

Perspectives on Software and their Priorities: Balancing Conflicting Stakeholder Views

Sebastian Barney

Blekinge Institute of Technology Licentiate Dissertation Series
No 2009:07

Perspectives on Software and their Priorities: Balancing Conflicting Stakeholder Views

Sebastian Barney



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

© 2009 Sebastian Barney
Department of Systems and Software Engineering
School of Engineering
Publisher: Blekinge Institute of Technology
Printed by Printfabriken, Karlskrona, Sweden 2009
ISBN 978-91-7295-164-8
Blekinge Institute of Technology Licentiate Dissertation Series
ISSN 1650-2140
urn:nbn:se:bth-00440

Nowdays people know the price of everything and the value of nothing.
Oscar Wilde (1854 – 1900)

Abstract

The sustainable development of a software product depends on a number of groups working together to achieve a common goal. However, each of the groups interacts with the product in different ways, and can have conflicting aims and objectives. For example, developers trying to correct issues in the software architecture, which will impact future releases of the product, can be stopped by a project manager who is charged with delivering a release on time and within budget. While the functional requirements of a software product are usually documented, there are a number of other investments in software development that are not always as explicitly agreed upon but are still essential to a product's long-term success. The major investment types include software product quality – a main control variable in software development, and intellectual capital (IC) – being the key input and tool used in software development.

As management requires measurement, it is necessary understand the priorities placed on investment options by the various groups involved in the development of a software product. The objective of this thesis is to develop a method capable of both determining the priorities of different groups, and the level of alignment between these groups in terms of their priorities.

Evolving the method from a study into the values used to select requirements for a release of software, Ericsson supported the development of a methodology to determine and compare the priorities of different groups for software product quality, and IC. The method elicited the required information from a series of case studies to build up a picture of the priorities placed on major investment options and constraints – features, quality, IC, time and cost. The results highlight strengths, and areas for improvement – through the identification of differing priorities and ambiguities in management of different aspects studied.

In conducting this research, systematic biases in the selection of requirements appear to be occurring, adding an objective to understand how bias impacts decision making in a requirements engineering context.

This thesis provides a method that determines the priorities on the level of investment on different options in the development of software products. It is concluded that people involved in the development of software need to be aligned on issues of software product quality as these priorities set expectations. The same was not found true for issues of IC, where groups can complete tasks without negatively impacting others, as long as the organisation works effectively as a single entity. On the issue of biases in the prioritisation of these aspects, prospect theory is found to apply to requirements selection in an academic experiment – suggesting people will prefer functionality over software product quality, and to meet the known requirements of customers over predicting general market requirements.

Acknowledgements

First I would like to sincerely thank my supervisor, Prof. Claes Wohlin, for his invaluable feedback, expertise and advice. Despite Claes' all too 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, Dr. 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 our research activities.

Recognition must also be given to my colleagues in the SERL group for creating a positive, supportive and enjoyable research environment. I would like to extend special thanks to my collaborators – Nina Fogelström, Mahvish Khurum and Dr. Anders Hederstierna, and my co-supervisor, Dr. Mikael Svahnberg.

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 his support, and both putting up both with me and without me.

This work was partly funded by *The Knowledge Foundation* in Sweden under a research grant for the project Blekinge Engineering Software Qualities (BESQ).

Overview of Papers

Papers in this Thesis

Chapter 2: Sebastian Barney, Aybüke Aurum, and Claes Wohlin. A Product Management Challenge: Creating Software Product Value through Requirements Selection. *Journal of Systems Architecture*, 54(6):576–593, 2008.

Chapter 3: *An extended version of* Sebastian Barney and Claes Wohlin. Software Product Quality: Ensuring a Common Goal. *Proceedings of the International Conference on Software Process (ICSP)*, May 2009.

Chapter 4: Sebastian Barney and Claes Wohlin. Balancing Software Product Investments. Submitted to *Empirical Software Engineering and Measurement (ESEM) Conference*, Lake Buena Vista, Florida, USA, October 2009.

Chapter 5: Nina D. Fogelström, Sebastian Barney, Aybüke Aurum, and Anders Hederstierna. When Product Managers Gamble with Requirements: Attitudes to Value and Risk. In *Proceedings of the International Workshop Conference on Requirements Engineering: Foundation for Software Quality (RefsQ)*, Amsterdam, Netherlands, June 2009.

Related Papers not in this Thesis

- Paper 1:** Sebastian Barney, Aybüke Aurum, and Claes Wohlin. Quest for a Silver Bullet: Creating Software Product Value Through Requirements Selection. *EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA)*, Cavtat, Croatia, pages 274–281, August 2006.
- Paper 2:** Sebastian Barney, Ganglan Hu, Aybüke Aurum, and Claes Wohlin. Creating Software Product Value in China. *IEEE Software*, 26(4), July–August 2009.
- Paper 3:** Sebastian Barney, Aybüke Aurum, and Claes Wohlin. The Relative Importance of Aspects of Intellectual Capital for Software Companies. Submitted to *EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA)*, August 2009.

Contents

Abstract	vii
Acknowledgements	ix
Overview of Papers	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	4
1.3 Research Questions	12
1.4 Methodology	13
1.5 Research Setting	17
1.6 Overview of Chapters	19
1.7 Contribution of Thesis	23
1.8 Future Work	25
1.9 Summary	26
2 Understanding Software Product Requirements Selection	29
2.1 Introduction	30
2.2 Background	31
2.3 Methodology	37
2.4 Data Analysis and Results	43
2.5 Discussion of Results	59

2.6	Validity Threats	64
2.7	Conclusions and Future Study	66
3	Balancing Software Product Qualities	69
3.1	Introduction	69
3.2	Background	71
3.3	Methodology	75
3.4	Case Study	80
3.5	Discussion	89
3.6	Conclusion	91
4	Balancing Software Product Investment Options	93
4.1	Introduction	93
4.2	Background	95
4.3	Methodology	100
4.4	Case Study	106
4.5	Discussion	111
4.6	Validity Threats	112
4.7	Conclusion	113
5	Attitudes to Value and Risk in Requirement Selection	115
5.1	Introduction	116
5.2	Background	118
5.3	Experiment Planning and Operation	123
5.4	Results and Analysis	127
5.5	Conclusions and Future Work	130
	Bibliography	133
	Appendix	145
	Software Product Quality Definitions	145
	Investment Type and Constraint Definitions	147

Chapter 1

Introduction

1.1 Overview

Getting any product to a consumer today involves many different groups of people. Each of these groups is involved with a product in a different way, and values different things from the same product. Many of these desires on the product solution can work together, but it is possible for that the desires of one group conflict with the desires of another group. However, these conflicts must be resolved. If an organisation is able to successfully exploit the diversity of ideas, strengths and options of the different groups involved in a product's development can also provide value to potentially all of the groups involved.

Take the simple example of a hospital sheet. Patients and healthcare professionals want the solution primarily to be hygienic – making disposable plastic sheets an obvious solution. But the patient also wants it to be comfortable – disposable silk sheets meet both criteria, and additionally silk is hypoallergenic. But those responsible for paying for healthcare require a more affordable solution than single use silk sheets. Laundering the sheets does help reduce the cost dramatically, but silk is still expensive and will not survive as many wash cycles as many alternatives. Fortunately sheets made from cotton blends are cheaper than silk, last longer, can be cleaned sufficiently and are acceptable to most patients. It is also reasonable to expect that a nurse would like fitted sheets as they are much easier to put on a bed, but with the amount of laundering the elastic would wear out quickly, resulting in an unacceptably short life for each sheet. Similarly colours might also help brighten up a sick patients day,

but the amount of washing combined with the sufficient cleaning agents having a bleaching effect means dull off-white sheets would quickly become the norm. Thus hospitals use white cotton blend sheets that are regularly laundered.

As the example shows, there are many people who have a stake in the provision of a hospital sheet. Each of these groups has different aims and concerns in the provision of the solution, but ultimately these need to be reconciled. The final solution to this problem meets the aims of each group involved, but each group has made some compromise – the patient did not get the most comfortable material, the payer did not get the cheapest solution and the nurse did not get elasticised sheets. However, ultimately the solution was acceptable to all groups involved.

The development of a software product faces many similar challengers to the example given of the hospital sheet, but the situation is much more complex. There are many groups involved in the development of a software product – for example there are end users, product managers, system architects, project managers, developers, testers and trainers. As each of these groups is involved with a software solution in a different way, they see the product differently have potentially conflicting desires and expectations on the final solutions. But as each group brings with it expertise and experience. If these groups are effectively leveraged a greater product can be achieved for all groups involved.

Exploiting the strengths and viewpoints of other people in decision-making to achieve a better outcome has become a natural part of our society. Most people reading this thesis will relate to Figure 1.1¹, as they will have either helped friends and family purchase a computer, or asked someone to help them. A diverse group of people has a better ability to understand problems and find possible solutions.

Ultimately the aim is to deliver the maximum amount of value to the groups that will decide the success or failure of a product. Someone visiting a patient in a hospital in the above example will have some response to the sheets, but ultimately the needs of the patient, healthcare professionals and payer for the medical services will overrule these concerns as success of the solution does not depend on the visitor.

To complicate the situation, software is not like other products. Both software, and the resources used to create it are intangible. A substantial amount of the cost of the hospital sheet will cover growing, harvesting, weaving, bleaching and distribution of the cotton. Very little cost goes into recovering the cost of designing the final solution. The opposite is true of software, most of the cost

¹This figure is reprinted with permission from xkcd.com.



Figure 1.1: “Shopping Teams” cartoon from xkcd.com [87]

goes towards the design, as the manufacture and distribution of software is very cheap. However, despite the different nature of software, the successful development of software products requires that different stakeholders requirements are balanced, and an outcome that is acceptable to all of the groups upon who success depends is created.

This thesis aims to understand the priorities of the groups upon whom the success of a software product depends. There are a number of different sets of issues known to impact the development of software products, which can be explored in terms of the issues raised. Specifically this thesis examines factors influencing the selection and prioritisation of requirements for a release of software, priorities on software product quality, and the balancing of different types

of investment and constraints software companies face – particularly features, quality, intellectual capital (IC), time and cost.

Additionally this thesis looks at how these groups are aligned in terms of their priorities, and the relative influence of the different groups. In order to achieve these first two research objectives a method is developed that is able to elicit the required information and process the results in a meaningful way.

Finally, given the complex nature of the decisions in this area, this thesis also explores how bias in decision-making may skew these priorities. This thesis looks at how prospect theory could impact decision makers selecting requirements for a release of software.

An introduction to the literature in the areas studied in this thesis are provided in Section 1.2. The research questions addressed in this thesis are presented in Section 1.3. Information about the methodologies used are presented in Section 1.4, with the research setting given in Section 1.5. An overview of the chapters that make up this thesis is given in Section 1.6, with the major contributions of this thesis presented in Section 1.7 Future work is presented in Section 1.8. Finally a summary of the thesis is made in Section 1.9.

1.2 Background

This section introduces the key concepts and related work used in this thesis. It covers different types of investments that must be made in software products and defines value for software products. This section then covers techniques to prioritise investments and introduces value-based software engineering (VBSE). It concludes with an introduction to biases when making decisions, with a focus on those those that will influence the prioritisation of investments.

1.2.1 Software Product Investments

Software is different from most other products. When purchasing software, the consumer normally only receives a box containing a CD and a booklet, which of course does not justify the price. Which raises the question, ‘where does the rest of the money go?’ This situation clearly contrasts from that of a car, which comes with a complex set of assembled parts, many of which have been tailor made to the particular make and model of the car, and even to the consumers individual preferences. While physical components do not make up the whole cost of the car, it does create a different perception for consumers.

The forgotten cost in both cases is for the creation of the product – covering design, testing, certification and in the case of software coding. The main difference is that software results in a physically intangible product, where most others additionally result in a physically tangible one as well.

The key input and tool used in the development of a software product is IC [95]. For an organisation IC comprises of the knowledge, skills and experiences of employees; the structures and processes within the organisation; and the relationships between the organisation and external parties who have some interest in the organisation.

Within an organisation, software is most commonly developed in projects. Four control variables have been identified for the management of software development projects – features, quality, time and cost [13]. Of these, features and quality are areas in which investments can be made, while time and cost constrain the possible investment that can be made. Features refer to the functionality the software provides – what it is able to do, while quality describes how well it does these tasks.

These area of IC, features and quality are described in the following sections.

Intellectual Capital

The standard model of IC used in research today was developed by Bontis [28]. It divides IC into human capital, structural capital and relationship capital.

Human capital describes the value brought to an organisation by its employees [28]. It covers employees' knowledge, experience, skills and abilities. Investments a company can make in human capital include strategic hiring of personnel, training, job rotation and collaboration.

Structural capital is most commonly described as the knowledge the organisation has when everyone goes home [86]. It refers to the structures and processes that allow the organisation to meet market requirements – for example copyright, patents, trademarks and data stored on information systems. Investments in structural capital include organisational restructures and taking a strategic approach to intellectual capital protection.

Relationship capital recognises the value of the knowledge held in the relationships between an organisation and external parties. External parties include customers, suppliers, alliances/partners, community, regulatory and competitors [74]. The knowledge built up in a relationship with an external party is valuable, and that is why an existing customer can be more valuable than a new one.

All of these areas require investment. Failure to invest in IC will be detrimental to the long-term financial performance of an organisation.

Features

Shifting the focus from the organisation to the products it produces, features are the functional requirements that a software product fulfils. For example, Google's main feature is search functionality.

The purpose of features in a software product is to create business value [44]. However, developing new features comes at a cost for organisations – requirements for features must be elicited from various sources, analysed to ensure they are consistent and clear, conflicts between requirements must be reconciled, specified, designed, coded and tested [79].

When selecting requirements to implement in a release of a software product there are many factors that must be balanced to create software product value [109, 110, 6]. There is no silver bullet to select requirements to maximise the value of a software product. This is because value creation strategies are highly contextual [75] and must be analysed as part of a multidimensional array of variables [91].

Software Product Quality

Another product issue requiring investment is software product quality.

Quality is a complex and multifaceted concept. It can be considered from a number of different perspectives [45], but the most common perspectives to consider in the development of a software product are that of the user and manufacturer [58, 71]. From the users' perspective quality is defined as fit for purpose, while the manufacturing perspective defines quality as conformance to specification.

Only considering the user and manufacturing perspectives is dangerous. Value-based software engineering recognises that there are many groups upon whom the success of a software product depends. While these groups will be discussed in more detail later in Section 1.2, it is important that some key groups are represented when deciding on how to invest in software product quality [71].

There is no single correct answer for the level investment different aspects of software product quality should receive. The issues involved in the quality of a software product will depend on a number of factors that define the context of the product [71]. However, it is important to realise that less than perfect software may in fact be ideal [113], but deciding how much less than than perfect

is a business decision – insufficient investment in software product quality means the product will fail to satisfy, while too much investment results in costs that will not pay-off.

Constraints

It is critical to recognise that all of these investment options are bound by time and cost. No organisation has unlimited resources, making it necessary to balance different investment options against each other. Increasing the investment in one area reduces the possible investment in other areas.

1.2.2 Value

In order to achieve the correct balance on investment options in the development of a software product it is necessary to understand how value is created. The notion of value is a rich one, with the word being used in many disciplines and having a wide diversity of meanings.

In economics value is used to describe what a customer is willing to pay for a product – use value, and the market value of a product – exchange value [84]. A consumers perceived value of a product is defined as their perceived benefits from using the product divided by the perceived price, which can be compared with alternative product offerings [107].

If the perceived value is greater than the price the customer will view the purchase as providing value. If the price is greater than the cost then the software company will make profit on sales.

An organisations success does not necessarily correlate with the level of investment in IT [105]. In order to create value, a software company must align the business, product, project and relationship perspectives with the business strategies that drive value [5, 4, 101]. The success of software companies depends on their ability to develop products that meet consumer requirements, while offering value for money [27, 82].

1.2.3 Perspectives in Software Development

Already there have been a number of references to different groups and perspectives involved in the development of a software product. The need to reconcile these groups has been stated. Relationship Capital, one part of IC, recognises that an organisation works with and is influenced by many parties outside the

organisation. Definitions of software quality consider the consumers and developers of software, while research into the creation of value recognise the need to align software consumers, software development organisations, software products and projects to develop software products.

Each software development organisation will have its own internal structures, but there are many roles that are undertaken in the development of a software product. These include product managers, project managers, designers, programmers and testers. Each of these stakeholder groups will bring a different perspective and priorities to the software development process [68, 23]. For example, a project manager considers whether they can deliver the required functionality on time and budget, a software company considers its profits, while a user considers whether the product meets their needs.

The different value propositions by the groups involved in the development of a software product are often not compatible, and therefore must be reconciled [23]. To complicate matters, these groups are often not aware of their own value propositions, or are unable to express them concretely, so a concerted effort is required to elicit the desires of each group [2].

Figure 1.2 shows an example of the possible misalignment between three different perspectives involved in the development of a software product. It is based on the methodology presented in Chapter 4. In this example the product manager has promised new features to customers within a given time period. As the product manager is responsible for ensuring the product is profitable they have further placed a high priority on cost. However, the architects and developers in this example do not have the same interaction with customers, but are aware of a number of internal quality issues affecting the code. While it is possible these issues do not currently affect the product, they can impact on the developers and can impact the product further down the line, and may require training and new skills to address. For these reasons we can see differences in the priorities of the three groups, with a greater focus on features time and cost by the product managers, while the architects and developers want to focus relatively more focus on quality and IC.

1.2.4 Prioritisation Techniques

There are a number of different techniques that can be used to prioritise different factors in the software development process to elicit the requirements of different people and groups. While academic literature has focused on the analytic hierarchy process (AHP), there are a number of techniques that can be used [17]. This section covers AHP, cumulative voting (CV) and hierarchical

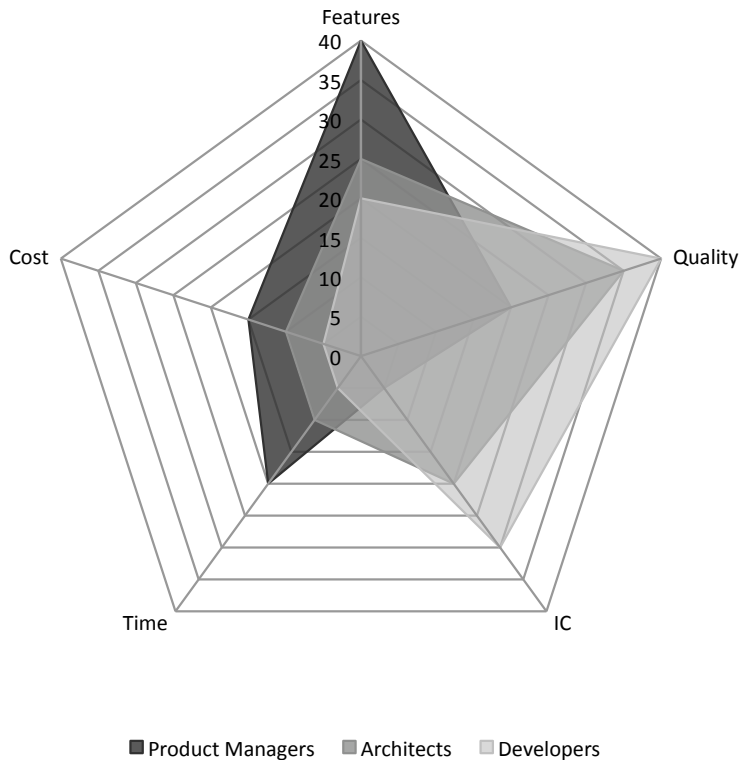


Figure 1.2: Misalignment between investment priorities of different perspectives

cumulative voting(AHP).

Analytic Hierarchy Process

AHP is a systematic decision-making technique that has been designed to solve complex problems with multiple aspects [15]. Users of this method make pairwise comparisons of all items being considered, rating their preference for one item over another on a nine-point scale. The technique recognises that people will not always be consistent in their use of the nine-point scale comparisons, and allows for this while calculating the users' final preferences. The final preferences are presented on a relative scale.

AHP was found unsuitable for prioritising large numbers of items due to the number of comparisons that must be made [15]. However, researchers have been able to develop techniques to remove redundant comparisons.

Cumulative Voting

CV is a simple and straightforward technique, where people are asked to split a specified number of points across a set of items provided [15]. This method is also known as the \$100 method, as people are commonly asked to split \$100 across a set of items to show their relative value.

For example, if someone thinks item *A* is twice as valuable as item *B* they may assign these items 20 and 10 points respectively, leaving 70 points to award to the remaining items. If they think item *C* has no value, they can assign it zero points (or \$0).

Like AHP, CV produces results on a relative scale [15].

Hierarchical Cumulative Voting

HCV is an extended version of CV. The CV technique asks participants to identify the relative priority of each item in a single list, while HCV asks people to complete a number of CV exercises and has a method for reconciling these separate lists into a single list showing the relative importance of each item.

Asking people to prioritise a single list of attributes can be troublesome as not all the items may be at the same level of extraction and may be difficult to compare as they are not directly related [11]. However, comparisons of this nature must be made – for example, when managers determine the level of investment in features, software product quality and IC.

In order to create a HCV exercise an author must first create a list of items for which they want to find the relative priority [16]. These items can then be logically grouped, creating a descriptive heading for each group. Each group then becomes its own CV exercise, with an additional CV exercise being generated from the list of descriptive headings. The relative importance of each item in the original list can then be calculated through multiplication of the items with their descriptive headings.

1.2.5 Value-Based Software Engineering

Being able to elicit the priorities of different groups is a key component of value-based software engineering (VBSE), which has become an area of increas-

ing interest to researchers. Software has traditionally involved a bespoke development for a single customer, but is increasingly moving towards market driven software product development – where a software product is made for the market-at-large. With bespoke development the customer was responsible for deciding what should go in to the software to create a satisfactory level of value. However, now the software companies need to take on this role as the sheer number of customers does not allow that level of involvement, and different customers will have different needs.

VBSE recognises that in order to maximise the value of a software product offering it is essential to reconcile the needs of different stakeholder groups [4, 109, 110]. Central to VBSE and this aim of reconciling the needs of different groups is Theory-W. Theory-W requires [26]:

1. Success-critical stakeholder groups to be identified – the groups upon whom the success of a product depends. This may change from development context to development context.
2. The requirements of these groups to be elicited.
3. Negotiation between the groups to create a win-win situation – this is a decision that all groups find acceptable. It is possible that it may not be an ideal outcome for each group individually, but it is important to consider these issues collectively.
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 lies together to understand each other's needs, compromise and agree. In order to be successful Theory-W must be managed to ensure the plans are achieved and any deviations from the plans are corrected [26]. 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 posed.

1.2.6 Decision-Making

It is important to recognise that people have biases, and these can affect the decisions that they make. When choosing between different investment options

there are a number of biases that may influence the decision made. These include [90]:

Hyperbolic discounting, which suggests people will have a preference for more immediate payoffs. With the aforementioned investment types this will most probably mean giving preference to features over quality and IC.

Prospect theory, which states that the relationship between money and value is not linear – a loss is felt more than an equivalent-sized gain, and doubling the size of a gain does not double the value it brings. This suggests decision makers will be conservative with revenue-generating decisions, and risk-taking with cost-generating decisions.

Omission bias, infers a tendency to judge harmful actions as worse than harmful inaction. As investment in quality and IC have immediate costs, but do not have immediate payoffs, it is likely this bias will lead to an under-investment in these areas.

The planning fallacy or tendency to underestimate task completion times, could also be impact the situation.

1.3 Research Questions

The development of software products involves a complex set of competing priorities and interests. Balancing these concerns is a difficult task, but is one that must be undertaken by software development companies. However, to manage this process and facilitate effective change, it is essential to be able to quantify the relative weight placed on different aspects that are managed. Thus the first main research question (MRQ) for this thesis is:

MRQ1: How can the priorities of success-critical stakeholders be determined in the software development process?

As the development of software products involves many different groups of people working together, it is of interest to see if they are working towards one single goal, or identify if they have competing priorities. Thus the second research question is:

MRQ2: Can the level of agreement between success-critical stakeholder groups be determined in terms of priorities on the software development process?

Further, as the decision-making process involved in prioritising different factors in the software development process is a human one, it is potentially subject to human biases. This raises the question:

MRQ3: How do biases affect the priorities placed on investment options in the development of software products?

The answer to these questions will allow researchers and practitioners to identify the priorities of different success-critical stakeholder groups, determine their alignment, and understand the biases that influence the balance as it is practised.

The chapters in which each MRQ is addressed is presented in Table 1.1.

Table 1.1: Relationship between chapters and main research questions

Chapter	MRQ		
	1	2	3
Chapter 2	×		
Chapter 3	×	×	
Chapter 4	×		
Chapter 5			×

1.4 Methodology

In order to answer the research questions posed in this paper it is essential to have a research methodology. A research methodology provides the link between research questions and the data used to answer them [77]. Thus a methodology must be chosen that will provide 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 methodology 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 [72]. 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 [97]. While the small sample size makes it difficult to impossible to generalise the results of case study research [73], the approach is useful in exploratory research where little is known about an area. Case studies are conducted within the natural setting [77]. 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 [77]. 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 employed to analyse the results.

Experiments

Experiments are situations created by researchers to investigate the relationship between different factors by controlling related variables [77]. 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 [77]. 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.

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 [77]. 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 [77]. 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 [77]. 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 research also needs to decide the subjects or context to be studied.

In 1994 Glass discussed the “software-research crisis,” concluding that the main problem with software research is that most is not relevant to industry [46]. 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, 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.

The number of systematic reviews of software engineering literature [69] has been growing steadily since 2004 [70].

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 1.2: Relationship between chapters and main research questions

Methods	Chapters			
	2	3	4	5
Quantitative	×	×	×	×
Qualitative	×			
Case study	×	×	×	
Survey	×			
Experiment				×
Industry	×	×	×	
Laboratory				×
Literature				

Chapter 2 used qualitative methods to discover how companies were 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.

In Chapter 3 a case study was conducted in industry using quantitative methods to determine the priorities placed on different aspects of software product

quality, and ambiguities in the management of these aspects. Qualitative analysis of the results was conducted in a workshop setting following the analysis of the quantitative data.

Chapter 4 presents an industrial case study using quantitative methods to determine the priorities on different investment types and constraints in the software development process. Again a workshop was held to analyse the results, employing qualitative methods.

Finally, Chapter 5 presents an experiment conducted in an academic setting to determine if prospect theory applies to the selection of requirements for a release of software. Quantitative methods were used to collect and analyse the data.

1.5 Research Setting

This thesis presents a number of empirical studies, set in both academic and industrial contexts. The three primary settings used in the collection of data are presented in the following subsections.

1.5.1 Anonymous Companies

Chapter 2 uses data collected from a set of companies. These companies are described here and in the corresponding chapter, but for reasons of confidentiality remain anonymous.

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

- The first company is based in Australia and operates primarily in the Australian market. The company provides a number of business services, which includes a number of software products. This company has been developing software for over ten years.
- The second company is based in Australia, 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 in 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 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 3 and 4 and Related Paper 3. Ericsson was active in this continued 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, one major software-intensive product has been the subject of the research activities presented in this thesis. It is developed within one part of the organisation, so the results are specific to this part of Ericsson and should not be considered representative of Ericsson more generally. The product is 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 an external perspective that is grounded in research to examine areas perceived beneficial to the company; providing the researcher with industry relevant topics, data and results.

1.5.3 Blekinge Institute of Technology

Unfortunately it is not always realistic or feasible to collect data from industry. Reasons of time, cost and risk can make it difficult-to-impossible to carry out some studies in this context. However, this does not mean that these studies do not have academic merit or the potential for industrial application. Experiments in an academic setting can help bridge the gap between ideas and industry being prepared to get involved.

Data was collected from students enrolled in a software metrics course as part of a software engineering programme at Blekinge Institute of Technology for the research presented in Chapter 5. The students involved in the research were considered to be well versed in concepts of market-driven software requirements engineering, with almost 95% completing their masters degrees and 69% having work experience in the software development industry. The students also come from a wide range of countries – Bangladesh, China, Germany, India, Iran, Jordan, Nepal, Nigeria, Pakistan and Sweden – giving the group collectively a wide range of academic, industrial and social experiences.

1.6 Overview of Chapters

This section provides a brief overview and highlights the major contributions of each chapter in this thesis.

1.6.1 Chapter 2: Understanding Requirements Selection

Much of the value of a software product is in the requirements it fulfils. 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 groups behind each perspective.

This chapter extends previous research by the author [6], and explores how creation of software product value through requirements selection is understood in an 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 CV task. The interviews seek to understand 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.

The results of the collective case study of anonymous companies show that two of the companies followed the same process changes to come to their current requirement management practices, despite coming from different countries and

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 than 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 that companies continue to improve in a step-wise process. Additionally, the values used to select requirements for a release vary with 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 profitable product; and the size of the customer base influences requirement selection.

Beyond this chapter, this research has been further extended, with another paper by the authors examining the values influencing the selection and prioritisation of companies operating in China [9]. 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.

1.6.2 Chapter 3: Balancing Software Product Qualities

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, software product 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.

The aim of Chapter 3 is to test a methodology proposed by the authors for determining the priorities placed on aspects of software product quality by the success-critical stakeholders and the degree to which these groups are aligned. This methodology additionally proposes a way to identify ambiguities in the responsibility and management of each aspect of software product quality.

A methodology is proposed using cumulative voting (CV) to determine the relative importance of factors drawn from ISO 9126. The pilot of this method found participants had trouble balancing some aspects, and the methodology was updated to use hierarchical cumulative voting (HCV) – which removes the need for direct comparisons between the areas participants felt unable, but still allows the areas to be brought back together for analysis.

The proposed methodology is applied to a case study within Ericsson. The methodology was able to elicit the values with respect to software product quality, and shows that the internal success-critical stakeholder groups are well aligned in the priorities placed on software product quality. Additionally the results show that in general qualities perceived as more important have fewer and less serious ambiguities in their responsibility and management, while the less important qualities have more ambiguities.

These results are encouraging on a number of counts. First, the proposed methodology was able to meet its objectives – identifying the values with respect to software product quality from success-critical stakeholders, determine the level of alignment between logical groupings, and determine ambiguities in management. While the results of Ericsson cannot be assumed to be indicative of the wider software development industry, they too are promising, with the success-critical stakeholder groups agreeing on the priorities of software product quality and ensuring that the most important things are the most effectively managed.

Given the success of the methodology, a follow-up study was conducted examining the priorities and alignment of success-critical stakeholder groups on aspects of IC [8]. Like Chapter 3, this paper also forms an input to Chapter 4.

1.6.3 Chapter 4: Balancing Software Product Investment Options

Developing a software product is more complex than just balancing software product qualities, but experience shows that people have trouble making comparisons between different types of investments. Features and software product quality are two obvious investments, but given IC is both the key input and tool in the development of software it is essential to develop this part of an organisation. Given that all investments are in some way constrained by time and cost, investing in any one area reduces the level of investment in another area – making it critical to get the balance right.

Chapter 4 aims to determine the relative importance of a number of key investment types – features, software product quality and IC – in the presence of

time and cost constraints. To achieve this goal, this study presents a methodology that aims to collect new data, and bring together the results from both the software product quality study presented in Chapter 3 and a study into IC [8].

HCV is used to bring together the priorities identified from previous studies, and allow them to be compared on one single scale.

The method is applied to the Ericsson case, and is able to bring together priorities identified in the different sets of data collecting to create a single prioritised list of investment types and constraints.

This work expands on the HCV method, and demonstrates that it is able to bring the results of separate studies together. This technique, for example, would allow specialists within particular areas to prioritise the factors closest to them, while still being able to create a complete prioritised list. The method supports people in making comparison between the relative importance of areas they would normally have trouble.

1.6.4 Chapter 5: Attitudes to Value and Risk in Requirements Selection

Finding a balance between commercial (customer specific, market pull and external quality requirements) and internal quality requirements is a recognised challenge in market driven software product development (MDSPD). Part of the problem is that any decision can be subject to human biases, and these biases can potentially result in decisions being made that do not lead to the optimal outcome. In order to address this challenge, and ensure that good decisions are made, it is important to understand the preferences and biases influencing decision makers selecting requirements for software releases.

Prospect theory has been successfully applied to many disciplines. Applying it to an MDSPD setting suggests decision makers will avoid risk when selecting between commercial requirements, take risk with internal quality requirements, and prefer commercial requirements over internal quality requirements to maximise their perceived value. This paper seeks to investigate this claim.

Chapter 5 presents an experiment into prospect theory, investigating whether the biases proposed by prospect theory can be seen operating in MDSP requirements engineering (RE). The methodology copies the original experiments in prospect theory, but places the student subjects in a MDSP RE setting.

The results indicate risk avoidance when dealing with commercial requirements, while greater risk is taken when dealing with internal quality requirements.

As this is the first paper to use prospect theory to explain requirements selection decisions, it presents an opportunity to educate people in the biases they bring to the RE process, and facilitate the creation of strategies for balancing the different requirements types.

1.7 Contribution of Thesis

The general contribution of this thesis is twofold:

- First is a methodology that is able to determine the value propositions of success-critical stakeholder groups and determine their alignment.
- Second is an insight into how the bias explained by prospect theory applies to the selection of requirements.

The methodology was developed to address the MRQ1 and MRQ2 – to determine the priorities of success-critical stakeholders in the software development process, and to determine the level of agreement between success-critical stakeholder groups in terms of priorities on the software development process. It is developed in Chapters 2, 3 and 4, and is successfully used to:

1. Elicit the motives behind the selection and prioritisation of requirements – addressing MRQ1 in Chapter 2.
2. Elicit the priorities on software product quality and determine the level of alignment between internal success-critical stakeholder groups – addressing MRQ1 and MRQ2 in Chapter 3.
3. Determine the relative priorities on features, aspects of software product quality, time, cost and aspects of IC – bringing together the results of Chapter 3 – addressing MRQ1 in Chapter 4.

The method has been shown to be robust – being successfully applied to a number of issues, and used to bring different issues together.

Being able to determine the relative priorities of different issues is a valuable capacity of the method. The research in Chapter 3 shows that people have trouble comparing issues of features, software product quality and project management – however, these are issues that decision makers must balance on a daily basis. Being able to both determine what is going on, and what people

perceive should be going on allows a deeper level on introspection and discussions between the success-critical stakeholder groups to achieve a better balance and hence better outcome.

The methodology can also bring a greater level of insight through the use of issue specific experts. Not all groups have the same knowledge and understanding of all parts of the software development process, making some groups more apt to describe an issue or set of issues. Similarly, it is unlikely any one group has a sufficient level of oversight on the entire software development process to provide the level of detail required on all issues. The methodology presented in Chapter 4 allows different groups to respond to different issues, and is able to bring these issues together at a higher level. This allows experts to describe issues at a high level of detail, but still make comparisons between issues that are not directly impacting the same groups.

This thesis presents the first industrial application of HCV to the best of the author's knowledge. While this method has been used in academic settings [16, 18], this thesis shows the method can be used by, makes sense to and can elicit the value propositions of industrial practitioners in describing their work setting.

In conducting this research a systematic biases in the selection of requirements appears to be occurring, adding an additional objective and research question to this research – MRQ3, to understand how bias impacts decision-making in a requirements engineering context. This question was addressed in Chapter 5, with a study to determine if prospect theory can be used to explain decisions made in the selection and prioritisation of requirements for a release of software.

The results indicate that prospect theory applies to the selection and prioritisation of requirements for a release of software. This will give decision makers a bias for functional requirements over non-functional requirements. This is because functional requirements are more commonly associated with gains, while non-functional requirements are more commonly associated with costs – at least in the short term.

Between functional requirements there will be a preference for lower risk, even if this means smaller revenues. Decision makers prefer to be responsible for a smaller and safer revenue stream than a larger and riskier one. Generating some revenue will reflect better on the decision maker, than taking a reasonable risk and getting nothing in return.

The same is not true when it comes to non-functional requirements. Here decision makers will take high risk options, as this reduces costs in the short term. However, this also increases the likelihood of greater costs in the future.

These insights provide a reminder that people responsible for the selection and prioritisation of requirements need to ensure that their decisions are aligned with the company strategy, and will help achieve short, medium and long-term goals and objectives.

1.8 Future Work

Future work was underway at the time of writing this thesis, but there remain a number of other research activities to be completed.

This thesis presents the first industrial uses of the hierarchical cumulative voting (HCV) method. This method potentially requires the use of weighing factors in bring together the different CV tasks that make up HCV [16], however, further guidelines are needed to support researchers implementing this method determine what weighting factors should be used. While some guidelines are presented by Berander and Jönsson [16], these authors also admit their guidelines are insufficient.

Expanding upon Chapter 3, a study was underway at the time of writing this thesis to determine the alignment of outsourced solution providers into the mix of success-critical stakeholders already presented. This use of outsourced solution providers has increased dramatically, and this has brought about a change in the work environment. Where a new employee is inducted into a support network of experienced developers, a subcontracted team potentially has no prior experience working on a product. While functional requirements are usually explicitly stated and contracted, this is much harder to do with non-functional aspects, making it even more important to ensure that outsourced solution providers are aligned with a company's strategy.

In the studies presented in this thesis, participants were asked both what they thought *was* happening today and what they thought *should be* happening today in terms of the priorities placed on software development. While a need for change was identified in each study, the reasons the groups studied have been unable to achieve this changes is unknown. In order to both better understand the current situations faced in the development of software, and help these groups achieve their aims and objectives it is essential to understand the barriers faced in prioritising investments in ways perceive most beneficial.

While this thesis presents a method for eliciting priorities, moving forward it is important to ensure that these priorities are aligned with the strategies of both the company and product in question. It is not sufficient just to ensure that the priorities realise the company and product strategies, as it is possible these

do not recognise the realities of sustainable software product development. It is also important to ensure that technical issues are appropriately balanced within the company and product strategies. To this end, this method can support the success-critical stakeholders determine what they want, to start a dialogue on why these requirements are needed, and how these factors should be balanced going forward.

Replication of these studies would allow any common problems or patterns in industry to be identified. This would allow researchers to focus on the areas causing the greatest conflict to industry.

On the topic of biases in the decision-making process future work needs to determine if industrial practitioners behave in the same manner as the students used in the experiment in Chapter 5. Further, as each product is different, determine if there are any other factors that influence the degree to which prospect theory biases the decisions made by software product managers.

Work also needs to be undertaken to determine what other biases affect decision makers in how they select and prioritise requirements for a release of software. While some biases that could impact the prioritisation of requirements are presented in Section 1.2.6, this list is not exhaustive and to the best of the author's knowledge has not been tested in a software engineering context.

Further it would be beneficial to understand the degree to which biases influence the prioritisation of requirements, and the situations where this leads to improved or sub-optimal outcomes. This information would allow decision support to reduce the negative effects of bias from decision-making processes.

1.9 Summary

This section provides an overview of this chapter. It reiterates the main concepts, relates each of the research objectives with the appropriate chapters and summarises the findings.

The sustainable development of a software product relies on both a number of groups working together, and appropriate level of investments in IC, features and quality. Finding an appropriate balance between the different investment options is a difficult task – each of the groups has a different view of the software, resulting in conflicts as to what is important.

However, a balance of investment options must be found, and the groups brought together to work towards a common goal. In order to understand this situation in industry, this thesis aims to find a method capable of eliciting the

priorities placed on aspects of IC, features and quality by groups critical to the success of a software product. Further the method should be able to identify the level of alignment between these groups.

In Chapter 2 the CV method is used to determine the priorities of success-critical stakeholder with respect to the selection of requirements for a release of software. Following the success of CV in Chapter 2, the same methodology was used to elicit priorities on software product quality in Chapter 3 – however, the participants found this method difficult to use, which lead to HCV to be employed successfully. The strength of HCV is then used to prioritise features, aspects of quality, aspects of IC, time and cost against each other in Chapter 4.

In conducting the studies associated with Chapters 2, 3 and 4 a possible systematic bias appeared to be occurring in the prioritisation of investment options. This is explored in Chapter 5, with an experiment testing the impact of prospect theory on requirements selection.

The results of Chapters 2, 3 and 4 find HCV to be a powerful method in eliciting the priorities of different groups on aspects of IC, features and qualities in the development of a software product. Not only can HCV be used to determine the level of alignment between the different groups, but this method allows experts to describe different areas of investment with the possibility to consolidate these responses together and produce a single view.

Chapter 5 indicates that prospect theory applies to the selection of requirements for a release of software. This means that decision makers will favour lower risk investments over ones that have a higher risk and greater reward. Conversely it means that the decision makers will preference high risk options in relation to cost, despite the possibility of incurring high costs later due to these decisions.

This thesis provides a method capable of determining the priorities on the level of investment on different options in the development of software products. While it is concluded that groups should be aligned on issues of quality, it is reasonable that the priorities on aspects of IC will change between groups due to the groups' different job functions. Further, it appears decision makers will be naturally biased towards certain decisions that can be defined in terms of losses and gains. Such biases need to be understood to ensure that decision makers can carry out their job in a balanced and unbiased manner.

Chapter 2

Understanding Software Product Requirements Selection

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.

2.1 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 [64, 32, 96]. 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 [22] 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 [31] 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 [27]. In other words, the alignment of product, project and business decisions is a major problem in the software industry.

The value-based approach in requirements engineering (RE) promotes the alignment of product, project and business decisions [5, 4], and the involve-

ment 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 [109, 110, 21]. Despite the fact that most release planning literature covers prioritization and dependencies between requirements [32, 64, 96], there has been little research into the criteria used in this decision-making process around requirement selection [109, 110, 59].

This chapter reports a continuation and extended version of the work presented in EuroMicro'06 [6]. 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. [6], 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 2.2 covers background knowledge on the value-based approach in software development. Section 2.3 presents the methodology used in the studies. Section 2.4 presents the results of data analysis. Section 2.5 provides a detailed discussion while Section 2.6 addresses validity threats. Finally, Section 2.7 concludes the chapter.

2.2 Background

2.2.1 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 [84]. '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 [102, 51].

The concept of the value-based approach in SE was introduced in late 1990s in the context of decision-making about product lines [43], managing investments in reusable software [44] and software economics [27]. 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 [110].

2.2.2 Defining Value

Value constructions in economic theory are based on customer satisfaction, loyalty and re-purchasing behaviour [51]. 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 [1]. 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. $perceivedvalue = perceivedbenefits/perceivedprice$, where $perceivedbenefits$ and $price$ are both measured relative to competing products [107]. 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 [52].

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 [27, 82]. 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 [5, 4, 101].

2.2.3 Value Based Approach in Requirements Process

Today software has a major effect on the cost, value and schedule of projects [22]. However, an organization's success in terms of profitability or market capitalisation does not necessarily correlate with their level of investment in IT [105]. 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 time-frames [22]. This is because projects are tracked by monitoring project cost and schedule [23]. 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 [22].

Favaro [44] 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 [47]. This is because value creation strategies are highly contextual [75] and must be analysed as part of a multidimensional array of variables [91]. 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 [75] 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 [23]. It is also important to consider that the value a market attaches to different requirements changes over time [81], but it is not understood how they will change [109, 110]. 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 [99] 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 [109, 110, 4]. 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 [47]. In this article we examine this topic in industry through three case studies.

2.2.4 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 [49] 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 [32]. 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 [81]. 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 [33]. Release planning can only be conducted after a product's require-

ments have been elicited, analysed and specified [110]. If release planning is done badly it increases risk [81]. 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. [64] suggested that release planning be approached through the prioritization of requirements. Carlshamre et al. [33] and Dahlstedt and Persson [39] furthered prioritization by recognising and accommodating the dependencies that exist between requirements. Wiegers [108] proposed a method of prioritization that recognised and combined benefits and penalties of proposed functionality between multiple stakeholder perspectives, relative costs and relative risks.

2.2.5 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 [5, 109, 110]. 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 [20]. 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 [109, 4]. 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 [68, 23]. 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 [23]. Boehm [23] 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 [2]. 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" [2]. Maurice et al. [81] 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 between an organisation and its stakeholder has found to create and improve the value of the organisation for shareholders [56]. While taking a more liberal definition of stakeholder management, Berman et al. [19] 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 [47]. Boehm [22] 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.

2.3 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. [6]. 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?

2.3.1 Research Model

The research framework in this study is built on Boehm's Theory W model [22] 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 [109, 110] 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 2.1. 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.

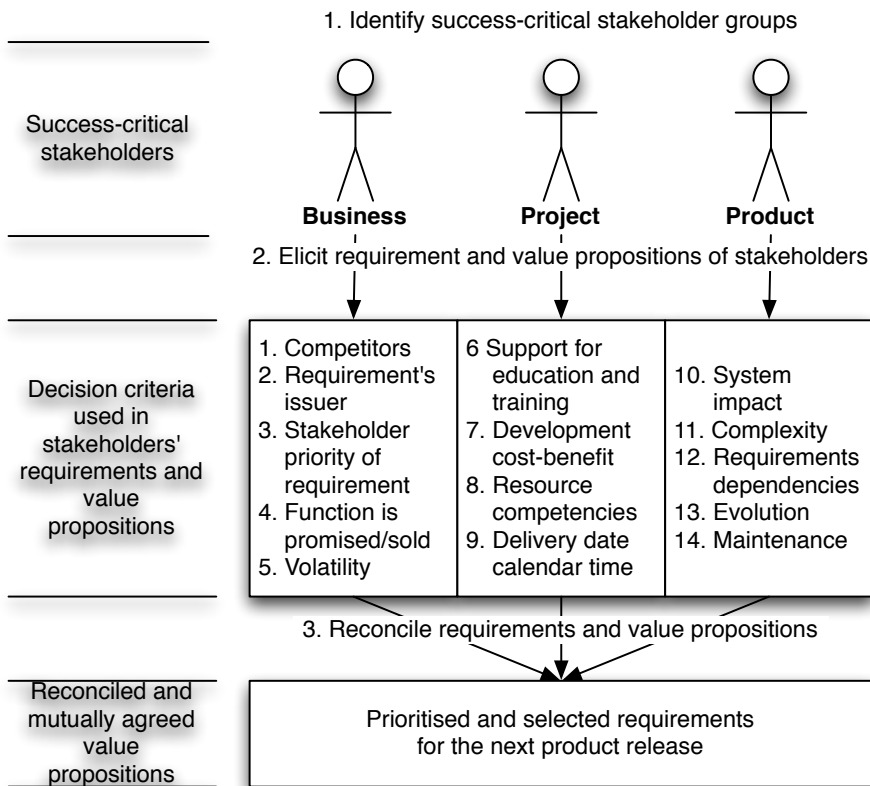


Figure 2.1: Value-based requirements engineering research model

This study used the criteria identified in Wohlin and Aurum [109, 110]. 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.

Additionally this chapter combined both the criteria identified by the Wohlin and Aurum [109, 110] 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” [77]. 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 [6] 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 planning 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 [6] 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 2.1, 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.

2.3.2 Case Studies

As illustrated in Table 2.1, 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.

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

Table 2.1: Descriptions of companies used in the collective case study

Company	Location	Market		Employees	
		Geographic	Industry served	R&D	Total
A	Australia	Australia	Telecommunications, finance, government, retail	200	1000
B	Germany	Europe	Telecommunications	All	100
C	Australia	Global	Corporate groupware, software development	All	55

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.

2.4 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.

2.4.1 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 were 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.

2.4.2 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 questionnaire is described in Table 2.2.

In their study in Sweden, Wohlin and Aurum [109, 110] 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 2.1.

Table 2.2: Questionnaire return rate

Product/Company	Distributed	Returned	Return Rate
Product A1	7	7	100%
Product A2	7	5	71%
Product B	16	9	56%
Company C	40	5	20%

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 2.3.

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 2.4.

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

Table 2.3: Relative importance of different criteria for Product A1 today and in future

Criteria	Perspective	Today (%)	Future (%)	Movement
3. Stakeholder Priority of Requirement	Business	11.5	10.4	-
2. Requirement's Issuer	Business	11.5	10.3	-
4. Function is Promised/Sold	Business	9.0	6.7	-5
11. Complexity	Product	8.9	6.4	-5
1. Competitors	Business	8.8	8.9	+1
10. System Impact	Product	7.8	8.8	+1
12. Requirements Dependencies	Product	7.6	7.3	-
7. Development Cost-Benefit	Project	7.3	9.8	+5
9. Delivery Date/Calendar Time	Project	7.3	7.8	+3
8. Resources/Competencies	Project	4.9	4.4	-3
14. Maintenance	Product	4.8	5.1	-
13. Evolution	Product	4.5	5.8	+2
6. Support/Education/Training	Project	3.4	3.8	-1
5. Volatility	Business	2.7	4.4	-3

Table 2.4: Relative importance of different criteria for Product A2 today and future

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

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 criteria considered of higher importance: (10) *the impact on the system*, and (11) *the complexity of the requirement*.

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 2.5.

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, ed-*

Table 2.5: Relative importance of different criteria for Product B today and future

Criteria	Perspective	Today (%)	Future (%)	Movement
4. Function is Promised/Sold	Business	14.1	9.6	-1
3. Stakeholder Priority of Requirement	Business	12.1	12.3	+1
2. Requirement's Issuer	Business	10.1	10.1	-1
9. Delivery Date/Calendar Time	Project	9.4	9.4	+1
7. Development Cost-Benefit	Project	9.1	9.1	-
8. Resources/Competencies	Project	6.6	6.6	-3
1. Competitors	Business	6.5	6.5	-3
11. Complexity	Product	5.9	5.9	-3
10. System Impact	Product	5.6	5.6	+1
12. Requirements Dependencies	Product	4.9	4.9	-4
14. Maintenance	Product	4.9	4.9	+5
13. Evolution	Product	4.6	4.6	+5
6. Support/Education/Training	Project	3.4	3.4	+3
5. Volatility	Business	3.0	3.0	-4

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 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.

comparison

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 2.6.

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 indi-

Table 2.6: Relative importance of different criteria in Company C today and future

Criteria	Perspective	Today (%)	Future (%)	Movement
3. Stakeholder priority of requirement	Business	11.2	10.7	-1
9. Delivery date/Calendar time	Project	9.8	7.8	-4
7. Development cost-benefit	Project	9.7	8.8	-1
10. System impact	Product	9.2	12.8	+3
12. Requirements dependencies	Product	8.3	5.3	-6
4. Function is promised/sold	Business	7.2	6.8	-1
8. Resources/competencies	Project	7.0	4.3	-5
11. Complexity	Product	6.7	5.4	-1
14. Maintenance	Product	6.7	8.6	+4
6. Support/Education/Training	Project	5.8	6.2	+2
2. Requirement's issuer	Business	5.3	3.3	-1
13. Evolution	Product	5.2	10.6	+9
5. Volatility	Business	4.1	4.1	-
1. Competitors	Business	3.8	5.3	+4

cate 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 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.

2.4.3 Perspectives Influencing Requirement Selection

The relative influence of the different stakeholder groups in each case can be seen in Table 2.7. The same information is also presented graphically in Figure 2.2 and Figure 2.3.

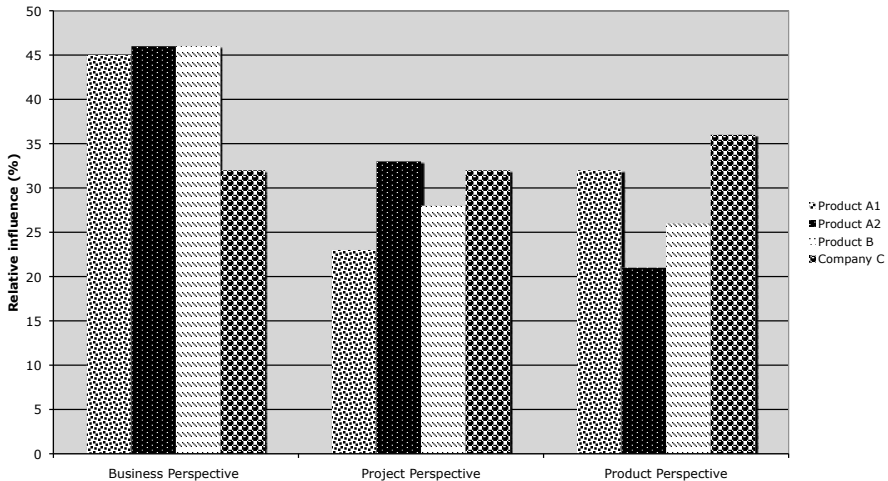


Figure 2.2: Stakeholders' perspective at business, project and product level (Today)

Table 2.7: Relative importance of different criteria in Company C today and future

Perspective	Product A1		Product A2		Product B		Company C	
	Today	Future	Today	Future	Today	Future	Today	Future
Business	45	42	46	40	46	40	32	30
Project	23	27	33	37	28	29	32	27
Product	32	31	21	22	26	31	36	43

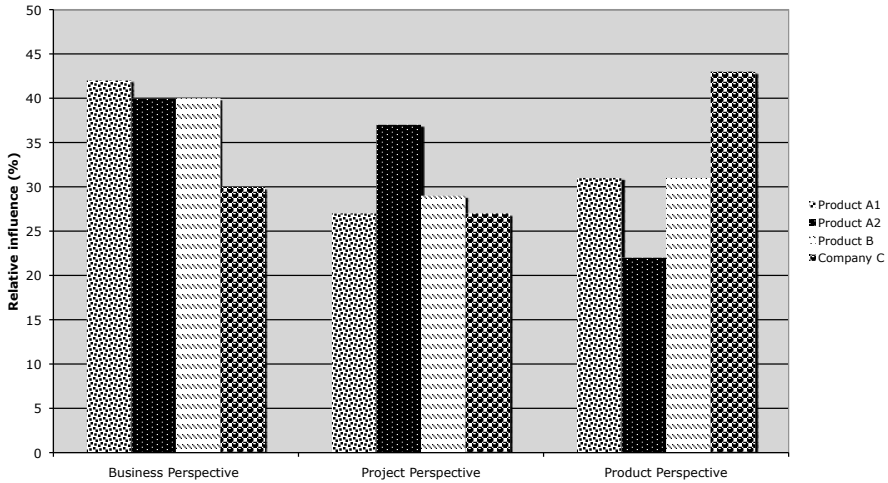


Figure 2.3: Stakeholders' perspective at business, project and product level (Future)

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.

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 2.7, 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 2.5.

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.

2.5 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.

2.5.1 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 “unknown waters”. This is inline with the research that says a companies strategy and product management will change with market demands [75].

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.

2.5.2 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.

2.5.3 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.

2.6 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 [111]. 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.

2.7 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 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 3

Balancing Software Product Qualities

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.

3.1 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 [13].

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 [71]. 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 [71]. 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 project will have time, resource and financial constraints, this information will allow the development effort to be focused in the most critical areas.

The importance of understanding who is responsible for setting quality goals and making sure they are achieved has also been recognised [71]. The management structures supporting features are generally more clearly defined than those supporting qualities. This creates a potential for key qualities to be overlooked, with no one taking responsibility or perceived responsibility.

The remainder of this chapter is outlined as follows. Section 3.2 presents

some background in terms of related work, objectives and the context of the case study. In Section 3.3, the method for studying qualities with respect to priorities and management in an industrial context is presented. The results of the case study are presented in Section 3.4. In Section 3.5, the applicability of the method and the findings from the case study are discussed. Finally, Section 3.6 presents the conclusions.

3.2 Background

In the software development process there are four variables that need to be controlled — cost, time, quality and scope [13]. 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 [13]. That said, quality does not need to be perfect [113] but the development process is much simpler when the success-critical stakeholders agree on what action should be taken [26].

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

3.2.1 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 [45]:

- *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. [71, 58]. 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 [26], while requirement specification reading techniques that take advantage of different perspectives have been found to catch 35% more defects than non-directed alternatives [12, 24], and value-based software engineering now recognises the value brought by different perspectives into the development process [71].

Software quality is not only defined by the relevant perspectives, but also by the context in which it exists [71]. 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 [113], but deciding how much less than perfect can only be decided in a given business context [71].

3.2.2 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 [71]. 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 [71]. 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 [83].

More recently work has been done to create an international standard for software product quality measurement — ISO 9126 [60]. 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 [83]. 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 [71].

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 [71]. 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 [40]. 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 [71]. 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.

3.2.3 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 [36]. 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 [54]. 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 [25]. Central to resolving conflict in VBSE is Theory-W, which requires [26]:

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 [26]. 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.

3.2.4 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?

3.3 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.

3.3.1 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 3.2 is a good starting point, but company specific needs have to be taken into account, as illustrated in the case study in Section 3.4.

3.3.2 Questionnaire

This method proposes the cumulative voting (CV) [78] 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.

3.3.3 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 3.1 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 3.1.
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

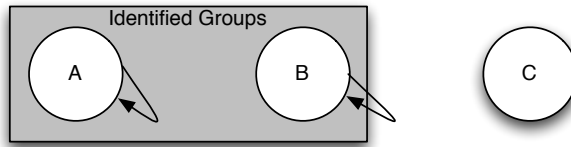


Figure 3.1: Example of (1b) pulling by multiple identified groups

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 3.2.
- (b) Combination of 2a and 3a, 3b or 3c.

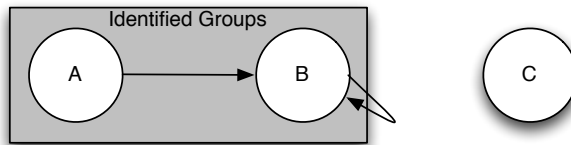


Figure 3.2: Example of (2a) push to pullers within identified groups

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 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 3.3.

- (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.

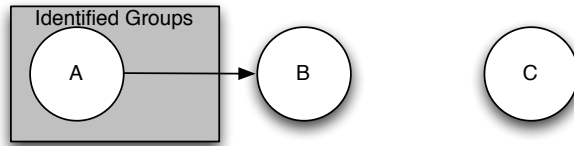


Figure 3.3: Example of (3a) push to a group not identified

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 3.4.

3.4 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.

3.4.1 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 3.1. 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.

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.

Table 3.1: Study response rate

Group	Candidates	Replacements	Responses
Strategic Product Management	15	1	6
Project Management	6	0	4
Tactical Product Management	9	0	9
Development and Testing	14	1	12
<i>Total</i>	<i>44</i>	<i>2</i>	<i>31</i>

3.4.2 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 development that are controlled. These are *time*, *cost*, *quality* and *scope* [13].

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 are reprinted in the appendix.

3.4.3 Pilot Study

A questionnaire was developed using the methodology described in Section 3.3 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 [16]. 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 [16]. 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 $category * quality * numberofqualities = 200 * 600 * 2 = 240,000$. 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 $result/totalsum * 1000 = 240,000/4,800,000 * 1000 = 50$.

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 [16].

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 [10].

3.4.4 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 3.2.

Table 3.2: Correlation matrix showing the degree to which the groups are aligned in how they perceive the priorities today

	SPM & PM	TPM	R&D
SPM & PM	1.00	0.80	0.90
TPM		1.00	0.86
R&D			1.00

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 3.3 show correlation values between 0.65 and 0.74 between the groups.

Table 3.3: Correlation matrix showing the degree to which the groups are aligned in how they perceive the priorities should be today (ideal)

	SPM & PM	TPM	R&D
SPM & PM	1.00	0.74	0.71
TPM		1.00	0.65
R&D			1.00

The similarities between the perceived situation today and the perceived ideal situation were striking. Looking at all responses the correlation between the two situations is 0.82. However, each group individually saw the need for more changes. The results in Table 3.4 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.

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.

Table 3.4: Correlation between perceived situation today and perceived ideal situation

Groups	Correlation
All groups	0.82
SPM & PM	0.72
TPM	0.77
R&D	0.62

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.

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*.

3.4.5 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 3.4, 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.

Ambiguity in who is ultimately responsible for or managing a quality can be presented in many different ways, as discussed in Section 3.3.3. 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 3.5, 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

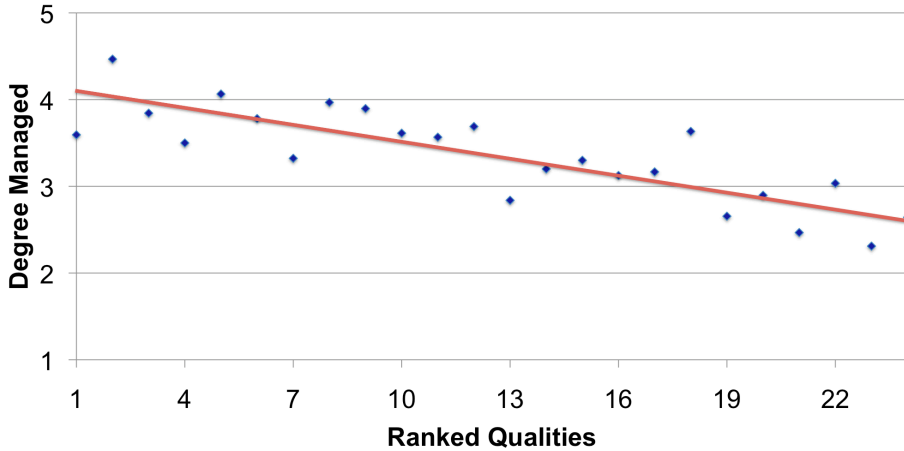


Figure 3.4: Importance of qualities today and the degree to which they are managed

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 3.5: Qualities with a significant number of blank responses

Quality (Rank: Today/Ideal)	Responsible	Managing
Maturity (19/23)	X	X
Understandability (21/19)	X	
Resource Behaviour (14/12)	X	
Analysability(24/22)	X	

Insufficient votes: Similarly four qualities received insufficient votes to pass the thresholds. The qualities are listed in Table 3.6, 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 3.6: Qualities with insufficient votes

Quality (Rank: Today/Ideal)	Responsible	Managing
Maturity (19/23)	11/23	
Understandability (21/19)	11/23	
Resource Behaviour (14/12)	12/27	
Analysability(24/22)	10/27	13/28

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 3.7 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 3.7: Pushing and pulling of ultimate responsibility and management

Management	Responsible	Managing
1a. Pull, single group	10	16
1b. Pull, multiple groups	7	6
2a. Push/Pull, push to puller(s)	6	2
2b. Combination of 2 & 3	0	0
3a. Push to unidentified group(s)	1	0
3b. Push to pusher(s)	0	0
3c. Combination of 3a & 3b	0	0

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 3.3, allows for identification of this type of situation and hence also appropriate actions.

3.5 Discussion

The methodology proposed in Section 3.3 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 3.4.3. 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 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 at-

tribute 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.

3.6 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 4

Balancing Software Product Investment Options

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.

4.1 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 [13] – *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 4.1. But people in the software development industry have trouble comparing the importance of the aspects that make up these areas [11]. 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.

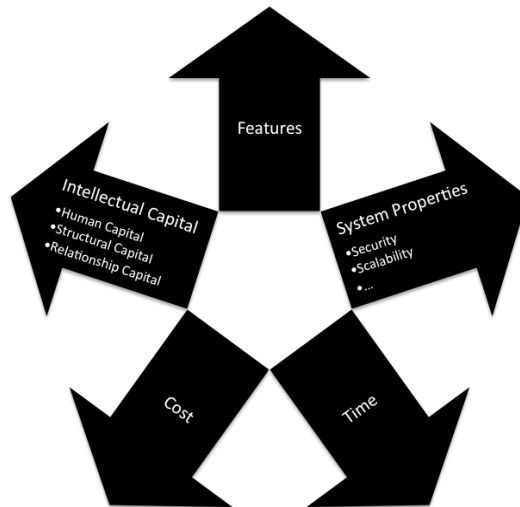


Figure 4.1: Conflict of investment types and constraints in software development

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 [11] and *IC* [8] 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 4.2 presents an overview of related work. Section 4.3 details the research questions and the method used to them. A case study is presented in Section 4.4, with a discussion of the results and method in Section 4.5. Threats to the validity are discussed in Section 4.6. Finally conclusions are drawn in Section 4.7.

4.2 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.

4.2.1 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* [13]. 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 [95]. *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 4.1. 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 [11], these are important decisions that managers must face. This only makes understanding the nature of these choices and decision support even more important.

4.2.2 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 [45]:

- *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 [58, 71]. But the increasing body of value-based software engineering (VBSE) literature recognises the importance of taking advantage of all perspectives involved in software development [71]. Central to VBSE is Theory-W [26], 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 [12, 24].

The concept of software quality also changes on the context in which it exists [71]. 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 [113], but deciding how much less than perfect can only be decided in a given business context[71].

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 [71]. 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 [71]. 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 [83].

More recently work has been done to create an international standard for software product quality measurement – ISO 9126 [60]. 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 [83]. 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 [71].

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 [71]. 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 [40]. 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 [71]. 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 *quality*, and any *quality* model must be considered in terms of other options, like *IC*.

4.2.3 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 [30]. Dow Chemical collaborated with Skandia to define the components of IC in terms of human capital, organisation

capital and customer capital [61].

Despite attempts to develop a general model of IC, organisations tend to customise models to suit their own context [50].

Four aspects of IC are defined by Brooking [29]: market assets, human centre assets, intellectual property assets and infrastructure assets. Lowendahl [80] identifies intangible assets in terms of competence and relational resources. Sullivan [103] 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 [28] 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 [104, 103, 41].

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 [85].

Over the years, several methods of measuring IC have also been developed. For example Skandia Navigator, Intellectual Capital Services' IC IndexTM and Philip M'Pherson's Inclusive Valuation Methodology (IVMTM) [57]. 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 [57].

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 [86, 34]. Most researchers have only examined aspects of these areas, recognising any list of attributes as incomplete. Moon and Kym [86] 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 [38]. 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 involve in the process as in many cases benefits achieved are difficult to quantify [38].

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 up-front investment on cumulative return in terms of money, effort and time [55]. Managers need to be able to understand the balance of the various aspects they are managing in order to make informed decisions.

4.3 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 4.4 presents the actual steps taken in the case study presented in this chapter.

4.3.1 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 4.1.

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 [11].

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 [11] and from an *organisational* perspective aspects of *IC* were studied [8]. 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 4.2, to understand how these areas interrelate in terms of importance. The areas covered in the case studies are discussed in more detail in Section 4.4, with a complete listing of the areas in Figure 4.4.

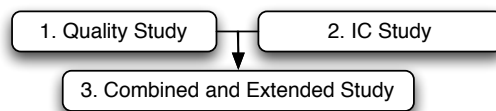


Figure 4.2: Study design

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?

4.3.2 Approach

To address the research questions a method has been developed, expanding on previous work by the authors in *quality* [11] and *IC* [8]. 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 4.4, 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 explain each of these phases in more detail.

4.3.3 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 4.2 is a good starting point, but company specific needs have to be taken into account, as illustrated in the case study in Section 4.4.

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 4.4. This model is discussed in more detail in Section 4.4.

4.3.4 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.

4.3.5 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 confus-

ing 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 4.3.

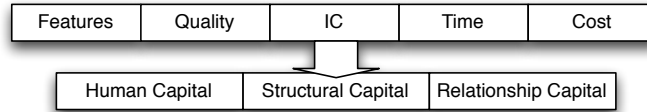


Figure 4.3: Example model of investment types and constraints

The hierarchical cumulative voting (HCV) method [16] 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 [11], 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 4.3, 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.

4.3.6 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 4.3 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 [16]. To ensure that aspects from CV tasks with many options are not underrepresented and that CV tasks with few aspects are not overrepresented 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:

$$\begin{aligned} IC \times \text{humancapital} \times \text{numberofaspectsofIC} \\ &= 200 \times 400 \times 3 \\ &= 240,000 \end{aligned}$$

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:

$$\begin{aligned} \text{result} \div \text{totalsum} \times 1000 \\ &= 240,000 \div 4,800,000 \times 1000 \\ &= 50 \end{aligned}$$

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.

4.3.7 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.

4.4 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* [11] and *IC* [8] – as shown in Figure 4.2. 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.

4.4.1 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 constraints, 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 engineering context.

Interview time constraints meant it was not possible to use the models exactly 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 hierarchical 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 [13] and the models created for the previous research into *quality* [11] and *IC* [8]. 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 4.4 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 4.4.

The first study asked participants to mark the relative importance of the detailed *quality* aspects, which are shown in the diagram without bolding. Similarly 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.

4.4.2 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 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 4.1.

4.4.3 Additional Analysis

The results of the *IC* study [8] contained several aspects of management that are outside the scope of this study as they are not directly related with *IC*. In order to manage this situation the aspects of management were removed. The

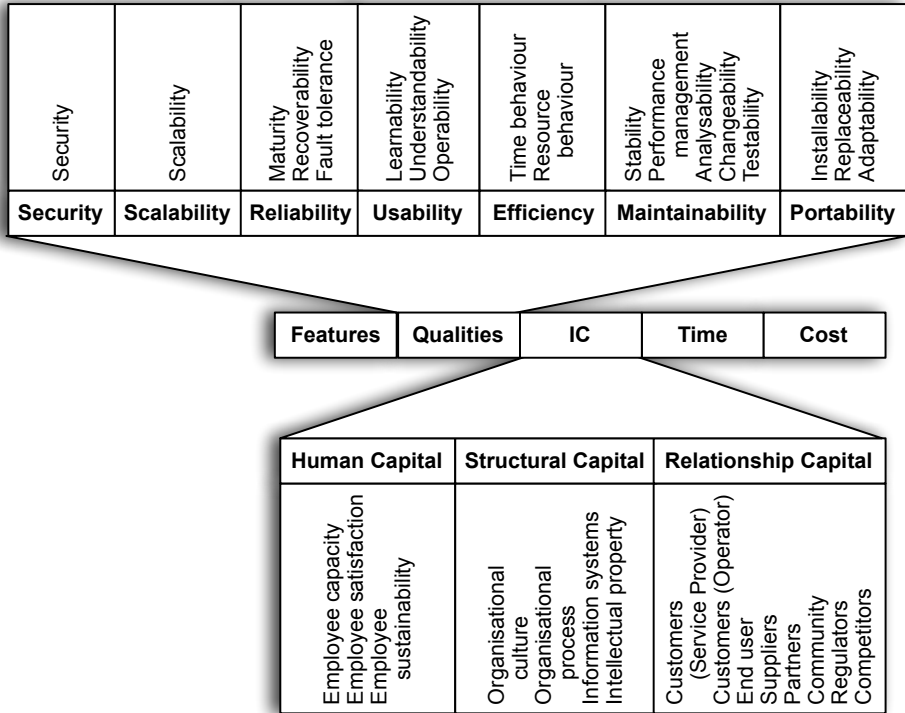


Figure 4.4: Model of investment types and constraints from case study

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 4.3.

4.4.4 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.

Table 4.1: Number and roles of participants in each study

Study	Participants	
	No.	Role(s)
Quality Study	31	Strategic Product Managers, Tactical Product Managers, Project Managers, Developers, Testers
IC Study	32	Strategic Product Managers, Tactical Product Managers, Project Managers, Line Managers, Developers, Testers
Combined & Extended Study	9	Line Managers

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 4.4. 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 4.4, 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 4.2.

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

Table 4.2: Priorities on investment types and constraints

Today	Ideal
1. Features	1. Time
2. Time	2. Features
3. Quality	3. Intellectual Capital
4. Cost	4. Quality
5. Intellectual Capital	5. Cost

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 4.3 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 4.4.

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 is split 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.

4.5 Discussion

The method presented in Section 4.3 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 [37], 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 [7, 11], but there is at least one exception [8].

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 [26].

4.6 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 [11], 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.

4.7 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 4.3 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 inline with research into management practices [37].

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.

Chapter 5

Attitudes to Value and Risk in Requirement Selection

Abstract

Finding a balance between commercial (customer specific, market pull and external quality requirements) and internal quality requirements is a recognized challenge in market driven software product development (MDSPD). In order to address this challenge it is important to understand the preferences and biases influencing decision makers selecting requirements for software releases. Prospect theory has been successfully applied to many disciplines. Applying it to MDSPD suggests decision makers will avoid risk when selecting between commercial requirements, take risk with internal quality requirements, and prefer commercial requirements over internal quality requirements in order to maximize their perceived value. This chapter seeks to investigate this claim. This chapter presents an experiment investigating whether the biases proposed by prospect theory can be seen operating in MDSP requirements engineering (RE). The results indicate risk avoidance when dealing commercial requirements, while greater risk is taken when dealing with internal quality requirements. As this is the first research paper to use prospect theory to explain requirements selection decisions, it presents opportunity to educate people in the biases they bring to the RE process, and facilitate the creation of strategies for balancing the different requirements types.

5.1 Introduction

Requirements engineering (RE) is a decision rich activity, and RE decision support as a field has received increasing attention as market-driven software product development (MDSPD) has taken off [88, 3, 92, 32]. Due to the growing popularity and complexity of MDSPD, it is important to understand how inherent biases influences decisions made in this area. This chapter explores one key bias presented through prospect theory, and how it impacts value and risk perception.

MDSPD is focused on the task of selecting a set of requirements that should be included in the coming releases of a product. The success of the product is measured in sales and is related to how much the product features are valued by the customers.

The requirements selection process in MDSPD is often perceived as a very complex activity. This complexity is explained by:

- A large number of requirements that need to be considered;
- A variety of different technical aspects that need to be taken into account before a selection of the requirements can be made; and
- The challenge associated with taking decisions based on the decision material of variable quality (such as uncertain cost and value estimations).

Thus, the authors of this chapter classify requirements selection as occurring in a high-risk environment.

Requirements selection in MDSPD often involves situations where a decision maker must choose between requirements of different types. Key requirement types include *commercial requirements* – originating from the market and key customers; and *system requirements* (internal quality requirements) – often connected with system maintenance, quality and architecture. The challenge here is to find a balance between commercial and system requirements, such as to allow satisfying the immediate market and customer needs, as well as assuring the healthy product architecture evolution.

Recent studies on MDSPD show that commercial requirements are perceived more important than system requirements [109, 110, 7, 9]. However, these studies also show that system requirements are generally undervalued. Considering the impact of requirements selection decisions on the business of a development company, it would be useful to have an understanding of the heuristics and cognitive biases behind these decisions, which according to the classical decision theory govern the decision outcome [90].

In this chapter prospect theory [63, 106] is used to explore biases behind requirement selection decisions in a MDSPD context. While this breaks normative assumptions about decision making in risky situations referenced in most engineering literature [76], decisions made by software project managers have found to be better modelled by prospect theory than the more common utility theory.

Prospect theory models decision makers' attitudes towards risk in terms of gains and losses [90]. It predicts that people will be risk adverse when selecting between options described in terms of gains, and risk taking when deciding between options described in terms of losses. Given that software requirements in MDSPD RE can be described in terms of revenues (gains) and/or costs (losses) it seems interesting to apply prospect theory in order to explain the choices of decision makers.

Assuming that in MDSPD commercial requirements are normally associated with revenue, while system requirements are associated with costs, applying prospect theory would suggest decision makers will take risks with system requirements. As these requirements are associated with cost, the preference is to delay spending in the short-term, despite the potential for increased costs later and thus taking a risk.

Conversely, since commercial requirements are associated with gains, decision makers will be more risk adverse when selecting and prioritizing these requirements. This means preference will be given to requirements with a more reliable revenue stream, even if the alternatives could yield greater ROI.

In the experiment presented in this chapter the authors investigate how well prospect theory applies to MDSPD RE. To the best of the author's knowledge this is the first time prospect theory has been applied to MDSPD. The primary objective is to investigate the decision maker's attitude towards risk, as well as perception of requirements value when it comes to choosing between alternatives of both cost and benefit.

The structure of this chapter is as follows; Section 5.2 presents the special characteristics of MDSPD and provides an overview of how prospect theory can be used to explain and model requirements selection decisions in an MDSPD environment; Section 5.3 details the research question and the experiment design; the experiment results and analysis are found in Section 5.4 and finally Section 5.5 presents conclusions and future work.

5.2 Background

The following section provides a brief summary of the characteristics of requirements selection process in MDSPD. This is followed by a presentation of prospect theory and an experiment that explore attitudes towards losses and gains. The section concludes with a discussion on how prospect theory can be used to explain requirements selection decisions in MDSPD.

5.2.1 Market Driven Software Product Development

In typical MDSPD the development organization is the main stakeholder and owner of a developed product. The product evolves over time with new functionality added and offered to the general market through product releases. In MDSPD the development organization takes most risks and is responsible for development costs of the product. Delivering the right set of functionality when the market is ready for it is critical for the success of the software product.

Market-driven development is largely product focused and many important activities are performed prior to the initiation of the development projects (i.e. pre-project). The goal of pre-project activities is to catch, analyse, select and plan requirements for future releases of the product. Pre-project activities should result in a prioritized list of requirements that are assigned to a specific release. These requirements are then realized through one or more development projects.

The importance of correct decision making pre-project is acknowledged and highlighted by both industry and academia. However, finding a perfect set of requirements is considered impossible due to a variety of different criteria that need to be taken into account and conflicting interests of stakeholders. Each stakeholder group is interested to see their requirement in the next release [66, 93, 89].

Different requirement types that are typically involved in requirements selection process are shown in Table 5.1 [92, 65, 48]. *Customer specific features* and *Market pull requirements*, which are often referred to as *commercial requirements* originate from the customers. In the case of *Customer specific features* the main stakeholder is one key customer, whereas for *Market pull requirements* the main stakeholder is often a marketing or sales unit representing a number of customers.

Innovation requirements can originate from *Key customers*, however mostly these requirements are defined by the R&D unit of a company.

The stakeholders of *External quality aspects* can be represented by both

Table 5.1: Requirement types in MDSPD

Requirement Type	Explanation
Customer specific features	Requirement originating from one of the key customers
Market pull requirements	Requirements originating from more than one customers that represent a certain market
Innovation	Requirement usually originating from the company's R&D. Examples of this may be innovative technology, patents, innovative features, etc.
External quality aspects	Requirements concerning usability, security, reliability and other quality issues
System requirements (Internal quality aspects)	These are system architecture specific requirements, connected to system architecture health, flexibility, and evolution.

external parties, such as customers and internal parties such as R&D. Finally, *System requirements* mainly focus on system architecture and evolution aspects and almost always originate from organisations' R&D units.

Recent studies on MDSPD show that commercial requirements are perceived more important than system requirements [109, 110, 7, 9]. However, these studies also show that system requirements are generally undervalued. The situation where system requirements are usually given a lower priority is troubling system engineers as it will lead to system architecture deterioration and collapse in the longer term, which can be connected to serious losses for the company.

Finding a trade-off and profitable balance between commercial and system requirements is very important for the success of a product. But before this trade-off can be found, the authors believe an understanding should be reached of the reasoning and decision frames that govern decision making process in MDSPD RE. This is where the authors believe prospect theory may be useful.

5.2.2 Prospect Theory and Related Works

Prospect theory was developed by Daniel Kahneman and Amos Tversky [63], for which Kahneman won a Nobel Prize in economics in 2002. It examines possible outcomes in terms of gains and losses to determine the value of these

outcomes [90]. When asked about the preferred choice between the “sure thing” and a risky option, it is assumed that the decision maker’s attitude to risk is revealed. If X is the monetary outcome, 0.5 is the probability of “winning” X and V is the value of a choice option, the preferences in Table 5.2 reveal the risk attitude of an individual.

Table 5.2: Risk preferences

Scenario	Risk Attitude	Risk Preference
$V(X) = V(0.5 * 2X)$	Risk neutral	Indifferent between the options
$V(X) > V(0.5 * 2X)$	Risk adverse	Prefers the “sure thing”
$V(X) < V(0.5 * 2X)$	Risk taker	Prefers the risky option

While prospect theory is built on utility theory, prospect theory differs in that it considers issues from within a reference frame. Utility theory looks at net wealth, where prospect theory considers gains and losses. Thus prospect theory can explain why a loss of \$500 is felt more strongly than a gain of \$500.

Prospect theory has many practical applications, even within software engineering. It has been used to understand the reasons behind software project escalation [67], explain why early involvement in software project bidding leads to higher quotes [62], and support with the pricing of software bundling [35]. What is interesting about prospect theory is the relationship between the outcome and value is not linear, as seen in Figure 5.1. The function is steepest around zero and flattens out as the losses and gains become greater. It is also steeper for losses than gains.

The properties of the function mean that \$500 loss is felt more strongly than a \$500 gain as the relative value for losses is greater than the value for a similar sized gain. This can be seen in Figure 5.1 by using the guide markings.

Given the choice between (a) a 50% chance of gaining \$1000 or (b) a sure gain of \$500, most people select the safer option, B [63]. Figure 5.1 shows that a doubling in gains is not equivalent to a doubling in value, so the value delivered by option B on average is actually less than option A (as the value derived from the average of no gain and a \$1000 gain is less than the value of a \$500 gain).

Conversely when presented with options to (a) lose \$1000 with 50% probability or (b) a sure loss of \$500, most people chose the riskier option A [63]. As more value is lost from \$0 to \$500 than from \$500 to \$1000, the expected value of the outcome is less than the guaranteed loss of \$500.

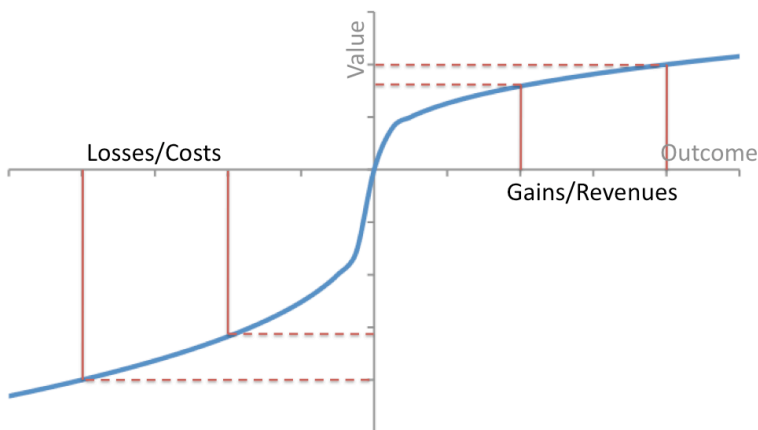


Figure 5.1: Value function from prospect theory

Another implication of prospect theory is that preferences will depend on how a problem is framed [90]. If the decision maker perceives the outcome as a gain they will accordingly act in a risk-averse manner, while acting in a risk-taking manner where the perceived outcome is a loss.

Framing has been found to play an important role in medicine. How the benefits of a vaccine are framed will yield very different take-up rates [76]. Patients who are told a vaccine is guaranteed to stop a virus are more likely to take it, than patients who are told a vaccine will stop two equally likely viruses with 50% probability.

However, these results would suggest that most people would not buy insurance. Replicas of the original experiments into prospect theory have found that most people act differently depending on whether the cost is framed as a gamble or an insurance [53, 100]. When a cost scenario is framed as insurance most people are risk averse preferring to pay the smaller fixed amount, while when it is presented as a gamble people prefer to take on the risk.

Kahneman and Tversky [63] also found that people overweighed low probabilities and underweighted higher probabilities. This explains why people are prepared to buy lottery tickets.

5.2.3 Using Prospect Theory to Explain Requirements Selection Decisions

Given prospect theory can be applied to decision-making processes in disciplines from finance to medicine [112], this section proposes how prospect theory can be used to explain the requirements selection decisions in MDSPD – in this context losses are described in terms of costs and gains in terms of revenues.

The requirements types presented in Section 5.2.1 are represented in Table 5.3 with information on whether each is perceived as generating costs or revenues. Commercial requirement types are marked with an asterisk, while system requirement types are not.

Table 5.3: Requirement associations and related risk

Requirement Type		Cost/Revenue Driver
Customer specific features	*	Revenue
Market pull requirements	*	Revenue
Innovation	*	Revenue
External Quality aspects	*	Revenue
System Requirements (internal quality)		Cost

Commercial and system requirements are obviously very different in nature. Commercial requirements are clearly associated with customers – and therefore with revenue, while system requirements are associated with costs. This result can be seen through the language used to describe the different requirement types. Commercial requirements are referred to as *business opportunities*, and are discussed in terms revenue generation and expanding markets. In contrast system requirements are discussed as costs and must be justified with either, “If we do not do this now it will lead to higher costs in the future,” or, “If we make this change now we will be able to lower our production costs.”

As commercial and system requirements can be described in terms of gains and losses, prospect theory has an obvious connection in how these different requirement types are valued. This is seen most clearly by replacing gains and losses with revenue and costs, and placing requirements of different types on the graph in Figure 5.1.

Looking at a commercial requirement with an expected gain, one can see that a doubling of the gain does not double the value of the requirement. Similarly halving the gain does not halve the value of the requirement. The results

from previous studies of prospect theory would therefore suggest that decision makers would be risk adverse when selecting commercial requirements, preferring guaranteed returns over larger riskier returns. Conversely decision makers would prefer to take more risk with system requirements, trying to minimize costs, even if these savings come at the risk of spending more money.

Different levels of risk are associated with the various requirement types. A customer specific requirement often has a customer willing to foot the entire cost, and one may reasonably assume that an individual customer is indicative of wider market demands. The return on investment of an innovation, however, does not have the same guaranteed revenue stream, but there is a potential for large revenues if the requirement proves popular. Internal system quality requirements are likewise risky as they involve investments in the longer-term sustainability of a software product. This could mean, for example, a requirement is developed allowing different database systems to be used with the product, but if this is never used the cost was unnecessarily incurred. It should also be noted that the customer is only aware of commercial requirements, with only the development organisation having clear visibility into internal system quality.

Given the clear link between prospect theory and MDSPD RE, the aim of the experiment presented in this chapter is to investigate whether prospect theory holds in an MDSPD RE setting. While the authors acknowledge MDSPD RE is a complex decision making setting, the focus of this chapter is on understanding the bias towards the perception of value as proposed by prospect theory, and how it may impact the decision making process.

5.3 Experiment Planning and Operation

This section presents the research question and major steps in the experiment planning and operation.

5.3.1 Research Question

The goal of the experiment presented in this chapter is to investigate if the results of the experiments conducted by Tversky and Kahneman [63, 106] can be seen in the MDSPD RE context - that is people are more risk taking when it comes to losses (costs) and risk adverse when it comes to gains (revenue).

In order to investigate this idea, the experiment follows the arrangement of the experiments conducted by Tversky and Kahneman [63, 106], but places

the participants in a MDSPD RE context. Thus the research question for the experiment is formulated as follows:

When selecting requirements, do product managers show more risk-adverse behaviour with requirements formulated in terms of revenue, than with requirements formulated in terms of cost?

5.3.2 Experiment Instrumentation

A questionnaire was developed to investigate whether the participants behave in a manner that can be predicted by prospect theory. A range of risk levels and monetary values was used. The questions included are presented in Table 5.4. In order to control the variable for value gains and losses are only described in monetary terms. While providing a richer context would better simulate reality, it is not easy to assess what value participants place on the different aspects that make up this context.

Each set of questions is designed in the similar style used in the original experiments by Tversky and Kahneman [63, 106]. Examples of the original questions can be found in Table 5.5.

From the questions presented in Table 5.4 and Table 5.5 it is easy to see the similarities between formulation of the questions from the authors' experiment and the original experiment. However, the question formulation differed between the original and authors' experiment with the original experiment using zero gain and zero loss as an option in each case. The effect of this difference in the design of the questions is discussed in Section 5.4.

A number of measures were undertaken to reduce systematic biases in the results. The order effect was addressed by systematically varying the order of the questions in the study. Additionally questions were placed between questions on revenue and cost to distract the participant from their previous answers.

A pilot of the questionnaire was completed prior to conducting the experiment. Two lecturers in software engineering completed the questionnaire, with both feeling they had a clear understanding of what was required, and were able to make informed decisions.

5.3.3 Experiment Subjects

The experiment involved 71 student participants completing either their final year of undergraduate studies or a masters in Software Engineering at

Table 5.4: Question sets for revenue and cost

Set	Revenue	Cost
1	<p>Select one requirement:</p> <p>A has a guaranteed revenue of €10,000.</p> <p>B could raise €2,400 revenue with 75% probability, and €32,800 revenue with 25% probability.</p>	<p>Select one requirement:</p> <p>A has a cost of €10,000.</p> <p>B could cost €200 with 30% probability and €14,200 with 70% probability.</p>
2	<p>Select one requirement:</p> <p>A has a guaranteed revenue of €10,000.</p> <p>B could raise €2,000 or €18,000 revenue with equal probability.</p>	<p>Select one requirement:</p> <p>A has a cost of €10,000.</p> <p>B could cost €2,000 or €18,000 with equal probability.</p>
3	<p>Select one requirement:</p> <p>A has a guaranteed revenue of €10,000.</p> <p>B could raise €200 revenue with 30% probability, and €14,200 revenue with 70% probability.</p>	<p>Select one requirement:</p> <p>A has a cost of €10,000.</p> <p>B could cost €2,400 with 75% probability and €32,800 with 25% probability.</p>

Table 5.5: Questions from related studies

Ref	Gain	Loss
[106]	Select one: <ul style="list-style-type: none"> • A sure gain of \$250. • A 25% chance to gain \$1000, and a 75% chance to gain nothing. 	Select one: <ul style="list-style-type: none"> • A sure loss of \$750. • A 75% chance to lose \$1000, and a 25% chance to lose nothing.
[63]	Scenario: In addition to whatever you own, you have been given \$1000. You are now asked to choose between the alternatives: <ul style="list-style-type: none"> • A 50% chance of gaining \$1000. • A sure gain of \$500. 	Scenario: In addition to whatever you own, you have been given \$2000. You are now asked to choose between alternatives: <ul style="list-style-type: none"> • A 50% chance of losing \$1000. • A sure loss of \$500.

Blekinge Institute of Technology, Sweden. The group consisted mostly of international masters students, who came from Pakistan, India, China, Iran, Nepal, Bangladesh, Nigeria, Germany and Jordan; with four undergraduate students, all Swedish.

Forty-nine of the participants had prior experience of working in the software industry. The Swedish students had experience from software engineering projects in terms of project courses which are run in tight cooperation with industry and very much resemble real development in industry. At the time of the experiment most of the students had completed the courses in Software Requirements Engineering and Software Verification and Validation, thus the students were assumed to be familiar with requirements engineering concepts.

5.3.4 Experiment Operation

The experiment was conducted at Blekinge Institute of Technology in the spring of 2008 during a single two-hour session.

Prior to starting the experiment the participants were introduced to the role of a software product manager and presented with the key performance indicators (KPIs) used to assess a person in this position. The list of KPIs included responsibility over the selection of requirements in a release to maximize the profit (return on investment) generated by the sales of a product. The KPIs remained visible for the duration of the experiment and the participants were encouraged to act in accordance to them.

5.4 Results and Analysis

This section presents the results, with an analysis and discussion.

5.4.1 Presentation of Results

The results of the experiment are presented in Table 5.6. The table shows the percentage of participants that chose the safe and risky options for the cost and revenue related questions in each of the question sets presented in Section 5.3.2.

Table 5.6: Participant responses (percentages)

Set	Revenue		Cost	
	Safe	Risk	Safe	Risk
1	0.69	0.31	0.49	0.51
2	0.68	0.32	0.68	0.32
3	0.56	0.44	0.52	0.48

As shown in Table 5.6, most participants were risk adverse when answering questions related to revenue, with most selecting the safer option. For the revenue questions presented in *Question Set 1*, 69% chose a safe option, in *Question Set 2* a similar 68% chose the safe option, while in *Question Set 3* 56% chose the safe option.

The results for the cost related questions show only a small difference in preference between the safe and risky options in *Question Set 1* and *Question*

Set 3. However, in *Question Set 2* the participants were more risk adverse, which was the opposite of what was expected (68% safe).

Comparing the answers between the revenue and cost related questions, the results show increased risk taking attitude in cost related questions for *Question Set 1* (from 31% to 51%) and *Question Set 3* (from 44% to 48%). The attitude towards risk is unchanged for *Question Set 2*.

5.4.2 Analysis

The results for revenue related questions show a preference for avoiding risk in each and every case, demonstrating alignment with aligned with the original experiment conducted by Tversky and Kahneman [63]. While the results showed more risk-taking behavior in the questions related to cost, the results were not as strong as the original prospect theory experiments.

The level of risk avoidance is higher for the revenue questions in *Question Set 1* and *Question Set 3* than for the questions regarding cost in the same question sets. When it comes to cost related questions, two of the three tested cases (*Set 1 & Set 3*) do not show a strong preference to take or avoid risk. These combined results, when considered relative to one another, indicate a more risk-adverse behaviour with requirements formulated in terms of revenue, than with requirements formulated in terms of cost.

The results for *Question Set 1* and *Question Set 3* show a clear change in attitudes towards risk between requirements termed as costs and requirements termed as revenue. While the result for *Question Set 2* did not show a strong difference, the majority of the cases support the application of prospect theory to software requirements selection.

5.4.3 Discussion

The results of the experiment indicate that decision makers will be more risk adverse when choosing between requirements formulated in terms of revenue, compared to when choosing between requirements formulated in terms of cost. This provides ground for concluding that prospect theory can be used to understand and explain decision making in MDSPD requirements selection situation.

The observed attitude towards risk taking for cost related questions was not as strong as expected, however, aspects of this may be explained by the experiment design. The participants of the experiment are students and do not have the same sense of ownership and responsibility towards the money of a real product manager. This could mean that they were not as sensitive to losing

money. For example, looking at the cost related questions in *Question Set 2*, it is reasonable to assume that the students did not see the value of taking the risky option because it was not their own budget, own reputation or job position that was at stake. Another difference is that in original experiments in prospect theory, the subjects were offered to decrease the loss to zero, which may have motivated more to risk as other psychological factors are involved with zero [98].

The authors expect an experiment involving professional software product managers would be more aligned with the original experiments as people in this role have a greater sense of ownership in the product for which they are responsible and will face a greater loss in reputation for failing to meet budget than the experiment participants. This assumption is supported by findings of the study on application of prospect theory on software project decisions, where project managers' decisions were found to be aligned with prospect theory [76].

5.4.4 Validity Threats

Internal and external validity threats are most important to address for experiments in software engineering field [111] and social sciences [94].

Internal validity is concerned with identifying whether the observed result of the experiment is influenced by other factors than the experiment treatment. For the experiment presented in this chapter an ordering effect (learning) and experiment instrumentation (the way questions are formulated) are the most significant threats.

Treatment order effect would mean the order the questions are presented will affect the participants' answers. To minimize this threat, the order of cost and revenue questions were systematically varied in the study.

To minimize the risk associated with the question formulations, a pilot of the experiment was conducted involving three participants. The intention of the pilot was to discover ambiguities and improve the formulation of the questions.

External validity is associated with the possibility to generalize the outcome of the experiment. While using students as experiment subjects usually puts a threat for generalizing the results [14], most of the students participating in the experiment have prior industrial experience and are trained in the requirements engineering field at a masters level. However, the participants may not have felt the responsibility for the success of the product in the same way that a product manager in industry would. As discussed earlier in Section 5.4.3 this provides ground for assuming that an experiment involving professionals will be more aligned with the original experiments in prospect theory.

MDSPD RE operates in a much more complex setting than that modelled in this experiment, potentially impacting the generalisability of the result. For example, while it can be argued that software requirement selection decisions are more commonly group decisions, and not up to individuals as modelled in this experiment. However, the application of the results in a group situation should still be possible as research has shown that individuals' attitudes to risk are translated into a groups attitudes to risk [42]. Similarly, while requirements are more complex than questions of cost and revenue, in order to control the participants perceived gains and losses and how these impact value, the problems were reduced to monetary terms.

The results presented in this chapter are inline with other findings from industry, recognising the importance commercial requirements with a fixed return over more variable market opportunities [7]. Similarly, literature recognises the risk-taking attitude towards system requirements [109, 110, 7, 9].

5.5 Conclusions and Future Work

The intention of the research presented in this chapter is to investigate if the ideas behind prospect theory, one of the prominent theories in decision making, can explain requirements selection decisions in market-driven software product development (MDSPD). The experiment presented in this chapter replicates the design of the original experiments into prospect theory, but places it in the context of requirements selection situations in MDSPD.

The experiment results show a clear potential for applying prospect theory in MDSPD setting. The participants consistently displayed risk adverse behaviour when selecting between requirements described in terms of revenue. In two of the three cases investigated the participants were more risk taking when selecting between requirements described in terms of cost when compared to the revenue situation. The increase in risk taking attitude was not as distinct as anticipated. However, the authors' expect that an experiment involving professionals would show larger difference between risk taking approach between the revenue and cost related situations.

To the best of the authors' knowledge the work presented in this chapter is the very first application of prospect theory in MDSPD requirements selection context. The insights found in this chapter provide contributions both to researchers and practitioners in this field, as they open up possibilities for explaining the behaviour of decision makers in different requirements selection situations. This provides an answer to why recent studies observe internal quality

requirements being consistently undervalued compared to commercial requirements, as well as help in finding ways for managing the balance between different types of commercial requirements originating from market pull and technology push.

Looking more closely at commercial requirements, prospect theory suggests preference favour for the guaranteed revenue stream over more uncertain options. Looking at the different requirements types, this would translate to a preference for customer specific features over innovation related requirements, as predicting market demands brings with it more risk.

The situation for system requirements is more complex. As these requirements are viewed as cost-drivers, they cannot be directly compared with the other types of requirements that are perceived as delivering revenue. This highlights the need to describe system requirements in terms of the gains that they deliver, so that a comparison between requirements of different types can be made. However, it should be noted that the risk associated with system requirements is still higher – so even when described in terms of gains, natural preference will be given to less riskier options. Putting this in MDSPD context implies that unless there is a clear strategy for balancing commercial and internal quality requirements, the later will be consistently down-prioritized until the point when the architecture issues will pose an impediment for the production of commercial features.

A number of actions should be undertaken as future work. A follow-up study is planned involving software product managers from companies working in a market-driven software development context, helping address issues faced in the experiment presented in this chapter. Additionally this work should be used to educate software project managers to the biases they bring to the development process, and will be used as an input to a model to help software product managers balance requirement types when selecting requirements for a release.

Bibliography

- [1] D. Alwis, V. Hlupic, and G. Fitzgerald. Intellectual capital factors that impact of value creation. In *Proceedings of the 25th International Conference on Information Technology Interfaces*, pages 411–416, June 2003.
- [2] J. C. Anderson and J. A. Narus. Business marketing: Understand what customers value. *Harvard Business Review*, 76(6):53–65, November–December 1998.
- [3] A. Aurum and C. Wohlin. The fundamental nature of requirements engineering activities as a decision-making process. *Information and Software Technology*, 45(14):945–954, 2003. Eighth International Workshop on Requirements Engineering: Foundation for Software Quality.
- [4] A. Aurum and C. Wohlin. A value-based approach in requirements engineering: Explaining some of the fundamental concepts. *Requirements Engineering: Foundation for Software Quality*, pages 109–115, 2007.
- [5] A. Aurum, C. Wohlin, and A. Porter. Aligning software project decisions: A case study. *International Journal of Software Engineering and Knowledge Engineering*, 16(6):795–818, December 2006.
- [6] S. Barney, A. Aurum, and C. Wohlin. Quest for a silver bullet: Creating software product value through requirements selection. *32nd EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA)*, pages 274–281, Aug. 2006.
- [7] S. Barney, A. Aurum, and C. Wohlin. A product management challenge: Creating software product value through requirements selection. *Journal of Systems Architecture*, 54(6):576–593, 2008.

- [8] S. Barney, A. Aurum, and C. Wohlin. Balancing aspects of intellectual capital. In *Submitted to Euromicro*, 2009.
- [9] S. Barney, G. Hu, A. Aurum, and C. Wohlin. Creating software product value in china. *IEEE Software*, 26(4), July–August 2009.
- [10] S. Barney and C. Wohlin. Software product quality questionnaire. <http://www.bth.se/tek/aps/sba>, 2008.
- [11] S. Barney and C. Wohlin. Software product quality: Ensuring a common goal. In Q. Wang, R. Madachy, and D. Pfahl, editors, *Proceedings of the International Conference on Software Process (ICSP)*, pages 256–267, 256–267, May 2009.
- [12] V. R. Basili. Evolving and packaging reading technologies. *Journal of Systems and Software*, 38(1):3–12, 1997. Achieving Quality in Software.
- [13] K. Beck. *Extreme Programming Explained: Embrace Change*. Addison-Wesley, Reading, Massachusetts, 2000.
- [14] P. Berander. Using students as subjects in requirements prioritization. *International Symposium on Empirical Software Engineering (ISESE)*, pages 167–176, August 2004.
- [15] P. Berander and A. Andrews. Requirements prioritization. *Engineering and Managing Software Requirements*, pages 69–94, 2005.
- [16] P. Berander and P. Jönsson. Hierarchical cumulative voting (hcv) — prioritization of requirements in hierarchies. *International Journal of Software Engineering and Knowledge Engineering*, 16(6):819–849, December 2006.
- [17] P. Berander, K. Khan, and L. Lehtola. Towards a research framework on requirements prioritization. In *Proceedings of the Sixth Conference on Software Engineering Research and Practise in Sweden (SERPS)*, pages 39–48, 2006.
- [18] P. Berander and M. Svahnberg. Evaluating two ways of calculating priorities in requirements hierarchies — an experiment on hierarchical cumulative voting. *Journal of Systems and Software*, 82(5):836–850, 2009.
- [19] S. L. Berman, A. C. Wicks, S. Kotha, and T. M. Jones. Does stakeholder orientation matter? the relationship between stakeholder management models and firm financial performance. *Academy of Management Journal*, 42(5):488–506, October 1999.

-
- [20] M. Berry and A. Aurum. Measurement and decision making. *Value-Based Software Engineering*, pages 155–177, 2006.
- [21] S. Biffi, A. Aurum, B. Boehm, H. Erdogmus, and P. Grünbacher, editors. *Value-Based Software Engineering*. Springer Berlin Heidelberg, 2006.
- [22] B. Boehm. Value-based software engineering: Overview and agenda. *Value-Based Software Engineering*, pages 3–14, 2006.
- [23] B. Boehm. Value-based software engineering: Seven key elements and ethical considerations. *Value-Based Software Engineering*, pages 109–132, 2006.
- [24] B. Boehm and V. Basili. Software defect reduction top 10 list. *Computer*, 34(1):135–137, January 2001.
- [25] B. Boehm and A. Jain. An initial theory of value-based software engineering. *Value-Based Software Engineering*, pages 15–37, 2006.
- [26] B. Boehm and R. Ross. Theory-w software project management principles and examples. *IEEE Transactions on Software Engineering*, 15(7):902–916, 1989.
- [27] B. W. Boehm and K. J. Sullivan. Software economics: a roadmap. In *ICSE '00: Proceedings of the Conference on The Future of Software Engineering*, pages 319–343, New York, NY, USA, 2000. ACM.
- [28] N. Bontis. Assessing knowledge assets: a review of the models used to measure intellectual capital. *International Journal of Management Reviews*, 3(1):41–60, 2001.
- [29] A. Brooking. *Intellectual Capital: Core Asset for the Third Millennium*. Cengage Learning EMEA, 1998.
- [30] M. Bucklew and L. Edvinsson. Intellectual capital at skandia. *The Foundation for Performance Measurement*, 1999.
- [31] J. Bullock. Calculating the value of testing: How to present testing as a business-process investment. *Software Testing and Quality Engineering*, 2(3):56–61, May–June 2000.
- [32] P. Carlshamre. Release planning in market-driven software product development: Provoking an understanding. *Requirements Engineering*, 7(3):139–151, 2002.

- [33] P. Carlshamre, K. Sandahl, M. Lindvall, B. Regnell, and J. Natt och Dag. An industrial survey of requirements interdependencies in software product release planning. In *Proceedings of the Fifth IEEE International Symposium on Requirements Engineering*, pages 84–91, 2001.
- [34] E. Carson, R. Ranzijn, A. Winefield, and H. Marsden. Intellectual capital: Mapping employee and work group attributes. *Journal of Intellectual Capital*, 5(3):443–463, 2004.
- [35] W.-L. Chang and S.-T. Yuan. A markov-based collaborative pricing system for information goods bundling. *Expert Systems with Applications*, 36(2):1660–1674, 2009.
- [36] L. Chung, B. Nixon, E. Yu, and J. Mylopoulos. *Non-Functional Requirements in Software Engineering*. Kluwer Academic, 2000.
- [37] M. D. Cohen, J. G. March, and J. P. Olsen. A garbage can model of organizational choice. *Administrative Science Quarterly*, 17(1):1–25, March 1972.
- [38] Cranfield. A report on the costs and the benefits of measuring intellectual capital assets. Technical report, Cranfield University School of Management, 2004.
- [39] Å. G. Dahlstedt and A. 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.
- [40] R. Dromey. Concerning the chimera. *Software, IEEE*, 13(1):33–43, January 1996.
- [41] L. Edvinsson. Developing intellectual capital at skandia. *Long Range Planning*, 30(3):320–321, 366–373, 1997.
- [42] H. S. Farber and H. C. Katz. Interest arbitration, outcomes, and the incentive to bargain. *Industrial and Labor Relations Review*, 33(1):55–63, October 1979.
- [43] S. R. Faulk, R. R. Harmon, and D. M. Raffo. Value-based software engineering (vbse): a value-driven approach to product-line engineering. In

- Proceedings of the First Conference on Software Product Lines: Experience and Research Directions*, pages 205–223, Norwell, MA, USA, 2000. Kluwer Academic Publishers.
- [44] J. Favaro. Value based management and agile methods. *Extreme Programming and Agile Processes in Software Engineering*, pages 1016–1016, 2003.
- [45] D. A. Garvin. What does “product quality” really mean? *Sloan Management Review*, 26(1):25–43, 1984.
- [46] R. Glass. The software-research crisis. *IEEE Software*, 11(6):42–47, November–December 1994.
- [47] J. Gordijn and J. M. Akkermans. Value-based requirements engineering: exploring innovative e-commerce ideas. *Requirements Engineering*, 8(2):114–134, 2003.
- [48] T. Gorschek and C. Wohlin. Requirements abstraction model. *Requirements Engineering*, 11(1):79–101, 2006.
- [49] D. Greer and G. Ruhe. Software release planning: an evolutionary and iterative approach. *Information and Software Technology*, 46(4):243–253, 2004.
- [50] D. Han and I. Han. Prioritization and selection of intellectual capital measurement indicators using analytic hierarchy process for the mobile telecommunications industry. *Expert Systems with Applications*, 26(4):519–527, 2004.
- [51] K. Heinonen. Reconceptualizing customer perceived value: The value of time and place. *Managing Service Quality*, 14(2/3):205–215, 2004.
- [52] S. Henneberg, C. Pardo, S. Mouzas, and P. Naudé. Value dimensions and strategies in dyadic ‘key relationship programmes’. In *Proceedings on the 21st IMP Conference*, 2005.
- [53] J. C. Hershey and P. J. H. Schoemaker. Risk taking and problem context in the domain of losses: An expected utility analysis. *Journal of Risk and Insurance*, 47(1):111–132, 1980.

- [54] G. Herzwurm, S. Schockert, and W. Pietsch. Qfd for customer-focused requirements engineering. *Requirements Engineering Conference, 2003. Proceedings. 11th IEEE International*, pages 330–338, September 2003.
- [55] W. A. Hetrick, C. W. Krueger, and J. G. Moore. Incremental return on incremental investment: Engenio’s transition to software product line practice. In *OOPSLA ’06: Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications*, pages 798–804, New York, NY, USA, 2006. ACM.
- [56] A. J. Hillman and G. D. Keim. Shareholder value, stakeholder management, and social issues: what’s the bottom line? *Strategic Management Journal*, 22(2):125–139, January 2001.
- [57] T. J. Housel and A. H. Bell. *Measuring and managing knowledge*. McGraw-Hill/Irwin, Boston, Mass., 2001.
- [58] R. W. Hoyer and B. B. Y. Hoyer. What is quality? *Quality Progress*, 34(7):53–62, July 2001.
- [59] G. Hu, A. Aurum, and C. Wohlin. Adding value to software requirements: 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, December 2006.
- [60] ISO9126. Software engineering – product quality – part 1: Quality model. International Standards Organization, 2001.
- [61] A. Jashapara. *Knowledge Management: An integrated Approach*. Financial Times Prentice Hall, Harlow, 2004.
- [62] M. Jorgensen and G. J. Carelius. An empirical study of software project bidding. *IEEE Transactions on Software Engineering*, 30(12):953–969, December 2004.
- [63] D. Kahneman and A. Tversky. Prospect theory: An analysis of decision under risk. *Econometrica*, 47(2):263–291, 1979.
- [64] J. Karlsson, C. Wohlin, and B. Regnell. An evaluation of methods for prioritizing software requirements. *Information and Software Technology*, 39(14–15):939–947, 1998.

- [65] L. Karlsson, A. G. Dahlstedt, J. Natt och Dag, B. Regnell, and A. Person. Challenges in market-driven requirements engineering — an industrial interview study. In *8th International Workshop on Requirements Engineering — Foundation for Software Quality (RefsQ)*, <http://crinfo.univ-paris1.fr/REFSQ/02/default.htm>, September 2002.
- [66] L. Karlsson, B. Regnell, J. Karlsson, and S. Olsson. Post-release analysis of requirements selection quality post-release analysis of requirements selection quality — an industrial case study. In *9th International Workshop on Requirements Engineering — Foundation for Software Quality (RefsQ)*, <http://crinfo.univ-paris1.fr/REFSQ/03/papers/REFSQ03-PreProceedings.pdf>, June 2003.
- [67] M. Keil and R. Mixon. Understanding runaway it projects: Preliminary results from a program of research based on escalation theory. *Proceedings of the Twenty-Seventh Hawaii International Conference on System Sciences*, 3:469–478, January 1994.
- [68] E. Keller. The value of software. *Manufacturing Systems*, 13(1):16, 1995.
- [69] B. Kitchenham. Procedures for undertaking systematic reviews. Joint technical report, Computer Science Department, Keele University (TR/SE-0401) and National ICT Australia Ltd. (0400011T.1), 2004.
- [70] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman. Systematic literature reviews in software engineering - a systematic literature review. *Information and Software Technology*, 51(1):7–15, 2009. Special Section - Most Cited Articles in 2002 and Regular Research Papers.
- [71] B. Kitchenham and S. Pfleeger. Software quality: The elusive target. *IEEE Software*, 13(1):12–21, January 1996.
- [72] B. Kitchenham, S. Pfleeger, L. Pickard, P. Jones, D. Hoaglin, K. El Emam, and J. Rosenberg. Preliminary guidelines for empirical research in software engineering. *IEEE Transactions on Software Engineering*, 28(8):721–734, August 2002.
- [73] B. Kitchenham, L. Pickard, and S. Pfleeger. Case studies for method and tool evaluation. *IEEE Software*, 12(4):52–62, July–August 1995.

- [74] D. J. Knight. Performance measures for increasing intellectual capital. *Strategy and Leadership*, 27(2):22–27, 1999.
- [75] P. Kotler. The major tasks of marketing management. *Journal of Marketing*, 37(4):42–49, October 1973.
- [76] T. W. Lauer. Software project managers' risk preferences. *Journal of Information Technology*, 11(4):287–295, 1996.
- [77] P. D. Leedy and J. E. Ormrod. *Practical Research: Planning and Design*. Prentice Hall, Upper Saddle River, N.J., 8th edition, 2005.
- [78] D. Leffingwell and D. Widrig. *Managing software requirements : a unified approach*. Addison-Wesley, Reading, Massachusetts, 1999.
- [79] P. Loucopoulos and V. Karakostas. *System requirements engineering*. McGraw-Hill Book Co, 1995.
- [80] B. Lowendahl. *Strategic Management of Professional Services Firms*. Handelshöjskolens Forlag, Copenhagen, 1997.
- [81] S. Maurice, G. Ruhe, O. Saliu, and A. Ngo-The. Decision support for value-based software release planning. *Value-Based Software Engineering*, pages 247–261, 2006.
- [82] D. G. Messerschmittis and C. Szyperski. Marketplace issues in software planning and design. *IEEE Software*, 21(3):62–70, May–June 2004.
- [83] D. Milicic. *Software Quality Attributes and Trade-Offs*, chapter Software Quality Models and Philosophies, pages 3–19. Blekinge Institute of Technology, 2005.
- [84] J. S. Mill and W. J. Ashley. *Principles of political economy : with some of their applications to social philosophy*. Longmas Green & Co., New York, 1909.
- [85] Montague. Measuring intellectual capital. Technical report, Montague Institute, 2008.
- [86] Y. J. Moon and H. G. Kym. A model for the value of intellectual capital. *Canadian Journal of Administrative Sciences*, 23(3):253–269, September 2006.

- [87] R. Munroe. Shopping teams, August 2007.
- [88] A. Ngo-The and G. Ruhe. Decision support in requirements engineering. *Engineering and Managing Software Requirements*, pages 267–286, 2005.
- [89] A. Ngo-The and G. Ruhe. A systematic approach for solving the wicked problem of software release planning. *Soft Computing — A Fusion of Foundations, Methodologies and Applications*, 12(1):95–108, 2008.
- [90] S. Plous. *The Psychology of Judgement and Decision Making*. McGraw Hill, 1993.
- [91] V. Poladian, S. Butler, M. Shaw, and D. Garlan. Time is not money: The case for multi-dimensional accounting in value-based software engineering. In *Fifth Workshop on Economics-Driven Software Engineering Research (EDSER-5)*, May 2003.
- [92] B. Regnell and S. Brinkkemper. Market-driven requirements engineering for software products. *Engineering and Managing Software Requirements*, pages 287–308, 2005.
- [93] B. Regnell, L. Karlsson, and M. Höst. An analytical model for requirements selection quality evaluation in product software development. *Proceedings of the 11th IEEE International Requirements Engineering Conference*, pages 254–263, September 2003.
- [94] C. Robson. *Real World Research: A Resource for Social Scientists and Practitioner-researchers*. Blackwell Publishing, 2002.
- [95] G. Roos and J. Roos. Measuring your company’s intellectual performance. *Long Range Planning*, 30(3):325, 413–426, 1997.
- [96] G. Ruhe and D. Greer. Quantitative studies in software release planning under risk and resource constraints. *Empirical Software Engineering, 2003. ISESE 2003. Proceedings. 2003 International Symposium on*, pages 262–270, September–October 2003.
- [97] P. Runeson and M. Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 14(2):131–164, April 2009.
- [98] K. Shampanier, N. Mazar, and D. Ariely. Zero as a special price: The true value of free products. *Marketing Science*, 26(6):742–757, 2007.

- [99] M. Shaw. Everyday dependability for everyday needs. In *Supplemental Proceedings of the 13th International Symposium on Software Reliability Engineering (ISSRE)*, pages 7–11. IEEE Computer Society, November 2002.
- [100] P. Slovic, B. Fischhoff, and S. Lichtenstein. *Question framing and response consistency*, chapter Response mode, framing and information processing efforts in risk assessment. New directions for methodology of social and behavioral science. Jossey-Bass, San Francisco, 1982.
- [101] D. Starovic, S. Cooper, and M. Davis. Maximising shareholder value: Achieving clarity in decision-making. Technical report, The Chartered Institute of Management Accountants, 26 Chapter Street, London SW1P 4NP, United Kingdom, November 2004.
- [102] K. Storbacka and J. R. Lehtinen. *Customer Relationship Management: Creating Competitive Advantage Through Win-Win Relationship Strategies*. McGraw-Hill Education, December 2001.
- [103] P. H. Sullivan. *Profiting from intellectual capital: Extracting value from innovation*. Wiley, New York, 1998.
- [104] K.-E. Sveiby. Intellectual capital and knowledge management. Technical report, Sveiby Knowledge Management, 1998.
- [105] J. Thorp. *The Information Paradox: Realizing the Business Benefits of Information Technology*. McGraw-Hill, Toronto, 2003.
- [106] A. Tversky and D. Kahneman. The framing of decisions and the psychology of choice. *Science*, 211(4481):453–458, 1981.
- [107] A. Weinstein and W. C. Johnson. *Designing and delivering superior customer value : concepts, cases, and applications*. St. Lucie, Boca Raton, Fla., 1999.
- [108] K. Wieggers. First things first: Prioritizing requirements. *Software Development*, 7(9):48–53, 1999.
- [109] C. Wohlin and A. Aurum. What is important when deciding to include a software requirement in a project or release? *International Symposium on Empirical Software Engineering*, pages 246–255, November 2005.

-
- [110] C. Wohlin and A. Aurum. Criteria for selecting software requirements to create product value: An industrial empirical study. *Value-Based Software Engineering*, pages 179–200, 2006.
 - [111] C. Wohlin, M. Höst, P. Runeson, M. C. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in Software Engineering: An Introduction*. Springer, 2000.
 - [112] G. Wu and A. B. Markle. An empirical test of gain-loss separability in prospect theory. *Management Science*, 54(7):1322–1335, 2008.
 - [113] E. Yourdon. When good enough software is best. *IEEE Software*, 12(3):79–81, May 1995.

Appendix

Software Product Quality Definitions

This section details the terms and definitions used in the study into investment types and constraints in Chapter 3.

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.

Investment Type and Constraint Definitions

This section details the terms and definitions used in the study into investment types and constraints in Chapter 4.

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 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: This covers both the time taken and the computing 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 modifications 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 (modifications to interfaces to adapt to a changing environment), perfective maintenance (enhancements to create new or modify existing functionality) 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 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: 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 individuals 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 individuals communication skills, entrepreneurship and creativity.

Employee satisfaction: The difference between the employees 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 employees 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 employees conscientiousness and willingness to engage in lifelong learning, increasingly critical attributes in the current dynamic marketplace. As it is difficult to change an individuals 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 organisations 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 (eg. patents, trademarks, copyright) and other documented materials (eg. 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.