (1), which no desire for change, to minus one (−1), which means the maximum amount of change is desired. The results indicate that not much change is desired by any one site, nor when the results of all sites are combined overall. The American site desires the greatest amount of change, with a correlation coefficient of 0.65.

<table>
<thead>
<tr>
<th>Site</th>
<th>Today Correlation</th>
<th>Today P-value</th>
<th>Ideal Correlation</th>
<th>Ideal P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overall</td>
<td>0.430</td>
<td>&lt; 0.001</td>
<td>0.296</td>
<td>&lt; 0.001</td>
</tr>
<tr>
<td>America</td>
<td>0.460</td>
<td>&lt; 0.001</td>
<td>0.253</td>
<td>&lt; 0.001</td>
</tr>
<tr>
<td>Australia</td>
<td>0.287</td>
<td>&lt; 0.001</td>
<td>0.280</td>
<td>&lt; 0.001</td>
</tr>
<tr>
<td>India</td>
<td>0.328</td>
<td>&lt; 0.001</td>
<td>0.301</td>
<td>&lt; 0.001</td>
</tr>
</tbody>
</table>

Table 8.10: Desired levels of change: Correlation coefficients and corresponding p-values

<table>
<thead>
<tr>
<th>Site</th>
<th>Correlation</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overall</td>
<td>0.79</td>
<td>&lt; 0.001</td>
</tr>
<tr>
<td>America</td>
<td>0.65</td>
<td>&lt; 0.001</td>
</tr>
<tr>
<td>Australia</td>
<td>0.88</td>
<td>&lt; 0.001</td>
</tr>
<tr>
<td>India</td>
<td>0.77</td>
<td>&lt; 0.001</td>
</tr>
</tbody>
</table>
8.7 Issues Affecting Alignment and Misalignment

The interviewees highlighted numerous factors perceived as negatively impacting the ability to achieve a common understanding of software quality between the sites. These are detailed in this section and confirmed through related literature.

8.7.1 Communication and Coordination

Operating product development across three time zones proves problematic for TravelBill, which negatively impacts the ability of the sites to reach a common understanding on quality issues. The American and Indian sites, which are not aligned in how they prioritize aspects of quality today, have no overlapping business hours. The Australian site is aligned with both the American and Indian sites, and shares approximately four business hours with the American site and three hours with the Indian site.

Multiple interviewees claimed that one of the main implications of the time zone difference is that it is very difficult to get the right people together, with sites missing out on key meetings. One of the business analysts further emphasized that when a site misses out on a meeting they are not always updated as to what was discussed, severely limiting the ability of the sites to reach a mutual consensus on software quality issues:

*It is much more difficult to communicate with people across various time zones. Like for instance, you could have a quality discussion in the US and if someone does not relate the information because they are not a participant in that meeting in a different time zone, it’s hard. It’s not always communicated across. Sometime, the out of sight, out of context comes into play.*

Communication and coordination issues are well documented for projects employing GSD practices (Herbsleb, 2007; Mockus and Herbsleb, 2001). This contrasts with co-located employees who share office hours, office space and can easily have planned and impromptu meetings. Herbsleb et al. (2005) also found it was more time consuming and difficult to communicate across globally distributed sites.

These communication and coordination issues were confounded with inadequate requirements documentation.
8.7 Issues Affecting Alignment and Misalignment

8.7.2 Inadequate Quality Requirement Documentation

A number of interviewees highlighted various inadequacies in the requirements documentation, specific to quality, that they felt affected the ability of the sites to reach a mutual understanding of the quality expectations.

In the user testing phases of the product development a significant portion of the defects identified, “about 20–30%,” related to “cosmetic changes.” An interviewee identified a “lack of documentation” as the prime cause of these defects, however, the authors believe this problem to be made worse by the limited ability for the developers at the Indian site to access TravelBill users during development phase.

Representatives from the business perspective recognized that they had made assumptions about the system that was to be developed that were not documented. This was also seen as negatively impacting the ability of the sites to come to a mutual agreement on what was expected in terms of quality.

\[\text{We went in with certain assumptions and we shouldn't have. So I think it should have been documented and lot of things should have been more clearly documented what they were. And it now affects our quality.}\]

Further, there were a number of changes to the TravelBill product that were not documented. As stated in Section 8.7.1, these discussions did not always involve all sites, and the missing sites were not always informed of decisions made. Coupled with a waterfall-style development methodology, this meant many problems were not picked up until the testing phase.

\[\text{We made decisions on the fly without documenting them. And that has burnt us. So we've incurred a lot more re-work because it breaks and it comes back in a bug report/issue.}\]

A business stakeholder blamed the poor documentation on “aggressive timelines” set by high level management.

These results are aligned with other case studies in GSD, which have documented problems with a lack of formal documentation (Ågerfalk and Fitzgerald, 2006).

8.7.3 Cultural Barriers

Cultural issues were identified by a number of interviewees as having a negative impact on the ability of the sites to align on issues of software quality.
Interviewees found collaboration between Australia and America easier than collaboration with India. An architect noted that “dealing with India, there are different issues from a cultural perspective and how people respond. That makes it a little tougher.”

However, these cultural differences were seen to become less problematic over the duration of the product development as relationships developed between individuals at the three sites:

> From my perspective, regardless of culture we have established personal relationships. And it’s easy to step past that. Once you establish that it makes it ... a lot easier to talk openly and gather information.

The biggest cultural issue for the product development related to power distance, or the perceived equality between colleagues. This case study confirmed the results of Hofstede (1980); workers at the American and Australian sites are prepared to challenge and question in a discussion made by a more senior colleague, while workers at the Indian site would accept and agree even if they did not understand or agree.

> In the US and Australia, a person doesn’t have a problem speaking up and saying, ‘hey, this doesn’t work right.’ Culturally in India, someone ... would never say that. They would leave that to their ... [senior manager] to bring that up. They will tell you, ‘yes, yes, yes, yes,’ they understand, when they don’t.

Hofstede (1980) found Indian culture to be similar on two cultural dimensions and different on two cultural dimensions when compared to American and Australian culture. All cultures are masculine, with weak uncertainty avoidance. However, American and Australian cultures have a small power distance, while India has a large power distance: and American and Australian culture is more individualistic, while Indian culture is more collectivist.

### 8.7.4 Domain Knowledge

Representatives of the business perspective identified a lack of domain knowledge on the part of the developers based in India to negatively impact quality. One of the business representatives explained how their language was interpreted differently by programmers to what they had intended:
I would use terminology from a finance background, which is different to what a programmer would use. I think there was a big gap and problem. I think it created gaps in our system.

Lack out domain knowledge is often cited as one of the key challenges in GSD settings (Kommeren and Parviainen, 2007). Thus, this finding is inline with the related literature.

8.8 Discussion

This section draws upon the issues affecting alignment and misalignment presented in Section 8.7 to explain the levels of alignment between the sites as shown in Section 8.6. In particular, this section seeks to understand why:

- The American and Australian sites and the Australian and Indian sites are aligned for the situation today, but the American and Indian sites are not.
- Each site is aligned when asked to describe the ideal situation.
- Participants in the US site agree more between each other when describing the ideal situation than when describing the situation today.

The Indian site is isolated from the users of TravelBill, with no representatives of this group onsite. This situation was compounded at the beginning of the product development, when there was no overlap in business hours between the two active sites. This situation can help explain the lack of alignment between the American and Indian sites in how they perceive software quality today.

It was much easier for the stakeholders at the Indian site to arrange meetings with users of the TravelBill system. The Australian and Indian business days overlap. However, the Indian site was still limited in the level of interaction that could be achieved; for example, it was not easily possible to sit down and go through screen layouts. Thus it was easier for the Indian site to develop a more mutual understanding of quality with the Australian site than for the American site. However, this level of interaction was not sufficient to overcome all of the challenges faced.

The organizational structure may also play a role in alignment on quality. The American and Australian sites are both part of the Global Finance organization, while Indian site is that of the outsource partner IndiaSoft. Further,
the American site hosts the head office of Global Finance, making both the Australian and Indian sites offshore locations. Thus Australia shares common elements in terms of onshore/offshore and inhouse/outsourced, while the American and Indian sites do not share any comment elements.

The Indian site was also much more vulnerable to deficiencies in the requirements documentation than the American and Australian sites. The workers at the Indian site generally came with much less domain knowledge than those at the other sites, and without representatives of Global Finance’s business perspective onsite it was much harder to clarify areas of ambiguity.

The Indian site was also the most hurt by the incomplete requirements documentation. With limited domain knowledge and access to business stakeholders, the Indian workers were very reliant on the information contained in these documents. This problem was compounded with changes to the system not always being documented or communicated. While the people in the American and Australian sites had knowledge, experience and expectations about the system, these were not effectively communicated and limited the ability of the Indian site to reach a common understanding of software quality with the other sites.

Over the course of the product development, through testing and bug reports, the requirements became more detailed. Thus the Indian site was able to develop a greater understanding as the project progressed. But ultimately this was too late for the project to be delivered on time and within budget.

Cultural differences also worked against the Indian site in the beginning. The American and Australian have a similar work culture, but this differs to aspects of the Indian work culture. However, the American and Australian sites felt that the cultural barriers decreased over the life of the product development as personal contacts were made and developed. This is another situation where it was unfortunate that the American and Indian sites do not share any part of their business day, limiting the communication that can occur, limiting the ability of the groups to breakdown the cultural barriers.

Finally, the questionnaire participants at the American site had much higher levels of disagreement between one-another when describing the priorities on software quality today. When these results were shown to representatives of this group they felt there were two sets of responses—the first class was created by respondents that described the priorities on software quality today, even if they were not being achieved; the second class was created by respondents that described what was being delivered today, which they felt did not represent the true priorities. However, there was much more agreement within the American site when it came to describing what should be happening today.
8.9 Conclusion

This paper presents a case study at Global Finance examining the level of alignment on aspects of software quality between sites in a GSD setting. The paper aimed to answer two research questions.

The first research question, RQ1, sought to determine the level of alignment between sites in the priorities given to aspects of software quality in a global software development setting. This was done twice—first, as the priorities were understood by the participants today; and then how they participants would prioritize the aspects of qualities in their perceived ideal situation. The results describing the situation today found the American Global Finance site to be aligned with the Australian Global Finance site, and the Australian Global Finance site to be aligned with Indian IndiaSoft site, however, the American and Indian sites were not aligned. When describing the ideal situation, all three sites were in alignment.

The second research question, RQ2, sought to identify the reasons behind key differences between the sites in how they prioritize aspects of software quality. The results highlighted a number of factors that can reduce the level of alignment between sites in how they understand software quality. The reasons identified in the case study presented are:

- **Insufficient access to key stakeholders** leads to mismatched expectations.
- **Time zone differences** limits casual and synchronous communication.
- **Late reviews and testing** leads to late identification of software quality issues.
- **Insufficient domain knowledge** limits the ability for key stakeholders to understand what is required.
- **Inadequate documentation** limits the ability for key stakeholders to understand what is required.
- **Failure to document and communicate changes** means they are not reflected in all parts of the product development.
- **Unexpected, unmanaged cultural differences** leads to poor communication.

While each of these issues has been identified more generally as problematic in GSD settings, this study explicitly links these issues to problems with issues of
software quality. Further this study highlights that companies employing GSD practices continue to face these challenges despite the work done in this area.

All of the reasons identified for reduced alignment on issues of software quality could have been more effectively managed. Project leaders in a GSD setting should ensure:

- Communication channels are open and key personnel are available at appropriate times. This may require locating key personal at other sites for critical stages of the product development.

- Reviews and tests are carried out regularly to ensure a common understanding and that expectations are being met.

- The staff selected to carry out specific activities have the knowledge and training required, or are given the opportunity to acquire these skills.

- The documentation is sufficient given the knowledge, understanding and skill of the stakeholders who will need to use it.

- All changes are communicated to correct parties in a timely manner.

- Cultural differences are expected and managed.

While this work reinforces and extends the research identifying challenges in GSD, it highlights the need for greater support for organizations to overcome these challenges. Further, the organizational setting in this paper is very specific, and work needs to be done to understand if other GSD settings are better or less able to support alignment on software quality. The method applied in this paper was found to be successful for meetings these aims, and could be reused in this future work.
Chapter 9

Balancing Software Product Investment Options

This chapter is published as:


Abstract

The long-term sustainability of a software product depends on more than developing features. Priorities are placed on aspects that support the development of software, like software product quality (eg. ISO 9126), project constraints – time and cost, and even the development of intellectual capital. A greater focus on any one aspect takes priority from another, but as each aspect delivers a different type of value managers have trouble comparing and balancing these aspects. This chapter presents a method to help determine the balance between key priorities in the software development process. The method is applied to a new case study, that also combines with results from previous studies. The results show it is possible to compare features, quality, time, cost and IC in a comprehensive way, with the case study showing that participants perceive a change from a shorter-term product perspective to a longer-term organisation.
beneficial to the business.

9.1 Introduction

The value of developing features for a software product is clear – if a product fails to satisfy the needs of its users the product is worthless. But developing successful and sustainable software requires investment in more than just features, with other investment types requiring time and funding. XP identifies four key areas to control in the development of software (Beck, 2000) – features, quality, time and cost. However, given that software development is human-intensive, it is crucial to also include intellectual capital (IC) when it comes to the investment in and evolution of a software organisation. These areas represent investment types – features, software product quality and IC, and project constraints – time and cost.

Ultimately software development managers must be able to balance these investment types and constraints against one another, as an over-investment in any one of these areas will come at the expense of the others – as shown in Figure 9.1. But people in the software development industry have trouble comparing the importance of the aspects that make up these areas (Chapter 5). A hypothetical example would be to ask people to state the relative importance of security, employee satisfaction and delivering on time, which is sure to bring confused faces.

Balancing investment types and constraints is difficult because the investments are intangible, and the aspects being examined are interrelated. For example, the act of writing code has the benefit of training the developer – providing benefits both in terms of the feature-base of the software product and the skills of an employee. Managers are interested in knowing the return on investment to be derived from a software product and/or software process improvement, but the value delivered by each investment type differs in who it impacts and how it impacts them. This means value cannot only be considered in monetary terms as there are temporal and human aspects, which make it much harder to make direct comparisons between different types.

Despite any difficulties software development managers may have, they must still make decisions that impact the balance of investment types and constraints. This chapter presents a quantitative empirical case study, which combines the results from previously completed studies in the areas of software product quality using ISO 9126 (Chapter 5) and IC (Barney et al., 2009a) with new data. The extra step presented in this chapter allows the previous results to be combined
and compared in a comprehensive way, through a single list that shows the relative importance of the aspects that make up features, quality, time, cost and IC. This is the first study to combine these aspects in such a way that they can be compared on a detailed level to the best of the authors’ knowledge. This method is used to both describe the current situation within the case study, and any changes the participants perceived ought to occur.

In this chapter Section 9.2 presents an overview of related work. Section 9.3 details the research questions and the method used to them. A case study is presented in Section 9.4, with a discussion of the results and method in Section 9.5. Threats to the validity are discussed in Section 9.6. Finally conclusions are drawn in Section 9.7.

## 9.2 Background

This section examines the investment types and constraints in more detail, with particular focus on quality and IC. Reasons why people have trouble balancing these concerns are discussed along with the reasons why it is important to get this balance right.
9.2.1 Investment Types and Constraints

Literature from XP identifies four variables that need to be controlled in the software development process—features, quality, time and cost (Beck, 2000). The features are described in terms of functional requirements, while quality is described as non-functional requirements. Constraints on development project time and cost are used to ensure that an economically feasible product is delivered at an appropriate time. However, this view only presents a project and product perspective of software development.

From an organisational perspective, IC must also be considered as an important investment type in the development of sustainable software products, as it is both the key input and tool used in the development of software (Roos and Roos, 1997). IC covers a range of issues from employee training and satisfaction to organisational processes and the knowledge held in the relationships with existing customers.

Thus features and quality present the product perspective; time and cost come from the project perspective; and IC represents the longer-term organisational perspective.

This chapter focuses on software product quality and IC in a greater level of detail than the other attributes, as the body of literature covering the prioritisation and selection of features is considerably more established and the project constraints—time and cost—are much more basic. The chosen areas of interest also fall within the scope of the line management position, supporting change within small and focused parts of the organisation. These areas are also of key interest to the authors’ industrial partners. Other investment types in the development of software products, such as the location of office space, are outside the scope of this chapter.

In most software development organisations resources are limited. This means funding one type of investment will come at the expense of one or more other types of investment, as shown in Figure 9.1. For example, to deliver a solution on time and budget, with a desired level of quality, it is a common method to exclude the least important features from the release plan.

While it is known that people have trouble comparing the relative importance of these different investment types against one another (Chapter 5), these are important decisions that managers must face. This only makes understanding the nature of these choices and decision support even more important.
9.2 Background

9.2.2 Software Product Quality

Quality is a complex multifaceted concept. Definitions will vary between people, their relationship to the product for which they are describing quality, and the context.

Multidisciplinary research has identified five perspectives from which quality can be described (Garvin, 1984):

- The *transcendental perspective* defines quality as something that can be recognised but not defined in advance.
- The *user perspective* defines quality as fit for purpose.
- The *manufacturing perspective* defines quality as conformance to specification.
- The *product view* defines quality in terms of essential characteristics of the product in question.
- The *value-based view* defines quality in terms of the amount a customer is willing to pay for it.

The *user* and *manufacturer* views are the most commonly taken in the development of software (Hoyer and Hoyer, 2001; Kitchenham and Pfleeger, 1996). But the increasing body of value-based software engineering (VBSE) literature recognises the importance of taking advantage of all perspectives involved in software development (Kitchenham and Pfleeger, 1996). Central to VBSE is Theory-W (Boehm and Ross, 1989), which states that success requires all of the success-critical stakeholders to compromise. Similarly requirement specification reading techniques that take advantage of different perspectives have been found to catch 35% more defects than non-directed alternatives (Basili, 1997; Boehm and Basili, 2001).

The concept of software quality also changes on the context in which it exists (Kitchenham and Pfleeger, 1996). Software quality must be planned to allow a development company to meet its business objectives. This means that different levels of software quality may be acceptable for software product offerings, just as there is a range of quality when it comes to cars. Less than perfect software quality may be ideal (Yourdon, 1995), but deciding how much less than perfect can only be decided in a given business context (Kitchenham and Pfleeger, 1996).

There are many models that seek to define software product quality. Some examples of these models are McCall’s quality model, Boehm’s quality model, Dromey’s quality model and ISO 9126.
The first of the modern software product quality models was that of McCall (Kitchenham and Pfleeger, 1996). The model uses a hierarchy of factors, criteria and metrics to address internal and external product quality. Eleven factors define an external or user perspective of quality. Each of these factors is linked to between two and five of 23 criteria that define an internal or development perspective of quality. Further metrics are associated with the factors allowing quality to be measured and managed.

McCall’s quality model was closely followed by Boehm’s quality model (Kitchenham and Pfleeger, 1996). Both models present product quality in a hierarchy, but Boehm’s model has three high level characteristics linked to seven intermediate factors, which are in turn linked to 15 primitive characteristics. Boehm’s model has a wider scope than that of McCall’s, with more emphasis on the cost-effectiveness of maintenance (Milicic, 2005).

More recently work has been done to create an international standard for software product quality measurement – ISO 9126 (2001). This standard is again organised in a hierarchy with six characteristics at the top level and 20 sub-characteristics with indicators used to measure the sub-characteristics. In addition to aspects of internal and external quality, covered by McCall’s and Boehm’s models, ISO 9126 includes quality characteristics of functionality (Milicic, 2005). Internal, external and functional qualities are also mixed at all levels of the hierarchy. However, ISO 9126 does not clearly state how quality should be measured (Kitchenham and Pfleeger, 1996).

None of the three models discussed present a rationale for the selection of characteristics to be included in the quality model and it is not possible to tell if a model presents a complete or consistent definition of quality (Kitchenham and Pfleeger, 1996). Further the placement of items are not motivated in ISO 9126, with no justification as to why Interoperability is not related to Portability, for example.

Dromey’s model attempts to address some of the issues presented with the other models and support developers achieve software product quality (Dromey, 1996). Dromey states that it is impossible to build high-level quality attributes like reliability or maintainability into a product, but developers must instead build properties that manifest in achieving these goals. The distinction this model makes is important, as using it will verify that it allows the quality required to be achieved (Kitchenham and Pfleeger, 1996). Before Dromey’s model can be successfully applied, the various groups involved in the development of a software product must agree on what quality attributes should be achieved and to what level. This process can be supported using other models.

The investment options in software development are more complex than
9.2 Background

quality, and any quality model must be considered in terms of other options, like IC.

9.2.3 Intellectual Capital

The origin of the IC concept can be traced to the balanced scorecard, developed by Skandia – a multinational insurance and financial services company based in Stockholm. Skandia proposed using IC as a management tool for the first time, with an objective to better manage intangible assets when creating further sustainable value for the organisation (Bucklew and Edvinsson, 1999). Dow Chemical collaborated with Skandia to define the components of IC in terms of human capital, organisation capital and customer capital (Jashapara, 2004).

Despite attempts to develop a general model of IC, organisations tend to customise models to suit their own context (Han and Han, 2004).

Four aspects of IC are defined by Brooking (1998): market assets, human centre assets, intellectual property assets and infrastructure assets. Lundqvist (1997) identifies intangible assets in terms of competence and relational resources. Sullivan (1998) develops a model based on human capital and defines human capital as the capabilities of stakeholders which is supported by structural capital – for example computers and information systems.

Bontis (2001) divided IC into human capital, relational capital and structural capital with several indicators that each described. Human capital is the capabilities of individuals who provide solutions to customers. These include knowledge, experiences, skills and abilities of employees, combined human ability to solve business problems. Structural capital refers to the structures and processes within the organisation that meets market requirements – for example patents, trademarks, information systems. Relational capital refers to an organisation’s relations with stakeholders including customers, suppliers and public. This model is one of the most frequently used models by practitioners and academics.

Previous research and business practices have led to various models and indicators of IC (Sveiby, 1998; Sullivan, 1998; Edvinsson, 1997).

There are several techniques, systematic processes and models used to measure IC of organisations. These include balanced scorecard, relative value, competency models, subsystem performance, benchmarking, business worth, business process auditing, knowledge bank, brand equity valuation, calculated intangible value, micro-lending, colorised reporting (Montague, 2008).

Over the years, several methods of measuring IC have also been developed. For example Skandia Navigator, Intellectual Capital Services’ IC Index™ and
Philip M’Pherson’s Inclusive Valuation Methodology (IVM™) (Housel and Bell, 2001). The starting point of every method is the identification of intellectual assets and grouping of these into categories. For example, Skandia’s Navigator model includes 112 indices of IC whereas Edvinsson and Malone measure IC by developing 140 indicators, using four perspectives, namely financial, customer, human, and renewals and development (Housel and Bell, 2001).

While many researchers have examined attributes of IC, there appears to have been no systematic attempt in the intellectual, structural or human capital literature to list the attributes that constitute these areas of study until very recently (Moon and Kym, 2006; Carson et al., 2004). Most researchers have only examined aspects of these areas, recognising any list of attributes as incomplete. Moon and Kym (2006) noted without a clear and comprehensive framework for IC, managers are likely to lack the detail required to effectively manage their organisations’ IC, so they created a model of IC by synthesising the attributes and model fragments from many studies. This model is used in this chapter.

Measuring IC has several benefits. For example it allows companies to assess the risk present and identify areas to develop and improve, it provides a systematic approach when comparing several units within a company or companies, it provides very useful information on companies’ future potential and helps providing a comprehensive company report.

Benefits can be direct, indirect or long term (Cranfield, 2004). Direct benefits improve financial performance of the organisation. Indirect benefits are related to changes in elements of performance which may be beneficial for the company – for example motivating staff members which allows managers to be more productive or increase in code reuse or reduction in testing time. Long term benefits include an improvement of the relationships within a company.

Although measurement of IC has several benefits to a company, this is a costly process because of the time needed to collect the data, analyse it and take actions on those measurements. In order to lower the cost, companies may need to automate the measurement and cut down on the number of people involved in the process as in many cases benefits achieved are difficult to quantify (Cranfield, 2004).

Additionally, the body of knowledge of investment in IC in a software engineering context is limited. The authors found that the majority of literature covers investment issues in software product line practice, and the impact of upfront investment on cumulative return in terms of money, effort and time (Hetrick et al., 2006). Managers need to be able to understand the balance of the various aspects they are managing in order to make informed decisions.
9.3 Methodology

This section presents the research questions, and describes the method to be used to answer them. The approach taken in this chapter to answer the research questions is presented in two ways. This section presents the general method, which can be applied to all cases. Section 9.4 presents the actual steps taken in the case study presented in this chapter.

9.3.1 Research Objectives

Finding the right balance between investment types and constraints is critical for the long-term success of all software development organisations. The major investment types come from the product perspective, with issues of features, quality; and from the organisational perspective with issue of IC; while time and cost represent the major project perspective constraints. Too much focus on any one of these areas will result in an under-investment in other areas as they compete with each other, as shown in Figure 9.1.

Getting people to identify the balance between investment types and constraints is a difficult task, but an essential one if these investment options and constraints are going to be successfully managed (Chapter 5).

Previous research by the authors has sought to find the balance between aspects of individual investment types – from a product perspective aspects of software product quality were studied using ISO 9126 (Chapter 5) and from an organisational perspective aspects of IC were studied (Barney et al., 2009a). However, these aspects must ultimately be understood and balanced together. The research presented in this chapter aims to bring these previous studies together with other feature and project perspective constraints, as shown in Figure 9.2, to understand how these areas interrelate in terms of importance. The areas covered in the case studies are discussed in more detail in Section 9.4, with a complete listing of the areas in Figure 9.4.

Thus the research questions addressed in this chapter are:
RQ1: Is it possible to identify the relative importance between aspects of *features*, software product *quality* using ISO 9126, the project constraints *time cost*, and *IC* for a given situation?

If it is possible to find the relative balance between this aspects, then it is of interest to ask:

RQ2: What is the balance between aspects of *features*, software product *quality* using ISO9126, the project constraints *time cost*, and *IC* for a given situation today?

However, being able to identify the current balance of investment types does not ensure that the correct balance has been achieved. Thus it is also important to identify what is perceived as the ideal balance for the current situation faced in an organisation. This is addressed by the third research question:

RQ3: What is the perceived *ideal* balance between aspects of *features*, software product *quality* using ISO9126, the project constraints *time cost*, and *IC* for a given situation today?

### 9.3.2 Approach

To address the research questions a method has been developed, expanding on previous work by the authors in *quality* (Chapter 5) and *IC* (Barney et al., 2009a). The previous studies were conducted to determine the level of alignment between key stakeholder groups over the priorities of aspects of *quality* and *IC* respectively, however, they also generated lists showing the relative importance of the aspects related to each study.

The method presented in this section expands on the work already done, to combine the previous results with new data to answer the research questions. While details of the case study are presented in Section 9.4, this section details a more general methodology to answer the research questions.

The process to determine the balance between different investment options and constraints in this chapter has five phases:

1. First identify the investment types and constraints that should be covered, and develop the model.

2. Then associate stakeholder groups to the different parts of the model to identify who should participate in the study.
3. Put together the questionnaires for eliciting the balance of investment types and constraints.

4. The results are analysed.

5. Finally hold a workshop with key stakeholders to develop a deeper understanding of the results.

The following sections explains each of these phases in more detail.

### 9.3.3 Investment Types and Constraints

The investment types and constraints vary depending on both the organisation and roles studied. Thus it is important that any study tailor the attributes for the context being studied. Using one or more of the quality models and IC models introduced in Section 9.2 is a good starting point, but company specific needs have to be taken into account, as illustrated in the case study in Section 9.4.

The selection and refinement of models to represent the investment types and constraints should be a consultative process between academia and industry to ensure that a representative and meaningful result is obtained. A workshop or series of workshop should be run to ensure the terms identified provide sufficient coverage and are clearly defined from the perspective of those participating in the study and those using the results.

However, it is important that the model developed is hierarchical in order to take advantage of the methodology presented in this chapter, or flat for a simple case. As an example, the model of investment types and constraints used in the case study presented in this chapter can be seen in Figure 9.4. This model is discussed in more detail in Section 9.4.

### 9.3.4 Stakeholders

When considering a range of investment options and constraints, it is likely that no one person or group will have detailed oversight of all of the areas of interest. Managers may be able to provide insight on high-level priorities, but not be able to go into the detail about technical qualities. Similarly a developer may not need to be aware of all of the external stakeholder groups with whom the company has a relationship.

This is where the value of the hierarchy comes into play. It is possible to ask different groups of people to be involved in the prioritisation of different parts of
the hierarchy, as the method then allows the separate responses to be brought together.

Participants should only be responsible for prioritising parts of the hierarchy for which they have understanding and oversight.

### 9.3.5 Questionnaire

Now it is possible to develop a questionnaire or series of questionnaires to answer the research questions. As discussing different parts of a hierarchy is confusing in the abstract, a *running example* will be used in the remainder of this methodology section, based on the model of investment types and constraints in Figure 9.3.

```
<table>
<thead>
<tr>
<th>Features</th>
<th>Quality</th>
<th>IC</th>
<th>Time</th>
<th>Cost</th>
</tr>
</thead>
</table>
```

Figure 9.3: Example model of investment types and constraints

The hierarchical cumulative voting (HCV) method (Berander and Jönsson, 2006) is used to elicit the relative importance of the investment option and constraint. This method has been found useful when asking people to make comparisons between areas in which they would normally have trouble (Chapter 5), as it breaks the problem up into a series of simpler tasks. It also allows different people to respond to different parts of the model of investment types and constraints.

HCV is made up of a number of cumulative voting (CV) tasks. CV asks participants to spend 1000 points across different sets of aspects to represent their relative influence. Using the top level of the hierarchy in Figure 9.3, for example, if a participant thought *IC* does not at all matter today and *features* was twice as important as *quality* they might award these qualities zero, 200 and 100 respectively with 700 points distributed amongst the remaining criteria.

Each CV task should contain a logical group of attributes at the same level of abstraction. The master CV task asks participants to compare the top level of the hierarchy. In the example, there would be two CV exercises – the one comparing the three types of *IC* and the master CV exercise asking participants to compare the five items in the top level of the hierarchy.
9.3 Methodology

9.3.6 Analysis

Analysing the results of the HCV task will answer the research questions.

The HCV task is analysed in two stages. The first stage looks at each individual CV task to see the relative weight and priority given to each option. The second stage is to combine the results of the individual HCV tasks into a single prioritised list of attributes, allowing comparisons between the CV tasks.

For the first stage, it is possible to average the points awarded to each aspect. Then each CV task can be analysed individually considering the rank order of the aspects covered and the range of points awarded to the CV set.

The second stage involves using the results of the first stage and converting the sets of CV tasks making up the HCV task into a single list of aspects, with each aspect assigned its own relative weighting. Remembering that HCV has a master CV task, and for each aspect in that master CV task there may be a related set of sub-aspects in their own CV task. The example in Figure 9.3 shows a master list containing features, quality, time, cost and IC; with another CV task asking participants to prioritise between three aspects that make up IC.

This set up allows the number of points awarded to each aspect in the master list to be multiplied with the number of points awarded to each aspect in the related CV task (Berander and Jönsson, 2006). To ensure that aspects from CV tasks with many options are not underrepresented and that CV tasks with few aspects are not over-represented it is also necessary to multiply each of these results by the number of aspects from the non-master CV task. Finally the set of numbers for each aspect can be scaled so that the sum is 1000.

For example, if 200 points are awarded to IC and 400 points are awarded to human capital, then:

\[ IC \times human \_capital \times number \_of \_aspects \_of \_IC \]

\[ = 200 \times 400 \times 3 \]

\[ = 240,000 \]

The scaling of this result then depends on the other values, but if the other values were to sum to 4,800,000 then the number would be scaled to:

\[ result \div total \_sum \times 1000 \]

\[ = 240,000 \div 4,800,000 \times 1000 \]

\[ = 50 \]
It is possible to deal with aspects in the master CV task with no related CV task, such as features in the running example. In this case create a dummy CV task that contains one item with the same name, and award it 1000 points.

9.3.7 Review

Finally the results should be reviewed with the participating organisation in order to gain a deeper understanding of what the results mean and the reasons behind them. This can be done in a workshop setting.

9.4 Case Study

This research has grown from two previous case studies that each looked at the relative importance of different aspects related to one investment types – quality (Chapter 5) and IC (Barney et al., 2009a) – as shown in Figure 9.2. While the original intent of these studies was to determine the degree to which priorities across the organisation were aligned and working towards a common goal, the results generated lists showing the relative importance of the various aspects that make up each investment type.

There was a desire, both academically and from within the industrial partner, to join these studies together to understand the relative importance of both sets of investment options together with features, time and cost. This is the aim of the study presented in this chapter, and involved the collection of additional data and an expansion of the methodology to join these areas together.

This case study was conducted during autumn 2008 for one product at Ericsson. Ericsson is a world leading company in telecommunication, providing a wide range of products and solutions. Products are developed and sold as generic solutions offered to an open market, although customised versions of the products are also developed.

9.4.1 Attribute Selection and Definition

In order to both validate the work previously done and be able draw greater understanding from the research it was decided to reuse the models from the previous research. But there was a desire to understand these concepts in terms of features and the project constraints time and cost, as previously discussed. Combined these areas represented the major concerns of line managers within
our case study and formed the base of the model of investment types and con-
straints, and to the best of the authors knowledge is the first time researchers
have attempted to find the balance between these aspects in a software engi-
neering context.

Interview time constraints meant it was not possible to use the models ex-
actly as presented in the previous studies as combining all of these elements
resulted in too many aspects to consider, however, as the models used are hier-
archical it is possible to move to a higher level of abstraction.

The first author created a new model of investment types and constraints
based on the control variables for XP (Beck, 2000) and the models created for
the previous research into quality (Chapter 5) and IC (Barney et al., 2009a).
A workshop was then held with people identified by Ericsson to ensure the new
model covers the issues of interest to the organisation, makes sense, and can be
understood. Some of the definitions were revised, but no major changes were
made to the model.

Figure 9.4 shows the model of investment types and constraints used across
the three studies that make up this research. This study asked participants to
complete the master CV task – comparing features, qualities, time, cost and
IC. However, as this study was conducted several months after the first two
studies, two additional CV tasks were added to check the priorities had not
changed. These consist of the aspects describing quality and IC that are bolded
in Figure 9.4.

The first study asked participants to mark the relative importance of the
detailed quality aspects, which are shown in the diagram without bolding. Sim-
ilarly the second study asked participants to mark the relative importance of
the detailed IC aspects, which are also shown in the diagram without bolding.

A questionnaire was developed based on the bolded sections of this model for
the third study. One CV task asked participants to compare aspects of IC, the
second asked participants to compare aspects of software product quality and
the final CV task asked participants to compare features, quality, intellectual
capital, time and cost. The final version of the definitions of the terms used in
the model can be found in the appendix.

9.4.2 Participant Selection

The authors were supported by the R&D management to select and identify
participants for this case study. It was decided that people in line management
positions would be able to best address the areas of greatest interest to the
company from this study. This group also has the greatest control over the
### 9.4.3 Additional Analysis

The results of the IC study Barney et al. (2009a) contained several aspects of management that are outside the scope of this study as they are not directly related to the balance being studied. However, this limited the number of possible participants as few people hold this role for the product studied. In total nine participants were selected to take part in the case study. The first author conducted one-on-one interviews with each participant to collect the data. All responses were used in the analysis.

However, as this study combines the results of the previous studies, the number of participants and their roles is presented in Table 9.1.
related with IC. In order to manage this situation the aspects of management were removed. The relative weights of the remaining aspects were scaled to 1000 using the same process used in the final stages of the method described in Section 9.3.

9.4.4 Results

While confidentiality prevents the prioritised list of attributes being published, the results are described and discussed in this section.

The results were fairly consistent between the participants for the additional data collected as part of the combined and extended study, with one exception that is discussed in this section.

Comparison to Previous Studies

This research aims to bring the results of three studies together. As there were some months between the collection of data for each study, some additional data was collected about aspects of quality and IC to ensure that the situation had not changed substantially between data collection points.

The quality study collected data about the 18 aspects of quality listed at the top of Figure 9.4. This study collected data about the six aspects of quality listed in the same figure in bold. The results of both studies showed strong parallels for both what people perceive is happening today and what people perceive should be happening in an ideal situation.
Similarly it was possible to make a comparison between this study and the IC study based on the three main areas of areas of IC shown in bold at the bottom of Figure 9.4, as both studies used HCV in similar ways. The averaged results for line managers in both studies placed these three aspects in the same order in both the situation today and the ideal situation.

**Overall**

Looking first at the prioritisation of the CV group containing features, quality, time, cost and intellectual capital there are a couple of interesting results. These results are presented in Table 9.2.

<table>
<thead>
<tr>
<th>Today</th>
<th>Ideal</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Features</td>
<td>1. Time</td>
</tr>
<tr>
<td>2. Time</td>
<td>2. Features</td>
</tr>
<tr>
<td>3. Quality</td>
<td>3. Intellectual Capital</td>
</tr>
<tr>
<td>4. Cost</td>
<td>4. Quality</td>
</tr>
<tr>
<td>5. Intellectual Capital</td>
<td>5. Cost</td>
</tr>
</tbody>
</table>

The results show that success is most dependent on delivering features within some constraint of time, with these two aspects coming out most important in the situation as it is perceived today, and how the participants would like to see the situation today.

However, some change was perceived beneficial to the situation today in the perceived ideal situation. The participants would ultimately like to see a stronger focus on intellectual capital. Additionally the participants would like to see a more equal focus on the different aspects, with a difference of 196 points between the most and least important aspects today, and a difference of 41 points in the perceived ideal situation.

It should be noted that participants were not consistent with their prioritisation of all five aspects, with quality being placed anywhere between most important to least important by different participants. The level of agreement on the other aspects in this group was much higher for the situation today. While the level of consensus was lower in how the participants prioritised what should be happening in the ideal situation today, the differences between the aspects studied was quite small.
Detailed Priorities

Using the method described in Section 9.3 the three CV results from the three studies were used to create a single list covering 36 aspects of investment types and constraints – features, all 18 aspects of quality, time, cost and all 17 aspects of IC. These can be seen in Figure 9.4.

In describing the situation today, the results show a strong product perspective, although the project perspective is important with respect to time. The top 10 aspects consist of eight different aspects of quality, features and time – suggesting the organisation likes to deliver functional requirements to schedule, with a certain set of quality requirements being met. The remainder of the list evenly across the different aspects, with no investment type group standing out.

It is worth noting that the two most important aspects of IC today represent the customer, with customer (operator) and customer (service provider). The third most important aspect of IC today is organisational process.

Participants were also asked to complete the tasks to show what they think should be happening today, in their perceived ideal situation. These results show no clear preference for any one investment type except features, which remains with a relatively high priority.

Ideally there are aspects of both quality and IC at the top, middle and bottom of the prioritised list. Looking at the difference in ranking between the situation today and the perceived ideal situation there are a number of interesting results:

- All but one of the aspects that participants identified should have a higher rank in the ideal situation were aspects of IC. In particular they were mostly aspects of human capital and relationship capital.

- A greater focus on the end user is one of the biggest changes desired.

- The aspects that fell the greatest number of places were from quality, but to a more limited extent features and time also fell.

- One aspect of IC fell a notable number of places. It was organisational process.

The results also show that the participants would ideally like the aspects to be more equal in their relative priority. In describing the situation today the most important aspect received 57 points and least important aspect received nine, giving a difference of 48 points. However, in the ideal situation there were only 27 points between the most and least important aspects as everything was brought closer together.
9.5 Discussion

The method presented in Section 9.3 appears to be robust. The method was able to combine priorities from the product, project and organisational perspectives – showing the relative importance of different aspects that make up features, quality, time, cost and IC.

The results of the case study show a strong product perspective within the organisation, with features and quality coming out as the most important investment types today. However, there is still a strong customer focus in terms of the priorities within IC.

Ultimately the participants in the case study would like to see a stronger organisational perspective taken, with a much greater emphasis on aspects of human capital and relationship capital. This means taking a longer-term focus when balancing the investment types and constraints against one-another, and being able to deliver value to customers both now and in the future. This result can be partially explained by a restructure within the organisation studied and the economic downturn. Both of these events act to make people feel less secure about their jobs, and the results show a desire for greater security and respect.

The one aspect of IC that bucked the trend set by all other was organisational process. Ideally the participants thought this should be less important. However, the organisation studied had also come to the same conclusion, and reducing processes and decision points was one of the major goals of the restructuring.

The perceived ideal situation remained customer focused, but saw a big increase in the importance of the end user. The end user in this case Ericsson’s customers’ customers.

The results of the third study, involving line managers, was interesting in that no major changes were proposed by the participants. This result is not entirely unexpected from a group of people in line management positions. This result is supported by the garbage can model (Cohen et al., 1972), which found that managers try to address issues with change, rather than solutions, and prefer to take smaller steps rather than bigger ones.

In the case study as it was described today there were 48 points between the most and least important aspects. The description of the perceived ideal situation, however, only had 27 points between the least and most important aspects. There are two possible explanation for this outcome; either the participants believe that the various aspects should be more equal in importance, or when presented with a list of important things, there is a desire to make everything more equal. This result has also been seen in previous studies (Chapter 3,
Chapter 5), but there is at least one exception (Barney et al., 2009a).

Finally, communication is critical for the groups involved in the development of software to work together effectively. Knowledge and priorities need to be shared and reconciled, as proposed by Theory-W (Boehm and Ross, 1989).

9.6 Validity Threats

While it may not be possible to generalise the results of the case study presented in this chapter to other organisations, or even Ericsson as a whole, the method shows potential for reuse in other settings. The authors’ research has found people have trouble making comparisons between aspects of different investment types (Chapter 5), however, the approach used in this chapter was able to balance aspects of IC against product and project perspective issues.

It should be noted that it is easier for the participants to agree with the set of criteria identified by the researchers than disagree in the workshops and questionnaire. This threat is partially taken care of by allowing the participants to assign a relative importance of zero to any aspect or set of aspects.

It is also difficult to know whether the respondents have understood the questions as intended and in a similar fashion to one another. This threat was partially addressed in the third stage of this study where the results were presented in the second workshop for confirmation and discussion with the researcher.

9.7 Conclusion

Being able to balance different investment types within the bounds of certain constraints is a difficult task for software managers, but one that must be undertaken on a daily basis. In this chapter the authors presented a method for determining the balance between different aspects of the software development process. The method brought together the results of three separate studies, and examined the balance between:

- The product perspective – with features and software product quality,
- The project perspective – with time and cost, and
- The organisational perspective – with IC.
The method proposed in Section 9.3 was successfully applied to a complex case, thus answering RQ1 – it is possible to identify the relative importance between aspects of features, software product quality using ISO9126, the project constraints time, cost, and IC for a given situation using the method proposed. Thus the method was also able to answer RQ2 and RQ3 for this case. That is RQ2 – What is the balance between aspects of features, quality, time, cost and IC for a given situation today? – and RQ3 – What is the perceived ideal balance between aspects of features, quality, time, cost and IC for a given situation today?

The answers resulted in a number of interesting findings:

• The organisation studied currently has a product focus in the priority placed on the aspects studied, but ultimately the participants would like to see a stronger organisational focus – with more emphasis on longer-term issues.

• The results of the study targeted at people in line management positions showed only a desire for small changes, which is in line with research into management practices (Cohen et al., 1972).

The results of this method can be used to start a dialogue within the organisation to ensure all success-critical stakeholder groups understand what activities and investments must be made in order for a company to achieve long term prosperity and sustainability. This will in turn help the different groups within the company work towards and achieve a common goal.

Going forward it is important to understand if the participants in the study have an organisational or individual perspective. While improving the work environment for employees will bring benefits to the organisation to a point, the right balance needs to be achieved between these aspects.

It is also essential to understand what stops organisations from achieving their ideal state, so that they can be supporting in reaching these end goals.
Bibliography


Åsa G. Dahlstedt and Anne Persson. Requirements interdependencies – moulding the state of research into a research agenda. In *9th International Workshop on Requirements Engineering – Foundation for Software Quality (RefsQ’03)*, pages 55–64, June 2003.


James, D. Herbsleb, Daniel, J. Paulish, and Matthew Bass. Global software
development at siemens: Experience from nine projects. In Proceedings of
27th International Conference on Software Engineering (ICSE), pages 524–
533, May 2005.

James D. Herbsleb. Global software engineering: The future of socio-technical
coordination. In 2007 Future of Software Engineering, FOSE ’07, pages 188–

Georg Herzwurm, Sixten Schockert, and Wolfram Pietsch. QFD for customer-
focused requirements engineering. Proceedings of the 11th IEEE International

William A. Hetrick, Charles W. Krueger, and Joseph G. Moore. Incremental
return on incremental investment: Engenio’s transition to software prod-
tect line practice. In Companion to the 21st ACM SIGPLAN Symposium on
Object-Oriented Programming Systems, Languages, and Applications (OOP-

Amy J. Hillman and Gerald D. Keim. Shareholder value, stakeholder manage-
ment, and social issues: What’s the bottom line? Strategic Management

Geert Hofstede. Motivation, leadership, and organization: Do American theories


Robert W. Hoyer and Brooke B. Y. Hoyer. What is quality? Quality Progress,

Ganglan Hu, Aybıike Aurum, and Claes Wohlin. Adding value to software re-
quirements: An empirical study in the chinese software industry. In S Spencer
and A Jenkins, editors, Proceedings of the 17th Australasian Conference on
Information Systems. Australasian Association for Information Systems, De-
cember 2006.


João Ramires, Pedro Antunes, and Ana Respício. Software requirements negotiation using the software quality function deployment. In Hugo Fuks, Stephan


Appendix A

Software Quality Model—Ericsson

This section details the model of software quality used for the case studies presented in Chapter 5, Chapter 6 and Chapter 7.

Features: Related to the existence of a set of functions and their specified properties; both stated and implied needs:

Suitability: The functionality is appropriate for its purpose and situation.
Accuracy: The functionality is correctly understood and implemented.
Interoperability: The system works with the other systems required.
Certification/Standardisation/Compliance: The system conforms to both standards and legislation.

Qualities: This area is made up of the following attributes:

Security: The system software is safe from danger or threat.
Scalability: Ability for the system to expand the computing solution to support a larger system load without impacting performance.
Availability/Reliability:
Maturity: The system has achieved an appropriate level of sophistication.
Recoverability: Able to regain operation after a fault or failure.
Containment/ISP/Fault tolerance: Capacity to endure adverse conditions and remain in operation.

Usability:
Learnability: The ease with which a user can gain or acquire knowledge about the system.
Understandability: Degree to which the system is as one expects.
Operability: Degree to which the system can be used.

Performance/Efficiency:
Time behaviour: The time taken to complete given activities.
Resource behaviour: The computing resources required to complete given activities.

Maintainability:
Robustness/Stability: Ability to operate without failure when there has been an unintended change as part of a maintenance activity.
Performance Management/Statistics: Ability of the system to provide statistics, logs and information to the user in order to implement a performance management process for continuous monitoring of capacity utilisation and success rates for applications, platforms and interfaces.
Analysability: Ability for the system to be examined methodically and in detail in order to understand the impact of and make changes to the system. For example, artefacts that may help with this process are system architecture diagrams, system documentation and comments in code.
Changeability: Ability for the system to be modified. This can be to complete any maintenance activity — corrective, adaptive, perfective and preventative.
Testability: Ability to confirm if an action, item, situation, etc ... is operating as desired.

Portability:
Installability: Capacity to place or fix into a new environment.
Upgradability/Replaceability: Ease with which the system can replace or be replaced by a new version of the system or a different system that performs the same task.
Configurability/Product Customisability/Adaptability: Degree to which the system can adjust to new conditions.

**Time:** Development time for a project and in particular release date.

**Cost:** Total cost for development, primarily in terms of person-hours, but also other costs.
Appendix B

Software Quality Model—Global Finance

This section details the model of software quality used for the case study presented in Chapter 8.

Features: Related to the existence of a set of functions and their specified properties; both stated and implied needs:

- Suitability: The functionality is appropriate for its purpose and situation.
- Accuracy: The functionality is correctly understood and implemented.
- Interoperability: The system works with the other systems required.
- Standardisation/Compliance: The system conforms to standards. E.g., standards in screen design.

Project Management:

- Time: Development time for a project and in particular release date.
- Cost: Total cost for development, primarily in terms of person-hours, but also other costs.

System Properties:

- Security: Attributes of software that relate to its ability to prevent unauthorized access, whether accidental or deliberate, to programs and data.
- System Accuracy: Attributes of software that bear on the provision of right or agreed results or effects.

- System Compliance: Attributes of software that make the software adhere to application related standards or conventions or regulations in laws and similar prescriptions

- Reliability: Related to the capacity of software to maintain its level of performance under stated conditions for a stated period of time:
  - Maturity: Attributes of software that relate to the frequency of failure by faults in the software
  - Recoverability: Attributes of software that relate to the capability to re-establish its level of performance and recover the data directly affected in case of a failure and on the time and effort needed for it.
  - Fault tolerance: Attributes of software that relate to its ability to maintain a specified level of performance in cases of software faults or of infringement of its specified interface

- Usability: A set of attributes that relate to the effort needed for use, and on the individual assessment of such use, by a stated or implied set of users.
  - Understandability: Attributes of software that relate to the users’ effort for recognizing the logical concept and its applicability
  - Learn-ability: Attributes of software that relate to the users’ effort for learning its application (for example, operation control, input, output).
  - Operability: Attributes of software that relate to the users’ effort for operation and operation control
  - Attractiveness

- Efficiency: A set of attributes that relate to the relationship between the level of performance of the software and the amount of resources used, under stated conditions.
  - Time behaviour: Attributes of software that relate to response and processing times and on throughput rates in performing its function.
  - Resource behaviour: Attributes of software that relate to the amount of resources used and the duration of such use in performing its function.
• Maintainability: A set of attributes that relate to the effort needed to make specified modifications.
  – Analysability: Attributes of software that relate to the effort needed for diagnosis of deficiencies or causes of failures, or for identification of parts to be modified.
  – Changeability: Attributes of software that relate to the effort needed for modification, fault removal or for environmental change.
  – Stability: Attributes of software that relate to the risk of unexpected effect of modifications.
  – Testability: Attributes of software that relate to the effort needed for validating the modified software.

• Portability: A set of attributes that relate to the ability of software to be transferred from one environment to another.
  – Adaptability: Attributes of software that relate to on the opportunity for its adaptation to different specified environments without applying other actions or means than those provided for this purpose for the software considered.
  – Installability: Attributes of software that relate to the effort needed to install the software in a specified environment.
  – Replaceability: Attributes of software that relate to the opportunity and effort of using it in the place of specified other software in the environment of that software.
  – Conformance: Attributes of software that make the software adhere to standards or conventions relating to portability.
Appendix C

Software Investment Model—Ericsson

This section details the terms and definitions used in the study into investment types and constraints in Chapter 9.

**Features:** New or changed functionality of the system. This should meet the needs of specific customers or the more general needs of the target market of the software product.

**Qualities:** This area is made up of the following attributes:

- **Security:** The system software is safe from danger or threat.
- **Scalability:** Ability for the system to expand the computing solution to support a larger system load without impacting performance.
- **Availability/Reliability:** Related to the capacity of software to maintain its level of performance under stated conditions for a stated period of time. It should have achieved an appropriate level of sophistication, endure adverse conditions and remain in operation, and regain operation after a fault or failure.
- **Maturity:** The system has achieved an appropriate level of sophistication.
- **Recoverability:** Able to regain operation after a fault or failure.
- **Containment/ISP/Fault tolerance:** Capacity to endure adverse conditions and remain in operation.
Usability: Related to the effort needed for use, and on the individual 
evaluation of such use, by stated or implied set of users. It covers 
the ease with which a user can acquire knowledge about the system, 
the degree to which it is as a user expects and the degree to which 
it can be used. This includes, for example, GUIs, CPI, training and 
alarm handling.

Learnability: The ease with which a user can gain or acquire know-
ledge about the system.

Understandability: Degree to which the system is as one expects.

Operability: Degree to which the system can be used.

Performance/Efficiency: This covers both the time taken and the com-
puting resources required to complete specified activities.

Time behaviour: The time taken to complete given activities.

Resource behaviour: The computing resources required to complete 
given activities.

Maintainability: Related to the effort needed to make specified modi-
fications to both the generic system and customer specific versions. 
The design of system should be robust, stable, analysable, changeable 
and testable. The system should also provide logs and statistics to 
allow continuous monitoring. This includes corrective maintenance 
(diagnosis and correction of errors), adaptive maintenance (modifi-
cations to interfaces to adapt to a changing environment), perfective 
maintenance (enhancements to create new or modify existing func-
tionality) and preventative maintenance (changes to improve future 
maintainability).

Robustness/Stability: Ability to operate without failure when there 
has been an unintended change as part of a maintenance activity.

Performance Management/Statistics: Ability of the system to pro-
vide statistics, logs and information to the user in order to im-
plement a performance management process for continuous mon-
itoring of capacity utilisation and success rates for applications, 
platforms and interfaces.

Analysability: Ability for the system to be examined methodically 
and in detail in order to understand the impact of and make 
changes to the system. For example, artefacts that may help 
with this process are system architecture diagrams, system doc-
umentation and comments in code.
Changeability: Ability for the system to be modified. This can be to complete any maintenance activity — corrective, adaptive, perfective and preventative.

Testability: Ability to confirm if an action, item, situation, etc ... is operating as desired.

Portability: Related to the ability of software to be installed, upgraded or replaced. This also covers the degree to which a product can be customised/configured/adapted to meet the needs on a specific clients setting.

Installability: Capacity to place or fix into a new environment.

Upgradability/Replaceability: Ease with which the system can replace or be replaced by a new version of the system or a different system that performs the same task.

Configurability/Product Customisability/Adaptability: Degree to which the system can adjust to new conditions.

Time: Development time for a project and in particular release date.

Cost: Total cost for development, primarily in terms of person-hours, but also other costs.

Intellectual Capital: This area is made up of the following attributes:

Human Capital: Human capital is held in the people employed by an organisation. It covers individuals’ education, experience, soft skills, and attitudes to work and life. It is affected by an employee’s satisfaction with their workplace. It consists of:

Employee capacity: An individual’s education, experience, tacit knowledge, competencies and soft skills. Education can be formal or informal, tacit knowledge is what people carry in their minds and is often difficult to write down, and soft skills refer to attributes like an individual’s communication skills, entrepreneurship and creativity.

Employee satisfaction: The difference between the employee’s job satisfaction and their perception of the offering. It impacts organisational commitment and the capacity for an organisation to retain staff. Many factors impact an employee’s satisfaction, for example salary, the physical work environment and how they feel fellow employees treat them.
Employee sustainability: Individuals’ attitudes to work and life. This encompasses an employee's conscientiousness and willingness to engage in lifelong learning, increasingly critical attributes in the current dynamic marketplace. As it is difficult to change an individual's attitude to work and life this issue is best addressed at the time of recruitment for a given role.

*Structural Capital:* Structural capital is described as the intellectual capital that is retained by the organisation when all the employees go home. It includes the legally protected intellectual property, organisational culture and organisational processes. This includes the shared perceptions, assumptions, understandings and ways of working in the context of the organisation.

Organisational culture: Shared perceptions of organisational practices within organisational units and shared assumptions and understandings, often at a non-conscious level.

Organisational process: Activities with defined actions that become part of an organisation's way of working and potential source of competitive advantage.

Information systems: Information systems can increase the value of intellectual capital to the business by successfully leveraging work processes and knowledge held within the organisation.

Intellectual property: This covers both legally protected intellectual capital (e.g. patents, trademarks, copyright) and other documented materials (e.g. strategy, competitive market intelligence).

*Relational Capital:* The relationships and networks that support an organisation are both formal and informal. Relational capital refers to the knowledge held in the relationships between a company and its customers (operators), customers (users), end users, suppliers, partners, community, regulators and competitors.

Customers (Service Provider): The legal entity that purchased the system. Accumulated knowledge between a company and a customer.

Customers (Operator): Expert users working at service provider. Accumulated knowledge between a company and a customer.

End user: Subscribers to the telecommunications services.

Suppliers: Relationship in supply chain management to achieve benefits — such as reduced costs, greater quality, flexibility, reliable
delivery – through methods like information sharing, technical assistance, training, direct investment.

Partners: Build reputation and industry standing through strategic alliances, collaborative relationships, business partnerships, joint ventures and industry associations. These may be used to fill a gap in a market offering.

Community: Trust based on corporate ethics, cooperation, collective action and reputation. Communities include both (i) professional, which the company can lead, share or take a more passive role; and (ii) social communities, including society-at-large.

Regulators: Intelligence and knowledge of the legal environment and lobbying skills and contacts.

Competitors: Critical understanding and intelligence about competitors — who they are and what are they doing.
ABSTRACT

Background: The software development environment is growing increasingly complex, with a greater diversity of stakeholders involved in product development. Moves towards global software development with onshoring, offshoring, insourcing and outsourcing have seen a range of stakeholders introduced to the software development process, each with their own incentives and understanding of their product. These differences between the stakeholders can be especially problematic with regard to aspects of software quality. The aspects are often not clearly and explicitly defined for a product, but still essential for its long-term sustainability. Research shows that software projects are more likely to succeed when the stakeholders share a common understanding of software quality.

Objectives: This thesis has two main objectives. The first is to develop a method to determine the level of alignment between stakeholders with regard to the priority given to aspects of software quality. Given the ability to understand the levels of alignment between stakeholders, the second objective is to identify factors that support and impair this alignment. Both the method and the identified factors will help software development organisations create work environments that are better able to foster a common set of priorities with respect to software quality.

Method: The primary research method employed throughout this thesis is case study research. In total, six case studies are presented, all conducted in large or multinational companies. A range of data collection techniques have been used, including questionnaires, semi-structured interviews and workshops.

Results: A method to determine the level of alignment between stakeholders on the priority given to aspects of software quality is presented—the Stakeholder Alignment Assessment Method for Software Quality (SAAM-SQ). It is developed by drawing upon a systematic literature review and the experience of conducting a related case study. The method is then refined and extended through the experience gained from its repeated application in a series of case studies. These case studies are further used to identify factors that support and impair alignment in a range of different software development contexts. The contexts studied include onshore insourcing, onshore outsourcing, offshore insourcing and offshore outsourcing.

Conclusion: SAAM-SQ is found to be robust, being successfully applied to case studies covering a range of different software development contexts. The factors identified from the case studies as supporting or impairing alignment confirm and extend research in the global software development domain.